Walking GPS: A Practical Solution for Localization in Manually Deployed Wireless Sensor Networks

Radu Stoleru, Tian He, John A. Stankovic
Department of Computer Science
University of Virginia
{stoleru, tianhe, stankovic@cs.virginia.edu}

Abstract

In this paper we present the design, implementation and evaluation of a simple, practical and cost effective localization solution, called Walking GPS, that can be used in real, manual deployments of WSN. We evaluate our localization solution exclusively in real deployments of MICA2 and XSM motes. Our experiments show that 100% of the deployed motes localize (i.e,. have a location position) and that the average localization errors are within 1 to 2 meters, due mainly to the limitations of the existing commercial GPS devices.

1. Introduction

Wireless Sensor Networks (WSN) have generated a lot of interest recently. The interest, initially in the academic community, was due to the challenging research problems posed by these devices with very limited resources (e.g. memory, processing power, radio bandwidth, energy). More recently, due to the tremendous potential of WSN in military, civil and industrial applications, real deployments of these networks have become imminent.

Many elegant and clever solutions have been proposed and evaluated in simulators and real system deployments, for several of the problems present in the WSN. Among these problems are energy conservation, efficient data placement and aggregation, programming paradigms and topology control. However, despite the attention it has received, accurate localization is a problem that remains unsolved in a real, ad-hoc deployment, without sophisticated, expensive hardware. In this paper we present a solution to this problem, when manual deployment is an option.

In many applications it is envisioned that WSN will be deployed from Unmanned Aerial Vehicles. In the meantime, manual deployments have been prevalent and the employed localization solutions have used some variant of associating the sensor node ID with prior knowledge of that ID's position in the field.

We propose in this paper a solution, called Walking GPS, in which the deployer (either person or vehicle) carries a GPS device that periodically broadcasts its location. The sensor nodes being deployed, infer their position from the location broadcast by the GPS device.

The main contribution of this paper is that we present the design, implementation and the real world evaluation of a solution for the localization of wireless sensor nodes that are deployed manually in the field. Our solution is simple, cost effective and has very little overhead. Despite the simplicity of the idea, many system design and implementation issues had to be addressed in order to make the solution work and be efficient. We further hypothesize that the lessons learned from our experience can be extended to aerial deployments, for an initial, coarse localization.

The rest of the paper is structured as follows. In the second section we present related work, emphasizing research that focused on real system implementations. We present the system design and architecture of our Walking GPS localization solution in section three and the implementation of the system in section four. We present our extensive experimental results in section five and conclude in section six by summarizing the main contributions and propose future extensions of our solution.

2. Related Work

Recent developments, driven by both industrial and academic research, continuously expand the applicability of sensor networks at an unprecedented momentum. The inherent interaction between sensor networks and the physical world makes location-awareness one of the essential services for many emerging applications such as location-based directory service (e.g., GHT) [1] and entity tracking [2]. In response, many algorithms have been proposed to address the localization problem in sensor networks.

Among these solutions, some of them are designed under certain assumptions and mostly evaluated in simulation environments. For example, the Amorphous positioning algorithm proposed in [3] uses offline hop-distance estimations and multilateration to estimate nodes' locations, assuming an isotropic RF radio. The APIT positioning algorithm [4] is a scheme in which a node infers its position based on the possibility of being inside or outside of a triangle formed by any three anchors . In this scheme, more powerful anchors, that cover the entire deployment area, are required. In the Received Signal Strength Indicator (RSSI) techniques (RADAR [5] and SpotOn [6]), either theoretical or empirical models are used to translate signal strength into distance estimates. These approaches show promising results in simulation and controlled laboratory environment. However, in practice, many empirical studies [7] [8] [9] demonstrate that in most environments, RF radio is not isotropic and there is, actually, no connection between the signal strength degradation and the distance an RF signal travels.

Another set of solutions use Time of Arrival (TOA) [10] and Time Difference of Arrival (TDOA) [1] [11] [12] techniques to obtain pair-wise distances. These techniques demonstrate high accuracy in localization in real deployment. However, they require extensive infrastructure support, such as GPS in [10], high per-node cost as in Cricket [11] [12] and AHLos [13]. In addition, the limited range of ultrasound (normally 7 – 10 meters) used in TDOA, imposes the requirement of a high-density anchor nodes, which makes the overall system cost for localization prohibitively high for large scale sensor networks.

Recently, mobile robots have been utilized as an effective instrument to localize sensor nodes [14] [15]. Although the ideas are similar to those in this paper, due to the significant discrepancy in the deployment configuration, their designs currently require a fine-tuning of parameters, in order to achieve a high accuracy. This tuning may prove to be problematic for a non-expert.

In view of various limitations exhibited by current localization schemes, we aim to find a practical solution that not only provides high localization accuracy in a running system, but also requires very low system cost.

3. System Architecture

In this section, we present the architecture of our system, and the design decisions we encountered. We support our decisions with background material and a description of the internals of our software components.

An alternative to the Walking GPS localization scheme is enabling each sensor node with GPS capabilities. This monolithic solution is both expensive and inefficient.

In the Walking GPS architecture, however, the system is decoupled into two software components: the GPS Mote and the Sensor Mote. A UML deployment diagram is shown in Figure 1.

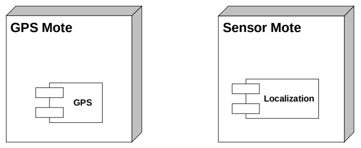


Figure 1. Software components for the Walking GPS scheme

The GPS Mote runs on a MICA2 mote. The mote is connected to a GPS device, and outputs its location information at periodic intervals. The Sensor Mote component runs on all sensor nodes in the network. This component receives the location information broadcast by the GPS Mote and infers its position from the packets received.

This architecture enabled us to push all complexity derived from the interaction with the GPS device to a single node, the GPS Mote, and to significantly reduce the size of the code and data memory used on the sensor node. Through this decoupling, a single GPS Mote is sufficient for the localization of an entire sensor network, and the costs are thus reduced.

A relatively simple design for the GPS Mote would have been to periodically broadcast the actual GPS location received from the GPS device. A GPS location is represented by a latitude and a longitude, which are angular measures from the Equator to North or South, and Prime Meridian to East or West, respectively. Due to the relatively small size of a sensor network (hundreds to a few thousand meters), the use of global (i.e. GPS) coordinates is very inefficient. The inefficiency stems from the size of the packets used for passing location information – a significant portion of the location is likely to be the same for all sensor nodes – as well as from the computational costs encountered when aggregating data, e.g., triangulation of several GPS coordinates for positioning a target.

In order to reduce the overhead incurred when exchanging data containing global GPS coordinates (the GPS coordinates take 11 bytes out of 29 bytes, which is the payload size of a TinyOS packet), we decided to use a local, Cartesian, coordinate system. This local coordinate system of reference, which uses linear units, is better suited for WSN, than a global coordinate system.

A local coordinate system is built from a global system, that uses GPS coordinates, in the following way: the local system of reference has an origin (called a Reference Point) specified in terms of global coordinates (GPS coordinates). The distance between this Reference Point (with coordinates λ_1 and φ_1) and another point, with a GPS location specified by λ_2 and φ_2 , can be computed as follows [16]:

$$\text{Distance} = \sqrt{\left(F_{lat} * \left(\varphi_1 - \varphi_2\right)\right)^2 + \left(F_{lon} * \left(\lambda_1 - \lambda_2\right)\right)^2} \qquad (1)$$

where:

$$F_{lat} = \frac{\pi}{180}\left(\frac{a^2b^2}{(a^2\cos^2\varphi + b^2\sin^2\varphi)^{3/2}} + h\right) \quad (2)$$

$$F_{lon} = \frac{\pi}{180}\left(\frac{a^2}{\sqrt{a^2\cos^2\varphi + b^2\sin^2\varphi}} + h\right)\cos\varphi \quad (3)$$

are conversion factors that represent the distances for 1 degree change in latitude and longitude, respectively. The unit of measure is meter/degree. The parameters in the above formulas are: *a*=6,378,137 meters, *b*=6,356,752.3142 meters and *h* is the height over the ellipsoid. The influence of *h* on the conversion factors is minimal and a value of 200 meters is assumed. The X and Y coordinates of the point with a GPS location specified by λ_2 and φ_2 are given by the two additive terms in (1). The Y-axis of the local coordinate system is oriented in the North/South direction and the X-axis in the East/West direction. All variables specified in (1), (2) and (3) (i.e., λ, φ and *h*) can be directly obtained from a commercial GPS device. The result of our design is that the GPS Mote transforms the global coordinates received from the GPS device into local coordinates and broadcasts these local coordinates.

The localization scheme that makes use of the Walking GPS solution has two distinct phases:

- the first phase is during the deployment of the sensor nodes. This is when the Walking GPS solution takes place. The carrier (soldier or vehicle) has a GPS-enabled mote attached to it; the GPS-enabled mote periodically beacons its location; the sensor nodes that receive this beacon infer their location based on the information present in this beacon.
- the second phase is during the system initialization. If at that time, a sensor node does not have a location, it will ask its neighbors for their location information. The location information received from neighbors is used in a triangulation procedure by the requester, to infer its position. This second phase enhances the robustness of the scheme.

The internals of the two components running on the GPS Mote and on the Sensor Mote, are described further in the following subsections.

3.1. GPS Mote

A GPS Mote without a Reference Point is in an Uninitalized state. No messages are sent by the GPS Mote, as long as it is in this state. A Reference Point can be obtained either through radio communication, or from flash memory. Once a Reference Point is obtained through radio, it is also stored in the flash memory. A GPS Mote with a Reference Point is in an Initialized state.

The state transition diagram for the GPS Mote is shown in Figure 2, below.

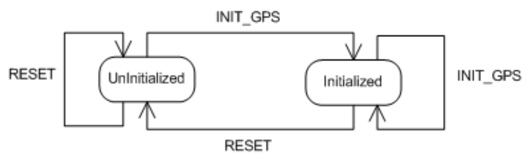


Figure 2. State transition diagram for the GPS Mote

The initialization of the GPS Mote is done by sending a packet of type INIT_GPS, with a format shown in Table 1. In addition to the latitude and longitude of the Reference Point, two more parameters are sent: the sending power and the sending period.

Table 1. INIT_GPS message format

Reference Point Latitude	Reference Point Longitude	Sending Power	Sending Period
4 bytes signed integer	4 bytes signed integer	1 byte unsigned	2 bytes unsigned integer

The sending power is used for limiting the communication range of the GPS Mote. The intent is that only the motes in the vicinity of the deployer receive the localization information. The sending period describes the frequency with which the localization packets are sent. This frequency is correlated with the speed of deployment.

The format for the latitude and longitude of the Reference Point is an optimized for space version of the more general d^{o}m'ss.ss" format (degrees, minutes, seconds). The unit for the 4-byte format that we use is the 1/1000 of a second and the formula for computing it, is the following:

$$\text{coord} = d * 36 * 10^5 + m * 6 * 10^4 + ss.ss * 10^3$$

When an INIT_GPS packet is received, its information is stored in the flash memory, to be available after a system reset. A packet of type RESET puts a GPS Mote back into an Unitialized state, and erases the portion of the flash memory that stores the Reference Point, Sending Power and Sending Period.

The GPS Mote sends location information by broadcasting an INIT_LOCALIZATION packet, with the format shown in Table 2.

Besides the X and Y coordinates, the GPS Mote broadcasts the GPS coordinates of the Reference Point as well. We chose this design because the base stations are also deployed using the Walking GPS solution.

Table 2. INIT_LOCALIZATION message format

Reference Point Latitude	Reference Point Longitude	X Coord.	Y Coord.
4 bytes signed integer	4 bytes signed integer	2 bytes signed integer	2 bytes signed integer

The base stations can be queried for the reference point of the local coordinate system, hence that is the reason for broadcasting it as part of the INIT_LOCALIZATION packet. Thus, the Reference Point information, present in the INIT_LOCALIZATION packets, is only used by base stations.

3.2. Sensor Mote

The Sensor Mote can also be in one of two states: Initialized (if location information is present) or Uninitialized. The state transition diagram for the Sensor Mote is shown in Figure 3, below:

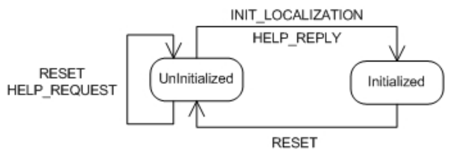


Figure 3. State transition diagram for the Sensor Mote

A Sensor Mote can become Initialized in one of two circumstances: a) if it receives an INIT_LOCALIZATION message (event that can happen during the Walking GPS phase of our localization scheme) or b) if the sensor node sent a HELP_REQUEST message and neighboring nodes reply with HELP_REPLY messages (event that can happen during the system initialization phase). If a sensor node enters the system initialization phase and it does not have a location (which was supposed to have been acquired as part of the Walking GPS), then the sensor node broadcasts a packet of type HELP_REQUEST. Neighboring nodes that have a location respond by broadcasting packets of type HELP_REPLY, with the format shown in Table 2. A sensor node that sent HELP_REQUEST messages, stores the HELP_REPLY responses in a buffer and computes its own location at the centroid of the locations received from its neighbors.

After obtaining the location information, a sensor node stores its location in flash memory, to be available after a system reset. If a RESET packet is received the location information is erased from the flash and the sensor node becomes Unitialized.

For Sensor Motes, we provided support for two methods of deployment:

- ON_AT_DEPLOYMENT, or the first deployment type, is the type of deployment in which a sensor node is powered on right before it is deployed (the deployer reaches the point of deployment, then turns on the power and places the mote on the ground).
- ON_ALL_THE_TIME, or the second deployment type, is the type of deployment where the sensor node is powered all the time. This second scenario is more convenient for the deployer (no mechanical switches to be turned on/off), and more likely to be used in a real world deployment, however it is more challenging, since the sensor node needs to infer its location from a set of beacons containing, most likely, different locations.

The complexity associated with the two deployment types, is different. For the case in which the sensor nodes are turned on right before the deployment, the solution is trivial. The first received INIT_LOCALIZATION packet provides the actual location. When the sensor nodes are on all the time, they need to infer from the RSSI value the time when they were deployed. In order to achieve this, we use a circular buffer with a window size of 4, which stores location information received in the INIT_LOCALIZATION messages. Two configurable parameters are used: a minimum RSSI value (called Lower Bound RSSI), below which the sensor mote no longer accepts localization packets (this reduces the risk of receiving messages from GPS Motes that are far away) and an interval RSSI (called Delta RSSI). We employ a moving average computation for the packets present in the circular buffer. A subsequent INIT_ LOCALIZATION message is accepted if its RSSI is in the interval [Avg. – Delta RSSI, Avg. + Delta RSSI]. In order to obtain its location, a sensor node goes back in the circular buffer, a configurable number of entries (currently two) and retrieves the location present in that entry in the buffer.

4. System Implementation

The Walking GPS localization scheme requires that the deployer has a GPS Mote attached to it. We built a prototype, called the GPS Mote assembly, that can be worn during the deployment. This prototype consists of a GPS device mounted on top of a bicycle helmet. The GPS device is connected through and RS232 cable to the GPS Mote that is attached with a velcro to a wristband. Figure 4 illustrates the prototype.

For the GPS device, we used the eTrex Legend [17] device produced by Garmin. The GPS device is WAAS (wide-area augmentation system) enabled, and it provides updated location information with high accuracy (error < 3 meters), at a rate of 1Hz. Our choice to use a commercial GPS device for experiments was due to its ease of use and seamless integration. We also implemented a miniaturized version of the GPS Mote using the MTS420CA sensor board from Crossbow Inc.

[18]. We have not performed extensive performance evaluation experiments to assess the accuracy of location information obtained from the miniaturized version of the GPS Mote. For our deployment method (used in the Walking GPS scheme) the miniaturization was not an important factor.

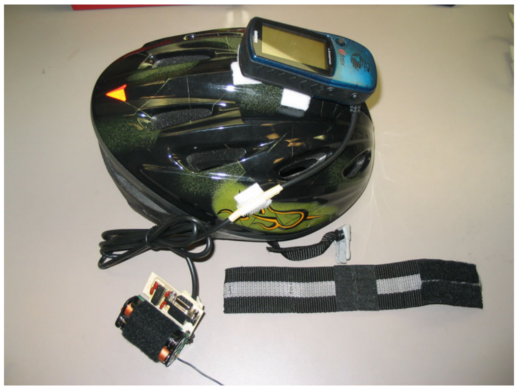


Figure 4. GPS Mote assembly

We performed tests on MICA2 motes and on the newer generation of motes, called XSM, from Ohio State University and Crossbow [19]. We used both platforms as Sensor Motes. For the GPS Mote, we only used MICA2 motes.

Several parameters were used for performance tuning. For the GPS Mote, the sending power of the radio was set for all experiments to 0xA and the rate at which the beacons were sent was set to 3Hz, unless the experiment (described in the following section) indicates a different value. For the Sensor Mote, two parameters were used: the Lower Bound on the RSSI value (packets with a signal strength smaller than the threshold were discarded), and the Delta RSSI, both explained in the preceding section. We provide the values of these parameters in an ADC_Count unit, which is a measure of the signal strength associated with a received radio packet. Both, MICA2 and XSM motes use a Chipcon CC1000 radio and the formulas for converting the ADC_Count unit to S.I. unit (dBm) is [20]:

$$V_{RSSI} = V_{battery} \times ADC\_Counts/1024$$
$$RSSI_{dBm} = -51.3 \times V_{RSSI} - 49.2$$

The values of the Lower Bound RSSI and Delta RSSI parameters, in ADC_Count units, are given in Table 3.

We implemented the Walking GPS localization scheme in nesC (approximately 1500 lines of code) for the TinyOS operating system. For the GPS Mote, the total code size was approximately 17KB and the data size was 595 bytes. The code size for the Sensor Mote module was 972 bytes and the data size was 117 bytes. The code has been released and it is available at [21] [22] .

Table 3. Sensor Mote parameter values used in experiments

	Lower Bound RSSI	Delta RSSI
MICA 2	200	50
XSM	35	5

5. Performance Evaluation

In this section we present the experimental results obtained from the evaluation of the Walking GPS localization scheme. In order to better understand the performance of each individual component of our system, we first evaluated each component separately. In the second part of our experiments, we evaluated the system in its entirety.

5.1. GPS Mote

The purpose of this experiment was to evaluate the accuracy of the GPS device and the precision of the transformation performed on the global GPS coordinates, in order to obtain the local coordinates (i.e., Cartesian). In this experiment we did not evaluate the Sensor Mote part of the Walking GPS scheme. The results are shown in the Figure 5.

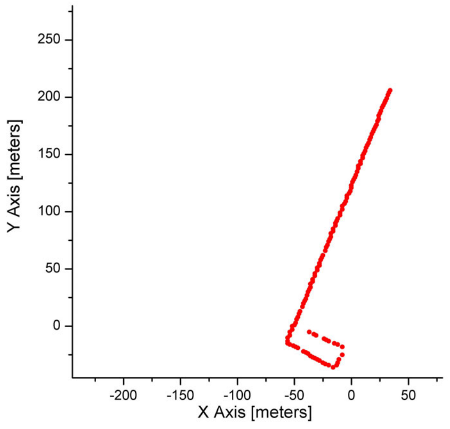


Figure 5. Walking in a straight path, then a loop

The experiment consisted in walking with the GPS Mote assembly in a straight alley, for approximately 215 meters, then turning left and following a U-shape path, going around a parking lot. The walk was performed at a speed of about 1.3 meters/sec and the rate with which the initialization beacons were sent was 2Hz. The average accuracy, as indicated by the Garmin GPS device, was approximately 4 meters. The beacons were received by a base station attached to a laptop, which were carried during the entire experiment. The starting point was in the upper right part of the graph:

In Figure 5, as well as in the figures that follow, the Cartesian coordinate system is aligned to the North-South and East-West directions. The X-axis represents the East-West direction and the Y-axis represents the North-South direction.

To further verify the validity of the experimental data, we superimposed the trajectory obtained in the experiment onto an aerial map [23] of the area where the experiment took place. The result in shown in Figure 6. A very good match can be observed, between the experimental data and the reality in the field.

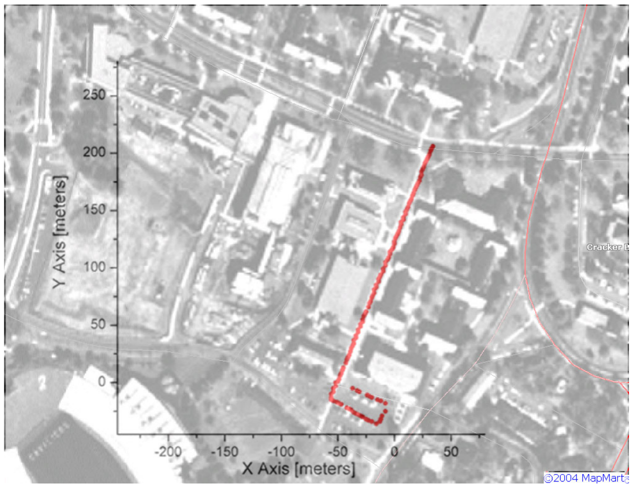


Figure 6. Super-imposing Figure 5 on an aerial map

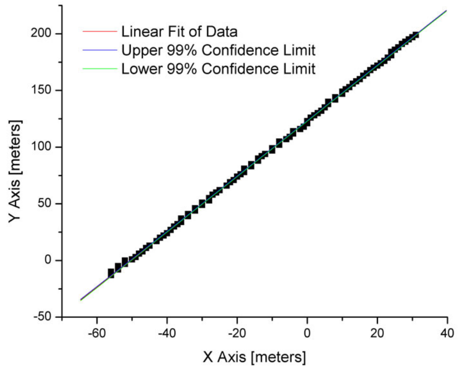


Figure 7. Linear fit for a portion of the path, shown in Figure 5.

For estimating the average error present in the localization, the portion of the experimental data, shown in Figure 5, that represented the walking in a straight path (not including the U-shape), was used in a regression linear fit. The result of the linear fit as well as the upper and lower 99% confidence limits, are shown in Figure 7.

From the linear regression analysis we obtained a coefficient of determination (R^2) of 0.99952, i.e., 99.9% of the variability can be explained by the linear model. The value for Mean Square Error (MSE) was 1.782m^2. This indicates an approximate error in localization on the order of 1 – 1.5 meters.

5.2. Sensor Mote

The purpose of this experiment was to evaluate the performance of the second component of the Walking GPS scheme, namely the Sensor Mote. In this experiment we did not evaluate the GPS Mote part of the Walking GPS scheme.

For this experiment we used 30 MICA2 motes as Sensor Motes. The test was performed outdoors. For this experiment, we configured the GPS mote to not use the actual readings of the GPS device. In this mode, called the "debug" mode, the X-coordinate is incremented with each broadcast, and the Y-coordinate has a value of 0 at all times. This allowed us to better identify the exact packet that is used by the Sensor Mote for inferring its location.

We performed the experiment 30 times, for each receiving mote. The beacons, containing location information, were sent at a rate of 1Hz. We used a lower rate because we wanted to control the timing when each mote was deployed. Using a lower rate enabled us to better synchronize the actual deployment with the occurrence of a particular beacon. The Sensor Motes were deployed at the same physical location, at approximately the time when the 18th initialization beacon was sent. In Figure 8, we show the number of motes for all X-coordinates that were obtained during the experiment.

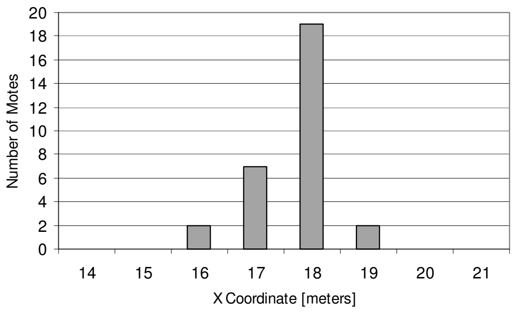


Figure 8. Number of motes for each X-coordinate

From Figure 8 it can be seen that, with a relatively good accuracy, the sensor motes inferred the correct location.

There is little variability in the value of the X-coordinate and this can be explained by our deployment process – we attempted to deploy each mote at the correct beacon number (the 18th) – however this procedure was not very rigorous. The small variability in the X-coordinate values, present in Figure 8, indicates a relatively good control in choosing which packet (among the ones present in the circular buffer) is used for determining the actual location. Empirical evaluations can provide insights regarding the length of the circular buffer and which entry from the buffer would best approximate the actual deployment location. For these evaluations, various deployment scenarios need to be considered (e.g., the speed with which the deployer moves, the rate at which the GPS device computes a new location and locations are broadcast, type of motes used in the deployment).

5.3. Deployment

The Walking GPS solution was evaluated in an open field, as shown in Figure 9. For an easier estimate of the localization error, we marked a 6x5 grid on the ground and we deployed the sensor motes in this grid. We want to emphasize the fact that the deployment being done in a grid was not used in any way during our localization. A deployment in any other regular geometric shape could have been performed. We used a grid because it was easy to create and it was easier to visually assess the performance.

Figure 9. Field where the experiments took place

We performed an experiment to verify, using only the GPS Mote, the grid marked on the ground. This experiment was similar in nature with the one shown in Figure 5. The walk was done on the grid marked in the field. This time the walk with the GPS assembly was for shorter distances and alternating directions, at a speed of about 1 meter/second. The localization packets were stored on a laptop, carried during the walk. The result of the experimental evaluation is shown in Figure 10.

The starting point was the upper-left corner, approximately at the (0, 0) coordinate. The path shown in Figure 10, was followed in all the following experiments as well. From Figure 10 we can observe a good fit between the experimental data and the path that was followed. In the experiments that follow, we provide numeric localization errors by performing a manual best fit of a strict grid with unit 10 meters, to the experimental data. The average localization error is defined as follows:

$$error = \frac{\sum_{i=1}^{N} \sqrt{(x_i - x_i^{grid})^2 + (y_i - y_i^{grid})^2}}{N}$$

where (x_i, y_i) is the coordinate obtained experimentally for the *i-th* mote, (x_i^{grid}, y_i^{grid}) is the coordinate of the *i-th* mote on the fitted grid, and N = 30 in our experiments.

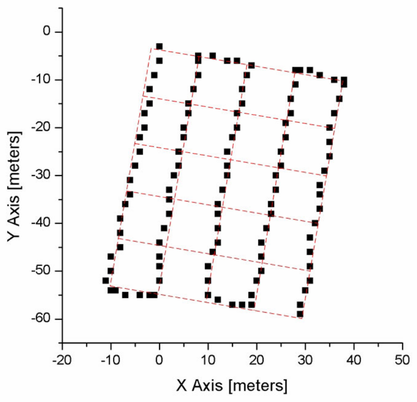


Figure 10. Walk with the GPS Mote along the grid

It is critical in understanding the following experimental results to note that the average location errors are not with respect to the "ground truth" location, but rather are relative to the known geometry of the deployment grid.

5.4. Integrated System - First Deployment Type

In this experiment we evaluated the entire system, consisting of 30 MICA2 motes that were deployed in the aforementioned grid. Each mote was turned on at its place of deployment, right before being deployed. This was described above as the First Deployment Type. The first localization packet provided the location of the receiving mote. The experimental results are shown in Figure 11.

The average localization error obtained from fitting a grid to the experimental data is 0.8 meters with a standard deviation of 0.5 meters. From Figure 11, as well as from the numerical results of the localization error, it can be observed a remarkably good fit. In this deployment type the errors are only due to the estimation of the global coordinate, done by the GPS hardware.

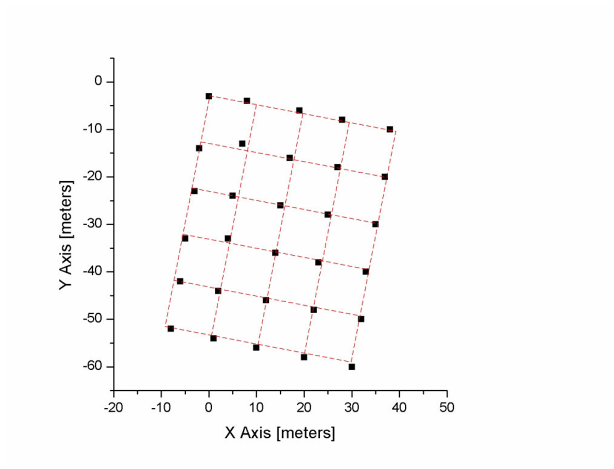


Figure 11. System deployment using the first deployment type

5.5. Integrated System - Second Deployment Type

The purpose of this experiment was to evaluate the performance of the second deployment type. In this experiment the sensor nodes were powered on all the time. The experimental results are depicted in Figure 12.

The localization error obtained from fitting a grid to the experimental data is 1.5 meters with a standard deviation of 0.8 meters. The average localization error is larger than in the experiment where the nodes were turned on right before deployment.

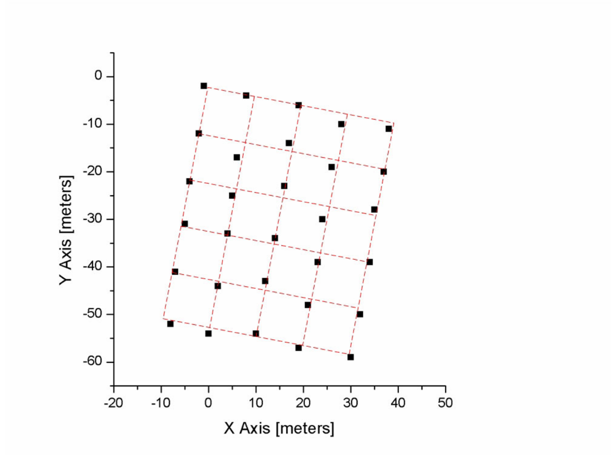


Figure 12. System deployed using the second deployment type

The less accurate location is due to the incorrect inference of the exact moment a sensor node was deployed. The same effect was observed and depicted in Figure 8, where there was some variance in the value of the X-coordinate. Nevertheless, an average localization error of only 1.5 meters is very good for deployments of sensor nodes not equipped with specialized hardware.

5.6. Integrated System – Dual Deployer

The purpose of this experiment was to evaluate the performance of the Walking GPS localization scheme when using two commercial GPS devices (the same model however). A GPS device, as any other hardware device is dependent on calibration. Even after stringent calibration procedures, some variability in the indicated location is expected. From the direct reading of the global GPS location as shown by two GPS devices positioned next to each other, differences on the order of 1/1000 of a minute and sometimes even 1/100 of a minute, were observed. It was anticipated that these differences will contribute to an even larger localization error.

The deployment in this experiment was done along the length of the grid field (lines containing 6 motes). Three of the vertical lines (the middle and the two extreme ones) were deployed using one of the GPS devices, the other two vertical lines were deployed using the second GPS device. The experimental results are shown in Figure 13.

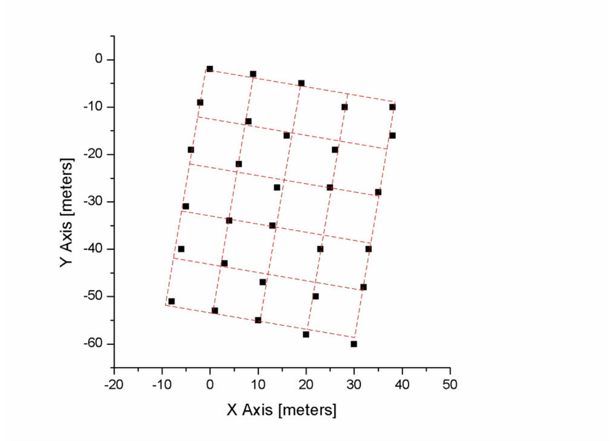


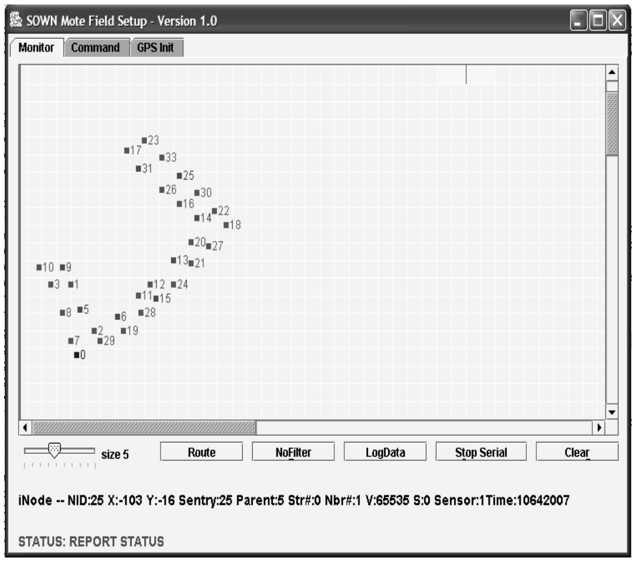
Figure 13. System deployment using two GPS devices

The localization error obtained from our fitting of a grid to the experimental data is 1.6 meters with a standard deviation of 0.9 meters. In this deployment scenario, the average localization error is the largest. In addition to the errors encountered in previous experiments, here, the GPS device calibration has an additional contribution. When comparing the results of this experiment with the previous one, in which only one GPS device was used, it can be observed that the effect the device calibration has on location error was relatively small, of about 0.1 meters.

6. Tracking Application

The proposed Walking GPS localization solution has been integrated and tested with a target tracking application [2] developed in our research group. A screenshot of the tracking application is shown in Figure

14. The experiment used 32 XSM motes, deployed in a parking lot, approximately 8 meters apart (spacing was not rigorous).



iNode -- NID:25 X:-103 Y:-16 Sentry:25 Parent:5 Str#:0 Nbr#:1 V:65535 S:0 Sensor:1Time:10642007

STATUS: REPORT STATUS

Figure 14. Target tracking application using Walking GPS

We observed a very good match between the locations reported by the tracking application and the real deployment of the nodes. We do not provide exact localization errors, due to the irregular deployment (the spacing was only approximate).

7. Conclusions and Future Work

In this paper we presented the design, implementation and the evaluation of a localization solution that can be used in situations in which WSN are deployed manually. The method to deploy sensor nodes manually is currently used in several projects and there are scenarios of real system deployments, where the manual deployment is, still, the only viable solution. Our proposed solution has very little overhead and it is cost effective.

The experience from the development of the current system can be further used in future research that will address the aerial deployment. Considering the sensor deployment rate, deployment altitude, sensor trajectory and the actual location at the beginning of deployment, some coarse location information can be inferred using our solution, giving a starting point for finer and more granular localization schemes.

8. Acknowledgements

This work was supported by the DARPA IXO office under NEST project (grant number F336616-01-C-1905) and by the NSF grant CCR-0098269. The authors would like to thank the anonymous reviewers for their invaluable feedback.

9. References

[1] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan and S. Shenker, “GHT: A Geographic Hash Table for Data-Centric Storage”, *1st International Workshop on Wireless Sensor Networks and Applications (WSNA)*, Atlanta, GA, 2002

[2] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, B. Krogh, “Energy-Efficient Surveillance System Using Wireless Sensor Networks”, *2nd International Conference on Mobile Systems, Applications, and Services (MobiSys)*, Boston, MA, 2004.

[3] R. Nagpal, H. Shrobe and J. Bachrach, “Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network”, *2nd International Workshop on Information Processing in Sensor Networks (IPSN)*, Palo Alto, CA, 2003.

[4] T. He, C. Huang, B. Blum, J. A. Stankovic, T. Abdelzaher, “Range-free localization schemes for large scale sensor networks”, *9th International Conference on Mobile Computing and Networking (MobiCom)*, San Diego, CA, 2003

[5] P. Bahl and V. N. Padmanabhan, “RADAR: An In-Building RF-Based User Location and Tracking System”, *19th Anual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM)*, Tel Aviv, Israel, 2000.

[6] J. Hightower, G. Boriello and R. Want, “SpotON: An indoor 3D Location Sensing Technology Based on RF Signal Strength”, *University of Washington CSE Technical Report #2000-02-02*, 2000.

[7] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin and S. Wicker “Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks”, *UCLA Technical Report UCLA/CSD-TR 02-0013*, 2002

[8] Y. J. Zhao and R. Govindan “Understanding Packet Delivery Performance in Dense Wireless Sensor Network”, *1st ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Los Angeles, CA 2003.

[9] G. Zhou, T. He and J. A. Stankovic, “Impact of Radio Asymmetry on Wireless Sensor networks”, *2nd International Conference on Mobile Systems, Applications, and Services (Mobisys)*, Bostan, MA, 2004

[10] B. Hofmann-Wellenhoff, H. Lichtenegger and J. Collins, “Global Positions System: Theory and Practice”, *Fourth Edition. Springer Verlag*, 1997.

[11] N. B. Priyantha, A. Chakraborty and H. Balakrishnan, “The Cricket Location-Support System”, *6th International Conference on Mobile Computing and Networking (MobiCom)*, Boston, MA, 2000.

[12] A. Smith, H. Balakrishnan, M. Goraczko, N. Priyantha, “Tracking Moving Devices with Cricket Localization System”, *2nd International Conference on Mobile Systems, Applications, and Services (*Mobisys), Boston, MA, 2004

[13] A. Savvides, C. C. Han and M. B. Srivastava, “Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors”, 7th *International Conference on Mobile Computing and Networking (MobiCom)*, Rome, Italy, 2001.

[14] P. Corke, R. Peterson and D. Rus, “Networked Robots: Flying Robot Navigation using a Sensor Net”, *11th International Symposium of Robotics Research (ISRR)*, Siena, Italy, 2003

[15] P.N. Pathirana, A.V. Savkin, S. Jha and N. Bulusu, “Node Localization Using Mobile Robots in Delay-Tolerant Sensor Networks”, *IEEE Transactions on Mobile Computing*, 2004

[16] Viacheslav Adamchuk, “Global Positioning System Data Processing”, url: pasture.ecn.purdue.edu/~abegps/web_ssm/web_GPS.html

[17] Garmin International Inc., “Owners’s Manual and Reference Guide” url: www.garmin.com/manuals/eTrexLegend_OwnersManual.pdf

[18] Crossbow Technologies, “MTS/MDA Sensor and Data Acquisition Boards User’s Manual”, Document 7430-0020-02, October 2003

[19] P. Dutta, “Echelon Documentation”, url: www.cse.ohio-state.edu/~duttap/echelon/

[20] Crossbow Technologies, “MPR/MIB Mote Hardware Users Manual”, Document 7430-0021-05, December 2003

[21] TinyOS CVS tree, url: www.sourceforge.net/projects/tinyos

[22] University of Virginia SOWN CVS tree, url: www.sourceforge.net/projects/uvasown

[23] IntraSearch Inc., url: www.mapmart.com

Spotlight: Low-cost Asymmetric Localization System for Networked Sensor Nodes

University of Virginia

Tian He, Radu Stoleru, John A. Stankovic
{ Tianhe,rs6bd,Stankovic}@cs.virginia.edu

Many middleware services for sensor network applications have been developed successfully. Localization - finding the position of sensor nodes - remains one of the most difficult research challenges to be solved economically and practically.

Our answer to this challenge is a localization system called Spotlight. This system employs an asymmetric architecture, in which field motes do not need any additional hardware than what they currently have. All sophisticated hardware and computation reside on a single Spotlight device. We demonstrate that this localization is much more accurate than the range-based localization schemes and that it has a much longer effective range than the solutions based on ultrasound/acoustic ranging. Meanwhile, since only a single sophisticated device is needed to localize the whole network, the amortized cost will be much smaller than the cost to add hardware components to individual sensors.

The general idea of the Spotlight technique is to use a sophisticated spotlight device to generate controlled events that assist the localization of sensor nodes. An event could be, for example, the presence of light in an area. Using the time when an event is perceived by a sensor node the sensor node can be inferred. We envision, and depict in Figure 1, a sensor network deployment and localization scenario as follows: wireless sensor nodes are randomly deployed from a helicopter. A device (e.g. helicopter) flies over the network and generates light events over predefine traces. The sensor nodes detect the events and report to a base station the times when the events were detected. The base station computes the location of the sensor nodes.

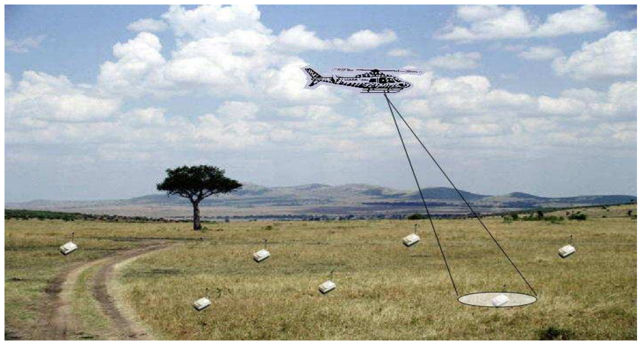


Figure 1: Localization of a sensor network using the Spotlight system

Formally, suppose that space *A* contains all sensor nodes *Ni*, and each node *Ni* is located at *Pi(x, y, z)*. To determine *Pi(x, y, z)*, a spotlight localization system needs to support three major functions as shown in Figure 2, namely a Detection Function *D(e)*, an Event Distribution Function *E(t)* and a Localization Function *L(t)*. The Detection Function *D(e)* is supported by the sensor nodes. It is a sensing module to determine whether an external event happens or not. The Localization Function *L(t)* and the Event Distribution Function *E(t)* are supported by a spotlight device. The Localization Function is an aggregation algorithm, which calculates the intersection of multiple sets of points. The Event Distribution Function *E(t)* describes the locations of the events (e.g. light spots) within the space A over a certain period of time. We implement multiple realizations of the *E(t)* function including Point Scan, Line Scan and Space Cover. Since the *E(t)* function is only realized at the spotlight device, its complexity is transparent to sensor nodes.

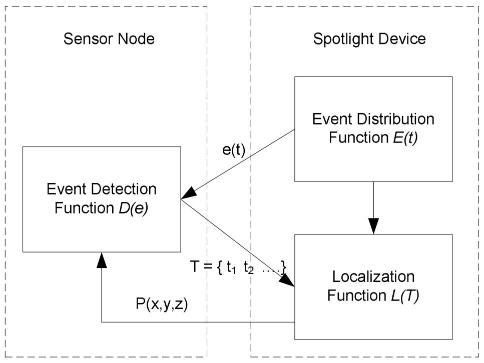

Sensor Node
Spotlight Device
Event Distribution Function E(t)
e(t)
Event Detection Function D(e)
T = { t1 t2 ... }
Localization Function L(T)
P(x,y,z)

Figure 2: Spotlight system architecture

With the support of the three functions, the localization scheme proceeds as follows:

1. A spotlight device distributes events into the space *A* over a period of time.
2. During the event distribution, sensor nodes record the time sequence $T = \{t_1, t_2 \ldots t_n\}$ when they detect the event.
3. After the event distribution, each sensor node sends the detection time sequences to the spotlight device.
4. The spotlight device estimates the location of sensor nodes, using the time sequence *T* and the known *E(t)* function.

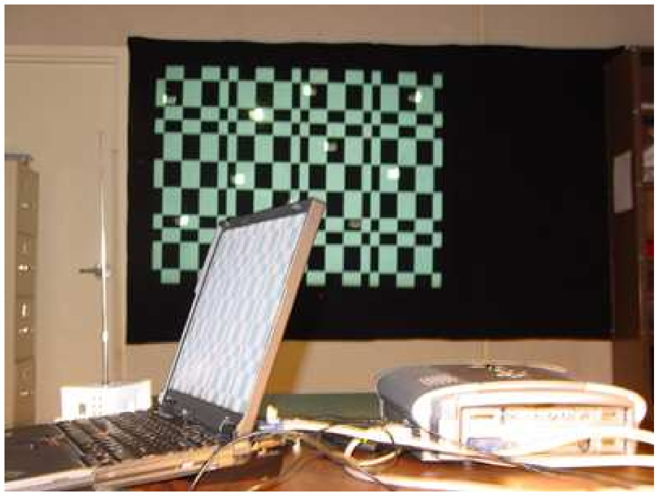


Figure 3: Spotlight System implementation

As shown in Figure 3, during this demonstration, we use a NEC MultiSync MT1030 projector connected to a DELL laptop. By projecting light to a vertical Veltex board, we distribute well-controlled events to the Mica2 motes attached to the board. Specifically, we will demonstrate the localization process of the point scan, line scan and area cover. We will also provide insights into how this localization scheme can be successfully applied to realistic outdoor environments.

Achieving Realistic Sensing Coverage in Wireless Sensor Networks

Joengmin Hwang, Tian He, Yongdae Kim
Department of Computer Science, University of Minnesota, Minneapolis

1 Introduction

Despite the well-known fact that sensing patterns in reality are highly irregular, researchers continue to develop protocols with simplifying assumptions about the sensing. For example, a circular 0/1 sensing model [2] is widely used in most existing simulators and analysis. While this model provides high-level guidelines, it could cause wrong estimation of system performance in the real world.

Our answer to this issue is a sensing area modeling technique, which obtains the coverage of sensor nodes through event training. The main idea is using *discrete controlled events* to identify the sensing coverage based on event detection results by individual sensor nodes. A key architectural advantage of this approach is a lightweight design in sensor node with minimal overhead. Besides communication, each sensor node only needs to support a simple detection function (with optional time synchronization requirement). We evaluate our model using a physical experiment on a testbed consisting of 40 MicaZ motes. Our evaluation results shows we can identify the sensing area with low error rate with simple training method.

2 Sensing Area Modeling

In this section, we introduce the design of our system at the architectural level.

2.1 Assumptions

We assume that we can generate controlled events with known time and location. This can be done through two approaches. First, events can be generated by using real targets. For example, one or multiple robots can move along predefined traces to activate PIR motion sensors in the field. Other events, such as vibration and sound, can be generated similarly. Second, the controlled events can be injected using special devices such as infrared radiation generator, Spotlight [1] and our prototype system described. Since the methods to generate controlled events are diversified, we intentionally describe our approach conceptually independent of the concrete method used. The targeted application scenarios are 1) to identify the coverage of motion sensors with a room, 2) to discover blind spots in a surveillance area, and 3) to compensate the irregularity of tiny proximity sensors used for paper-edge detection in printers.

2.2 Main Idea

The main idea of our training-based modeling approach is to generate *controlled events* in the space where the sensor nodes are deployed. An event could be, for example, the presence of an object in an area or a light spot projected on a plate of sensors. Formally, an event can be defined as a detectable phenomenon $e(t,p)$ that occurs at time t and at location $p \in A \subset \mathbb{R}^k$ $(k = 1,2,3)$. Without loss of generality, we use $k = 2$ in the rest of the paper. An event is said to be *controlled* if we can enforce a relationship between the time t and location p. In other words, a set of controlled events can be described as the event locations over the discrete time: $G : \mathbb{R} \to \mathbb{R}^2$, where $G(t) = p_t = (x_t, y_t)$ where $t \in \{t_1, t_2, ..., t_n\}$.

It consists of two major parts: an event generator G and a set of sensor nodes n_i $(i \in N)$. The event generator G could be a single device or multiple distributed devices that can generate a sequence of controlled events $e(t,p)$ with known spatiotemporal correlation $G(t) = p(x_t, y_t)$. We define $S_i(t,p)$ as the detection function of node n_i. If node n_i can detect event $e(t,p)$, $S_i(t,p) = 1$, otherwise $S_i(t,p) = 0$. In case of detection, sensor nodes store the timestamp t locally. By the end of training, a sensor computes the location of all events it detects by inputting the timestamps into $G(t)$. Therefore, a set of timestamps $T_i = \{t_1^i, t_2^i, \dots, t_n^i\}$ stored in node n_i can be converted to a set of locations $P_i = \{p_1^i, p_2^i, \dots, p_n^i\}$ within the sensing area. The location set P_i can be directly used to describe the sensing area of node n_i.

2.3 Design of Event Generator $G(t)$

Since the overhead and accuracy of the sensing modeling is largely determined by $G(t)$, it is important to consider several solutions to optimize $G(t)$ under different system configurations.

2.3.1 *Regular $G(t)$*

To illustrate the basic functionality of an event generator, we start with a simple sensor system where the sensing area of a node is a line segment. We shall find out the portion of the line included in the sensing ranges of sensor node n_1 and n_2. To achieve this, the event generator creates discrete point events along this line $[0,L]$ with constant speed v with same interval D. Formally, $G(t) = t \cdot v$ where $t = kD/v$ and $0 \le k \le L/D$. For example, a sensor node n_1 collects a set of six timestamps $T_1 = \{t_1, t_2, \dots, t_6\}$ at which the events are detected. From function G, the actual locations of events are converted to a set of locations $P_1 = \{t_1 v, t_2 v, \dots, t_6 v\}$. The sensing coverage of sensor n_1 can be defined as the line segment that covers P_1. For the sensor n_2, it reports timestamps $T_2 = \{t_4, t_5, t_6, t_7\}$ and the sensing coverage of sensor n_2 is defined as the line segment that covers $P_2 = \{t_4 v, t_5 v, t_6 v, t_7 v\}$. The intersection of T_1 and T_2, $T_1 \cap T_2 = \{t_4, t_5, t_6\}$ indicates the coverage of two sensors is overlapped. The regular training can be generalized to the case when the events occur in a plane by dividing the plane into several lines with a certain interval.

In addition to the progressive scanning, the $G(t)$ function of the regular training can generate events with an arbitrary sequence as long as every point in the area is covered. Also, as long as we can match the event and its position we can use any training method for $G(t)$ which includes unsupervised event.

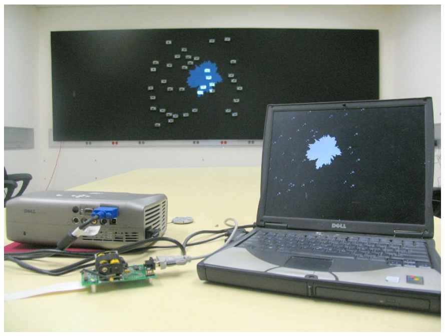


Figure 1. System Setup

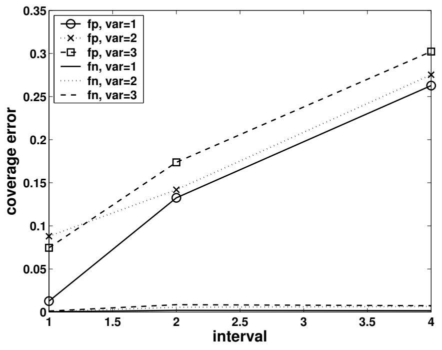


Figure 2. Errors in regular $G(t)$ with varying interval and irregularity

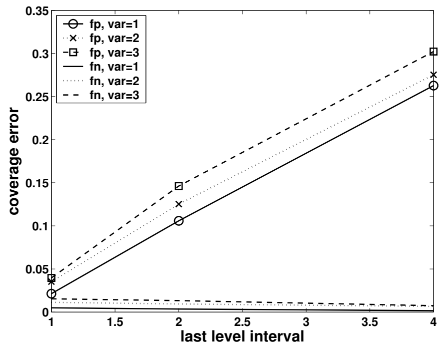


Figure 3. Errors in hierarchical $G(t)$ with varying interval and irregularity

Algorithm 1 Hierarchical G(t) process

```
Output: $P_i$: The sensing area of $n_i$.
 1: $G(t)$ starts with level-1 events $e(t,p)$ (The number of level-1 event is decided by the minimum sensing area)
 2: Node $n_i$ reports $S_i(t,p)$ for all level-1 events
 3: repeat
 4:   for all level-$k$ adjacent pair $e(t_m,p_m)$ and $e(t_n,p_n)$ do
 5:     if any node detects only one of events && no event generated at position $\frac{p_m+p_n}{2}$ before then
 6:       Generate a level-$(k+1)$ event at position $\frac{p_m+p_n}{2}$
 7:     end if
 8:   end for
 9:   $k=k+1$
10: until ($k$ = Maximum Level)
11: $P_i$ is a set of positions $p$ where $S_i(t,p)=1$
```

2.3.2 *Hierarchical $G(t)$*

Hierarchical G(t) is motivated by the observation that the boundary area of a sensing coverage requires more detail than the area in the middle of coverage. With the hierarchical $G(t)$, we can reduce the number of events required to obtain the same accuracy as regular $G(t)$. A level-1 event divides the area into four sub-areas, and level-2 events divide the area into 16 sub-areas. In general, level-i events divide an area into 4^i sub-areas. If an event is a level-i event, it is also a level-j event, where $j \geq i$. Two events are said to be *adjacent*(or a pair) if they are neighboring each other vertically, horizontally or diagonally (e.g., an event could have maximal eight adjacent events). Two adjacent events are said to be a *boundary pair* if only one of two adjacent events is within a sensing range of some node. form a boundary pair). The event in the boundary pair is called *boundary event*. The main idea of Hierarchical $G(t)$ is to *recursively generate new events in the middle of boundary pairs.* It works in a way similar to the binary search within a two-dimensional space. We describe the step by step operation of Hierarchical $G(t)$ in Algorithm 1.

3 System Implementation

We design and implement a complete version of our system which includes regular and hierarchical training on the TinyOS/Mote platform. The NesC language is used to program the motes and Java is used to build the regular and hierarchical generators. The compiled image of a full mote implementation occupies 14,500 bytes of code memory and 605 bytes of data memory. For each event we generate, we assign a unique ID. By using these IDs, we eliminate the need for time synchronization. We use an *oracle algorithm* that assumes the knowledge of the sensing area of the nodes. Basically, this algorithm activates a sensor node (e.g., through projecting light to a sensor), if the controlled event $e(t,p)$ is within the sensing area of the node. We want to emphasize that the oracle algorithm and generated ground truth is used *only for the purpose of evaluation*. This knowledge is not used in any part of our proposed algorithm. Figure 1 shows the implementation setting. After each run, the training results are visualized on the board and compared with the ground truth.

In experiments, we investigate the impact of training interval (resolution) and sensing irregularity in both training methods. We divide error into two types: (1) **false positive fp:** the area measured as a part of sensing coverage but is not a part of real sensing coverage, or (2) **false negative fn:** the area which is measured as not in the sensing coverage but is a part of real sensing coverage. Figure 2 and 3 shows the coverage error in regular training and hierarchical training respectively. The degree of irregularity of sensing coverage is denoted by *var* where higher value means more severe irregularity. Our result shows that with a small training interval, we can achieve very precise coverage modeling, fp is almost 0% and fn is at 1% to 8% and even if we increase the interval, fp is still almost 0%. The performance of hierarchical training is almost similar to the regular training, which means we can do an efficient training while saving the cost of event generation.

4 Conclusion

This paper intends to draw attention to the sensing irregularity issue known but largely ignored by many designers. We contribute to this area by designing and implementing two training-based methods that accurately identify the sensing patterns of nodes using controlled events. Our design has been fully implemented and evaluated in a test-bed consisting of 40 MicaZ motes. We hope this work motivates our community to seriously consider the reality issues existed in the sensor networks.

5 References

[1] R. Stoleru, T. He, J. A. Stankovic, and D. Luebke. High-Accuarcy, Low-Cost Localization System for Wireless Sensor Networks. In *Sensys'05*, November 2005.

[2] T. Yan, T. He, and J. A. Stankovic. Differentiated Surveillance Service for Sensor Networks. In *SenSys'03*, November 2003.

Achieving Realistic Sensing Coverage in Wireless Sensor Networks

Joengmin Hwang, Tian He, Yongdae Kim
Department of Computer Science and Engineering, University of Minnesota
{jhwang, tianhe, kyd}@cs.umn.edu

Research Issue:

Despite the well-known fact that sensing patterns in reality are highly irregular, researchers continue to develop protocols with simplifying assumptions about the sensing. For example, a circular 0/1 sensing model is widely used in most existing simulators and analysis. While this model provides high-level guidelines, it could cause wrong estimation of system performance in the real world.

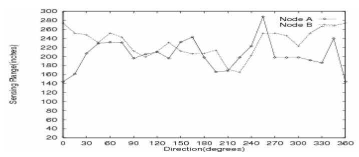


Fig 1. Directional range

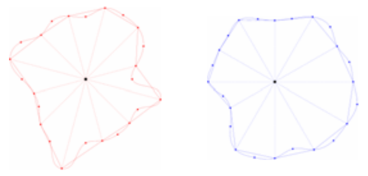


Fig 2. (1) Sensing area of node A (2) Sensing Area of node B

Project Overview

Our answer to this issue is a sensing area modeling technique, which obtains the coverage of sensor nodes through event training. The main idea is using discrete controlled events to identify the sensing coverage based on event detection results by individual sensor nodes. A key architectural advantage of this approach is a lightweight design in sensor node with minimal overhead. Besides communication, each sensor node only needs to support a simple detection function (with optional time synchronization requirement). We evaluate our model using a physical experiment on a testbed consisting of 40 MicaZ motes. Our evaluation results shows we can identify the sensing area with low error rate with simple training method.

Assumptions:

We assume that we can generate controlled events with known time and location. This can be done through two approaches. First, events can be generated by using real targets. For example, one or multiple robots can move along predefined traces to activate PIR motion sensors in the field. Other events, such as vibration and sound, can be generated similarly. Second, the controlled events can be injected using special devices such as infrared radiation generator, Spotlight and our prototype system described. Since the methods to generate controlled events are diversified, we intentionally describe our approach conceptually independent of the concrete method used. The targeted application scenarios are 1) to identify the coverage of motion sensors with a room, 2) to discover blind spots in a surveillance area, and 3) to compensate the irregularity of tiny proximity sensors used for paper-edge detection in printers.

Architecture:

We employ an architecture to model sensing coverage.

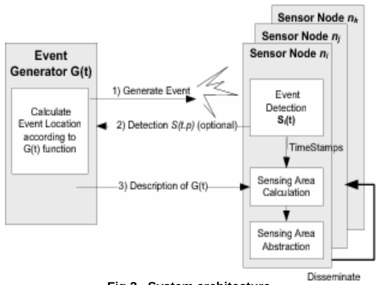


Fig 3. System architecture

Regular Training:

The sensing area of a node is a line segment. We shall find out the portion of the line included in the sensing ranges of sensor node n1 and n2. To achieve this, the event generator creates discrete point events along this line [0, L] with constant speed v with same interval D. Formally, G(t)=tv where t = kD/v and 0 < k < L/D. The sensing coverage of sensor n1 can be defined as the line segment that covers P1={t1 v, t2 v, … , t6v}.

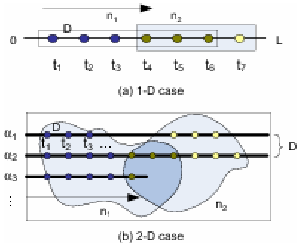


Fig 4. Regular training

Hierarchical Training:

With the hierarchical G(t), we can reduce the number of events required to obtain the same accuracy as regular G(t).

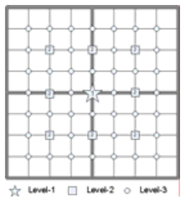


Fig5. Hierarchical partition

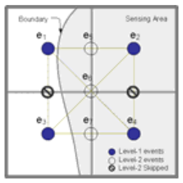


Fig 6. Level of detail

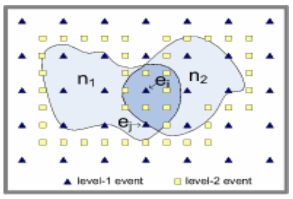


Fig 7. Hierarchical training

Implementation:

We have implemented a complete version of Regular Training and Hierarchical Training on Berkeley TinyOS/Mote platform, using 40 MicaZ motes as shown below.The compiled image of a full mote implementation occupies 14,500 bytes of code memory and 605 bytes of data memory.

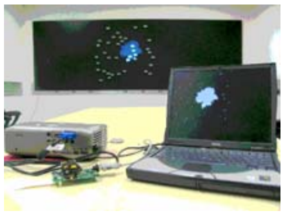


Fig 8. System Implementation

Evaluation:

We evaluate our architecture with physical system as well as an extensive simulation with 1,000 nodes.

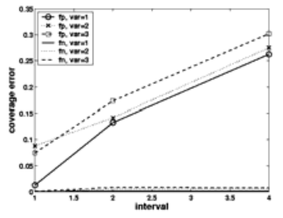


Fig 9. Error in Regular G(t) with varying interval and irregularity

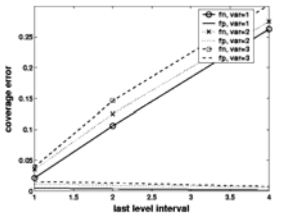


Fig 10. Error in Hierarchical G(t) with varying interval and irregularity

Conclusion:

This paper intends to draw attention to the sensing irregularity issue known but largely ignored by many designers. We contribute to this area by designing and implementing two training-based methods that accurately identify the sensing patterns of nodes using controlled events. Our design has been fully implemented and evaluated in a test-bed consisting of 40 MicaZ motes.

Acknowledgements:

This work is supported by NSF Nets NOSS Program
More information can be found at
Minnesota Embedded Sensor System Group
Http://mess.cs.umn.edu

uScan: A Lightweight Two-Tier Global Sensing Coverage Design

Yu Gu and Tian He
Department of Computer Science and Engineering, University of Minnesota
{*yugu,tianhe*}*@cs.umn.edu*

1 Introduction

Wireless Sensor Networks (WSNs), consisting of thousands of low-cost sensor nodes, have been used in many application domains such as military surveillance [1], habitat monitoring [4] and scientific exploration [6]. Limited power supplies and difficulties in harvesting ambient energy make energy conservation a critical issue to address. As one of solutions, energy-efficient sensing coverage extends system lifetime by leveraging on the redundant deployment of sensor nodes. Existing algorithms [5, 7, 8, 9] are designed to be well distributed and localized with solid performance gains. While the state-of-the-art is encouraging, we believe there are some aspects that need further investigation. In most algorithms, extending system lifetime is achieved essentially through coordination among neighboring nodes. The local node density, therefore, imposes a theoretical upper bound on the system lifetime, if a continuous sensing coverage or a partial coverage is required. Such a bound can be surpassed through global scheduling. However, the overhead of global scheduling would increase significantly if the coordination among the nodes goes beyond the neighborhood.

To address these issues, we introduce a two-tier global scheduling method, called uScan. At the first level, coverage is scheduled to activate different portions of an area. We propose an optimal scheduling algorithm to minimize area breach. At the second level, sets of nodes are selected to cover active portions. Interestingly, we show that it is possible to obtain optimal set-cover results in linear time if the layout of areas satisfies certain conditions. We have implemented and evaluated our design on the Berkeley TinyOS/Mote platform [2], using 30 MicaZ motes. The results indicate that uScan is a lightweight solution with significant energy savings, compared with localized solutions.

2 uScan Design

uScan is a two-level schedule algorithm, which works as follows: Suppose we provide sensing coverage to a given area using uScan. First, uScan divides the area into small regions, and decides the working schedules for these regions. This level of scheduling is conceptually independent of the deployment of the nodes. At the second-level, we assign nodes to cover the active regions at different time intervals, using a set-cover technique. By combining the first-level schedule and the set-cover assignment, we can decide the working schedule of individual nodes.

2.1 Assumption

We assume that nodes are time-synchronized and their locations are known. These are common assumptions for many sensor network applications [1, 4, 6]. Accuracy of time synchronization and localization do not need to be precise, because clock drift can be resolved by slightly extending the active duration and localization error can addressed using the method proposed in [8]. For the clarity of the protocol description in the rest of the paper, we refer the sensing area of a node as a circle with a nominal radius r centered at the location of the node. However, our design works under irregular sensing areas as long as nodes are aware of their sensing areas.

2.2 Definition of the Node Schedule

In essence, a sensing coverage algorithm decides the working schedule of individual sensor nodes. Specifically, uScan describes the behavior of a node using two parameters, namely the schedule bits S and switching rate R.

- **Schedule bits S:** It is an infinite binary string in which 1 denotes the active state and 0 denotes the inactive state. Since the sensing coverage schedule is usually periodic, follows a certain pattern. Therefore, we can express S with a regular expression. For example, $(0010)^*$ can be used to denote a repeated off-off-active-off schedule.
- **Switching rate R:** It defines the rate of toggling between states. For example, a switching rate of 0.5HZ requires a node to read one bit from the schedule S every 2 seconds (when consecutive bits have the same value, there is no need to change power state of the node physically).

2.3 Level I: Tile Scheduling

In uScan, we partition an area under surveillance into some small regions of the same shape, a process called tessellation. These small regions are called *tiles*, which can be regular triangles, rectangles or regular hexagons in a 2-D space. The size of tiles is set to be smaller than the minimum target size, so that a target is detected as long as a portion of a tile is covered. In this section, we discuss two methods for the tile-level scheduling. They differ in the energy consumption rate and the detection delay.

- **Line Scan:** Instead of trying to cover all tiles, we only cover a column/row of tiles in a certain interval of time during one round of scan. The covered columns/rows are increasing or decreasing consecutively.
- **Systolic Scan:** Systolic Scan emulates the cardiac cycles of a beating heart. Over the area under surveillance, we sense the tiles from the inner layer to the outer layer continuously.

Both line scan and systolic scan specify only the set of tiles need to be activated (covered) at a given point of time. The task of covering each tile set is accomplished by the second-level node scheduling, which is described in the next section.

2.4 Level II: Node Scheduling

Tile-level scheduling determines the set of active tiles TS_i at the time interval i. In this section, we describe how we can

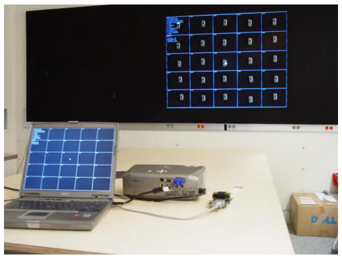


Fig. 1. Test-bed Setup

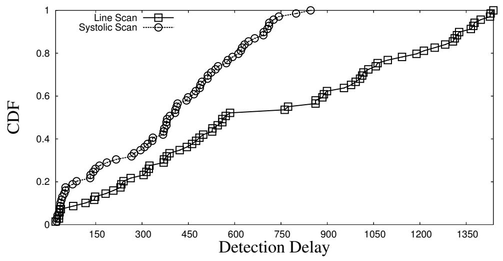


Fig. 2. Detection Delay

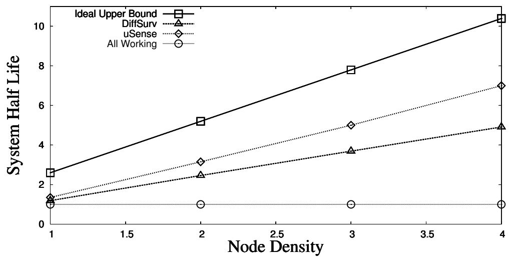


Fig. 3. System Half Life vs. Densities

translate a known tile schedule into a corresponding node schedule bits S, which can be interpreted directly by a generic switching algorithm.

The main idea of our approach is to find the optimal set of nodes which could cover all the tiles that need to be active at time interval i. Before node scheduling, we first map physical node coverage into the coverage bipartite graph according to the coverage relationship. Then we divide node scheduling into two steps. First, for a tile set TS_i, we keep identifying 1-cover set with minimal number of nodes, until the size of 1-cover set is above a certain threshold. Secondly, we create schedules for nodes such that each identified 1-cover set provides coverage to TS_i in a round-robin fashion.

The generic Minimum Set Cover problem has been proven NP-Hard [3]. Fortunately, we find line scan coverage is a special case of the generic set cover problem, because a node only needs to cover a continuous segment of tiles. By mapping the coverage bipartite graph into a directed acyclic graph with following rules:

1. Map N tiles in TS_i into N vertex $V = \{v_1, ..., v_N\}$ and add one extra vertex v_{N+1}.
2. If a node covers a set of tiles $\{T_i, ..., T_{i+n}\}$, we create n directional edges (v_i, v_j) where $v_j = v_{i+1}, ..., v_{i+n+1}$. Each edge has a unit cost.

We reduce the tile set cover problem to the problem of finding out the shortest paths from v_1 to v_{N+1}, with the overall runtime of $O(|V|) + O(|E|)$.

We note that the proposed polynomial algorithm does not apply to generic tile scheduling, where a tile set does not form a continuous curve or where a node can cover multiple segments of a tile set simultaneously. In these cases, we adopt a greedy set-cover method by choosing the node that covers the most number of tiles first.

In order to support line scan or systolic scan in a 2-D space, we need to identify cover sets for the whole area (not just for a single tile set). Thus a node may need to cover multiple tile sets TS_i. To effectively handle the cases where we have to select cover sets for multiple TS, we designed an algorithm that each node maintains a counter SC which records how many times it has been selected into a unique cover sets. While selecting cover sets for each tile set TS, instead of solely consider the number of nodes in the cover sets, the algorithm calculates the minimum cover set among the nodes whose SC counter values are as small as possible.

After obtaining cover sets for every tile set TS_i, we build the final schedule of node according to all the cover sets it's belonged to, which can be executed directly by our generic switching algorithm.

3 Implementation and Evaluation

We have implemented a complete version of uScan on Berkeley TinyOS/Mote platform, using 30 MicaZ motes as shown in Figure 1. The compiled image of a full implementation occupies 21,040 bytes of code memory and 907 bytes of data memory. The results showed that uScan is a lightweight and efficient coverage design. To reveal the system performance at scale, we have conducted some initial large scale simulations with 10,000-node. Under full coverage mode as shown in Figure 3, we demonstrated that our global scheduling algorithms provide significant energy savings over previous protocols such as DiffSurv [8] under metrics such as Half-life, Coverage Overtime and Node Energy Consumption. In the future, we plan to investigate the performance under partial coverage mode at scale as well, with additional metrics such as Detection Delay for Static Targets and Worst-Case Breach (WCB) for Mobile Targets.

4 Conclusion

The major contribution of this work is a two-level global scheduling algorithm called uScan. In the first level, we propose two tile-level scheduling algorithm. In the second level, we propose a linear algorithm to address the set-cover problem when the layout of tiles satisfies certain conditions. We evaluate our architecture with a network of 30 MicaZ motes, an extensive simulation with 10,000 nodes, as well as theoretical analysis. We believe our work has successfully provided flexibility and efficiency for the sensor network coverage problem.

5 References

[1] A. Arora and et al. A Wireless Sensor Network for Target Detection, Classification, and Tracking. *Computer Networks (Elsevier)*, 2004.

[2] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System Architecture Directions for Networked Sensors. In *ASPLOS'00*, 2000.

[3] V. T. Paschos. A Survey of Approximately Optimal Solutions to Some Covering and Packing Problems. In *ACM Computing Surveys*, June 1997.

[4] R. Szewczyk, A. Mainwaring, J. Anderson, and D. Culler. An Analysis of a Large Scale Habit Monitoring Application. In *SenSys'04*, 2004.

[5] D. Tian and N. Georganas. A Node Scheduling Scheme for Energy Conservation in Large Wireless Sensor Networks. *Wireless Communications and Mobile Computing Journal*, May 2003.

[6] G. Tolle, J. Polastre, R. Szewczyk, N. Turner, K. Tu, S. Burgess, D. Gay, P. Buonadonna, W. Hong, T. Dawson, and D. Culler. A Macroscope in the Redwoods. In *Sensys'05*, November 2005.

[7] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks. In *Sensys'03*, November 2003.

[8] T. Yan, T. He, and J. A. Stankovic. Differentiated Surveillance Service for Sensor Networks. In *SenSys'03*, November 2003.

[9] F. Ye, G. Zhong, S. Lu, and L. Zhang. PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks. In *Proc. of International Conference on Distributed Computing Systems (ICDCS)*, May 2003.

uScan: A Lightweight Two-Tier Global Sensing Coverage Design

Yu Gu and Tian He
Department of Computer Science and Engineering, University of Minnesota
{Yugu, tianhe}@cs.umn.edu

Research Issue:

Wireless Sensor Networks (WSNs), consisting of thousands of low-cost sensor nodes, have been used in many application domains such as military surveillance, habitat monitoring and scientific exploration. Limited power supplies and difficulties in harvesting ambient energy make energy conservation a critical issue to address. As one of solutions, energy-efficient sensing coverage extends system lifetime by leveraging on the redundant deployment of sensor nodes. Existing algorithms are designed to be well distributed and localized with solid performance gains. While the state-of-the-art is encouraging, we believe there are some aspects that need further investigation. In most algorithms, extending system lifetime is achieved essentially through coordination among neighboring nodes. The local node density, therefore, imposes a theoretical upper bound on the system lifetime, if a continuous sensing coverage or a partial coverage is required.

Project Overview

To address these issues, we introduce a two-tier global scheduling method, called uScan. At the first level, coverage is scheduled to activate different portions of an area. We propose an optimal scheduling algorithm to minimize area breach. At the second level, sets of nodes are selected to cover active portions. Interestingly, we show that it is possible to obtain optimal set-cover results in linear time if the layout of areas satisfies certain conditions. We have implemented and evaluated our design on the Berkeley TinyOS/Mote platform [2], using 30 MicaZ motes. The results indicate that uScan is a lightweight solution with significant energy savings, compared with localized solutions.

Assumptions:

We assume that nodes are time-synchronized and their locations are known. These are common assumptions for many sensor network applications. Accuracy of time synchronization and localization do not need to be precise, because clock drift can be resolved by slightly extending the active duration and localization error can addressed using the method proposed by DiffSense. For the clarity of the protocol description, we refer the sensing area of anode as a circle with a nominal radius centered at the location of the node. However, our design works under irregular sensing areas as long as nodes are aware of their sensing areas.

Asymmetric Architecture:

We employ an asymmetric architecture to improve the flexibility and extensibility of the design.

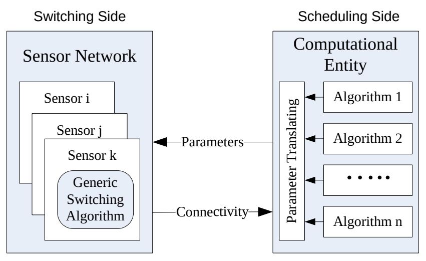


Fig 1. Asymmetric System Architecture

Specification of Node Schedule:

In essence, a sensing coverage algorithm decides the working schedule of individual sensor nodes. Specifically, uScan describes the behavior of a node using two parameters, namely the schedule bits S and switching rate R.

Schedule bits S: It is an infinite binary string in which 1 denotes the active state and 0 denotes the inactive state.

Switching rate R: It defines the rate of toggling between states.

Tier I: Tile Scheduling:

Tile-level scheduling determines the set of active tiles at a certain time interval. Nodes within a sensor network only support a generic switching algorithm, which has neither the concept of tiles nor the partition information of the tiles.

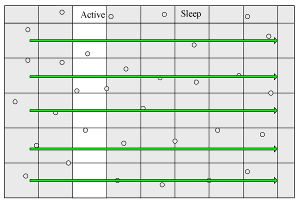


Fig 2. Line Scan

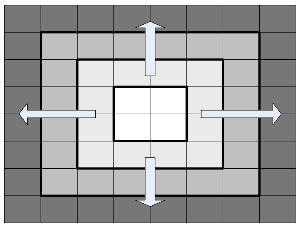


Fig 3. Systolic Scan

Tier II: Node Scheduling

Node Scheduling translates a known tile schedule into a corresponding node schedule bits S. The main idea of our approach is to find the optimal set of nodes which could cover all the tiles that need to be active at time interval i. Before node scheduling, we first map physical node coverage into the coverage bipartite graph according to the coverage relationship. Then we divide node scheduling into two steps. First, for a tile set TS_i, we keep identifying 1-cover set with minimal number of nodes, until the size of 1-cover set is above a certain threshold. Secondly, we create schedules for nodes such that each identified 1-cover set provides coverage to TS_i in a round-robin fashion.

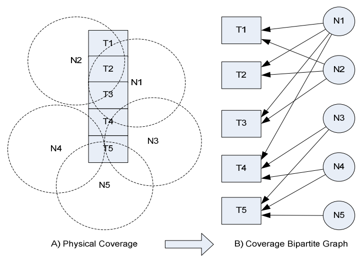


Fig 4. Set-Cover Based Scheduling

Implementation:

We have implemented a complete version of uScan on Berkeley TinyOS/Mote platform, using 30 MicaZ motes as shown below. The compiled image of a full implementation occupies 21,040 bytes of code memory and 907 bytes of data memory.

Fig 5. System Implementation

Evaluation:

We evaluate our architecture with physical system as well as an extensive simulation with 10,000 nodes.

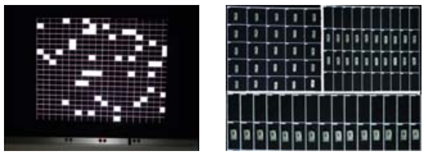


Fig 6. Evaluation Set-up

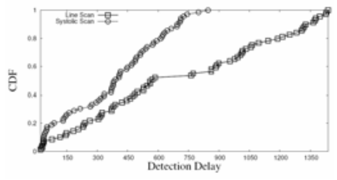


Fig 7. Detection Delay

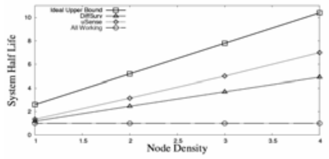


Fig 8. System Life-Time over Different Node Densities

Conclusion:

The major contribution of this work is a two-level global scheduling algorithm called uScan. In the first level, we propose two tile-level scheduling algorithm. In the second level, we propose a linear algorithm to address the set-cover problem when the layout of tiles satisfies certain conditions. We evaluate our architecture with a network of 30 MicaZ motes, an extensive simulation with 10,000 nodes, as well as theoretical analysis. We believe our work has successfully provided flexibility and efficiency for the sensor network coverage problem.

Acknowledgements:

This work is supported by NSF Nets NOSS Program

More information can be found at
Minnesota Embedded Sensor System Group
Http://mess.cs.umn.edu

Achieving Real-Time Target Tracking Using Wireless Sensor Networks

Tian He[§], Pascal Vicaire[†], Ting Yan[†], Liqian Luo[‡], Lin Gu[†], Gang Zhou[†], Radu Stoleru[†], Qing Cao[‡], John A. Stankovic[†] and Tarek Abdelzaher[‡]

[†]Department of Computer Science, University of Virginia
[§]Department of Computer Science and Engineering, University of Minnesota
[‡]Department of Computer Science, University of Illinois, Urbana-Champaign

Target tracking systems need to meet certain real-time constraints in response to transient events, such as fast-moving targets. While the real-time performance is a major concern in these applications, it should be compatible with other important system properties such as energy consumption and accuracy. This work presents the real-time design and analysis of VigilNet, a large-scale sensor network system which tracks, detects and classifies targets in a timely and energy efficient manner. Based on a deadline partition method and theoretical derivations to guarantee each sub-deadline, we are able to make guided engineering decisions to meet the end-to-end tracking deadline. The results from 10,000-node simulation and 200 mote field test reveal the effectiveness of our design.

Categories and Subject Descriptors: C.3. [**Special-Purpose and Application-Based Systems**]: Real-Time and Embedded Systems; C.2. [**Computer Communication Networks**]: Distributed Systems

General Terms: Design, Performance, Experimentation

Additional Key Words and Phrases: Sensor Networks, Real-Time, Tracking, Energy Conservation

1. INTRODUCTION

Recent developments in sensor techniques make wireless sensor networks (WSNs) available to many application domains [Dutta et al. 2005; He et al. 2004; Juang et al. 2002; Simon et al. 2004; Xu et al. 2004; Arora and et al. 2005]. Most of these applications, such as battlefield surveillance, disaster and emergency response, deal with various kinds of real-time constraints in response to the physical world. For example, surveillance may require a sensor node to detect and classify a fast moving target within 1 second before it moves out of the sensing range. Compared with the traditional distributed systems, the real-time guarantee for sensor networks is more challenging due to the following reasons. First, sensor networks directly interact with the real world, in which the physical events may exhibit unpredictable *spatiotemporal* properties. These properties are hard to characterize with traditional methods. Second, although the real-time performance is a key concern, it should be performance compatible with many other critical issues such as energy efficiency and system robustness. For example, it is not efficient to activate the sensors all the time only for the benefit of a fast response. This naive approach severely reduces the system lifetime [He et al. 2004]. Third, the resource constraints restrict the design space we could trade off. For example, the limited computation power in sensor nodes makes the Fast Fourier Transformation not quite suitable for real-time detection. All these issues challenge us with two questions. *How to make the design of a large-scale real-time sensor network system manageable?* And *how to trade off among system metrics while maintaining the real-time guarantee?* Our answer to these questions, presented in this paper, is a case study of the VigilNet system, a real-time outdoor tracking system using a large-scale wireless sensor network.

Our contribution lies in the following aspects: 1) This work addresses a real-world application with a running real-time system, designed and implemented over the last few years. 2) We demonstrate how to guarantee the end-to-end tracking deadline in a complex sensor system. For a given sub-deadline partition, we identify the system configurations that meet the sub-deadlines without compromising other important system properties. 3) The real-time design and tradeoffs are validated by a large-scale field evaluation with 200 XSM motes and an extensive simulation with 10,000 nodes. These evaluations reveal quite a few practical design suggestions that can be applied to other real-time sensor systems.

The remainder of the paper is organized as follows: Section 2 introduces the tracking process in VigilNet. Section 3 identifies the real-time requirements. Section 4 provides a real-time analysis of VigilNet's tracking performance and its tradeoffs. In Section 5 we describe the implementation of the VigilNet system. In Section 6, we evaluate the real-time performance of VigilNet in an outdoor field test. In Section 7, we conduct a large-scale simulation to further validate and analyze the real-time issues in VigilNet. Section 8 discusses the related work. Section 9 concludes the paper.

2. OVERVIEW OF VIGILNET TRACKING OPERATIONS

VigilNet is an energy-efficient surveillance and tracking system, designed for spontaneous military operations in remote areas. In these areas, the events of interest happen at a relatively low rate, i.e. the duration of significant events (e.g., intruders) is very short, compared with the overall system lifetime (e.g., 5-minute tracking per day). According to our empirical results [He et al. 2006], nearly 99% of energy is consumed during the idle-waiting period for potential targets. Therefore to conserve energy, the most effective approach is to selectively turn a subset of nodes off, and wake them up on demand in the presence of significant events. This power management technique fundamentally shapes the VigilNet tracking process. It introduces a set of new delays that traditional tracking systems do not experience.

In this section, we give a brief overview of the VigilNet tracking operation, serving as a background for the real-time design and analysis in the following sections. As shown in Figure 1, after a target enters the area, it activates the first sensor node that can confirm the detection, then other nodes nearby are awakened to form a group to deliver the aggregated reports to the base. More specifically, the VigilNet tracking operation has six phases:

A. **Initial Activation:** VigilNet stays in the power management state when there are no targets. The power management protocol puts nodes into either one of two states: *sentry* and *non-sentry*. In brief, a node becomes a sentry node

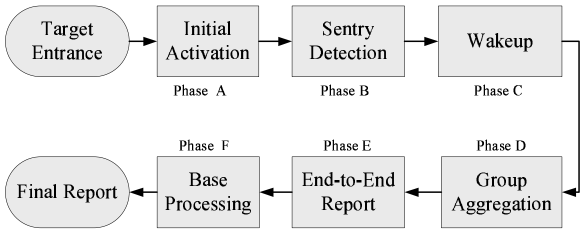


Fig. 1. The Delay Breakdown in Tracking Operation

if it is a part of the routing infrastructure or it needs to provide the sensing coverage. Otherwise, it becomes an inactive non-sentry node. The details of sentry selection can be found in [He et al. 2004]. If the sentry nodes are awake 100% of time (i.e. the deployed area is always covered), any incoming target is covered by at least one sentry node immediately. On the other hand, if the sentry nodes have a certain duty cycle (i.e. they go to sleep and wake up periodically to save energy), there will be an initial activation delay, denoted as $T_{initial}$, before the first sentry node starts to sense the incoming target.

B. **Initial Target Detection:** After the initial activation, it takes a certain delay, defined as $T_{detection}$, for the first sentry node to *confirm* the detection. This delay consists of the hardware response delay, the discrete sampling delay and the delay to accumulate a sufficient number of samples before a detection algorithm recognizes the target.

C. **Wake-up:** Normally, the detection from a single sentry node does not provide sufficient confidence in detection and classification, therefore a group-based tracking is designed in VigilNet. In order to form a group with a reasonable size, non-sentry nodes need to be awakened after the initial target detection by a sentry node in Phase B. We define the wake-up delay T_{wakeup} as the time required for a sentry node to wake up other sleeping non-sentry nodes. This delay is determined by the toggle period of none-sentry nodes.

D. **Group Aggregation:** Once awakened, all nodes that detect the same target join the same logic group in order to establish a unique one-to-one mapping between this logic group and the detected target. Each group is represented by a leader which maintains the identity of the group as the target moves through the area. Group members (who by definition can sense the target) periodically report to the group leader. A leader reports a detection to the base after the number of member reports exceeds a certain threshold, defined as the degree of aggregation (DOA). We use $T_{aggregation}$ to denote the group aggregation delay, which is the time required to collect and process the detection reports from the member nodes.

E. **End-to-End Report:** After group aggregation, the leader node reports the event to the nearest base. Multiple bases are used to partition a network into several sections, in order to bound the end-to-end delivery delay T_{e2e}.

F. **Base Processing (T_{base}):** A base is in charge of processing the reports from the leaders of different logic groups. Since the reports from the same logic group are spatiotemporally correlated, a string of consecutive reports can be further analyzed and summarized for end users. For example, taking the time stamps and the locations of targets as the inputs, a base uses the least-square estimation to obtain the velocity of each target.

3. REAL-TIME REQUIREMENTS IN VIGILNET

To ensure the effectiveness of target tracking, VigilNet must meet a certain real-time constraint. Specifically, VigilNet should detect, classify and analyze the incoming targets within a certain end-to-end deadline (e.g., 5 seconds from Phase A to F). As shown in Section 2, the end-to-end deadline is affected by many system parameters, which form a high-dimensional design space where the number of possible configurations increases exponentially with the number of system parameters. It is intractable to identify a system-wide global optimal solution within this design space. Therefore, we adopt the deadline partition method to divide the end-to-end deadline into multiple sub-deadlines. By confining the real-time decisions within each phase, we make the end-to-end analysis manageable in a lower-dimensional design space. For a given end-to-end deadline, a designer can make an initial partition, according to the workload, system resources available and preliminary estimation of the delay in each phase. As a concrete example, supposing a target enters the field with a speed up to 20 mph, to guarantee that this target can be detected by the first sentry node with a probability higher than 90%, we need to design a detection algorithm with a sub-deadline less than 1 second, assuming the detection range is 10 meters. After the initial partition, one can identify a system configuration to guarantee the sub-deadlines, based on the analysis in this work. As long as the individual sub-deadlines are met, we have a certain guarantee on the end-to-end delay.

4. VIGILNET REAL-TIME TRACKING ANALYSIS

The description in this section follows the natural order of VigilNet's tracking operations presented in Section 2. Such design and analysis is validated later with a real system implementation consisting of 200 XSM nodes as well as a large-scale simulation in Section 6 and 7, respectively.

4.1 Initial Activation Delay and Its Tradeoffs

In a duty-cycle-based power management scheme, the sentry nodes go to sleep and wake up periodically. In this case, the initial activation delay $T_{initial}$ may not be zero, because the sentry nodes near the target's entry point may be asleep when the target enters the field. In this section, we identify a quantitative relationship between the energy savings and the $T_{initial}$, which helps us make decisions to guarantee that the initial activation finishes within a given sub-deadline $D_{initial}$.

In our VigilNet design, all sentry nodes agree on a common sentry toggle period P and a common sentry duty cycle SDC. The starting time of a period is randomized in each node. For each period, a sentry wakes up and stays awake for $P \cdot SDC$, then goes to sleep for $P \cdot (1 - SDC)$. Assuming a target enters the tracking area from point s for L meters till it reaches the point e, as shown in Figure 2(a), we first derive P_r, the probability that a single sentry node detects this target. Obviously, the nodes that may detect the target must be in the rectangle or the

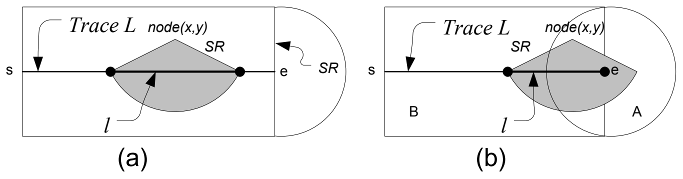


Fig. 2. Detection Probability for fast targets

semi-circle shown in the Figure 2(a). The size of the area is $2SR \cdot L + \pi \cdot SR^2/2$, where SR is the Sensing Range. For a single node located at (x, y) in this area, the probability that the node detects the target $P(x, y)$ is $SDC + l(x, y)/(P \cdot TS)$ if $SDC + l(x, y)/(P \cdot TS) \leq 1$, where $l(x, y)$ is the overlapping length of the node's sensing range and the target's trace, and TS is the Target Speed. If we consider all possible locations in this area, we can get P_r in Equation 1 by integrating and normalizing $P(x, y)$ over the area. We note that when (x, y) is in the circle (area A), as shown in Figure 2(b), $l(x, y) = \sqrt{SR^2 - y^2} + L - x$. When (x, y) is in area B, $l(x, y) = 2\sqrt{SR^2 - y^2}$.

$$\begin{aligned} P_r &= \frac{\int\limits_A (SDC + \frac{\sqrt{SR^2-y^2}+L-x}{P \cdot TS})ds + \int\limits_B (SDC + \frac{2\sqrt{SR^2-y^2}}{P \cdot TS})ds}{(2SR \cdot L + \pi SR^2/2)} \\ &= SDC + \frac{\pi \cdot SR \cdot L}{(2L + \pi \cdot SR/2) \cdot TS \cdot P} \end{aligned} \tag{1}$$

We note that P_r calculated by Equation 1 is valid only when the target speed is faster than or equal to $2SR/(P - P \cdot SDC)$. We define this as a *fast target*. For a target with a speed slower than $2SR/(P - P \cdot SDC)$, which we define as a *slow target*, it may happen that for a node located at (x, y), the corresponding $l(x, y)$ is greater than $(P - P \cdot SDC) \cdot TS$, so that $SDC + l(x, y)/(P \cdot TS) > 1$. For the node at this location, the probability that it detects the target is 1 instead of $SDC + l(x, y)/(P \cdot TS)$. Therefore, we need to revise the result in Equation 1 for slow targets. We define variable a such that $SDC + 2a/(TS \cdot P) = 1$. In Figure 3, it can be easily proven that if a node appears inside area C bounded by the dashed arc and lines, the probability that it detects the target is 1. Note that the distance between the dashed line and the target trace is $\sqrt{SR^2 - a^2}$. The dashed arc is centered at $(L - 2a, 0)$ and its radius is SR. The rest of the area is divided by the circle with radius SR centered at $(L, 0)$ into area A' and area B'. The detection probabilities for nodes in area A' and area B' have the same forms as those for nodes in area A and area B in the fast target case, correspondingly. Then we have

$$\begin{aligned} P_r &= \frac{\int\limits_{A'} (SDC + \frac{\sqrt{SR^2-y^2}+L-x}{P \cdot TS})ds + \int\limits_{B'} (SDC + \frac{2\sqrt{SR^2-y^2}}{P \cdot TS})ds + \int\limits_C 1 ds}{(2SR \cdot L + \pi SR^2/2)} \\ &= SDC + \frac{\pi \cdot SR^2 \cdot L + min[(L-a)k(SR,a),0]}{(2SR \cdot L + \pi \cdot SR^2/2) \cdot TS \cdot P}, \end{aligned} \tag{2}$$

in which $k(SR, a) = 2a\sqrt{SR^2 - a^2} - 2SR^2 \cos^{-1}(a/SR)$.

In the paper, we omit the intermediate derivation, for those interested, more information can be found at [Cao et al. 2005].

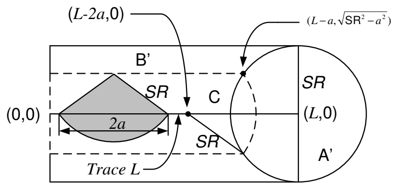


Fig. 3. Detection Probability for slower targets

Now we are ready to provide a statistical real-time guarantee for the initial activation process, i.e., we need to ensure a target is detected before the sub-deadline $D_{initial}$. Equivalently, a target should be detected before it enters for $L = TS \cdot D_{initial}$ meters. Obviously, $P(T_{initial} < D_{initial})$ equals $P(T_{initial} \cdot TS < L)$, where $P(T_{initial} \cdot TS < L)$ is the probability that at least one of nodes in the area (A+B) detects the target. If there are n nodes in the area, the probability that at least one of them detects the target is $1 - (1 - P_r)^n$. Suppose the sentry density is D_s and n conforms to a Poisson distribution with parameter λ $=(2SR \cdot L + \pi \cdot SR^2/2)D_s$, therefore, the probability that the initial activation finishes before sub-deadline $D_{initial}$ is:

$$P(T_{initial} < D_{initial}) = P(T_{initial} \cdot TS < L) = 1 - e^{-P_r \cdot \lambda} \tag{3}$$

Equation 3 identifies a feasible region for us to decide the system parameters such as sentry duty cycle (SDC) and sensing range (SR) to ensure the real-time property in Phase A. In addition, we can obtain the expected value of $T_{initial}$ from the formula $E(T_{initial}) = \int_0^\infty (1 - P(T_{initial} < t))dt = \int_0^\infty (1 - P(SD < TS \cdot t))dt$. According to Equations 1 and 3, we have the expected delay for a target whose speed is greater than or equal to $2SR/(P - P \cdot SDC)$:

$$E(T_{initial}) = \frac{e^{-SDC \cdot \pi \cdot SR^2 \cdot D_S/2}}{(2SR \cdot SDC \cdot TS + \pi SR^2/P)D_S} \tag{4}$$

Similarly, for a target whose speed is lower than $2SR/(P - P \cdot SDC)$, we have

$$E(T_{initial}) = \frac{e^{-SDC \cdot \pi \cdot SR^2 \cdot D_S/2}[1 - \frac{k(SR,a)e^{-(2SR \cdot SDC \cdot TS \cdot P)(1-SDC)D_S/2}}{2SR \cdot SDC \cdot TS \cdot P + \pi SR^2 + k(SR,a)}]}{(2SR \cdot SDC \cdot TS + \pi SR^2/P)D_S} \tag{5}$$

One caveat in the analysis needs some attention. Above we derive the expected detection delay for a duty cycle based system with *random deployment*. However, sentry nodes are located more evenly than totally randomly case [He et al. 2004]. Fortunately, we can prove that the random deployment case provides a theoretical upper bound for the sentry-based deployment case. It can be easily proved that if for all t, $P(T_1 < t) > P(T_2 < t)$, we must have $E(T_1) < E(T_2)$. For $0 < P_r < 1$, $1 - (1 - P_r)^n$ is a strictly concave function of n. Therefore, $E(1 - (1 - P_r)^n) \leq 1 - (1 - P_r)^{E(n)}$, and the left side of the equation equals the right side if and only if n is a constant. Given the same $E(n)$, the more scattered the distribution of n is, the smaller the value of $E(1-(1-P_r)^n)$ is. Since the sentry nodes are selected more uniformly than the random case, $P(T_{initial} < D_{initial})$ for the sentry based system is greater than a totally randomly distributed system, and therefore the expected

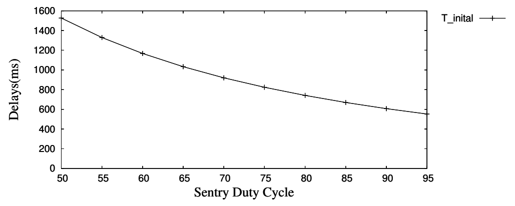


Fig. 4. Initial Delay vs. SDC

delay is smaller. The expected delay for the random case can be used as an upper bound for the expected detection delay for a more evenly distributed system. Later, we will see from the simulation that the analytical result overestimates the $T_{initial}$ by 15%.

We can further take the detection delay $T_{detection}$ into account, since a successful detection in Phase B activates a full tracking process. In this case, we establish an equivalent model for $T_{initial}$. Specifically, in Equation 4 or Equation 5, we substitute SDC with the effective sentry duty cycle $SDC_{eff} = SDC - T_{detection}/P$ and substitute SR with the effective sensing range $SR_{eff} = \sqrt{SR^2 - (T_{detection} \cdot TS/2)^2}$. Figure 4 gives a more concrete view of the tradeoff between SDC and expected $T_{initial}$. We take parameters from the real VigilNet implementation: $D_S = 0.01 node/m^2$, $P = 10s$, $SR = 10m$, $TS = 10m/s$ and $T_{detection} = 1000ms$. This result is consistent with what we obtained from the real experiments and simulations.

4.2 Sentry Detection Delay and Its Tradeoffs

After the initial delay in Phase A, a target approaches the vicinity of a sensor which begins to observe a different signal pattern than that without a target. With the current sensing algorithms, the signal pattern can be amplitude, frequency, or a combination of the two. We call the signal pattern corresponding to a target *a target signature*. The recognition of a target signature indicates a sensor-level detection, and produces data for higher-level detection and classification algorithms.

As defined before, $T_{detection}$ is the time for a detection algorithm to recognize a target signature. This delay must be smaller than a certain sub-deadline $D_{detection}$. Multiple reasons contribute to this delay. First, the sensor hardware has a response delay for the physical signals that the target generates. Second, the sensing circuitry requires special operations with a further delay. For example, the magnetometer in MICA2 node [CrossBow 2003] takes about $35ms$ to stabilize after the potentiometer adjustment. Third, the sampling is discrete and periodic, not continuous, which leads to sampling delay. Fourth, the target signature itself may be time related (e.g., a certain frequency), which can not be recognized from just one sample. Finally, the VigilNet system is designed for outdoor deployment. It must adapt to environmental noise and dynamics, such as the change of temperature, the motion of small plants on windy days, and the sound of animals. Hence, noise filtering is a key step in the recognition of the target signal pattern. Such filtering usually needs to accumulate and analyze measurements over a period of time, and imposes a delay in detection.

Now we describe how to decide the sub-deadline $D_{detection}$. Obviously, a detection algorithm must finish before a target moves out of the sensing range of a node.

Suppose that the nominal sensing area is a circle with a fix sensing range SR, the amount of time a target stays in a node's sensing range can be derived from the speed of the target, TS, and the minimum distance from the target's trajectory line to the sensor node. Since the target trajectory intersects with the sensing circle randomly, we assume this minimum distance is uniformly distributed within $[0, R)$, therefore the probability that a target stays in one sensing circle for at least $D_{detection}$ seconds can be calculated as

$$\begin{array}{rl} P(t > D_{detection}) = \sqrt{1 - \frac{(TS \cdot D_{detection})^2}{4SR^2}} & D_{detection} < \frac{2SR}{TS} \\ P(t > D_{detection}) = 0 & D_{detection} \geq \frac{2SR}{TS} \end{array} \tag{6}$$

According to Equation 6, the sub-deadline $D_{detection}$ can be decided by choosing a desired $P(t > D_{detection})$ value.

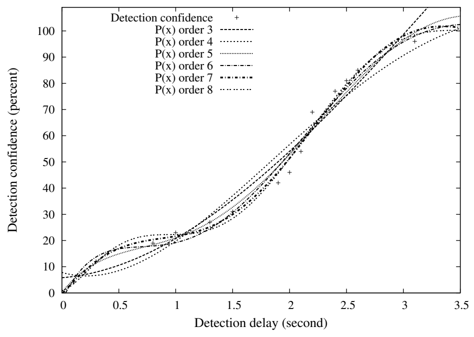


Fig. 5. Detection Confidence vs. Detection Delay

In addition, we desire to know how a detection algorithm performs under a given sub-deadline $D_{detection}$. We define the *Detection Confidence* (DC), as the confidence on the target detection, i.e., 100% DC indicates this sensor has no doubt about the existence of the target. Normally, the longer $D_{detection}$ is, the more information about target signature a sensing algorithm can obtain, and therefore, it can achieve a higher detection confidence DC. Such a relationship depends on the type of sensors. In order to quantitatively analyze the relation between DC and $D_{detection}$ as a case study, we performed experiments on XSM motes with the magnetic sensing algorithm detecting a moving vehicle in an outdoor environment. We approximate the sensing range as 7 meters around the sensor node, according to empirical data. Figure 5 plots the relation between the detection confidence and the detection delay, based on the experiments. As we can see from the figure, DC does not have a linear relation to $D_{detection}$. Based on experimental measurements, we use a polynomial to characterize DC versus $D_{detection}$. Figure 5 shows a series of polynomials of different orders that fit the points representing the relation between the detection confidence and the detection delay. The plotting indicates that the polynomials of an order higher than 5 are fairly close to each other and fit the points well. Hence, we choose the polynomial of order 5 to characterize the relation, as shown below.

$$DC = f(D_{detection}) = \sum_{i=0}^{5} a_i D^i_{detection} \tag{7}$$

The coefficients of the polynomial calculated from the curve fitting are $a_5 = 1.0999, a_4 = -13.1138, a_3 = 51.3443, a_2 = -73.2343, a_1 = 54.6671, a_0 = 0.2402$. The polynomial $f(D_{detection})$ characterizes the relation of the detection confidence and the imposed sub-deadline $D_{detection}$ when the vehicle is moving at a relatively low speed. In the scenarios where the vehicles move faster, the detection delay tends to be shorter and detection confidence will be higher because the targets impose a faster change to the sensor readings. Hence, $f(D_{detection})$ represents a conservative estimation of the detection confidence, given a certain amount of time available to the sensor node to capture and process the target signals.

We note that in the analysis of the time-related properties of the sensing algorithms, we choose such a conservative-case approach instead of a worst-case approach. In many cases, the worst-case scenario is a rare event that the system is not designed to handle well. For example, with the magnetic sensing algorithm, the worst case of detection delay is infinity – if a vehicle moves extremely slowly, it provides a low-frequency signal just as the background noise, resulting in a non-detection for that target. We note that an analysis with such a worst-case scenario provides little insight into the system. To represent a reasonably practical scenario, we study a conservative case in which a target can be detected.

In conclusion, we must provide a detection algorithm that finishes before a given sub-deadline $D_{detection}$. According to Equations 6 and 7, when running a detection algorithm with a sub-deadline $D_{detection}$, one node can detect $P(t > D_{detection})$ percent of targets with DC percent of the confidence in detection. This analysis justifies the benefits of fast detection algorithms and the need for group aggregation to improve the detection confidence.

4.3 Wake-up Delay and Its Tradeoff

Once a target is detected in Phase B, we need more nodes to join in order to increase the confidence in detection. We design a wake-up service to activate the non-sentry nodes after the sentry nodes detect the incoming targets. Different target speeds impose different sub-deadlines D_{wakeup} to the wake-up services.

Normally the wake-up service can be supported either through hardware or software. Several hardware solutions have been proposed in [Dutta et al. 2005; Gu and Stankovic 2004]. Since the wake-up circuits accumulate the ambient energy slowly, the current hardware solutions are not fast enough for the real-time target tracking. Therefore, we propose a software-based wake-up strategy, which has a short average delay and a predictable worse-case delay.

The wake-up operation goes as shown in Figure 6. A non-sentry actually does not sleep all the time. It periodically wakes itself up, quickly senses the radio activity at a particular frequency. If no radio activity is detected, this node goes back to sleep, otherwise it remains active and starts to sample the environment. We control the non-sentry operation through two parameters: *Toggle Period (TP)* and *Channel Clear Access duration (CCA)*. The toggle period is defined as the time interval

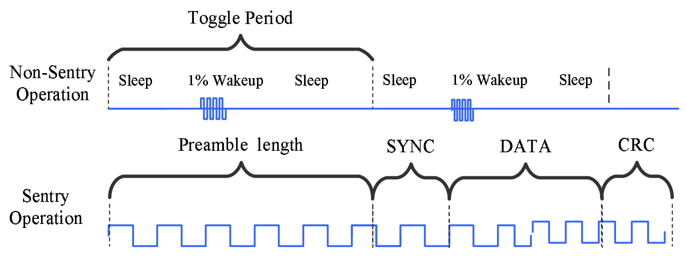


Fig. 6. The Wake-up Operation

between two consecutive wake-up instances. The CCA is defined as the minimal time for a radio module to detect the existence of the radio signal. For example, the CC2420 radio transceiver takes at least $2ms$ (8 symbol periods, as specified by 802.15.4 [IEEE]) to access the radio activity. Based on TP and CCA, we can get the Non-Sentry Duty Cycle ($NSDC$) as $\frac{CCA}{TP}$. At the sentry side, once a sentry detects a target, it broadcasts a radio message with a long preamble. This long preamble is guaranteed to be sensed by neighboring non-sentry nodes as long as this preamble has a length equal to or longer than the toggle period of non-sentry nodes. The worst case wake-up delay WC_{Delay} equals TP. In other words, the sub-deadline D_{wakeup} can be ensured trivially in our design by setting $TP = D_{wakeup}$. Let the power consumption for an active node during a unit of time be E, the energy consumption for a non-sentry node is $\frac{E \times CCA}{TP}$. Since the amount of time to check the radio activity (CCA) is constant for a specific radio hardware, the length of the toggle period determines the energy consumption rate in non-sentry nodes. In general, a long toggle period TP leads to a low energy consumption, however, it also leads to a long delay in waking up the non-sentry nodes. Figure 7 shows such a tradeoff, using the CC1000 radio transceiver for MICA2/XSM motes as an example. As shown in Figure 7, a sub-deadline of 200ms lead to a 99% energy saving for the non-sentry nodes.

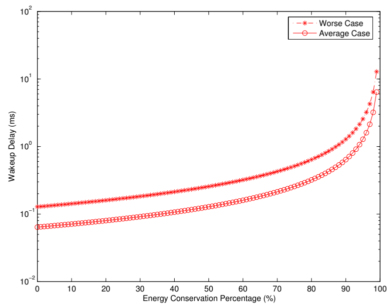


Fig. 7. Wakeup Delay Vs. Non-Sentry Energy Saving

4.4 Aggregation Delay and Its Tradeoffs

Once all nodes near the target are awakened in Phase C, the group-based tracking begins. To avoid an excessive power consumption, instead of relaying every detection message back, VigilNet sends only aggregates to the base stations for further processing. Such an online aggregation process is subject to a certain sub-deadline $D_{aggregation}$ determined by the target speed and the node density.

Specifically, we organize the nodes in the vicinity of a target into one group. We use a semi-dynamic leader election [Luo et al. 2005] to minimize the delay. Nodes that detect the target become the group members, which, upon detection, immediately report their own locations and sensing data to a leader. The leader then averages the locations of members as the estimates of the target positions, and sends such estimates to a base station. To filter out the sporadic false alarms of individual nodes, we introduce a configurable parameter, DOA (Degree of Aggregation), which forces the leader to withhold reports to a base station until the number of received member reports reaches DOA. To achieve a high confidence in target detection, one should set a high DOA value (e.g., 4). On the other hand, a higher DOA value inevitably introduces a longer group aggregation delay since the leader waits longer to expect more member reports. This tradeoff allows us to choose appropriate DOA to meet the sub-deadline $D_{aggregation}$.

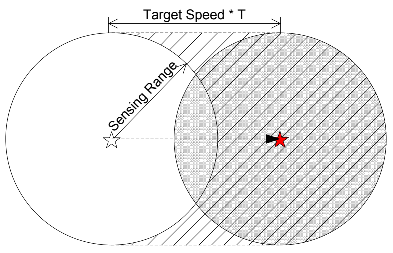


Fig. 8. The Detection Areas Before and After Movement

The relation between DOA and the group aggregation delay is complicated by various factors, e.g., the sensing range, the target speed, and the node density. Therefore, we make several assumptions to simplify the analysis, including a circular sensing range, a straight target trajectory and randomly distributed nodes. Based on these assumptions, Figure 8 depicts the movement of a target with a speed TS for a time period T. Again, the sensing range of a sensor node is SR. The white circle and the grey circle denote the detection area of the target before and after movement, respectively. Nodes located in the diagonally lined area are the new detectors of the target, which contribute to DOA by sending reports to the leader. To guarantee a certain sub-deadline $D_{aggregation}$, the number of new detectors must exceed or equal DOA before the sub-deadline $D_{aggregation}$:

$$D_{aggregation} \geq T_{aggregation} = \frac{DOA}{2 \cdot SR \cdot TS \cdot D} \tag{8}$$

where D represents the node density. Note that after the wake-up process, not only the sentry nodes but also the non-sentry nodes participate in the tracking. Equation 8 quantitatively reveals a feasible region for us to guarantee the sub-deadline $D_{aggregation}$. For example, if the network density (D) and the sensing range (SR) are fixed, we can exploit a feasible solution, using different DOA values under different target speeds. Figure 9 gives a more concrete design space by depicting the group aggregation delay for varied DOA values and target speeds when the sensing range is 10m, the node density is 1 per 100 m^2. We note that this result is consistent with the results obtained from the large-scale simulations presented in Section 7.

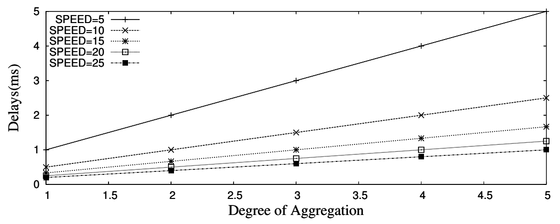


Fig. 9. Minimal Group Aggregation Delay for Varying DOA and Target Speed

4.5 Communication Delay and Its Tradeoff

After group aggregation in Phase D, the leader delivers the aggregated tracking reports to a nearby base. Suppose the end-to-end communication sub-deadline is D_{e2e} and one-hop worst case communication delay is T_{WC_MAC} [He et al. 2003], we need to ensure that the number of hops is smaller than D_{e2e}/T_{WC_MAC}. For a given node density, the hop length L_{hop} can be estimated through Kleinrock-Silvester formula [L.Kleinrock and J.Slivester 1978], which gives the correlation between the hop length L_{hop}, the communication range CR and the number of neighbors N as:

$$L_{hop} = CR \times (1 + e^{-N} - \int_{-1}^{1} e^{-\frac{N}{\pi}(arccos(t) - t\sqrt{1-t^2})} dt) \tag{9}$$

Therefore, to guarantee a sub-deadline D_{e2e}, when we deploy the network, we should ensure that every node can reach a base within a radius of L_{e2e}:

$$L_{e2e} = \frac{D_{e2e} \cdot L_{hop}}{T_{WC\_MAC}} \tag{10}$$

In VigilNet, the sub-deadline D_{e2e} is guaranteed by partitioning the whole network into multiple sections based on the Voronoi diagram [Okabe et al. 2000]. Specifically, a network with n bases is partitioned into n Voronoi sections such that each section contains exactly one base and every node in that Voronoi section is closer to its base than to any other base inside the network.

4.5.1 *Base Deployment Strategy.* We have shown that an ideal deployment should ensure that each node is able to reach a base within a distance of L_{e2e}, so that the sub-deadline D_{e2e} can be satisfied. This possesses an implicit requirement on the number of base stations and their positions. We therefore provide a detailed analysis regarding this requirement and compare the performances of different strategies.

We model the area S with each side as D. Suppose the total number of deployed base stations is N, each serving nodes located within a radius of L. We assume that a large number of other non-base nodes are deployed in the area as well. The problem is, what is probability that every non-base node can reach a base within a distance of L, given a certain deployment strategy? Furthermore, what is the best deployment strategy available? We shall analyze three different deployment strategies: random, grid and optimal. In particular, we show that the optimal strategy is a special case of the grid deployment.

We first consider random deployment. We derive the probability in question as follows. Consider an arbitrary point Q under question. Since each base can serve a radius of L, once a base station is deployed, we know that the point Q has a probability of $\frac{\pi L^2}{S}$ of being located inside this base's service radius. Therefore, once N bases are deployed, the coverage probability for an arbitrary point Q is $1-(1-\frac{\pi L^2}{S})^N$. Notice that this derivation does not take into account the boundary effect. This approximation is valid when $S \gg \pi L^2$, as verified in our experiment.

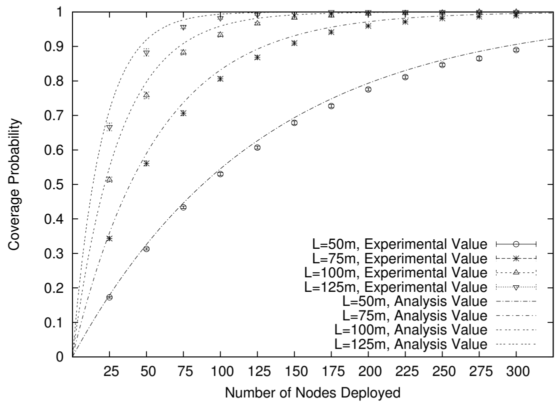


Fig. 10. Random Deployment Performance

Figure 10 validates our analysis on the performance of the random deployment strategy. In particular, we assume that $D = 1000m$. The number of bases N and the serving distance L are both adjustable. All simulation results are plotted based on 50 rounds of data, and the confidence interval is 95%. As shown in this figure, the analysis result roughly fits the experimental data, with certain inconsistencies. These inconsistencies are introduced by the boundary effect: the bases deployed near the boundary have a service area less than πL^2, therefore, the observed coverage probability is slightly lower than the predicted coverage probability.

An interesting problem regarding deployment strategies is *redundancy.* Since typically more bases than needed are provided, it is interesting to consider the ratio between the number of base stations deployed to the minimum number of base stations required. For example, when $L = 100m$, using the random deployment, we observe that roughly 150 bases are needed to provide each potential node real-time service (the coverage probability is more than 98%). The redundancy can be calculated at $\frac{\pi L^2 \times N}{D^2}$, which is 4.71. This is indeed quite high. We, therefore, discuss more efficient deployment strategies, assuming we can position the base stations at desired places accurately.

We focus on two types of grid strategies, square based and hexagon based. These strategies are shown in Figure 11.

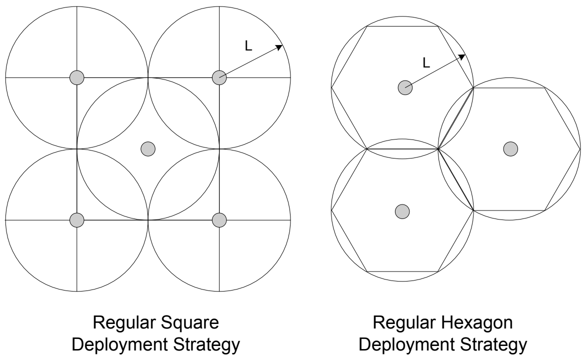

L
L
Regular Square Deployment Strategy
Regular Hexagon Deployment Strategy

Fig. 11. Regular Deployment Strategies

In the first type of grid deployment, the base stations form a regular square structure. The redundancy can be determined to be about 1.57. The second type of grid deployment forms the *honeycomb* structure, where regular hexagons are used. Notice that this figure also shows the Voronoi diagram partitions associated with the honeycomb structure. The second grid deployment has a redundancy of $\frac{2\pi}{3\sqrt{3}}$, or approximately 1.208.

Previous literature [Williams 1979] has proved that the optimal redundancy ratio for any circle covering is exactly $\frac{2\pi}{3\sqrt{3}}$. This result indicates that the honeycomb based node deployment is the optimal strategy. Indeed, boundary effect also exists in this type of deployment, however, when the area is considerably large, the actual redundancy ratio should approach the optimal bound.

4.6 Base Processing Delay and Its Tradeoffs

After a base receives the reports delivered in phase E, it performs the high-level processing such as the velocity estimation. In order to do so, a base node needs to accumulate several reports from the network. The delay to accumulate the reports T_{base} is subject to its sub-deadline D_{base}. We defined the minimal number of reports needed by the base as K. This value can be one, if the in-networking processing is sufficient. The frequency of reports depends on the speed of the target and the aggregation of locations from nodes at different locations. From the analysis in Section 4, we know that after the target enters the system for time t, the expected number of nodes that can sense the target is $(\pi \cdot SR^2/2 + 2SR \cdot TS \cdot t)D$.

Obviously, if the target goes further for Δt, the expected number is increased by $2SR \cdot TS \cdot \Delta t$. Considering the detection delay $T_{detection}$, only nodes that are $\sqrt{SR^2 - (T_{detection} \cdot TS/2)^2}$ meters away from the target trajectory can recognize the target. Therefore, we can estimate the number of reports (NR) generated before the sub-deadline D_{base} as:

$$NR = (2TS \cdot D \cdot \sqrt{SR^2 - (T_{detection} \cdot TS/2)^2}) \cdot D_{base} \tag{11}$$

Alteratively, to guarantee D_{base}, we need to select K, the minimal number of reports needed by the base, to be a value smaller than NR.

Now we consider how the selection of K impacts the accuracy in velocity estimation. Since each location report is an approximation of the target location, there is an error in the result of velocity estimated using the least square method. Without loss of generality, we first consider the velocity along the x-axis. Statistics has established the variance of the estimated slope in a two-variable least square linear regression as:

$$\frac{\sigma^2}{\sum_{i=1}^{K}(x_i - \bar{x}^2)},$$

where σ is the standard deviation of the disturbance, which in our case is the detection error of a single report; x_i in our case is a timestamp. It is hard to get the distribution of $\sum_{i=1}^{K}(x_i - \bar{x})^2$, but a rough estimation can be obtained by a simplification so that the values of x_i are evenly distributed and $x_i = i/(2D \cdot SR \cdot TS \cdot P_R)$. Thus we can get an estimation of the standard deviation of the velocity:

$$\frac{4\sigma \cdot D \cdot SR \cdot TS \cdot P_R}{\sqrt{3K(K+1)(K-1)}}, \tag{12}$$

where σ is the standard deviation of the location error of a single report. Equation 12 reveals the tradeoff between the accuracy in tracking and the delay in base processing. In brief, T_{base} increases linearly with the number of reports required and the standard deviation of the velocity estimation reduces approximately linearly with $K^{-3/2}$.

4.7 Summary of the Analysis and Tradeoffs

Dealing with the physical world, many sensor-based systems must respond to external stimuli within certain time constraints. Such constraints could change over time with the changes of the application objectives. For example, a surveillance system should be able to track fast vehicles at a high energy budget as well as slow personnel at a smaller budget. So it is desirable for a system designer to have the ability to trade off the system parameters to satisfy certain real-time constraints. In this section, we use the deadline partition method to guarantee the sub-deadline of each phase, consequently guaranteeing the end-to-end deadline. This approach makes the real-time design for a complex sensor network manageable. Since VigilNet aims at various tracking scenarios, for a given end-to-end deadline, the actually partition among the phases would vary significantly. Our analysis is independent of how the sub-deadlines are assigned, which gives the designer more flexibility to choose appropriate partition. Currently, the deadline partition is done statically, and we shall investigate the solutions that allow dynamic online partition in the future.

We note our analysis provides a set of generic design guidelines for other tracking systems with or without certain features. For example, the tracking system presented in [Arora et al. 2004] does not consider the power management, which makes the analytical results of $T_{initial}$ and T_{wakeup} trivially zero, while other analytical results are still applicable. Other notable insights from our analysis are: First, to guarantee the same sub-deadline, a higher node density is desired in the slow-target case, however a slower duty cycle can be tolerated without jeopardizing the detection. Second, it is very beneficial to increase the wake-up delay, when possible, in exchange for the energy saving. Third, fast detection algorithms are essential. Fourth, a low network density increases the group aggregation delay, which indirectly reduces the detection confidence. Fifth, theoretically, honeycomb is the optimal base placement strategy.

We also note due to the unpredictable and statistical nature of environmental inputs (e.g., a target could move infinitely slowly, and sensing and communication ranges could be highly irregular), VigilNet is not quite amenable to the traditional precise worst-case real-time analysis. Nevertheless, the analytical results we provide can assist the designer to provide soft real-time guarantee and make guided decisions on system configurations. In the Section 6 and Section 7, we validate our real-time design and analysis through a physical test-bed with 200 XSM motes as well as a large-scale simulator with 10,000 nodes, respectively.

5. SYSTEM IMPLEMENTATION

A large portion of code of VigilNet is written in NesC [Gay et al. 2000], a module oriented extension of the C programming language. Since the concept of traditional OS kernels does not exist in TinyOS [Hill et al. 2000], a NesC programmer can directly access the hardware devices, which facilitates the time analysis within a single node [Mohan et al. 2004]. The network infrastructure in VigilNet is a multi-path diffusion tree rooted at bases. The contention-based B-MAC protocol [Polastre and Culler 2004] is the default media access control protocol, which has certain uncertainty in the communication delay. Three detection algorithms are designed separately for acoustic, magnetic and motion sensors. They identify the target signatures through a lightweight classification scheme as described in [Gu et al. 2005]. VigilNet consists 40,000 lines of code, supporting multiple existing mote platforms including MICA2, MICA2dot and XSM. The compiled image occupies 83,963 bytes of code memory and 3,586 bytes of data memory.

Among 30 protocols implemented within VigilNet, we only describe the time-related services here. Other information can be found at [He et al. 2006; He et al. 2006; He et al. 2004]. VigilNet needs a millisecond-level synchronization to coordinate the operations among the nodes. In addition, to obtain precise timing measurements in the experiments, we need a network-wide synchronization between a base and other nodes within the field. Several well-known schemes are able to achieve a high synchronization precision, however they do not match well with VigilNet requirements. GPS-based schemes [Wellenhoff et al. 1997] typically achieve persistent synchronization with a precision of about 200 ns. However, GPS devices are expensive and bulky. The reference broadcast scheme (RBS) proposed in [Elson and Romer 2002] maintains information relating the phase and frequency of each pair of clocks in the neighborhood of a node. While RBS achieves a precision of about 1 μs, the message overhead in maintaining the neighborhood information is high and may not be energy-efficient in large systems. We believe that fine-grained clock synchronization achieved by costly periodic beacon exchanges may not be suitable for the energy-constrained surveillance system. Therefore, we modified the FTSP time synchronization protocol [Maroti et al. 2004] to synchronize the motes only during the initialization phase, using a synchronization beacon broadcast by the base station at the beginning of each initialization cycle. Since the underlying MAC layer provided by TinyOS does not guarantee reliable delivery, the base station retransmits the synchronization beacon multiple times. The synchronization beacons are propagated across the network through limited flooding with timestamp values reassigned at intermediate motes immediately prior to the transmission of the timestamp. This eliminates the uncertainty in MAC contention delay. Receivers take the timestamp from the beacon plus a fixed hardware delay as their own local time. The timer drift that accumulates over time is rectified by a new system cycle (i.e., a repeated initialization phase). The frequency of re-initialization is a configurable parameter, which can be calculated based on the rate of clock drift and the desired accuracy of time synchronization. As for the current VigilNet system, the accuracy of tens of milliseconds is sufficient, which leads to about once per day synchronization.

6. EVALUATION OF REAL SYSTEM PERFORMANCE

In the evaluation, we validate the analytical results as well as provide more insights into the timing issues from the real system and simulation perspectives.

6.1 Experimental Settings

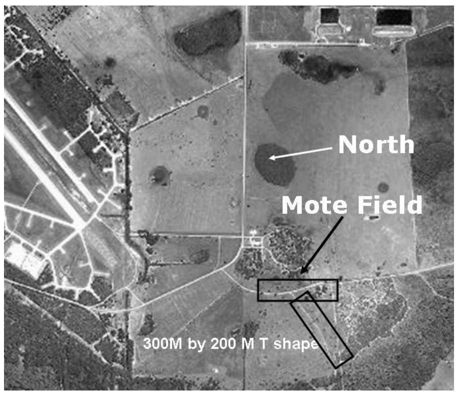


Fig. 12. Deployment Site

As a real-time online tracking system, VigilNet is designed to complete detection, classification and velocity estimation within 4 seconds. The field test was done on a T-shape dirt road in Florida as shown in Figure 12 from the aerial view. We deployed 200 XSM motes which are equipped with CC1000 radio, magnetic, acoustic, photo, temperature and passive infrared sensors (PIR). Along the road, nodes were randomly placed roughly 10 meters apart, covering one 300-meter road and one 200-meter road. Through a certain localization [Stoleru et al. 2004; He et al. 2003; Stoleru et al. 2005], nodes were aware of their positions. In order to measure various kinds of delay, all nodes within VigilNet synchronized with the base within 1∼10 milliseconds using the techniques described in [Maroti et al. 2004]. The time stamps of various actions such as initial detection were sent back to the base, so that we can calculate the delay. We used a Ford Explorer that weighted about 4000 lbs. as the target.

6.2 Delay Measurements

When a car enters the surveillance area at about 10 meters per second (22 mph), a detection report is issued first, followed by classification reports. Finally, after sufficient information is gathered, velocity reports are issued. Figure 13 illustrates the cumulative distribution of different delays. The communication delay (leftmost curve) is much smaller compared with other delays. About 80% of detections are done within 2 seconds. Over 80% of the classification and velocity estimations are made within 4 seconds. The empirical results from most runs are consistent with our analysis in Section 4 and the simulation results in Section 7.

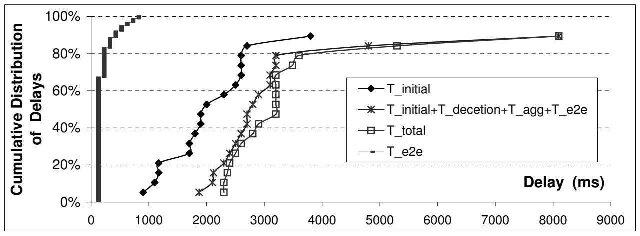


Fig. 13. Various Delays Measurements from Field Test

We emphasize here that field experiments indicate that VigilNet meets its real-time requirement and our real-time analysis can approach the reality with a reasonable precision, despite the amount of complexity within the VigilNet (30 protocols integrated). On the other hand, we acknowledge that due to various physical constraints, field experiments can only exploit a very limited design space and obtain a limited amount of data. Therefore, to understand the real-time properties in VigilNet at scale with a much larger context, we provide a large-scale simulation in the next section.

7. LARGE-SCALE SIMULATION

Our simulator is a discrete simulator, written in C++. It emulates the tracking operations as shown in Figure 1. We distribute 10,000 nodes randomly within a 100,000 m^2 rectangle area, assuming nominal circular sensing and communication ranges. We run each experiment 30 times with different random numbers. The figures are plotted with the average value as well as the 95% confidence interval.

7.1 Experiment Setup

We note that our evaluation does not choose deadline/sub-deadline miss ratios as the major metrics, because such an approach reveals less information about the tradeoff between actual delays and other system performance parameters. Since the mean value and 95% confidence intervals of the delays are plotted in the figures, one can determine the appropriate system settings for a given deadline requirement.

In our experiments, we study several system-wide parameters that directly affect the real-time properties of VigilNet. These parameters are: 1) the target speed (TS), 2) the physical delay in detection ($T_{detection}$), 3) the sentry duty cycle (SDC), 4) the non-sentry duty cycle (NSDC), 5) the required degree of aggregation (DOA), 6) the sensing range (SR) and 7) the required number of reports for base processing (K). We match the simulations with the analysis to see how well they fit with each other.

We use the settings from the VigilNet system as the default values for these system parameters, which are listed in Table I. Unless mentioned otherwise, the default values in Table I are used in all experiments. The metrics used to measure the system performance are mainly the six types of delays discussed in Section 2, the end-to-end delay and the energy consumption per day per node.

Table I. Key System Parameters

Parameter	Definition	Default Value
TS	Target Speed	10 m/s
SDC	Sentry Duty Cycle	50%
NSDC	Non-Sentry Duty Cycle	1%
DOA	Degree of Aggregation	1%
SR	Sensing Range	10
K	Reports required by the base	1
D	Node Density	0.01 m^2

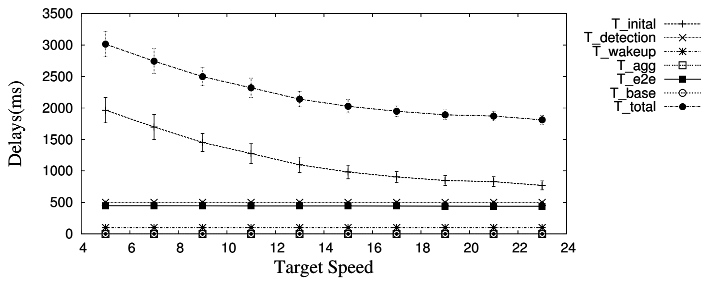


Fig. 14. Delays vs. Target Speed

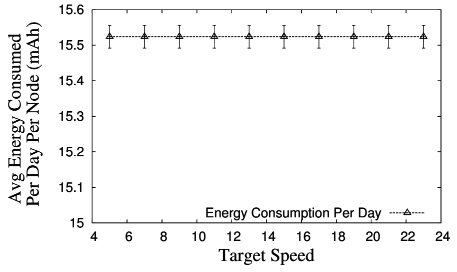


Fig. 15. Energy Consumption vs. Target Speed

7.2 Performance vs. Target Speed

The target speed determines the spatiotemporal distribution of events over a certain time period. It is crucial to understand its impacts on the tracking performance. In this experiment, we incrementally increase the target speed (TS) from 5m/s to 15m/s in steps of 1 meter. As expected from our analysis in Section 4, $T_{initial}$ and $T_{aggregation}$ decrease with the target speed, as shown in Figure 14. One interesting observation is that the descend rate of $T_{initial}$ diminishes when TS becomes larger. This is because a node needs a sufficient sensing time to ensure detection. It is possible that a quick target passes one sensor without detection, which negatively affects the $T_{initial}$. Since VigilNet deals with a rare event model, the energy consumed during the tracking is not perceptibly affected by the target speeds as shown in Figure 15.

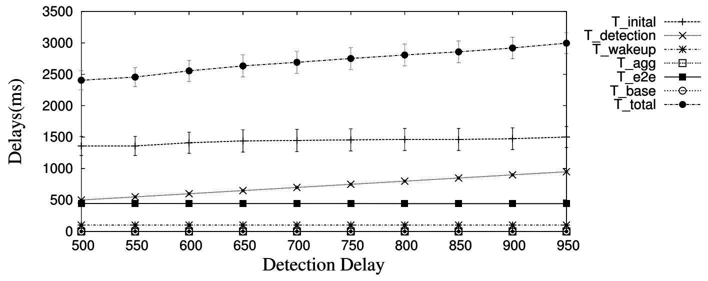


Fig. 16. Delays under Varying Detection Delay

7.3 Performance vs. Detection Delay

Different tracking systems use different sensing devices and detection algorithms, which have various detection delays $T_{detection}$. In this experiment, we increase the delay in the detection algorithm $T_{detection}$ from 500 ms to 1000 ms in steps of 50 ms. It is interesting to observe in Figure 16 that at a speed of 10m/s, the detection delay has a small impact on the initial delay, however it contributes most significantly to the overall increase of the total tracking delay. Again, since the detection time is relatively small, this system parameter does not noticeably affect the overall energy consumption, as shown in Figure 17.

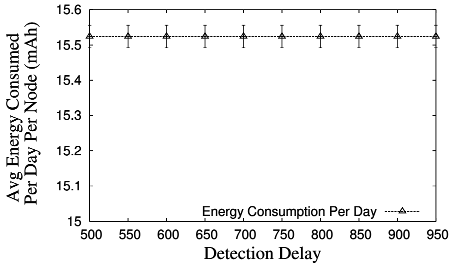


Fig. 17. Energy Consumption vs. Detection Delay

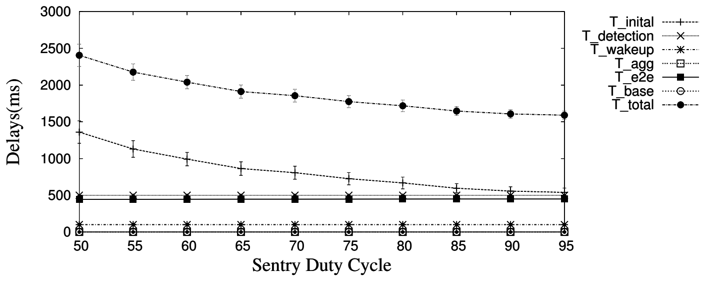


Fig. 18. Delays vs. Sentry Duty Cycle

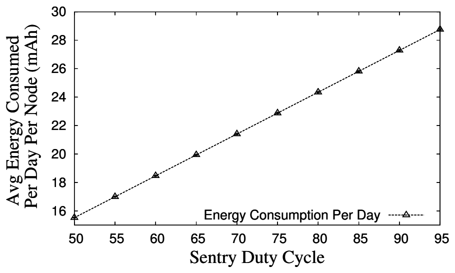


Fig. 19. Energy Consumption vs. Sentry Duty Cycle

7.4 Performance vs. Sentry Duty Cycle

From the analytical results in Section 4, we obtain an analytical delay curve between $T_{initial}$ and SDC in Figure 4. In this experiment, we obtain a set of other curves (Figure 18) through the simulation. By comparing these two results, we conclude that they are consistent with each other. For example, at a default 50% duty cycle, $T_{initial}$ obtained from the analysis in Figure 4 is 1600ms, while $T_{initial}$ obtained from the simulation (Figure 18) is 1360ms (Note that our analysis is relatively conservative). In addition, Figure 19 reveals that the energy consumption escalates linearly with the SDC, which indicates that an efficient sentry scheduling algorithm is beneficial.

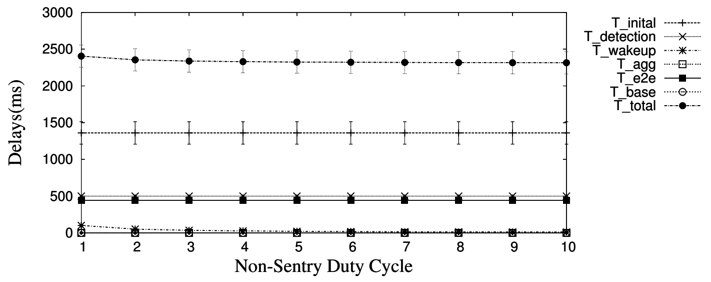


Fig. 20. Delays vs. Non-Sentry Duty Cycle

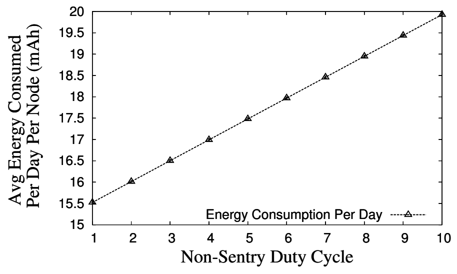


Fig. 21. Energy Consumption vs. Non-Sentry Duty Cycle

7.5 Performance vs. Non-sentry Duty cycle

Here, we evaluate the impact of the wake-up operation on the delay and energy consumption. First, the simulation results confirm that the average wake-up delay is approximately half of the toggle period as predicted in Section 4.3. Since the wake-up delay T_{wakeup} is one order of magnitude smaller than other delays such as $T_{initial}$, a slight decrease in the wake-up delay, shown in Figure 20, does not noticeably impact the overall delay. However, interestingly a slight increase of the Non-Sentry Duty Cycle leads to a significant increase in energy consumption as shown in Figure 21. This is because the non-sentry nodes are by far the majority, so a duty-cycle increase for the non-sentry nodes leads to a quick increase in the total energy. This result indicates that it is beneficial to increase the wake-up delay, when possible, in exchange for the energy saving.

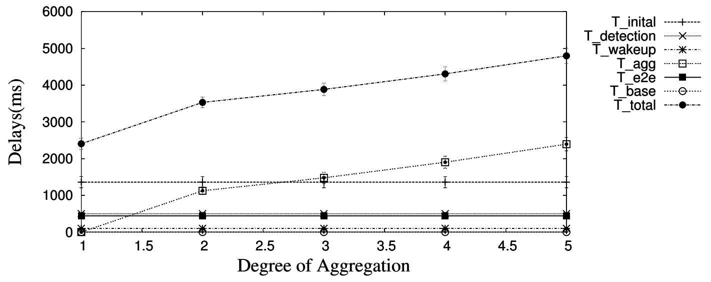


Fig. 22. Delays vs. DOA

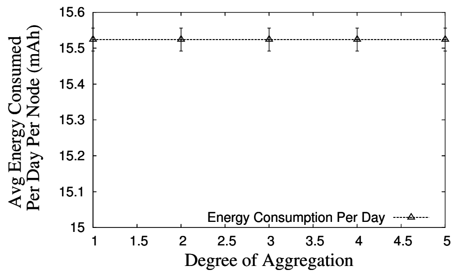


Fig. 23. Energy Consumption vs. DOA

7.6 Performance vs. DOA

In-network processing through data aggregation can reduce the amount of data transmitted over the network and can increase the confidence in target detection. However, to accumulate enough report, it inevitably introduces a certain delay. This experiment studies the effects of data aggregation. We gradually increase the DOA threshold for a leader to report to the base. Since the DOA value only affects the tracking phase, which has a small energy consumption, DOA's impact on the energy consumption is not noticeable. On the other hand, with a larger DOA value, it takes more time for a leader to collect the member reports. For example as shown in Figure 22, it takes as long as 2.39 seconds to achieve DOA value of 5. We note that this simulation result is again consistent with the analytical results shown in Figure 9, which has an estimated delay of 2.5 seconds.

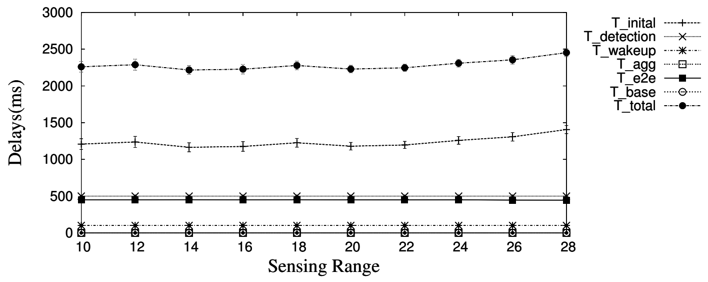


Fig. 24. Delays vs. Sensing Range

7.7 Performance vs. Sensing Range

To accommodate various requirements in detection and classification, different tracking systems use sensors with different ranges. Figure 24 and Figure 25 investigate the impact of sensing range to the tracking performance and energy consumption. With a large sensing range, a smaller number of sentries is required. Therefore, the total energy consumption decreases quickly. For example in Figure 25, the energy reduces by 75% when the sensing range increases from 10m to 28m. It is interesting to see that the initial delay $T_{initial}$ actually slightly increases. This is because the number of sentry nodes reduces while the coverage per sensor

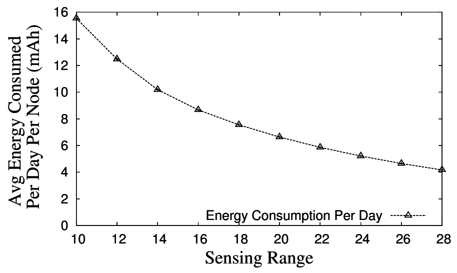


Fig. 25. Energy Consumption vs. Sensing Range

increases, the total coverage by all sentry nodes remains the same. We can derive from Equation 3 that the expected $T_{initial}$ is higher when the sensing range is smaller, given the same coverage in both cases. This analytic result is confirmed by the simulation results shown in Figure 24. Due to the space constraints, we omit the detailed derivation here.

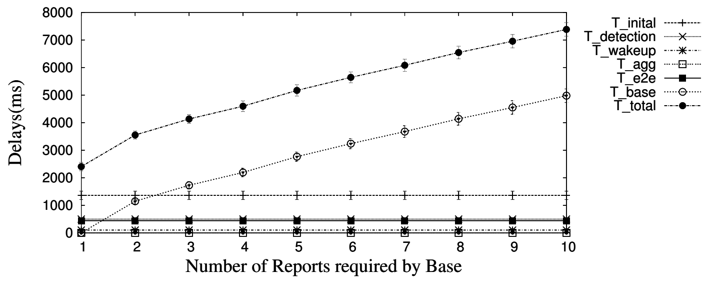


Fig. 26. Delays vs. Num of Required Reports

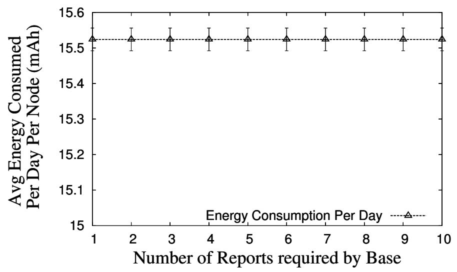


Fig. 27. Energy Consumption vs. Num of Required Report

7.8 Performance vs. Number of Reports

To improve the estimation of target velocity and to classify targets with a high confidence, a base node normally needs to accumulate a certain number of spatiotemporally related reports from the same logic tracking group. This experiment investigates the impact of the number of reports required by a base on the tracking delays. Obviously, this only affects T_{base}. Figure 26 shows that T_{base} approximately increases linearly with the number of reports, which is expected from our analytical results in Section 4.6. Since the operation is done at the base, there is no energy impact to the sensor network, as shown in Figure 27.

8. RELATED WORK

Real-time protocols have been designed at different layers to guarantee the effectiveness of the interactions between wireless sensor networks and the physical world. At the MAC layer, RAP [Lu et al. 2002] uses a novel velocity monotonic scheduling to prioritize the real-time traffic and enforce such prioritization through a differentiated MAC Layer. Woo and Culler [Woo and Culler 2001] propose an adaptive rate control scheme to achieve fairness among the nodes with different distances to a base station. Li [Li et al. 2005] proposes a SLF message scheduling algorithm to exploit spatial channel reuse, so that deadline misses can be reduced. Carley [Carley et al. 2003] designs a periodic message scheduler to provide a contention-free predicable medium access control. At the network layer, He [He et al. 2003]et al. support a soft real-time communication service with a desired delivery speed across the sensor network, using feedback-based adaptation algorithms that enforce per-hop speed in face of unpredictable traffic. Felemban [Felemban et al. 2005] presents a novel packet delivery mechanism, called multi-path and multi-speed routing protocol, for probabilistic QoS guarantee in wireless sensor networks. At the aggregation layer, Vasudevan [Vasudevan et al. 2003] proposes an application-specific compression for time delay estimation in sensor networks, and He [He et al. 2004] adaptively performs application independent data aggregation in a time sensitive manner. The real-time solutions at the application is highly diversified. Huang [Huang et al. 2003] et al. propose the Mobicast protocol to provide just-in-time information dissemination to nodes in a mobile delivery zone. Given the complete knowledge of traffic pattern, Somasundara [Somasundara et al. 2004] proposes a mobile agent scheduling algorithm to collect the buffered sensor data, before the buffer overflow occurs at the sensor nodes. Nam [Nam et al. 2005] proposes time-parameterized sensing task model for real-time tracking. Yang [Yang and Vaidya 2004] proposes a wakeup scheme that assists balancing energy saving and end-to-end delay. The Lightning protocol [Wang et al. 2004] localizes the acoustic source with a bounded delay regardless of the node density.

Besides the real-time protocol design, several researchers have focused on the time analysis for sensor networks. Lu [Lu et al. 2005] studies how to minimize the communication latency given that each sensor has a duty cycling requirement of being awake for only $\frac{1}{k}$ time slots on average. In [Mohan et al. 2004], Mohan et al. provides a cycle-accurate WCET analysis tool for the applications running on the Atmega Processor Family. Abdelzaher [Abdelzaher et al. 2004] derives a real-time capacity bound for multi-hop wireless sensor networks. It is a sufficient schedulability condition for a class of fixed priority packet scheduling algorithms. Using this bound, one can determine whether a certain traffic pattern can meet its real-time requirement beforehand.

With advances in the sensor techniques, several large-scale sensor systems have been built recently. The GDI Project [Szewczyk et al. 2004] provides an environmental monitoring system to record animal behaviors for a long period of time. The shooter localization system [Simon et al. 2004] collects the time-stamps of the acoustic detection from different nodes within the network to localize the positions of the snipers. These systems mention some timing issues, however they do not treat real-time as a major concern. Our previous publications on VigilNet [He et al. 2004; He et al. 2006] focus on the middleware services and overarching system integration. To the best of our knowledge, this work is the first to analyze the real-time performance and its tradeoffs in a real-world large-scale wireless sensor system.

9. CONCLUSION

In this paper, we demonstrate the feasibility to design a complex real-time sensor network, using the deadline partition method, which guarantees an end-to-end tracking deadline by satisfying a set of sub-deadlines. We also analytically identify the tradeoffs among system properties while meeting the real-time requirements. We validate our design and analysis through both a large-scale simulation with 10,000 nodes as well as a field test with 200 XSM nodes. We contribute a set of tradeoffs that are useful for the future development of real-time sensor systems. Given real-time constraints, a system designer can make guided engineering judgments on the system parameters. Here we just name a few. First, to guarantee the same sub-deadline, a higher node density is desired in the slow-target case, however a slower duty cycle can be tolerated without jeopardizing the detection. Second, it is beneficial to increase the wake-up delay, when possible, in exchange for the energy saving. Third, fast detection algorithms are essential. Fourth, a low network density increases the group aggregation delay, which indirectly reduces the detection confidence. Fifth, theoretically, honeycomb is the optimal base placement strategy to meet the communication sub-deadline.

Finally, we acknowledge that although it is amenable to provide the worst-case real-time analysis for a certain protocol such as the wake-up protocol in Section 4.3, however, due to the dynamic and unpredictable nature of the sensor networks, it is a long-term research goal for us to achieve precise worst-case real-time analysis across the whole system.

10. ACKNOWLEDGEMENTS

This work was supported in part by the DAPRPA IXO offices under the NEST project (grant number F336615-01-C-1905), the MURI award N00014-01-1-0576 from ONR, NSF grant CCR-0329609, CCR-0325197 and CNS-0435060. The authors specially thank the NEST program manager Dr. Vijay Raghavan for his valuable contributions.

REFERENCES

Abdelzaher, T. F., Prabh, S., and Kiran, R. 2004. On Real-Time Capacity Limits of Multihop Wireless Sensor Networks. In *IEEE RTSS*.

Arora, A., Dutta, P., Bapat, S., Kulathumani, V., Zhang, H., Naik, V., Mittal, V., Cao, H., Demirbas, M., Gouda, M., Choi, Y., Herman, T., Kulkarni, S., Arumugam, U., Nesterenko, M., Vora, A., and Miyashita, M. 2004. A Wireless Sensor Network for Target Detection, Classification, and Tracking. *Computer Networks (Elsevier)*.

Arora, A. and et al. 2005. Exscal: Elements of an extrem scale wireless sensor network. In *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2005)*.

Cao, Q., Yan, T., Abdelzaher, T., and Stankovic, J. 2005. Analysis of target detection performance for wireless sensor networks. In *DCOSS'05*.

Carley, T. W., Ba, M. A., Barua, R., and Stewart, D. B. 2003. Contention-free periodic message scheduler medium access control in wireless sensor/actuator networks. In *IEEE RTSS*.

CrossBow 2003. *Mica2 data sheet*. CrossBow. Available at `http://www.xbow.com`.

Dutta, P., Grimmer, M., Arora, A., Biby, S., and Culler, D. 2005. Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events. In *IPSN'05*.

Elson, J. and Romer, K. 2002. Wireless Sensor Networks: A New Regime for Time Synchronization. In *Proc. of the Workshop on Hot Topics in Networks (HotNets)*.

Felemban, E., Lee, C., Ekici, E., Boder, R., and Vural, S. 2005. Probabilistic qos guarantee in reliablility and timeliness domains in wireless senosr networks. In *IEEE INFOCOM 2005*.

Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E., and Culler, D. 2000. The nesC Language: A Holistic Approach to Networked Embedded Systems. In *Proceedings of Programming Language Design and Implementation (PLDI) 2003*.

Gu, L., Jia, D., Vicaire, P., Yan, T., Luo, L., A.Tirumala, Cao, Q., T. He, J. A. S., T.Abdelzaher, and Krogh., B. 2005. Lightweight Detection and Classification for Wireless Sensor Networks in Realistic Environments. In *SenSys'05*.

Gu, L. and Stankovic, J. A. 2004. Radio-Triggered Wake-Up Capability for Sensor Networks. In *Proceedings of RTAS*.

He, T., Blum, B. M., Stankovic, J. A., and Abdelzaher, T. F. 2004. AIDA: Adaptive Application Independent Data Aggregation in Wireless Sensor Networks. *ACM Transactions on Embedded Computing Systems, Special issue on Dynamically Adaptable Embedded Systems*.

He, T., Huang, C., Blum, B. M., Stankovic, J. A., and Abdelzaher, T. 2003. Range-Free Localization Schemes in Large-Scale Sensor Networks. In *MOBICOM'03*.

He, T., Krishnamurthy, S., Luo, L., Yan, T., Stoleru, R., Zhou, G., Cao, Q., Vicaire, P., Stankovic, J. A., Abdelzaher, T. F., Hui, J., and Krogh, B. 2006. VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance. *ACM Transaction on Sensor Networks*.

He, T., Krishnamurthy, S., Stankovic, J. A., Abdelzaher, T., Luo, L., Stoleru, R., Yan, T., Gu, L., Hui, J., and Krogh, B. 2004. An Energy-Efficient Surveillance System Using Wireless Sensor Networks. In *MobiSys'04*.

He, T., Stankovic, J., Lu, C., and Abdelzaher, T. 2003. SPEED: A Stateless Protocol for Real-Time Communication in Ad Hoc Sensor Networks. In *ICDCS'03*.

He, T., Vicaire, P., Yan, T., Cao, Q., Zhou, G., Gu, L., Luo, L., Stoleru, R., Stankovic, J. A., , and Abdelzaher, T. 2006. Achieving Long-Term Surveillance in VigilNet. In *IEEE Infocom*.

Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D. E., and Pister, K. S. J. 2000. System Architecture Directions for Networked Sensors. In *Proc. of Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 93–104.

Huang, Q., Lu, C., and Roman, G.-C. 2003. Spatiotemporal Multicast in Sensor Networks. In *SenSys 2003*.

IEEE. IEEE Wireless Medium Access Control(MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs).

Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L., and Rubenstein, D. 2002. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In *Proc. of ASPLOS-X*.

Li, H., Shenoy, P., and Ramamritham, K. 2005. Scheduling Messages with Deadlines in Multi-hop Real-time Sensor Networks. In *RTAS'05*.

L.Kleinrock and J.Slivester. 1978. Optimum transmission radii for packet radio networks or why six is a magic number. In *national Telecomm conference*.

Lu, C., Blum, B. M., Abdelzaher, T. F., Stankovic, J. A., and He, T. 2002. Rap: A real-time communication architecture for large-scale wireless sensor networks. In *IEEE RTAS*.

Lu, G., Sadagopan, N., Krishnamachari, B., and Goel, A. 2005. Delay efficient sleep scheduling in wireless sensor networks. In *IEEE INFOCOM 2005*.

Luo, L., He, T., Abdelzaher, T., and Stankovic, J. 2005. Design and comparison of lightweight group management strategies in envirosuite. In *DCOSS '05: International Conference on Distributed Computing in Sensor Networks*.

Maroti, M., Kusy, B., Simon, G., and Ledeczi, A. 2004. The Flooding Time Synchronization Protocol. In *SenSys'04*. 39–49.

Mohan, S., Mueller, F., Whalley, D., and Healy, C. 2004. Timing Analysis for Sensor Network Nodes of the Atmega Processor Family. In *IEEE RTSS*.

Nam, M.-Y., Lee, C.-G., Kim, K., and Caccamo, M. 2005. Time-parameterized sensing task model for real-time tracking. In *IEEE RTSS 2005*.

Okabe, A., Boots, B., Sugihara, K., and Chiu, S. N. 2000. *Spatial Tessellations:Concepts and Applications of Voronoi Diagrams*. Wiley.

Polastre, J. and Culler, D. 2004. Versatile Low Power Media Access for Wireless Sensor Networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*.

Simon, G., Maroti, M., Ledeczi, A., Balogh, G., Kusy, B., Nadas, A., Pap, G., Sallai, J., and Frampton, K. 2004. Sensor Network-Based Countersniper System. In *SenSys'04*.

Somasundara, A. A., Ramamoorthy, A., and Srivastava, M. B. 2004. Mobile Element Scheduling for Efficient Data Collection in Wireless Sensor Networks with Dynamic Deadlines. In *IEEE RTSS*.

Stoleru, R., He, T., and Stankovic, J. A. 2004. Walking GPS: A Practical Solution for Localization in Manually Deployed Wireless Sensor Networks. In *EmNetS-I*.

Stoleru, R., He, T., Stankovic, J. A., and Luebke, D. 2005. High-Accuarcy, Low-Cost Localization System for Wireless Sensor Networks. In *Third ACM Conference on Embedded Networked Sensor Systems (SenSys 2005)*.

Szewczyk, R., Mainwaring, A., Anderson, J., and Culler, D. 2004. An Analysis of a Large Scale Habit Monitoring Application. In *SenSys'04*.

Vasudevan, L., Ortega, A., and Mitra, U. 2003. Application-Specific Compression for Time Delay Estimation in Sensor Netowrks. In *ACM SenSys 2003*.

Wang, Q., Zheng, R., Tirumal, A., Liu, X., and Sha, L. 2004. Lightning: A Fast and Lightweight Acoustic Localization Protocol Using Low-End Wireless Micro-Sensors. In *IEEE RTSS*.

Wellenhoff, B. H., Lichtenegger, H., and Collins, J. 1997. *Global Positions System: Theory and Practice,Fourth Edition*. Springer Verlag.

Williams, R. 1979. Circle Coverings. In *The Geometrical Foundation of Natural Structure: A Source Book of Design, pp. 51-52*.

Woo, A. and Culler, D. 2001. A Transmission Control Scheme for Media Access in Sensor Networks. In *Proc. of Mobile Computing and Networking (Mobicom)*.

Xu, N., Rangwala, S., Chintalapudi, K. K., Ganesan, D., Broad, A., Govindan, R., and Estrin, D. 2004. A Wireless Sensor Network for Structural Monitoring. In *SenSys 2004*.

Yang, X. and Vaidya, N. H. 2004. Awakeup scheme for sensor networks: Achieving balance between energy saving and end-to-end delay. In *IEEE RTAS 2004*.

EnviroSuite: An Environmentally Immersive Programming Framework for Sensor Networks

LIQIAN LUO and TAREK F. ABDELZAHER
University of Illinois at Urbana-Champaign
TIAN HE
University of Minnesota
and
JOHN A. STANKOVIC
University of Virginia

Sensor networks open a new frontier for embedded distributed computing. Paradigms for sensor network programming in the large have been identified as a significant challenge towards developing large-scale applications. Classical programming languages are too low-level. This paper presents the design, implementation, and evaluation of *EnviroSuite*, a programming framework that introduces a new paradigm, called *environmentally immersive programming*, to abstract distributed interactions with the environment. Environmentally immersive programming refers to an object-based programming model in which individual objects represent physical elements in the external environment. It allows the programmer to think directly in terms of environmental abstractions. EnviroSuite provides language primitives for environmentally immersive programming that map transparently into a support library of distributed algorithms for tracking and environmental monitoring. We show how nesC code of realistic applications is significantly simplified using EnviroSuite, and demonstrate the resulting system performance on Mica2 and XSM platforms.

Categories and Subject Descriptors: C.2.4 [**Computer-communication Networks**]: Distributed Systems—*Distributed Applications*; D.2.11 [**Software Engineering**]: Software Architectures—*Domain-specific Architectures*; D.3.2 [**Software Engineering**]: Language Classifications—*Specialized Application Languages*

General Terms: Design, Experimentation, Languages, Performance

Additional Key Words and Phrases: Abstractions, embedded systems, middleware, programming models, tracking, sensor networks

1. INTRODUCTION

This paper presents *EnviroSuite*, the first sensor network programming framework for environmentally immersive programming. The need to facilitate software development for sensor networks motivates new high-level abstractions for programming-in-the-large. These abstractions must hide the details of distributed monitoring and

This work is supported in part by Microsoft and NSF grants EHS-0208769, ITR-0205327, and EHS-0509233.

tracking algorithms, capture the unique properties of these networks such as their distributed interactions with a physical environment, and address the issue of scale.

Traditional programming languages such as Java and C, as well as their sensor network adaptations, such as nesC [Gay et al. 2003] and its extension galsC [Cheong et al. 2003; Cheong and Liu 2005], are too low-level. Their basic computation, communication and actuation unit is typically the sensor node. Programmers must think in terms of single node activities and explicitly encode interactions among nodes. For example, programmers are exposed to reading sensing data from appropriate sensor devices, aggregating data pertaining to the same external stimulus, deciding where to send it, and communicating with actuators if needed. If the monitored activity moves in the environment, programmers are responsible for spatial and temporal correlation of measurements obtained about the activity across a changing set of sensor nodes, and associating that data with event progress.

A more desirable approach would be for the programmers to encode only overall network behavior, leaving it to the underlying system to decompose such behavior into node-level algorithms. Examples of higher-level abstractions that address this concern include logical-node-based primitives [Gummadi et al. 2005], token-based languages [Newton et al. 2005], database-centric abstractions [Madden et al. 2003; Yao and Gehrke 2002; Madden et al. 2005], event-based systems [Li et al. 2004], group-based primitives [Blum et al. 2003; Whitehouse et al. 2004; Welsh and Mainland 2004; Liu et al. 2003], state-centric approaches [Liu et al. 2003] and virtual machines [Levis and Culler 2002; Boulis et al. 2003]. These paradigms offer logical nodes, tokens, queries, events, sensor node groups, and logical state, respectively, as the underlying abstractions with which the programmer operates.

EnviroSuite is an object-based programming system. Its abstractions revolve directly around elements of the environment as opposed to sensor network constructs such as regions, neighborhoods, or sensor groups. The existence of the sensor network is thus made more transparent. EnviroSuite is different from other object-based systems in that its objects are representations of elements in the external environment. Dynamic object instances are created automatically by the run-time system when the corresponding external elements are detected and are destroyed when these elements leave the network. A unique mapping between object instances and the corresponding environmental elements is maintained by the system. Object instances float across the network following (geographically) the elements they represent. The execution of object code at the location of the corresponding physical element is ideal for sensing and actuation tasks. Objects encapsulate the aggregate state of the elements they represent, making such state available to their methods. These objects (as opposed to the individual nodes) are therefore the units that encapsulate program data, computation, communication, sensing and actuation. Classical objects (that do not represent any environmental elements) are also supported. We call the above model, *environmentally immersive programming* (EIP).

This paper presents the first comprehensive design and implementation of an environmentally immersive programming framework. EnviroSuite abstractions are supported by an underlying library called *EIPLib*, which is implemented in nesC on TinyOS [Hill et al. 2000], an operating system designed specifically for sensor networks. We evaluate EnviroSuite and several applications written in it on TOSSIM [Levis et al. 2003] as well as on a mote-based sensor network. TOSSIM is an emulator that runs the actual nesC service code, emulating on a PC the behavior of programs on the Berkeley motes [U. C. Berkeley 2005]. The framework extends a previous tracking middleware service by the authors, called EnviroTrack [Abdelzaher et al. 2004], which introduced a network address space where representations of environmental entities are the addressable objects.

Finally, two remarks are in order on what EnviroSuite is *not*. First, EnviroSuite is not a replacement to other emerging programming paradigms such as group-based primitives, database-centric abstractions, event-based systems, and virtual machines. This paper does not argue for a single approach to the exclusion of others. The most appropriate abstractions are often a personal choice that depends on subjective programmer preferences as well as application specifics. Ultimately, it is the availability of multiple programming alternatives that induces more software development. EnviroSuite is therefore presented and evaluated for its own merits, and not as a substitution for other high-level paradigms.

Second, EnviroSuite is not a programming language in itself. EnviroSuite is a framework that extends other programming languages with environmentally immersive programming primitives. This extension takes two different forms. First, the programmer is allowed to define and use variables that summarize elements of a potentially distributed environmental state (such as the average temperature of a region or the current location of a moving target). Second, the programmer may define code that is geographically distributed and associate the time and place of its execution with the occurrence of certain environmental events. Both the aggregate variables and distributed code are encapsulated within simple objects. As with other distributed computing paradigms, remote communication is allowed between objects. The purpose is to abstract the distributed aspects of environmental interactions and computation.

With distribution hidden from the programmer, logical computation can be performed using the native programming language. The current implementation of EnviroSuite extends nesC. However, there is nothing in its design and general abstractions that is nesC specific. The implementation can be easily re-targeted to support other programming languages. nesC was chosen due to its wide popularity in the sensor network community and due to the availability of a compiler for the current mote hardware.

The remainder of the paper is organized as follows. Section 2 introduces the overall architecture of the EnviroSuite framework. Section 3 provides a detailed description of the exported abstractions. Section 4 presents the design of the essential algorithms underlying these abstractions. Section 5 highlights the implementation details. Section 6 presents and analyzes performance evaluation results. Section 7 discusses related work. Section 8 concludes the paper.

2. SYSTEM ARCHITECTURE

EnviroSuite lets the programmer think in terms of objects in the external environment that are relevant to the application. An environmental object may refer to a localized entity (such as a moving vehicle) or a distributed region of the en-

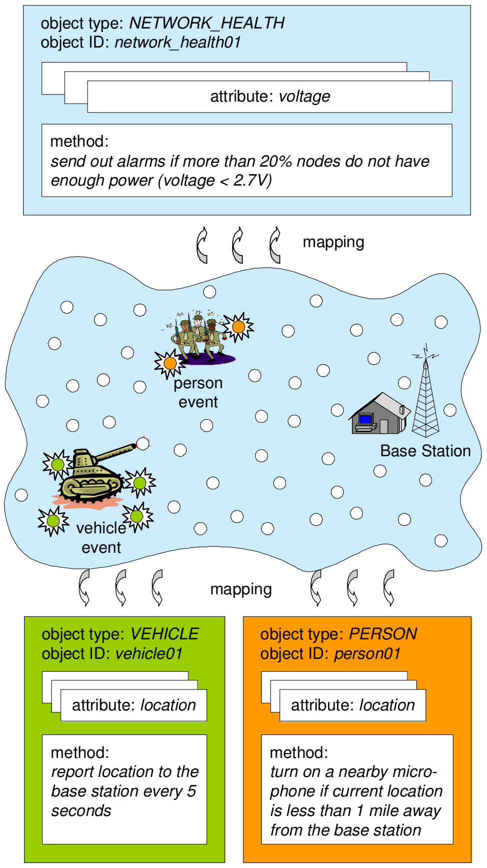

object type: *NETWORK_HEALTH*
object ID: *network_health01*
attribute: *voltage*
method:
send out alarms if more than 20% nodes do not have enough power (voltage < 2.7V)
mapping
person event
Base Station
vehicle event
mapping
object type: *VEHICLE*
object ID: *vehicle01*
attribute: *location*
method:
report location to the base station every 5 seconds
object type: *PERSON*
object ID: *person01*
attribute: *location*
method:
turn on a nearby micro-phone if current location is less than 1 mile away from the base station

Fig. 1. One-to-one mapping between physical events and event objects

vironment (e.g., a geographic area or an area specified by some sensory signature such as high temperature). Typically, the system must keep some state or other information about each object. This state is maintained in variables encapsulated within the object that can be accessed using object methods. Hence, each object is given by (i) a sensory or geographic signature that defines its boundaries or location, (ii) a set of data variables to be collected in its vicinity, and (iii) a set of methods that can be performed in its context. The fact that the obtainment of values stored in the encapsulated variables and the execution of local methods may need distributed computation (such as data aggregation) is hidden from the programmer. The programmer may also define regular objects that are not linked to objects in the environment. We call them *function objects*. Environmental objects and function objects seamlessly coexist in the programmers' world and can invoke each other's methods using a remote object invocation mechanism. An example of such mapping is depicted in Figure 1.

The example in Figure 1 represents a surveillance application that monitors vehicle and person movement in a hostile territory (e.g., behind enemy lines). Each tracked vehicle or person is mapped into a dynamically instantiated object with

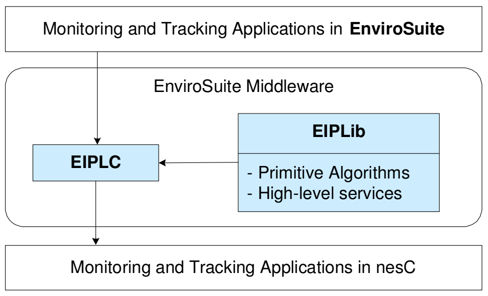

Monitoring and Tracking Applications in **EnviroSuite**
EnviroSuite Middleware
EIPLib
EIPLC
- Primitive Algorithms
- High-level services
Monitoring and Tracking Applications in nesC

Fig. 2. Relation of EnviroSuite, EIPLC & EIPLib

a unique label, denoted by an object ID in EnviroSuite. Desired event attributes such as location can be returned for the object. This application also periodically monitors the health of the network by collecting information on nodes that are alive and their remaining power. The network is thus mapped into an object that maintains aggregate health statistics. Computation, communication and actuation can be logically attached to these objects. Examples include reporting vehicle location by vehicle objects, turning on microphones in their vicinity for tracking purposes, or sending out alarms if system health fails to meet an acceptable threshold.

These primitives are supported by the environmentally immersive programming library (*EIPLib*), which provides a series of algorithms containing the detailed implementations (such as sensor data processing, object maintenance, and inter-object communication) and some other higher level services. A compiler (*EIPLC*) is introduced to translate EnviroSuite applications into nesC. The relation among EnviroSuite, EIPLC and EIPLib is depicted in Figure 2.

Programmers design and implement environmental monitoring and tracking applications using a combination of EnviroSuite and nesC. Taking such implementations as input, the compiler (EIPLC) configures and restructures services and protocols in EIPLib to automatically produce as output the corresponding implementations in the nesC language. The resulting code can be compiled on TinyOS and uploaded to the motes. In the following sections, we describe in more details the abstractions of EnviroSuite, the services and protocols provided in EIPLib to support these abstractions, and the translation of these abstractions carried out by EIPLC.

3. ENVIRONMENTALLY IMMERSIVE PROGRAMMING SYNTAX

When a programmer develops a monitoring application using EnviroSuite, the programmer creates a virtual world with a set of logical objects, which attempts to reflect the real world with a set of physical objects. Each object is defined by a sensory signature such that contiguous groups of nodes that satisfy that signature will be given a unique object ID. These object IDs constitute a global name space available to the programmer. Various data operations can then be performed on different locales defined by the corresponding object IDs. These operations are typically coded as methods encapsulated in the corresponding objects. Observe that EnviroSuite is only concerned with (i) grouping together nodes that satisfy programmer-defined sensory signatures, (ii) giving global names to those groups,

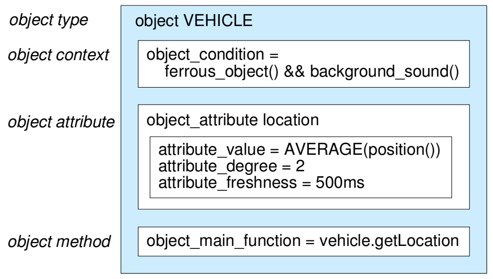

object type
object VEHICLE
object context
object_condition =
ferrous_object() && background_sound()
object attribute
object_attribute location
attribute_value = AVERAGE(position())
attribute_degree = 2
attribute_freshness = 500ms
object method
object_main_function = vehicle.getLocation

Fig. 3. Object declaration of object VEHICLE

(iii) executing programmer-defined data operations within each named group, and (iv) storing programmer-defined group state in variables encapsulated within the named objects. The correspondence between the groups and meaningful aspects of the environment is the programmer's responsibility. For example, there is inherent uncertainty regarding whether or not a motion and magnetic signature defined by the programmer truly signals the presence of a vehicle. EnviroSuite is concerned only with tracking the defined signature. The uncertainty in the interpretation of the signature must be handled at a higher-level.

Syntactically, an EnviroSuite program consists of a list of object declarations such as the one shown in Figure 3. Each declaration specifies a user-defined object *type*, an object *condition* statement, declaration of object *attributes*, and the object *methods*.

The object condition statement creates a mapping between the software object and the corresponding environmental element. For example, it can specify the sensory signature of an external tracked entity, or a geographic area defining a physical region. An object is created for each contiguous region where the object condition is true. A contiguous region is one that is not partitioned. In other words, there exists a path between any two nodes in the region that has no intermediate hops outside the region. We call this region the object *context*. A null object condition specifies that this object is not a representation of an environmental element (e.g., a pure computational object), which is called a function object, as mentioned above.

Specifications of object contexts are followed by declarations of encapsulated data to store the state of the object. Such data typically refers to aggregates of sensory measurements or node attributes over the object context. They can be thought of as query results over the context. The declaration specifies the method of aggregation together with confidence and freshness parameters. Finally, as in traditional object-based systems, an object *main* function specifies the name of a default constructor method to be automatically executed when the object is created. Other methods can be defined to be executed when called. Object methods can access the attributes of their encapsulating object and perform remote method invocations on other objects.

Objects are instantiated either statically or dynamically. The former is useful to represent fixed environmental elements such as topological features of the terrain. The latter is useful, for example, to represent dynamically arriving targets in the

Table I. Keywords for basic EnviroSuite object declaration

Object Context	`object_condition`

Object Attributes	`object_attribute`
	`attribute_value`
	`attribute_degree`
	`attribute_freshness`

Object Methods	`object_main_function` `object_function`

environment. As described later in this paper, special care is taken to ensure unique representation (i.e., that a single object is instantiated to refer to a single target, even though the target causes multiple sensor hits).

EnviroSuite keywords for basic object declarations are listed in table I. More detailed discussion on EnviroSuite object contexts, attributes, and methods is presented in the following subsections respectively, using the object declaration example depicted in Figure 3.

3.1 Defining the Object Context

In EnviroSuite, the object condition statement defines the object context, which is the continuous region where the object condition is true. EnviroSuite includes a library of sensor data processing algorithms (called the *condition library*) designed by domain experts for purposes of defining object contexts. These algorithms return (possibly) filtered or otherwise processed sensor outputs (e.g., `temperature()`), or identify specific boolean environmental conditions (e.g., `ferrous_object()` or `vehicle_sound()`), or return node attributes (e.g., `position()` or `voltage()`). A boolean expression of such conditions can then define the region of object context. We call it the object condition statement. For example, the following declaration defines the condition that represents the potential presence of a vehicle:

```
object_condition = ferrous_object() && vehicle_sound();
```

In this example, `ferrous_object()` is a function that returns true when the magnetometer output indicates a significant disturbance to the earth magnetic field (consistent with the passage of a large ferrous object), and `vehicle_sound()` indicates microphone output of energy and pitch consistent with the sound of a passing vehicle. Implementation of such functions is described in [Gu et al. 2005].

The idea is to compile a library of such conditions to abstract the specifics of sensor processing away from the programmer in much the same way device drivers abstract the details of device I/O away from application code. The separation between high-level application code and low-level sensor processing comes at the cost of increased condition library size, since many different algorithms need to be written to identify a sufficient range of useful conditions for each sensor type. This is not unlike the proliferation of device drivers (one for each version of every possible device) in contemporary operating system installations. The success of device drivers as a means for separating concerns leads one to believe that the condition library will considerably simplify application development in sensor networks. An object executes when and where the conditions defined in its condition statement become true.

Observe that conditions can also be parameterized. For example, the condition:

```
object_condition = altitude()>500 && temperature()<32;
```

defines the region (i.e., object context) that satisfies freezing temperatures on top of a local hill. The case of `object_condition = NULL` specifies a function object not associated with an environmental element (region or physical entity).

3.2 Defining Object Attributes

The main purpose of objects invoked in response to environmental conditions such as those mentioned above is usually to monitor attributes of environmental events, targets or regions. These attributes are measurements collected and aggregated by nodes in the object context. Specification of attributes requires specification of (i) the sensor measurements in question, and (ii) optionally, their method of aggregation. Aggregation is always performed over all nodes within the object context. The sensor measurements to be aggregated could be any environmental measurements, or node attribute measurements such as remaining battery power or node position, for which a measurement function exists in the condition library described above. A library, called the *aggregation method library*, is supplied, which lists a set of aggregation methods such as `AVERAGE`, `MAX` and `RANGE` on attributes. For example, to define an aggregate attribute, `targetLocation`, EnviroSuite programmers can simply specify the corresponding node measurement, `position()`, from the condition library, and the name of the appropriate aggregation method, say `AVERAGE`, from the aggregation method library, in an object attribute clause, such as:

```
object_attribute targetLocation {
  attribute_value = AVERAGE(position());
}
```

Within the declaration of an attribute, EnviroSuite allows the programmer to specify the minimum aggregation degree, `attribute_degree`. The aggregate attribute is valid only when it is the aggregation result from at least as many nodes as `attribute_degree`. This knob allows programmers to control the confidence in retrieved information. The feature is especially useful in reducing false alarms. Another important property of attributes is freshness. Most monitoring applications have temporal data validity constraints. Usually, stale information is of no use. EnviroSuite allows programmers to define `attribute_freshness`, which determines how often aggregate attributes are to be sampled and updated by the mechanisms that compute them in EIPLib.

3.3 Defining Object Methods

Sensor network applications can have more complex functionality than merely monitoring attributes. In general, computation, communication or actuation could be encapsulated into the definition of an object. EnviroSuite tries to make full use of existing general-purpose languages, such as nesC, and their existing modules, such as those exported by TinyOS, by separating real object method implementation from object declaration. In object declaration, EnviroSuite programmers are required to denote the name of functions implementing in a general language the object methods. Such functions can use the EnviroSuite communication primitives (using keyword ES_IOC and ES_IOCRESULT) and read values of encapsulated aggregate attributes of the object (using keyword ES_GETATTRIBUTE). The separation of object method declaration and object method implementation retain independence of EnviroSuite abstractions from the underlying programming language.

There are two types of object methods that can be encapsulated within an object. Those object methods specified in object_main_function statements are functions which will be automatically executed upon the creation of the corresponding object. In contrast, object methods specified in object_function statements will be executed only when they are called by other objects. Assuming programmers choose nesC as the general language to implement object methods, the following clause specifies that the implementation of the main object method can be found in nesC command getLocation within interface vehicle.

```
object_main_function = vehicle.getLocation;
```

To facilitate communication and coordination beyond the scope of one object, we introduce a RPC-like mechanism in EnviroSuite, called the *Inter-Object Call* (IOC). IOC is different from traditional RPC in several aspects. First, both the caller and the callee of IOC can be migrating across nodes as the location of the external object changes. Such migration is transparent to programmers, who simply specify the callees instance name (to be stated below) and never worry about which physical nodes these objects are located on. Second, IOC is asynchronous. Callers do not block themselves to wait for results. Instead, results declare their arrivals by interrupts. The keyword for IOC is ES_IOC and ES_IOCRESULT. The former is used for executing an IOC and declaring its handler and the latter for receiving IOC result interrupts. All object methods defined in an object can be remotely called by any other objects by using its reference. The underlying low-level communication protocol and routing extensions to support IOC have been previously published in [Blum et al. 2003] and are thus not described in this paper.

3.4 Defining Static Object Instances and Global Variables

The above discussion covered declaration of object types. Objects that represent fixed environmental elements, such as topological features of the terrain, can be statically instantiated. These static instances can be used, for example, as the destinations of IOCs that invoke object methods. Object types that do not have static instances will be instantiated dynamically at run-time when their object conditions become true. They would have to send their handle to any other objects that need to communicate with them.

EnviroSuite also allows programmers to define globally shared static variables in (static) object declarations and to access defined static variables in object method implementation by using EnviroSuite keyword, ES_READ and ES_WRITE.

The next section gives a complete tracking application implemented in EnviroSuite, including code samples for static instances and static variables.

3.5 A Tracking and Monitoring Application in EnviroSuite

A typical tracking and monitoring application written in EnviroSuite (and some nesC) is shown in Figure 4. The main function of this application is to estimate the current location of a tracked vehicle, update the estimates every 500 ms and report

Object Declarations

```
object VEHICLE {
    object_condition = ferrous_object()&&vehicle_sound();
    object_attribute location {
        attribute_value = AVERAGE(position());
        attribute_degree = 2;
        attribute_freshness = 500ms; }
    object_main_function = vehicle.getLocation; }
```

```
object NETWORK_HEALTH {
    object_condition = TRUE;
    object_attribute energyLevel {
        attribute_value = voltage();
        attribute_freshness = 20m; }
    object_main_function = networkHealth.getEnergyLevel; }
static NETWORK_HEALTH networkHealthInstance;
```

```
object MONITOR {
   object_condition = NULL;
   object_main_function = monitor.start;
   object_function = monitor.reportLocation;
   static int vehicleNumber = 0; }
static MONITOR monitorInstance;
```

Implementations of Object Methods

object method implementation of object VEHICLE

```
Triple_float_t *currentLocation;

command result_t vehicle.getLocation() {
    call ES_WRITE(monitorInstance.vehicleNumber,
                        monitorInstance.vehicleNumber +1);
    return call Timer.start(TIMER_REPEAT, 500); }

event result_t Timer.fired() {
    currentLocation = call ES_GETATTRIBUTE(location);
    ES_IOC report = call monitorInstance.monitor.
                                reportLocation(currentLocation);
    return SUCCESS; }

ES_IOCRESULT report(bool result) {
    //deal with remote call results here
    return; }
```

object method implementation of object NETWORK_HEALTH

```
uint16_t currentEnergyLevels[MAX_NODE_NUMBER];

command result_t networkHealth.getEnergyLevel() {
    return call Timer.start(TIMER_REPEAT, 1200000); }

event result_t Timer.fired() {
    currentEnergyLevels = call ES_ATTRIBUTE(energyLevel);
    //deal with obtained node IDs and voltage values here
    return SUCCESS; }
```

object method implementation of object MONITOR

```
command result_t monitor.start() {
    return SUCCESS;  }

command bool monitor.reportLocation(Triple_float_t
  Location) {
    //deal with received target location here
    return TRUE; }
```

Fig. 4. An EnviroSuite application

the estimated location to the base station every 500 ms. The total number of vehicles is counted. At the same time, voltage values for individual nodes are collected every 20 minutes to obtain system health information. This application illustrates the main abstractions supported by the framework, as well as the programming style.

The application declares three object types `VEHICLE`, `NETWORK_HEALTH` and `MONITOR` which refer to a dynamically instantiated object, a geographic region object and a function object, respectively (lines 1 - 20).

For object type `VEHICLE`, the `object_condition` statement (line 2) specifies its sensory signature as `ferrous_object()` and `vehicle_sound()`. The `object_attribute` statements (lines 3 - 6) define an aggregate attribute `location` for which the value is the average of positions of more at least 2 nodes, updated every 500 ms. The `object_main_function` statement (line 7) states that main object method implementation can be found in interface `vehicle` that includes command `getLocation`.

For object type `NETWORK_HEALTH`, the `object_condition` statement (line 9) specifies its object context as `TRUE` to include all nodes in the network. The `object_attribute` statements (lines 10 - 12) define an attribute `energyLevel` as the voltage values of individual nodes with an update rate 20 minutes. The `object_main_function` statement (line 13) defines that main object method is command `getEnergyLevel` in interface `networkHealth`, which obtains an array of node IDs and voltage values. Line 14 creates a static instance `networkHealthInstance` for object type `NETWORK_HEALTH` so that it will be instantiated statically in system initialization and IOCs can be made through this reference.

For object type `MONITOR`, the `object_condition` statement (line 16) specifies `NULL` as the object context since the object is not mapped to any environmental element. The `object_main_function` statement (line 17) specifies the command `start` in interface `monitor` as the main object method. Finally, the `object_function` statement (line 18) defines that command `reportLocation` in interface `monitor` can be remotely called by any other objects by using IOC and its static instance `monitorInstance` (line 20). Line 19 defines a static variable `vehicleNumber` which is globally accessible by any object through `ES_READ` and `ES_WRITE`.

In the object method implementation of object `VEHICLE`, it is defined that static variable `vehicleNumber` is increased by one whenever a new instance of `VEHICLE` is created (line 23). For each instance, every 500 ms (line 24) the current value of aggregate attribute `location` is fetched (line 26) and sent to the base station by using `ES_IOC` (line 27) to remotely call method `monitor.reportLocation` located in static instance `monitorInstance`. In line 29, `ES_IOCRESULT` keyword is used to receive IOC interrupts of `ES_IOC report`. The interrupt handler name must be the same as `ES_IOC` which is `report` and the parameters should be of the same type as the returned value of remote called method `reportLocation` which is `bool`.

In the object method implementation of object `NETWORK_HEALTH`, it is defined that every 20 m (line 33) the current values of individual voltages are collected (line 35) and analyzed (not included) to monitor system health.

The object method implementation of object `MONITOR` includes the implementation of its constructor method `monitor.start` (lines 37 - 38) and its exported method `monitor.reportLocation` (lines 39 - 40).

This application is used as a running example throughout this paper. It is compiled and evaluated on an actual sensor network as well as on TOSSIM.

4. OBJECT MAINTENANCE ALGORITHMS

To support EnviroSuite abstractions, the main question is how physical state, events, and activities can be uniquely and identically mapped into objects despite of distribution and possible mobility in the environment. This section gives extensive answers.

While all objects in EnviroSuite have the same declaration syntax and programming interface, underneath the common API, EnviroSuite supports three different implementations of objects, namely, *event objects* (created for mobile events defined as those that dynamically change their geographical locations), *region objects* (mapped to static or slowly moving regions), and *function objects* (not mapped to an environmental element).

To alleviate the programmers burden, EnviroSuite can automatically determine the best category for each object based on the keywords used in the `object_condition` statement. Conditions defined in terms of volatile measurements (such as motion sensing) typically give rise to dynamic contexts with rapidly changing node membership, which are more appropriately implemented as event objects. In contrast, conditions defined in terms of slowly changing measurements (such as temperature) result in more stable groups that can be implemented as region objects. Taking sensor type into account therefore allows the compiler to make intelligent guesses about the most appropriate group management protocols to use for object implementation. The programmer is allowed (although not required) to annotate the object as event or region object, overriding the compilers intelligent guess. An incorrect annotation, however, will result in impaired performance. Function objects are similar to region objects, except that they do not interact with the physical environment. In the following, we describe the three different object maintenance protocols, which determine how and when to form the object context, what group management protocols are involved, where to execute object code, and how to compute object attributes.

4.1 EVENT OBJECT MAINTENANCE

Typically, event objects are created dynamically in response to environmental events that may be mobile and usually fast moving. (A compile-time warning is generated if a static instance is declared for such objects.) In the current implementation and in the discussion below, only localized events are supported. By a localized event, we mean those with a geographically limited sensory signature, such as moving vehicles. We call such localized events, *targets*. Supporting events with a large signature that move quickly is challenging because of the high overhead. However, we do support slowly moving large-signature events as described in region objects.

The core component of our event object implementation is the *multi-target group management protocol* (MGMP). When the condition statement of an event object evaluates to true in a new contiguous region, MGMP creates a new globally unique address, *object ID*, and associates it with the geographically contiguous group of sensor nodes which sense the environmental event. The movement of the contiguous region associated with the event results in dynamic changes to group membership. The protocol ensures that the same object ID is maintained for the event object despite mobility and membership changes, so that it can always be addressed via its uniquely assigned object ID. Dynamically created event objects are aware of their ID and must explicitly send it to other objects if they want to be contacted. Observe that the internal details of MGMP are transparent to the programmer. From the perspective of application code, the only visible effect of MGMP is the

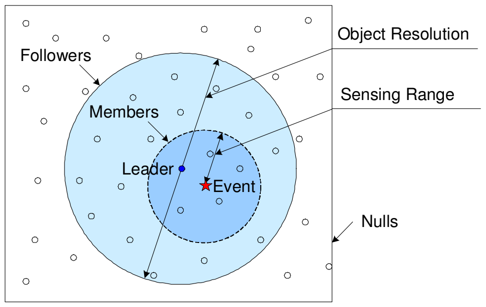


Fig. 5. States of nodes around an environmental event

dynamic creation and deletion of object instances (in response to environmental conditions). These instances encapsulate aggregate object attributes as defined by the programmer.

Internally, MGMP elects a leader in each object context to maintain a persistent and unique object ID, collects raw data from group members in the context, performs aggregation functions on the leader to compute object attributes, and coordinates computation and actuation tasks as defined in object methods.

In the following, we discuss how MGMP maintains object uniqueness (one-to-one mapping of external events to logical objects) and object identity (immutability of the mapping function) for fast moving targets.

4.1.1 *State Machine Representation.* MGMP treats each node as a state machine. The sensor network around an environmental event might have the state distribution shown in Figure 5. It should be noted that although we use circles to indicate sensing areas, we do not assume sensing areas are circular.

All nodes sensing an event constitute the *member* set. A single *leader* is elected by MGMP among the member set. The leader sends periodic *heartbeats* to nodes within half an `object_resolution` (default half is two times the sensing range) away from itself to claim its leadership and to inform them of the existence of the event. Note that the sensing range can be statically derived from the sensor characteristics, and, if the event is detected by a combination of multiple sensors, the shortest one is used. Heartbeats are disseminated through limited flooding, and later on, members communicate to the leader through reverse paths of flooding. The period of these messages, called the *heartbeat period*, is one of the key parameters of MGMP. As we show in the evaluation section, this period can be chosen automatically by EnviroSuite from a high-level specification of the maximum abject creation latency.

All nodes that cannot sense the event themselves but know of its existence through received nearby leader heartbeats are said to be in the *follower* state. All MGMP control messages are transmitted to nodes within half `object_resolution` away from senders. Thus, half the `object_resolution` must be no less than two times the sensing range, since nodes within the same sensing area must communicate with each other to agree on a single leader. The minimal tolerable object resolution in EnviroSuite is therefore four times the sensing range.

At any point of time, a node stays at a single state from a set of states S_s:

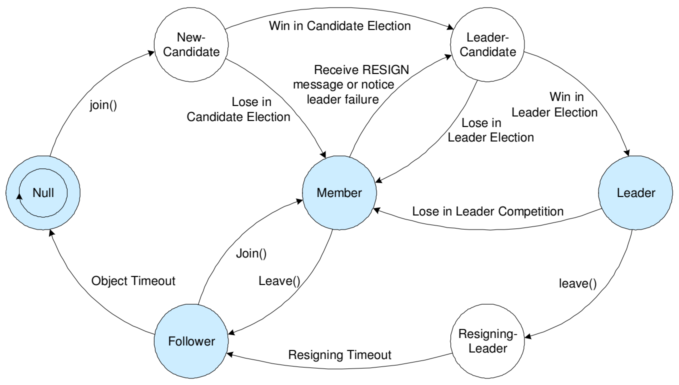


Fig. 6. State machine in MGMP

$$S_s = \{Null, Follower, Member, NewCandidate, LeaderCandidate, Leader, ResigningLeader\}$$

To make MGMP suitable for sensor devices with limited computation and storage ability, we allow each node except leaders to maintain only one object/object ID to reduce algorithm complexity both in time and space. For instance, a node cannot act as `member` of two different objects/object IDs. Figure 6 depicts the general state machine algorithm of MGMP without providing details of associated objects/object IDs.

4.1.2 *Maintaining Object Uniqueness.* Object uniqueness can be compromised in several cases. The first is at the time when a new event causes the creation of a new object. Multiple object IDs for one event may be created since there is no agreement on a single leader initially. To solve the problem we employ a *delayed object creation* mechanism, which delays the creation of a new object by an amount called the *candidate period*, until we are of high confidence that the group of nodes has elected a single `leader` node. In this mechanism, `null` nodes, when sensing an event, transit their states to `newCandidate` and begin to send periodic `CANDIDATE` messages at the heartbeat period, containing sequence numbers and their own node ID. To prolong system lifetime, instead of using the fixed heartbeat period, we can enhance energy balancing by using a dynamic period inversely proportional to remainder energy of nodes. Hence, nodes with a higher energy will become candidates first and will have a higher chance of being elected. In the case of re-transmissions, candidates with a higher energy can back-off less, hence having a higher chance of successfully claiming leadership. The node with the smaller sequence number or, if sequence numbers are equal, with the bigger node ID is forced to quit from the `newCandidate` state and transition to state `member`. This procedure is called *candidate election*, which finally results in only one node at the `newCandidate` state. After a given delay (namely, the candidate period) this node transits to the `leaderCandidate` state. The candidate period is measured in the number of periodic `CANDIDATE` messages sent before one `newCandidate` node can transit to the `leaderCandidate` state. The candidate election algorithm ensures a single `leaderCandidate` in the absence of message loss. Even if messages can be lost, by increasing the candidate delay, a single `leaderCandidate` can be generally guaranteed since the possibility of consecutive message loss is small. In the evaluation section, we determine a good choice for the candidate delay, such that the programmer need not be involved in the decision.

The next problem that compromises uniqueness occurs during leader re-election. When tracked events move out of the current leaders sensing ranges, these leaders must handover their leadership to other nodes, which is called *object migration*. Object migration, especially frequent object migration caused by fast moving events, challenges the maintenance of object uniqueness. MGMP solves this problem by introducing the `follower` state. Through heartbeats from leaders, `follower` nodes know in advance the event objects associated with incoming events. When these nodes come to sense these events, they join the existing objects as `member` instead of creating spurious objects. It was shown in [Abdelzaher et al. 2004] that this mechanism is successful in maintaining object uniqueness as long as object velocity is below some maximum limit.

The third case that challenges object uniqueness is when multiple events of same signatures become closer than defined object resolution, or even cross each others path. To simplify the situation, we assume that event crossing does not coincide with event disappearance. In the previous cases without event crossing, the delayed object creation mechanism and the introduction of the `follower` set ensures object uniqueness. Here, we need only to prevent accidental object termination during event crossing, so that object uniqueness is maintained. The leadership handoff mechanism used in MGMP prevents object termination as long as the object is maintained by one `leader` node and at least one `member` node. Thus, the key in maintaining object uniqueness during event crossing is to balance `member` nodes between merging objects to assign at least one `member` for each object, which is detailed below.

To show the *member balancing* mechanism, Figure 7 depicts part of the state machine, which describes how `member` nodes choose their corresponding objects. $\texttt{Member}_x$ denotes `member` state with object ID x. A simple way to balance Member nodes is to divide `member` nodes based on leader position. When a `member` node receives heartbeats from multiple objects, it chooses to join the one with the nearest leader since there is a higher possibility that this node is sensing the same event as that leader. However, such division is not accurate since leader positions are not identical to event locations. It is also possible that a `member` node is actually sensing the same event as the farther leader. For this reason, a new state called `freeMember` is introduced into the state machine. The continuous reception of n continuous heartbeats from `object 2` can transit $\texttt{member}_1$ to `freeMember` and then to $\texttt{member}_2$ even if the last heartbeat was from a closer leader (`object 1`). The introduction of `freeMember` allows wrong choices to be corrected, thus ensuring correctness of member balancing. The member balancing mechanism prevents object termination successfully, therefore enhancing object uniqueness in the third case.

4.1.3 *Maintaining Object Identity.* While object uniqueness refers to maintaining a single object representation for each external target, maintaining object iden-

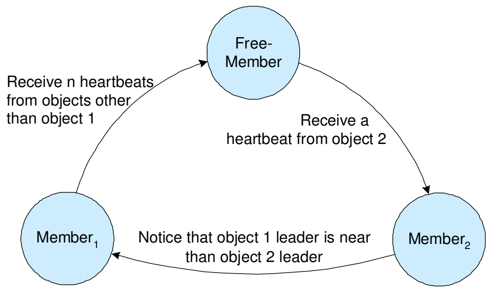

Free-Member
Receive n heartbeats from objects other than object 1
Receive a heartbeat from object 2
Member1
Notice that object 1 leader is near than object 2 leader
Member2

Fig. 7. Member balancing mechanism

tity refers to keeping the *correct* association between external targets and their representing objects. In the case where events with same signatures are closer than one object resolution, an extra mechanism is required to maintain identity since member balancing only ensures object uniqueness. EnviroSuite makes the default extra assumption that targets tend not to change direction abruptly. This assumption, for example, allows disambiguation of crossing targets based on their path. The default assumption can be customized by programmers if needed.

EnviroSuite keeps a record of the recent trajectory of each target (storing it within its representing object). To reduce system overhead, instead of using position information of group members to estimate target locations, EnviroSuite takes leaders positions as an approximation. When transferring leadership, each leader also transfers its maintained history of the last n−1 old leaders positions plus its own. When two events E_1 and E_2 cross each others path, each object leader is able to receive the heartbeat from the other. Each object leader marks itself by the concatenation of the old object ID (O_1) and the new object ID (O_2) as its temporary object ID (O_1O_2). The two leaders exchange their event trajectories such that each remembers both. After separation, a disambiguation algorithm is used, based on recorded history and current locations to chose the ID assignment most consistent the default (straight path) assumption.

4.2 REGION AND FUNCTION OBJECT MAINTENANCE

Region object maintenance differs from event object maintenance since region objects are associated with a relatively fixed set of nodes. What we implement for region object maintenance is a spanning-tree based information collection structure described in [He et al. 2004]. Like event objects, the details of region object maintenance are transparent to the programmer. The application code is only aware of the object and its encapsulated aggregate attributes.

When a region object is initialized (statically at system deployment time or dynamically, depending on whether a static instance is declared), a default leader node disseminates tree construction requests to the object context with a running hop-count initialized to zero. Requests are flooded outward with hop-count incremented at every intermediate hop. After receiving tree construction requests, nodes establish multiple reverse paths towards the sending nodes. As a result, a multi-parent diffusion tree is constructed with the leader residing at the root. Spanning tree construction stops when nodes are reached that do not satisfy the region object condition statement. Such nodes become the outer boundary of the tree and serve a role similar to followers in event objects. If these nodes ever satisfy the condition statement they become members and recruit other followers for which the statement is not satisfied. Also, if tree leaves cease to satisfy the object condition, they truncate themselves from the tree and become outer boundary nodes. Hence, membership of the tree can change slowly over time. Measurements needed to compute object attributes can flow up the tree from members towards the leader and get aggregated along intermediate hops. We do not provide details of aggregation algorithms here, since similar mechanisms have been described in previous literature such as directed diffusion [Intanagonwiwat et al. 2000] and TAG [Madden et al. 2002]. Our contribution lies in the uniform programming abstractions presented on top of such mechanisms.

One aspect where our region object maintenance algorithm differs from previous work is that we automatically migrate the root of the aggregation tree to the location that minimizes communication and aggregation overhead, as well as to a higher energy node, periodically escaping energy depleted regions. This load balancing flexibility is made possible in our programming model since we implement the program inside the network, alleviating external bottlenecks. After each migration, a (possibly partial) tree reconstruction is done to form a new spanning-tree rooted in the new host node.

The introduction of region objects enables EnviroSuite to support not only tracking functions, but also region monitoring functions such as contour finding and system health monitoring, thus making EnviroSuite applicable to a broader set of applications.

Function objects are quite similar to region objects except that there are no object contexts and object attributes in function objects. There is no need for object context maintenance and object attribute collection since function objects do not interact directly with the physical environment. In EnviroSuite, the leader of a function object always migrates to the gravity center of all other objects which have recently communicated with the function object through IOC or global variable access.

Like in event objects, leaders in region objects and function objects are responsible for object method execution.

5. IMPLEMENTATIONS IN NESC

In this section, we take nesC, the most popular language in sensor network area, as the general language that implements EnviroSuite. EnviroSuite object declarations (defined by programmers) and object methods (assumed to be written in nesC by programmers) are to be automatically translated by EIPLC into a whole nesC application by selecting and integrating primitive algorithms provided in EIPLib. This section describes how we design EIPLib to simplify the work of EIPLC and how we implement the compiler EIPLC with the help of EIPLib. Although the implementation details are specific to nesC, most design decisions we make in this section are portable to other languages.

All nesC applications consist of a set of *components*. A component provides and uses *interfaces*, as defined in the components *provides* and *uses* clauses. An interface describes the parameters of a set of *commands* and *events*. There are two types of components: *modules* and *configurations*. Modules provide application code, implementing one or more interfaces. Configurations connect interfaces used by components to interfaces provided by other components. The action of connecting component interfaces is called component *wiring*. It is the main mechanism for building large applications from smaller modules. Wiring is done at compile time, and offers no run-time overhead. We use wiring extensively to connect application components to components implemented by our language libraries.

5.1 EIP Service and Protocol Library (EIPLib)

EIPLib contains a series of primitive algorithms to be used by EIPLC to build comprehensive applications in nesC, currently including: sensor data processing algorithms (condition library), aggregation algorithms (aggregate method library), object maintenance algorithms and inter-object communication protocols. It also contains higher level services as potential consumers of primitive algorithms, including: object context determination components, object attribute collection components and object method execution components.

Each condition such as `temperature()` and `vehicle_sound()` is associated with a sensor data processing algorithm in EIPLib, which returns processed sensor outputs either as a meaningful value or a boolean either immediately or in a phase-splitting way. However, the association and ways of accessing are hidden in EIPLC and programmers are only aware of available condition names and their purposes. Also, each aggregation method such as `AVERAGE` is associated with an aggregation algorithm which implements the method. In the current version, object maintenance algorithms contain separate implementations for three object categories: event objects, region objects and function objects. As stated in the beginning of Section 4, the object categories are transparent to programmers and are determined by EIPLC based on the `object_condition` statement. (Advanced APIs are provided for sophisticated programmers to override default rules.) Inter-object communication protocols provide supports for maintaining links between dynamic objects, which is required to implement IOCs and global variable access. All these primitive algorithms are implemented as nesC components with standard interfaces.

Object context determination components determine whether the current node should join some object context based on object declarations. Object attribute collection components collect raw object attributes from member nodes, apply aggregation methods to form aggregate attributes in leader nodes, and support access to aggregate attributes. Object method execution components are responsible for executing object methods in leader nodes whenever corresponding objects exist. These higher level components are also implemented in the form of nesC components, yet differ from usual nesC components in many ways, including:

(1) They are not pre-wired since object declarations and object method implementations are not available until compile time. Wiring is left for the compiler so that primitive algorithm components may be freely selected and wired into higher level components to construct any EnviroSuite applications defined by programmers.

(2) They contain special clauses that are recognizable only by the compiler. In

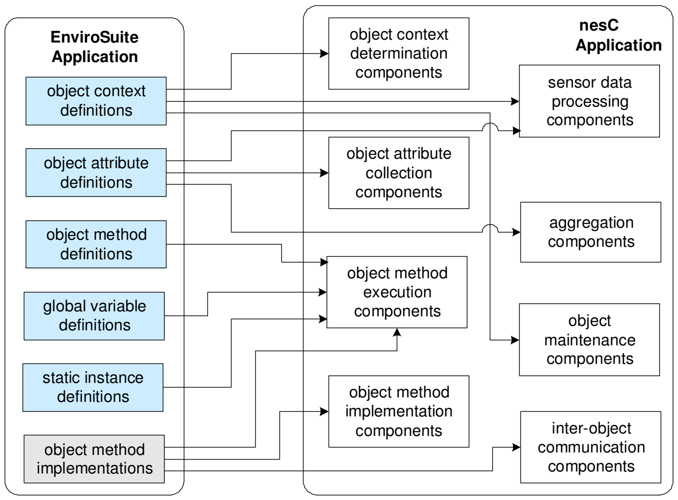

EnviroSuite Application
object context definitions
object attribute definitions
object method definitions
global variable definitions
static instance definitions
object method implementations
nesC Application
object context determination components
sensor data processing components
object attribute collection components
aggregation components
object method execution components
object maintenance components
object method implementation components
inter-object communication components

Fig. 8. Translate an EnviroSuite application into a nesC application

many cases, such clauses are necessary to guide language translation. For example, the configuration of object context determination components may include a special clause ({__ES_COMPONENTS}) which indicates the position where necessary sensor data processing components are to be listed by EIPLC. These clauses greatly simplify the implementation of EIPLC by giving some hints.

The hierarchical structure between primitive algorithms and high level components is also critical. In this structure, various configurations can be achieved by changing only the high level components while other components can remain unchanged, thus reducing the complexity of the compiler.

5.2 EnviroSuite Compiler (EIPLC)

EIPLC is essentially a translator that takes EnviroSuite code as input and outputs desired environmental monitoring applications in nesC, which then can be compiled by a standard nesC compiler and uploaded to the motes. EIPLC is implemented in Perl, a language with powerful built-in support for text processing. Current implementation of EIPLC contains 1533 lines.

EnviroSuite application code consists of two parts, object declarations and object method implementations. The detailed translation of both parts is illustrated in Figure 8.

EIPLC analyzes object declarations line by line, making corresponding configurations and integrations. As depicted in Figure 8, for object context definitions, EIPLC identifies all conditions, locates the corresponding sensor data processing components by searching condition library, which lists all condition names and the corresponding implementations, wires them into object context determination components. A feature of EIPLC is that it automatically determines the best category for each object based on these conditions and integrates the corresponding object maintenance algorithms.

For object attribute definitions, besides identifying conditions and wiring corresponding components into object attribute collection modules, EIPLC also wires aggregation components. Additional work includes setting attribute refreshing timers based on `attribute_freshness` definition and validating resulted aggregate attribute based on `attribute_degree` definition. Based on object method definitions, EIPLC wires the implementations into object method execution components. EIPLC also copies global variable definitions into object method execution components and enables remote access to global variables by implementing local read and write commands, which respond to received remote calls. For each static object instance, EIPLC randomly selects a node as the default leader, which initially executes the main object function, and migrate the leader to a more power-efficient position later.

EIPLC also filters object method implementations for keywords, translating `ES_GETATTRIBUTE` into command calls to object attribute collection components and `ES_IOC`, `ES_IOCRESULT`, `ES_READ` and `ES_WRITE` into command calls and event handlers of inter-object communication components.

As seen above, EIPLC successfully bridges between low-level implementations in EIPLib and high-level abstractions exported by EnviroSuite by making several intelligent steps that are transparent to the programmers: selecting sensor data processing algorithms; automatically identifying object categories and applying corresponding maintenance algorithms; and automatically collecting and aggregating attributes from multiple nodes.

Observe that, one clause in an EnviroSuite application may result in multiple changes in higher level components, and one higher level component from EIPLib may be changed multiple times by multiple EnviroSuite clauses, which means EIPLC may need to change the same file in EIPLib repeatedly. Considering such phenomenon, instead of creating corresponding new code line by line, we store the resulting changes in a hash of hashes, so that already changed code can be further changed easily. The hash of hashes stores, for each file and each special clause such as {`__ET_COMPONENTS`}, their corresponding nesC code. Only after analyzing the entire EnviroSuite application, EIPLC changes files from EIPLib based on the resulted hash of hashes. The storage space needed by the hash of hashes may be very large. However, we consider it acceptable since EIPLC runs on a PC and therefore does not have severe storage constraints.

6. PERFORMANCE EVALUATION

This section provides a detailed quantitative analysis of EnviroSuite. We begin by evaluating the performance of a series of micro-benchmarks on simulators, which analyze the primary features of EnviroSuite: object uniqueness and identity maintenance, and inter-object communication support. The first set of benchmarks tests object uniqueness and identity management during object creation, object migration and object crossing (which is the most challenging case). The second set of benchmarks tests inter-object communication. We then move to real platforms to evaluate the performance of a surveillance system built using the EnviroSuite framework. Both tracking performance and monitoring performance are evaluated to demonstrate event objects and region objects. The evaluated system is the one described in Section 3. Its abbreviated code is given in Figure 4.

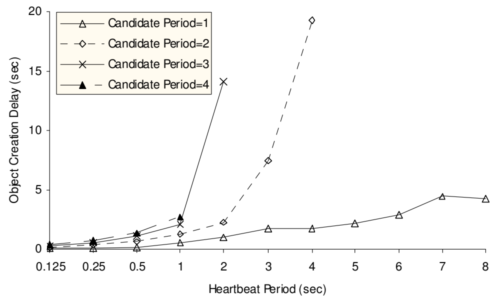


Fig. 9. Object creation delay for varied heartbeat period and candidate period

6.1 Performance of Object Operations

To evaluate the performance of primitive object operations, we choose TOSSIM since EnviroSuite produces real nesC code for motes and TOSSIM can emulate the execution of the real code on the motes without the need for deployment. The radio model simulated in TOSSIM is almost identical to the 40 Kbit RFM-based stack on the motes. To control per-hop message loss at the packet level we added an external program component. We focus on fast-moving objects (event objects), since their real-time maintenance offers the most challenge to the EnviroSuite infrastructure.

In our emulated experiments, we set the sensing range to 100 feet (approximately 30 meters). Current sensor devices such as the micropower impulse radar [Azevedo and McEwan 1996] can detect objects up to 50 meters away. Radio range is set to 300 feet. Current sensor network products such as the Mica2 and Mica2Dot motes [U. C. Berkeley 2005] have a maximum outdoor radio range of 500 feet to 1000 feet under ideal conditions when sending with full power. Sensor nodes are placed on a grid 100 feet apart.

6.1.1 *Experiment 1 - Object Creation.* EnviroSuite associates a logical object with each physical event. It is critical that such association should be done as soon as possible to reduce the inconsistency between the physical world and the logical world exported by EnviroSuite. In the first experiment, we measure object creation delay, defined as the difference between the time the first node senses an external stimulus and the time an object ID is created for the corresponding object. The external entity tracked, in this case, is a vehicle. The tracking code is given in Figure 4.

The delay of object creation is decided by both the candidate period, which indicates how many candidate messages must be sent before creating objects, and the heartbeat period, which determines candidate message intervals. In the following we show the experimental data that allow these parameters to be selected automatically by EnviroSuite from a high-level specification of the maximum tolerable object creation delay. Figure 9 plots object creation delay versus heartbeat period for different candidate periods.

From Figure 9 we observe that object creation delay increases with the increase in both the candidate period and the heartbeat period. The plots show only those points for which a non-zero number of objects are created. A candidate period of 1 performs best in terms of object creation delay. However, it is undesirable since

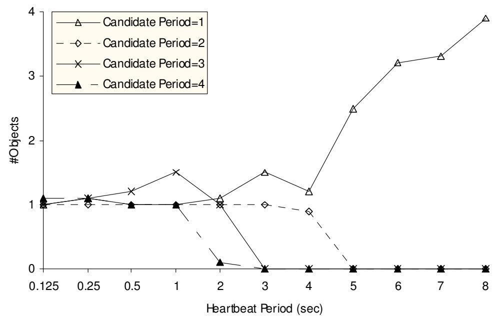


Fig. 10. Number of objects created for varied heartbeat period and candidate period

it causes spurious objects at higher heartbeat periods as stated below.

The candidate period and the heartbeat period affect not only object creation delays but also object uniqueness. Figure 10 shows the impact of the candidate period and the heartbeat period on object uniqueness by plotting the average number of created objects. Ideally, only one object should be created per experiment, since the only target is deployed.

Figure 10 shows that with shorter heartbeat periods, candidate periods 2, 3 and 4 perform similarly. However, longer candidate periods result in a longer object creation delay, so that when the heartbeat period exceeds a certain threshold, objects cannot be formed in time before the vehicle moves out of their sensing ranges. Figure 10 shows that for candidate period 4, it is difficult to create objects after the heartbeat period exceeds 2 seconds, while for candidate period of 2, objects can be created up to a heartbeat period around 5 seconds. We therefore choose 2 as the default candidate period in EnviroSuite. A longer candidate period should be chosen in the presence of message loss.

Given the default candidate period (of 2), the object creation delay can be chosen anywhere from a small fraction of a second to multiple seconds depending on the choice of heartbeat period, as shown in Figure 9. The programmer should therefore specify a maximum tolerable value of object creation delay. This specification stems easily from application domain knowledge. For example, in a vehicle tracking application, a delay of 1-2 seconds between vehicle entry into the field and the creation of a corresponding event object is quite tolerable. EnviroSuite then uses Figure 9 to compute the corresponding heartbeat period. Observe that a smaller heartbeat period implies more communication, more energy consumption, and consequently a shorter lifetime. Hence, a trade-off exists between system responsiveness (object creation delay) and lifetime.

6.1.2 *Experiment 2 - Object Migration.* The core part of EnviroSuite is to uniquely and identically map physical events to logical objects. In this experiment, we reveal how fast object migration could be performed without breaking object uniqueness and identity. Object migration is caused by the movement of associated events. Hence, from the perspective of applications, the velocity limit of object migration is more meaningfully expressed by the maximum tolerable event velocity. It is defined as the maximum velocity of events, which can be uniquely and

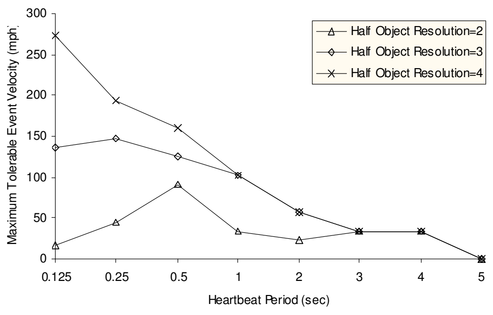


Fig. 11. Maximum tolerable event velocity for varied heartbeat period and object resolution

identically mapped to logical objects. Observe that for a given maximum object migration speed (in hops per second), the corresponding maximum event velocity depends on the radio range (distance per hop). The data presented below is for the range parameters mentioned in Section 6.

We explore several factors, which affect maximum tolerable event velocity, including heartbeat period and object resolution (in multiples of sensing range). As is shown in Figure 11, the maximum tolerable velocity increases when the heartbeat period decreases, since a shorter heartbeat period results in a shorter leader re-election delay and thus a higher trackable velocity. This trend is reversed when heartbeat period becomes short enough to cause message loss or congestion as shown in Figure 11 (for a half object resolution of 2 sensing ranges) when heartbeat period falls below 0.5 s. Increasing object resolution has positive impact on the maximum tolerable velocity since a bigger set of followers allows the vehicle to go farther without causing new object creation. Similar results were reported in [Abdelzaher et al. 2004]. We stress, however, that results reported in [Abdelzaher et al. 2004] were obtained from algorithm simulation in GloMoSim. In contrast, results presented in this paper test the performance of actual nesC code generated by our functional EIPLC compiler for the application in Figure 4.

Next, we evaluate how robust the object uniqueness guarantee is against message loss during object migration. TOSSIM does not provide message loss models at the packet level. Thus, we add a simple external program to control per-hop packet loss ratio. Figure 12 depicts the average number of objects formed per run as a function of target velocity in the presence of different degrees of packet loss. As before, the ideal number should be 1 object per run.

From Figure 12, we see that EnvoroSuite can completely tolerate a 10% loss ratio since we get similar results to those with 0% loss ratio. EnviroSuite can also tolerate a loss ratio of up to 30% when event velocity does not exceed 68 mph. Larger velocities or loss percentages, however, cause spurious objects to emerge. Observe that at a very high event velocity, the number of formed objects decreases again, which might seem like an anomaly. The explanation lies in that very high speed objects do not have enough time to form in the first place.

6.1.3 *Experiment 3 - Object Crossing Performance.* Next, we explore the efficacy of EnviroSuite in maintaining object uniqueness and identity when two objects of the same sensory signature (i.e., fulfill the same `object_condition` statement)

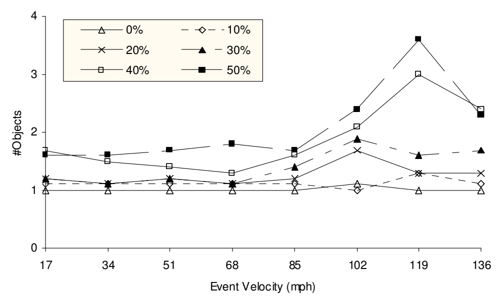


Fig. 12. Number of objects created for varied event velocity and per-hop packet loss ratio

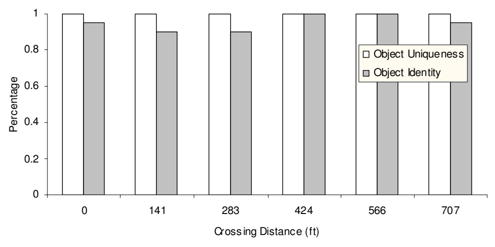


Fig. 13. Achieved object uniqueness (white) and identity (shaded) for varied crossing distance

cross paths. In this experiment, two vehicles are moving straight along crossing diagonals with the same speed of 24 mph. The diagonals cross in the center of the field. However, these objects may not start at the same time, and hence may not reach the crossing point together. We vary their relative start times to vary the shortest distance reached between the two objects at the crossing point (which we call, the *crossing distance*). We show the percentage of runs where object uniqueness and identity are maintained as a function of crossing distance. As shown in Figure 13, object uniqueness and identity are ensured in most cases even when the two targets cross the center point at the same time (crossing distance is 0).

Each bar in Figure 13 represents the average of more than 10 runs. The tracked trajectory for one run with crossing distance 0 is shown in Figure 14. After passing the center point, although object identity is lost for a while, the system successfully recovers from the confused state after accumulating enough history.

The results prove the relative success of our adopted direction disambiguation algorithm. It also shows that defensive programming is advisable. While we elevate the level of abstraction to that of objects representing environmental elements, the programmer should expect such objects to be occasionally confused. The application code may chose to implement its own disambiguation on top of EnviroSuite object IDs.

6.1.4 *Experiment 4 - Inter-object Communication.* Programming for communication and coordination between objects becomes very simple by using IOC and global variables. In this experiment, we evaluate a vehicle counting application, which counts the total number of vehicles in a global variable, to analyze the performance of inter-object communication. As seen in the code from Figure 4, whenever

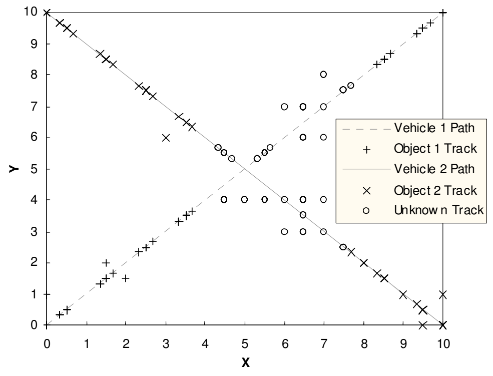


Fig. 14. Reported target tracks with crossing distance 0

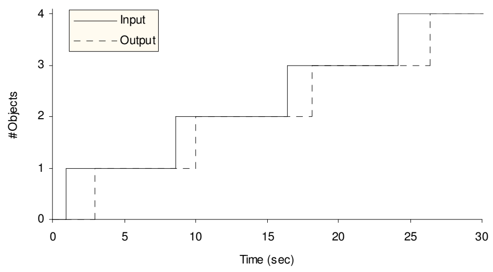


Fig. 15. Vehicle counting application results

a vehicle appears, the global variable `vehicleNumber` is increased by one through an `ES_WRITE` call from the corresponding `vehicle` object.

In this scenario, four vehicles enter the coverage field one by one, maintaining the same speed of 35 mph and thus the same distance. The first one goes straight from (-1, 1) to (16, 1); the second from (-5, 5) to (16, 5); the third from (-9, 9) to (16, 9); the last from (-13, 13) to (16, 13).

Figure 15 plots the counter values as a function of time in this application. Input represents real numbers of vehicles. Output represents the counting results achieved by the application. Delay between the input and output curves represent the end-to-end performance of remote object invocation. These delays reflect the sum of object creation delay and inter-object communication delay.

6.2 A Surveillance System

Finally, we test the complete surveillance application written in EnviroSuite, described in Section 3. This surveillance system tracks all in-field vehicles, counts their number and monitors system health at the same time. The EnviroSuite code of this application can be translated by EIPLC into a nesC application. Emitted nesC code size of different services in the translated application is listed in Table II.

Table III compares EnviroSuite code and emitted nesC code of the same application in terms of module number, code length and size. The code size of the nesC version gives a good estimation of required programming effort if the whole system

Table II. Services and code sizes of the nesC application translated from the EnviroSuite application

Service Name	Code Size (KB)
Sensing Data Processing	10.8
Event Object Maintenance	25.6
Region and Function Object Maintenance	28.5
Inter-object Communication	15.0
Other Service (Aggregation, etc.)	25.1
Object Method Components	6.0

Table III. Code Comparison of EnviroSuite Version and nesC Version

	Module Number	Code Length (lines)	Code Size (KB)
EnviroSuite Version	3	218	5.9
nesC Version	12	3692	111.0

is to be programmed directly in nesC. As is seen from Table III, the code size of the nesC version is more than ten times of that of the EnviroSuite version. Thus, the estimated programming effort with EnviroSuite is roughly an order of magnitude less. The result reflects the efficiency of EnviroSuite compared with node-based languages, such as nesC.

6.2.1 *Tracking Performance.* In this experiment, we evaluate the efficiency of EnviroSuite in terms of tracking performance and power consumption by comparing it to a simple baseline. This baseline is to plot the trajectory of a tracked target at a base station located in $(0, 0)$. In the EnviroSuite implementation, members, who are sensing the target, report to the current leader their own positions every 0.5 seconds. The leader aggregates these positions and reports the average to the base station twice per second. The baseline has a simple implementation of the same application. Each node that senses the target sends its own position to the base station every 0.5 seconds. The base station averages received positions twice per second. In both the EnviroSuite version and the baseline, a minimum aggregation degree of 2 is enforced to reduce false alarms.

The actual testbed for this experiment consists 40 Mica2 motes laid out in a 10×4 grid with integer (x, y) coordinates ranging from $(0, 0)$ to $(9, 3)$. The goal is to track a rectangular object, 1 square grid in size, moving straight along the middle of the longer axis, with a speed of 0.5 grid per second. This testbed does not take into account errors in localization and time synchronization services. To ensure enough tracking accuracy for real applications, we require that localization errors not exceed half grid and time synchronization errors be kept in the order of ms. Many existent techniques support such precision.

Figure 16 compares the target trajectory obtained by the EnviroSuite application to the one resulting from the baseline. Some tracking error is seen because our sensor devices have no notion of proximity to the target. As shown in Figure 16, the EnviroSuite version has a smaller average tracking error compared with the baseline although it introduces a little more variability. The underlying reason is that in the baseline, position reports from nodes may not be in order when they arrive at the base station, since they may have traversed multiple hops, which results in more inaccuracy.

Figure 17 depicts the number of packets sent or forwarded by each node in a slice

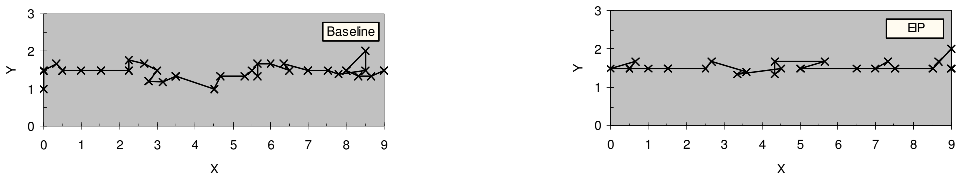


Fig. 16. Tracked target trajectory comparison

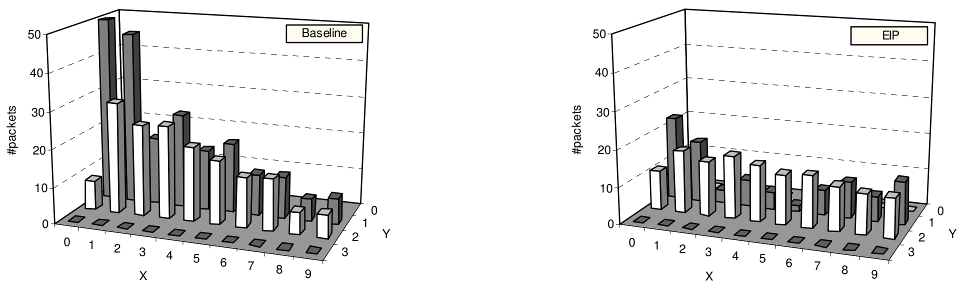


Fig. 17. Transmitted packet number comparison

of the network over the duration of the experiment (where X, Y is the coordinate of each node). Each bar in this figure represents the average of 15 runs to ensure a statistical significance at the 0.05 level. The number of packets is important as it is proportional to power consumption. It can be seen that the EnviroSuite version achieves its comparable tracking performance with much less power consumption in terms of the number of transmitted packets. Hence, our tracking algorithms are more energy-efficient.

In the baseline test, most packet transmissions occur on nodes with Y coordinates between 1 and 2 since only these nodes can forward the packets to the base station. The nodes with smaller X coordinates in the baseline send much more packets than those in the EnviroSuite version since each node sensing the target sends packets directly to the base station located in $(0, 0)$. Hence, a greater number of packets have to be forwarded by nodes with smaller X coordinates. In the EnviroSuite version, position reports are aggregated locally by leaders, amounting to much fewer packets forwarded to the base station.

6.2.2 *Monitoring Performance.* In this experiment, we utilize the `NETWORK_HEALTH` object coded in Figure 4 to monitor the health of the network by collecting information on nodes that are alive and their remaining power. Alarms will be sent out if a big portion of the network is dead or lacks power.

We carry out this experiment on a network of 27 XSM motes [Dutta et al. 2005] deployed in a grassy field. The system performs the function of vehicle tracking as well as health monitoring. For system health monitoring, the `NETWORK_HEALTH` object is determined as a region object by EIPLC, thus a multi-parent spanning tree is automatically constructed at object initialization to collect power information of each node every 20 minutes. The system is tested for several hours. Figure 18 depicts the collected power information, where the black bars represent initial voltage

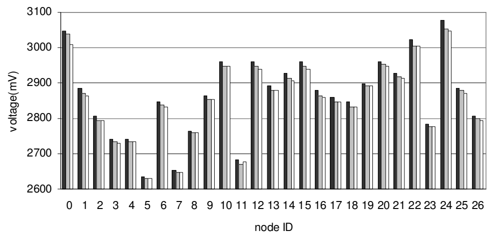


Fig. 18. Power level of each node for different times

reported by the region object, the grey ones show voltage reported after 20 minutes and the white ones shows voltage reported after 40 minutes. Node 0 is the base node, which consumes the most power.

7. RELATED WORK

EnviroSuite opens a new category of distributed programming paradigms. It differentiates itself from traditional paradigms such as CORBA [Vinoski 1997], Microsoft's COM [Microsoft 1994], and remote procedure calls [Birrell and Nelson 1984] by combining within its programming abstractions objects and events in the physical world.

Several communication and programming models have been proposed for sensor networks in recent years. These include node-based languages, virtual machines, database-centric abstractions, event-based models, and group-based primitives. EnviroSuite is different in that its abstractions are not centered about computational constructs such as queries or sensor groups. Instead, these abstractions are centered around elements of the physical environment. The aspiration is that at the highest level of abstraction, the existence of the sensor network itself should be entirely transparent.

Node-based languages such as nesC [Gay et al. 2003] and galsC [Cheong et al. 2003; Cheong and Liu 2005] are too low-level since they typically take the sensor node as basic computation, communication and actuation unit. To address this issue, higher-level languages that export logical nodes [Gummadi et al. 2005] were proposed to abstract away from physical sensors. EnviroSuite successfully raises the abstraction level to logical objects mapped from physical elements, thus expedite the procedure of design and programming compared with node-based languages.

Virtual machines such as Mate [Levis and Culler 2002] and SensorWare [Boulis et al. 2003] allow large sensor networks to be reprogrammable frequently by writing application scripts, replicating them through the network and executing them automatically. However, they usually concentrate on issues related to code replication and auto-execution rather than raising programming abstraction levels. For example, to reduce energy cost of code replication, Mate even provides an instruction-like language to shorten code length, which actually puts extra burden on programmers shoulders.

Database-centric abstractions such as TinyDB [Madden et al. 2003] and Cougar [Yao and Gehrke 2002] view sensor networks as databases that allow users to express requirements as queries, and to distribute and execute these queries. Comparatively, our work, instead of providing a specific data collection and aggregation model, attempts to support a wider range of applications by encapsulating not only computation and communication units but also actuation units into its programming abstractions.

Event-based models such as [Li et al. 2004] are similar with database-centric abstractions except that they view the sensor field as an active entity that automatically push data streams to users when defined events are triggered instead of a passive database which only responds upon queries.

Group-based primitives such as Hood [Whitehouse et al. 2004] and Abstract Regions [Welsh and Mainland 2004], provide neighbor discovery and neighborhood data sharing mechanisms. Compared with EnviroSuite, these abstractions are passive. In contrast, EnviroSuite abstractions are active objects that encapsulate local code and aggregate state, as well as share data across neighborhoods or regions.

Another group-based paradigm, State-centric programming [Liu et al. 2003], described a programming abstraction mostly related to our work. However, it is implemented and evaluated only on Pieces simulator built in Java and Matlab, which can not simulate some critical features of wireless communication including message collision. In contrast, our paper presents a detailed implementation in nesC on TinyOS, an operating system for real sensor network devices, and provides comprehensive evaluation results both in TOSSIM and real sensor devices. Furthermore, the underlying group management protocol [Liu et al. 2003] differs in its mechanisms for object classification and identity management.

An earlier paper by the authors [Abdelzaher et al. 2004] presented a programming paradigm focusing on tracking applications. In this paper, we expand this idea and present programming abstractions that successfully support a broader set of applications including not only event tracking but also regional monitoring applications.

Finally, we should mention that a criticism of current high-level programming languages has been that they are too application specific. Hence, intermediate-level languages such as [Newton et al. 2005] were proposed as a step towards macroprogramming. EnviroSuite attempts to cater to a general application pool by diversifying the supported object types.

8. CONCLUSION

In this work, we describe an environmental immersive programming paradigm for application developers in sensor networks. We present the design, implementation and evaluation of a framework implementing this paradigm. The EnviroSuite framework successfully exports high-level abstractions, such as objects and inter-object calls. It implements low-level distributed protocols such as sensing data processing, group management and inter-object communication in an underlying library EIPLib, transparent to programmers, thus resulting in a considerable potential to reduce development costs of deeply embedded systems. This paper describes the first comprehensive design and implementation of all EIP abstractions including objects, their attributes, methods and inter-object calls (The concept of EIP was described earlier in [Blum et al. 2003]). This paper also presented the first comprehensive evaluation of the performance of real nesC code generated by the EnviroSuite compiler from EnviroSuite source files. This is to be distinguished from prior initial results, which reported some GloMoSim simulations.

REFERENCES

Abdelzaher, T., Blum, B., Cao, Q., Evans, D., George, J., George, S., He, T., Luo, L., Son, S., Stoleru, R., Stankovic, J., and Wood, A. 2004. Envirotrack: Towards an environmental computing paradigm for distributed sensor networks. In *ICDCS '04: Proceedings of the International Conference on Distributed Computing Systems*.

Azevedo, S. G. and McEwan, T. E. 1996. Micropower impulse radar. *Science and Technology Review*.

Birrell, A. D. and Nelson, B. J. 1984. Implementing remote procedure calls. *ACM Trans. Comput. Syst. 2,* 1, 39–59.

Blum, B., Nagaraddi, P., Wood, A., Abdelzaher, T., Son, S., and Stankovic, J. 2003. An entity maintenance and connection service for sensor networks. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*. ACM Press, New York, NY, USA, 201–214.

Boulis, A., Han, C.-C., and Srivastava, M. B. 2003. Design and implementation of a framework for efficient and programmable sensor networks. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*. ACM Press, New York, NY, USA, 187–200.

Cheong, E., Liebman, J., Liu, J., and Zhao, F. 2003. Tinygals: a programming model for event-driven embedded systems. In *SAC '03: Proceedings of the 2003 ACM symposium on Applied computing*. ACM Press, New York, NY, USA, 698–704.

Cheong, E. and Liu, J. 2005. galsc: A language for event-driven embedded systems. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*. IEEE Computer Society, Washington, DC, USA, 1050–1055.

Dutta, P., Grimmer, M., Arora, A., Bibyk, S., and Culler, D. 2005. Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In *IPSN '05: Proceedings of the Fourth International Conference on Information Processing in Sensor Networks*.

Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E., and Culler, D. 2003. The nesc language: A holistic approach to networked embedded systems. In *PLDI '03: Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation*. ACM Press, New York, NY, USA, 1–11.

Gu, L., Jia, D., Vicaire, P., Yan, T., Luo, L., Tirumala, A., Cao, Q., Stankovic, J. A., Abdelzaher, T., and Krogh, B. 2005. Lightweight detection and classification for wireless sensor networks in realistic environments. In *Sensys*. San Diego, CA.

Gummadi, R., Gnawali, O., and Govindan, R. 2005. Macro-programming wireless sensor networks using kairos. In *DCoSS*.

He, T., Krishnamurthy, S., Stankovic, J. A., Abdelzaher, T., Luo, L., Stoleru, R., Yan, T., Gu, L., Hui, J., and Krogh, B. 2004. Energy-efficient surveillance system using wireless sensor networks. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*. ACM Press, New York, NY, USA, 270–283.

Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K. 2000. System architecture directions for networked sensors. In *ASPLOS-IX: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*. ACM Press, New York, NY, USA, 93–104.

Intanagonwiwat, C., Govindan, R., and Estrin, D. 2000. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM Press, New York, NY, USA, 56–67.

Levis, P. and Culler, D. 2002. Mat: A tiny virtual machine for sensor networks. In *ASPLOS-X: Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*.

Levis, P., Lee, N., Welsh, M., and Culler, D. 2003. Tossim: accurate and scalable simulation of entire tinyos applications. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM Press, New York, NY, USA, 126–137.

Li, S., Lin, Y., Son, S. H., Stankovic, J., and Wei, Y. 2004. Event detection services using data service middleware in distributed sensor networks. *Telecommunication Systems, Special Issue on Information Processing in Sensor Networks 26,* 2-4.

Liu, J., Chu, M., Liu, J., Reich, J., and Zhao, F. 2003. State-centric programming for sensor-actuator network systems. *Pervasive Computing, IEEE 2,* 4, 50–62.

Liu, J., Liu, J., Reich, J., Cheung, P., and Zhao, F. 2003. Distributed group management for track initiaition and maintenance in target localization applications. In *IPSN '03: Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks*.

Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. 2002. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev. 36,* SI, 131–146.

Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. 2003. The design of an acquisitional query processor for sensor networks. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. ACM Press, New York, NY, USA, 491–502.

Madden, S. R., Franklin, M., Hellerstein, J., and Hong, W. 2005. Tinydb: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems 30,* 1 (March).

Microsoft. 1994. Ole2 programmers reference.

Newton, R., Arvind, and Welsh, M. 2005. Building up to macroprogramming: An intermediate language for sensor networks. In *IPSN '05: Proceedings of the Fourth International Conference on Information Processing in Sensor Networks*.

U. C. Berkeley. 2005. the motes. `http://www.tinyos.net/scoop/special/hardware#mica`.

Vinoski, S. 1997. Corba: Integrating diverse applications within distributed heterogeneous environments. *IEEE Communications Magazine 32,* 2 (February), 46–55.

Welsh, M. and Mainland, G. 2004. Programming sensor networks using abstract regions. In *NSDI '04: Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation*.

Whitehouse, K., Sharp, C., Brewer, E., and Culler, D. 2004. Hood: a neighborhood abstraction for sensor networks. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*. ACM Press, New York, NY, USA, 99–110.

Yao, Y. and Gehrke, J. 2002. The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec. 31,* 3, 9–18.

Received October 2004; revised May 2005; accepted September 2005

Models and Solutions for Radio Irregularity in Wireless Sensor Networks

GANG ZHOU, TIAN HE, SUDHA KRISHNAMURTHY, JOHN A. STANKOVIC
Department of Computer Science
University of Virginia

In this paper, we investigate the impact of radio irregularity on wireless sensor networks. Radio irregularity is a common phenomenon which arises from multiple factors, such as variance in RF sending power and different path losses depending on the direction of propagation. From our experiments, we discover that the variance in received signal strength is largely random; however, it exhibits a continuous change with incremental changes in direction. With empirical data obtained from the MICA2 and MICAZ platforms, we establish a radio model for simulation, called the Radio Irregularity Model (RIM). This model is the first to bridge the discrepancy between spherical radio models used by simulators and the physical reality of radio signals. With this model, we investigate the impact of radio irregularity on several upper layer protocols, including MAC, routing, localization and topology control. Our results show that radio irregularity has a relatively larger impact on the routing layer than the MAC layer. It also shows that radio irregularity leads to larger localization errors and makes it harder to maintain communication connectivity in topology control. To deal with these issues, we present eight solutions to deal with radio irregularity. We evaluate three of them in detail. The results obtained from both the simulations and a running testbed demonstrate that our solutions greatly improve system performance in the presence of radio irregularity.

Categories and Subject Descriptors: C.2.1 [**Computer Communication Network**]: Network Architecture and Design; I.6 [**Computer Methodologies**]: Simulation and Modeling

General Terms: Design, Algorithms, Measurement, Performance, Experimentation
Additional Key Words and Phrases: Sensor networks, wireless communication, radio irregularity, sending power, path loss, link asymmetry, packet loss, localization, topology control

1. INTRODUCTION

Radio irregularity is a common and non-negligible phenomenon in wireless sensor networks. It results in irregularity in radio range and variations in packet loss in different directions, and is considered as an essential reason for asymmetric links as viewed by upper layers in the protocol stack. Several empirical studies [Ganesan et al. 2002][Woo et al. 2003][Zhao and Govindan 2003][Cerpa et al. 2003] on the Berkeley mote platform have shown that the radio range varies significantly in different directions and the percentage of asymmetric links in a system varies depending on the average distance between nodes.

The impact of radio irregularity on protocol performance can be investigated through a running system. However, few researchers have actually pursued this direction, because of two reasons: First, the complexity and cost of performance evaluations on a running system escalate, when sensor networks scale up to thousands or more nodes. Second, repeatable results of radio performance are extremely hard to obtain from uncontrolled environments, hence leading to difficulties in system tuning and performance evaluation. As a result, simulation techniques are used as an efficient alterative to evaluate protocol performance. Unfortunately, most existing simulations do not take radio irregularity, a common phenomenon in wireless communication, into account. The spherical radio patterns assumed by simulators such as [Zeng et al. 1998] may not approximate real radio properties well enough and hence may lead to an inaccurate estimation of application performance.

Several researchers [Ganesan et al. 2002][Woo et al. 2003][Zhao and Govindan 2003][Cerpa et al. 2003] have already shown extensive evidence of radio irregularity in wireless communication. Their main focus is to observe and quantify such phenomena. This paper is distinguished from the previous ones for the initiative in bridging the gap between spherical radio models used by simulators and the physical reality of radio signals. We first verify the presence of radio irregularity using empirical data obtained from MICA2 and MICAZ platforms. The results demonstrate that the radio pattern is largely random; however, it exhibits a continuous change with incremental changes in direction. Based on experimental data, a radio model for simulations, called the Radio Irregularity Model (RIM), is formulated. RIM takes into account both the anisotropic properties of the propagation media and the heterogeneous properties of devices.

With the help of the RIM model, we explore the impact of radio irregularity on MAC, routing, localization and topology control performance. Among the protocols we evaluate, we find that radio irregularity has a significant impact on routing, localization and topology control protocols, but a relatively small impact on the MAC protocols. We also find that location-based routing protocols, such as Geographic Forwarding (GF) [Karp 2000] perform worse in the presence of radio irregularity than on-demand protocols, such as AODV [Perkins and Royer 1999] and DSR [Johnson and Maltz 1996]. We propose several potential solutions to deal with radio irregularity in wireless sensor networks. We evaluate the Symmetric Geographic Forwarding solution in simulation, and implement the Asymmetry Detection Service as well as the Bounded Distance Forwarding solution in running systems with 27$\sim$60 MICA2 devices. Our results illustrate that our solutions do succeed in alleviating the performance penalties due to radio irregularity.

The rest of this paper is organized as follows: we briefly analyze the causes and impact of radio irregularity in Section 2. In Section 3, we describe experimental data collected from the Berkeley mote platform and make some general conclusions about radio irregularity. Based on these conclusions, we propose the RIM radio model in Section 4. We then use the RIM model in simulations to analyze the impact of radio irregularity on MAC protocols in Section 5, routing protocols in Section 6, localization protocols in Section 7 and topology control protocols in Section 8. Solutions to deal with radio irregularity are proposed and evaluated in Section 9. Finally, we conclude the paper in Section 10.

2. ANALYSIS OF RADIO IRREGULARITY

In this section, we first identify the causes of radio irregularity, and then briefly discuss the impact of irregularity on the different protocol layers.

2.1 Causes of Radio Irregularity

Radio irregularity is caused by two categories of factors: devices and the propagation media. Device properties include the antenna type (directional or omni-directional), the sending power, antenna gains (at both the transmitter and receiver), receiver sensitivity, and the Signal-Noise-Ratio (SNR) threshold. Media properties include the media type, the background noise and some other environmental factors, such as the temperature and obstacles within the propagation media.

In general, the radio irregularity is caused by the anisotropic properties of the propagation media and the heterogeneous properties of devices. Among all these factors, we focus on the anisotropic path losses and the differences in signal sending power, which are commonly regarded as the key causes of radio irregularity.

—*Anisotropic Path Losses*: The variance in the signal path loss is one of the major causes of radio irregularity. When a signal propagates within a medium, it may be reflected, diffracted, and scattered [Shankar 2001]. Reflection occurs when an electromagnetic signal encounters an object, such as a building, that is greater than the signal's wavelength. Diffraction occurs when the signal encounters an irregular surface, such as a stone with sharp edges. Scattering occurs when the medium through which the electromagnetic wave propagates contains a large number of objects smaller than the signal wavelength. The medium is normally different in different directions. Consequently, radio propagation exhibits anisotropic patterns in most environments.
Another significant reason for anisotropic path loss is hardware differences. A node may not have the same antenna gain along all propagation directions, possibly due to hardware manufacturing. Hence, the anisotropic antenna gain of each node also contributes to the anisotropic path loss.

—*Heterogeneous Sending Powers*: Sensor devices may transmit RF signal at different sending powers, even though they are the same kind of devices. This difference may arise from some random factors during the manufacture of sensor devices. In addition, after the sensor devices are deployed, the batteries of different sensor devices deplete at different rates, due to different workloads and different environments in which they are deployed. Heterogeneous sending powers result in variable communication ranges, and cause anisotropic connectivity.

Environment and hardware differences are the two major sources of radio irregularity, and we propose a radio model to simulate these two factors. We are also aware that there are methods to adjust for differences in output power, noise, and manufacturing differences. Readers can refer to [Whitehouse and Culler 2002][Hoff and Azuma 2000][Hightower et al. 2000] for details. Our focus of this paper is to simulate the hardware differences, rather than to calibrate hardware differences.

2.2 Impact of Radio Irregularity

Radio irregularity is a non-negligible phenomenon in wireless systems. It's an essential reason for asymmetric radio interference and asymmetric links in upper layers. It can directly or indirectly affect many aspects of upper layer performance.

Asymmetric radio interference between neighboring nodes affects the correctness of MAC layer functions. For example, in the presence of radio irregularity, a node might not be able to successfully reserve the wireless channel through RTS and CTS handshaking, because those neighboring nodes of the receiver, which cannot hear the CTS control packet, might disrupt the receiving node. So, radio irregularity increases the chance of channel reservation failure and reduces the delivery ratios of data frames at the MAC layer.

Radio irregularity can also affect the performance and even correctness of networking protocols such as [He et al. 2003][Johnson and Maltz 1996][Karp and Kung 2000][Karp 2000]. For example, link asymmetry is one of the ways in which radio irregularity manifests itself at the higher layer. Link asymmetry has an adverse impact on protocols that use path-reversal techniques to establish an end-to-end connection.

Actually, the impact of radio irregularity is not only confined to the MAC and routing layers, radio irregularity also influences other protocols, such as localization, sensing coverage and topology control protocols.

Localization protocols such as DV-HOP [Niculescu and Nath 2003] and Centroid [N. Bulusu and Estrin 2000] assume a spherical radio range. The study in [He et al. 2003] shows that the performance of such protocols degrades when the radio range becomes irregular. The sensing coverage scheme in [Yan et al. 2003] assumes that sensing and communication ranges are spherical. In the presence of radio irregularity, they might not be able to guarantee full coverage and blind areas would occur. The topology control scheme in GAF [Xu et al. 2001] builds a communication mesh based on the assumption of a spherical range. This might lead to network partition in the presence of a non-spherical range. We note that some other topology control protocols, such as ASCENT [Cerpa and Estrin 2002] and Span [Chen et al. 2001] do not depend on such an assumption, however, performance evaluations of those protocols considering radio irregularity would be interesting future work.

In the rest of the paper, we evaluate the impact of radio irregularity on many upper layer protocols, including the MAC layer, the routing layer, localization protocols and topology control protocols.

3. RADIO IRREGULARITY IN REALITY

We conduct several experiments[1] to study the irregularity of the radio using MICA2 motes, and in this section we discuss some of the experimental results we obtain from an outdoor environment. Our results confirm that radio propagation is largely anisotropic and exhibits a continuous variation with incremental changes in direction.

3.1 Experimental Setup

We use a pair of MICA2 motes for our experiments. One of the motes periodically transmits probing beacons and the other mote samples its ADC port while receiving these beacons. The ADC reads the signal on the analog pin of the Chipcon transceiver [ChipconCC1000] and converts it into a 10-bit voltage value. The voltage reading is mapped into the received signal strength in dBm according to the specification in [ChipconCC1000]. All experiments are conducted in an open parking lot near a building, and all devices are equipped with whip antennae with the length of a quarter of the radio (433MHz) wavelength.

[1]This work is proposed to study and simulate the degree of radio irregularity, not the exact radio pattern. Readers can refer to [RF Chamber] on how to use an RF chamber to measure highly accurate radio patterns in special labs.

3.2 Experimental Results

In this section, we demonstrate the presence of radio irregularity using three different metrics: 1) the received signal strength, 2) the packet reception ratio and 3) the communication range.

3.2.1 *Anisotropic Signal Strength.* In the first experiment, the receiver is placed 10 feet away from the sender (both on ground) and the received signal strength is measured in four different geographical directions by sampling 100 beacons received in each direction.

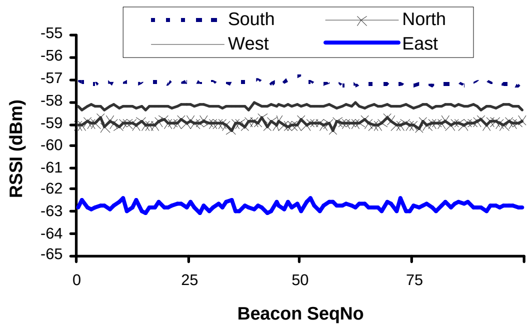
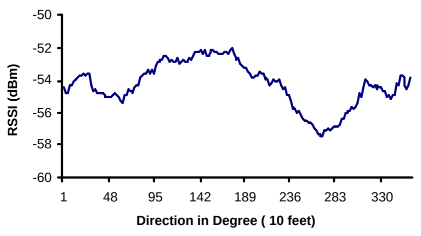


Fig. 1. Signal Strength over Time in Four Directions

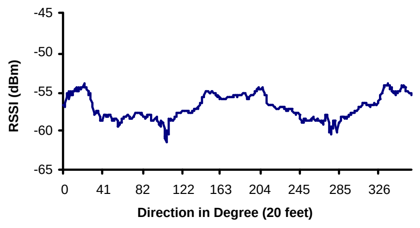
Figure 1 shows that the received signal strength in each direction is relatively stable over time (The small variance comes from the fading effect [Shankar 2001]). However, the signal strength received in the south is much higher than that received in the east, although nodes have the same distance from the sender. We also measure the variation of signal strength with the changes in the angular direction of the receiver with respect to the sender. Figure 2 shows the variation of the received signal strength as a function of the angular direction with respect to the sender, when the distance between the sender and receiver is 10 feet and 20 feet, respectively. These results show that the received signal strength varies continuously[2] with the direction. In other words, incremental changes in direction result in incremental variation in the received signal strength.

3.2.2 *Anisotropic Packet Loss Ratio.* Figure 3 shows how the packet reception ratio varies in different directions. When the sender and receiver are placed 10 feet apart, the packet reception ratio is nearly 100% in all the directions, because the signal is still strong in all the directions. However, when they are placed 20 feet apart, there is a 90% packet loss in the east direction. This result is consistent with the results shown in Figure 1, which demonstrates that the received signal strength measured in the east is lower than that in the other three directions.

[2]We call the variation continuous if and only if the maximum received signal strength percentage variance per unit degree change in the direction of radio propagation is smaller than 0.05.



(a) Measured at 10 feet



(b) Measured at 20 feet

Fig. 2. Signal Strength Values in Different Directions

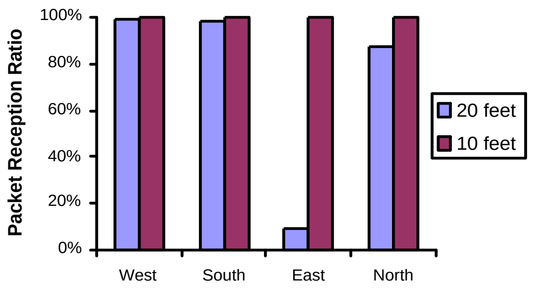


Fig. 3. Anisotropic Packet Reception

3.2.3 *Anisotropic Radio Range.* Another aspect in which we demonstrate the irregularity is to show that the communication range of a mote is not uniform in all directions. In the experiment, we fix the received signal strength threshold at -55.5 dBm and -59 dBm, respectively. Then with such thresholds, we measure the communication ranges in different directions. Figure 4 shows the communication range of a mote as the receiver direction varies from degree 0 to degree 359. The range map shown in Figure 4 is another confirmation of radio irregularity in a wireless medium.

3.2.4 *Range Irregularity with Varying Sending Power.* We also investigate the received signal strength when the sending power varies due to different battery status and hardware differences. In Figure 5(a), we use the same sender and receiver, placed 10 feet apart. We change the batteries at the sender side each time. The result indicates that different battery status at the same sender can affect the received signal strength. In Figure 5(b), we use the same batteries, but in different senders each time. The same receiver is used, placed 10 feet apart from the sender. The result shows that different senders with the same batteries can also affect the received signal strength.

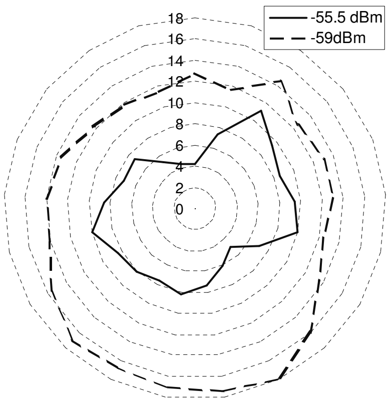
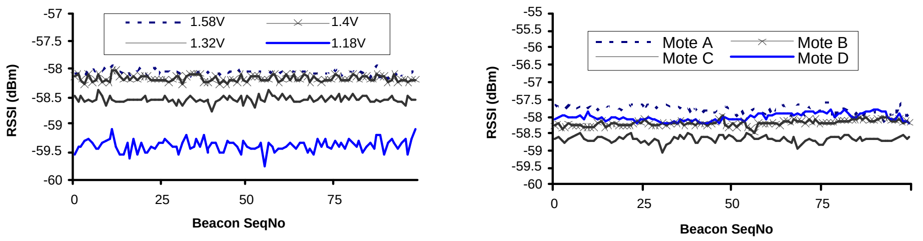


Fig. 4. Anisotropic Range



(a) One mote with different battery status (b) Different motes with the same battery status

Fig. 5. Radio Irregularity with Sending Powers

3.3 Summary of Experimental Results

From the experimental results, we infer that the radio of sensor devices has the following main properties:

(1) *Anisotropy:* The radio signal from a transmitter has different path losses [3] in different directions (Figure 1 and Figure 2).

(2) *Continuous variation:* The signal path loss varies continuously with incremental changes of the propagation direction from a transmitter (Figure 2 and Figure 4).

(3) *Heterogeneity:* Differences in hardware property and battery status lead to different signal sending powers, hence different received signal strengths (Figure 5).

[3]Figure 2 shows that the received signal strength varies greatly in different propagation directions, while Figure 1 tells us that the fading effect does not cause much variation in the received signal strength for a specified direction. Accordingly, it is reasonable to believe that different path losses in different directions are the main reason for the received signal strength variations in different propagation directions.

4. MODELING RADIO IRREGULARITY

As we have shown in our experiments as well as demonstrated in other research results [Ganesan et al. 2002][Woo et al. 2003][Zhao and Govindan 2003][Cerpa et al. 2003], radio irregularity is a common phenomenon in wireless sensor networks. Therefore, it is essential for simulations of wireless systems to capture such effects. This section describes our effort to model such a phenomenon in simulation environments.

4.1 Isotropic Radio Models

In isotropic radio models, the received signal strength is usually represented with the following formula:

$$Received\,Signal\,Strength \;=\; Sending\,Power - Path\,Loss + Fading \qquad (1)$$

The $SendingPower$ of a node is determined by the battery status and the type of transmitter, amplifier and antenna. $PathLoss$ describes the signal's energy loss as it travels to the receiver. Many models are used to estimate the $PathLoss$, such as the free-space propagation model, the two-ray model and the Hata model [Shankar 2001]. All these models are isotropic, meaning that the signal attenuates exactly the same in all directions. However, our experience as well as results obtained by others [Ganesan et al. 2002][Woo et al. 2003][Zhao and Govindan 2003][Cerpa et al. 2003] all indicate that the isotropic models do not hold well in practice.

4.2 Radio Irregularity Model (RIM)

The RIM model we propose here is an extension to isotropic radio models. It enhances isotropic radio models by approximating three main properties of radio signals: anisotropfy, continuous variation and heterogeneity, as we summarized in Section 3.3. These properties are normally ignored by previous isotropic radio models.

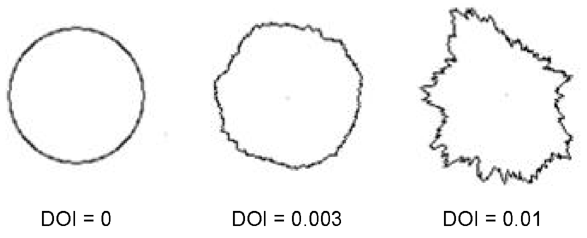
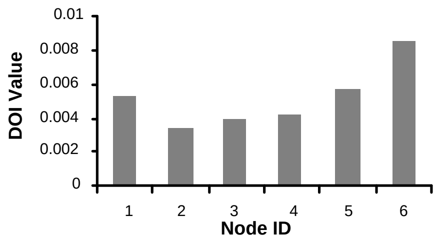
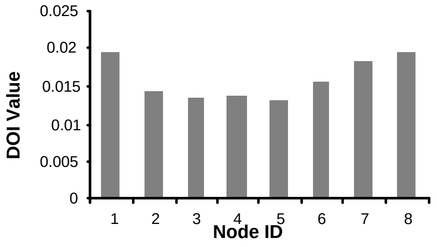


Fig. 6. Degree of Irregularity

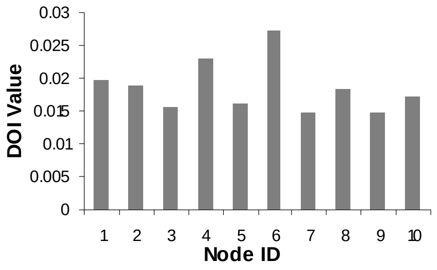
To denote the irregularity of a radio pattern, the parameter DOI (degree of irregularity) is introduced into the RIM model. The DOI parameter is defined as *the maximum path loss percentage variation per unit degree change in the direction of radio propagation.* As shown in Figure 6, when the DOI is set to 0, there is no range variation, and the communication range is a perfect sphere. However, when we increase the DOI value, the communication range becomes more and more irregular.



(a) A Short Antenna MICA2 System



(b) A Long Antenna MICA2 System



(c) A MICAZ System

Fig. 7. DOI Values from MICA2 Experiments

The RIM model is a general radio model which can default to the isotropic model when the DOI value is 0. The RIM model is established based on data from real sensor devices. It is a hybrid approach, which introduces real data (DOI value) into simulations, so that the radio irregularity pattern in reality can be approximated well. We repeat the experiments shown in Figure 2 on 6 MICA2 devices in a vehicle tracking system, and calculate the corresponding DOI values, according to the DOI definition. The experimental results depicted in Figure 7(a) inform that the variances of the received signal strength with incremental changes in directions are small, which validates our conclusion about continuous variation.

In order to investigate possible DOI variance in different types of devices, we repeat the experiments with new MICA2 motes that have double length antennae, 1/2 the radio wavelength. Each node's DOI value is calculated and presented in Figure 7(b), which shows comparatively larger DOI values than those in Figure 7(a). This is because the new MICA2 devices have longer and more powerful antennae that amplify existing hardware differences.

To explore the RIM model's applicability across platforms and radios, we repeat the experiments with MICAZ [CROSSBOW] motes, each of which has a whip antenna of 1/4 radio wavelength. MICAZ motes use the CC2420 radio [ChipconCC2420], which follows the IEEE 802.15.4 [IEEE 802.15.4 1999] standard and is different from the CC1000 [ChipconCC1000] radio used in MICA2 motes. As Figure 7(c) illustrates, the RIM model applies in MICAZ devices and the measured DOI values have the range from 0.015 to 0.03. Comparing Figure 7(a) with Figure 7(c), we observe that MICAZ motes exhibit a higher degree of radio irregularity than MICA2 motes, when both types of devices use 1/4 wavelength whip antennae. This is because the CC2420 radio in MICAZ is more powerful than the CC1000 radio in MICA2.

4.2.1 *Anisotropy Property in the RIM Model.* Many models are used to estimate path loss, such as the free-space propagation model, the two-ray model and the Hata model [Shankar 2001]. These models are isotropic in the sense that the path losses in different directions are the same. To reflect the two main properties of radio irregularity, namely anisotropy and continuous variation, we adjust the value of path loss models in Equation 1 based on DOI values, resulting in the following formula:

$$\begin{aligned} Received\,Signal\,Strength = \; & Sending\,Power - DOI\,Adjusted\,Path\,Loss + Fading \\ \text{where} \quad & DOI\,Adjusted\,Path\,Loss = Path\,Loss \times K_i \end{aligned} \tag{2}$$

Here K_i is a coefficient to represent the difference in path loss in different directions [4]. Specifically, K_i is the i^{th} degree coefficient, which is calculated as follows:

$$K_i = \begin{cases} 1 & \text{if } i = 0 \\ K_{i-1} \pm Rand \times DOI & \text{if } 0 < i < 360 \wedge i \in N \end{cases}$$
$$\text{where } \mid K_0 - K_{359} \mid \leq DOI \tag{3}$$

We can generate 360 K_i values for the 360 different directions, based on Equation 3, by randomly fixing a direction as the starting direction represented by $i = 0$. For the direction which does not have an integer value of angle from the start direction, we interpolate the K_i value based on the values of the two adjacent directions which have integer angles from the starting direction.

$$\begin{aligned} K_i \; = \; & K_s + (i - s) \times (K_t - K_s) \\ & \text{where } s = \lfloor i \rfloor \wedge t = \lceil i \rceil \bmod 360 \wedge 0 < i < 360 \wedge i \notin N \end{aligned} \tag{4}$$

The statistical analysis of our experimental data indicates that the variance of received signal strength (mainly because of path loss variation since Figure 1 shows that fading is pretty small) in different directions fits the Weibull [Devore 1982] distribution. The Weibull distribution can be used to model natural phenomena such as variation of wind speed, scattering of radiation, etc. The Rayleigh distribution, which is commonly used for modeling multi-path fading in wireless communication, is a special case of the Weibull distribution. Analysis details are provided in Appendix A. In Equation 3, we generate a random number according to the Weibull distribution.

We conduct experiments to evaluate the RIM model's ability to generate radio patterns that have specified degree of radio irregularity. We input to the RIM model the degree of irregularity value DOI=0.01821 [5], which is measured in a MICA2 device and illustrated as the value of column 7 in Figure 7(b). Then the

[4]Coefficient K_i is used to adjust the path loss in a specified direction. In this specified direction, we can also propose to adjust K_i's value in a small range based on the distance the receiver is from the transmitter, because when the distance increases, the signal travels in a larger environment area. So it may suffer different reflection, diffraction and scattering and has different path loss. We leave this as future work.

[5]Here, we keep 5 digits after the point to illustrate how accurate the RIM model is in simulating the degree of radio irregularity. As shown in later sections, maintaining 3 digits after the point is good enough for performance evaluation.

DOI values of generated radio patterns from the RIM model are calculated and compared with the input DOI value. Figure 8 presents the result.

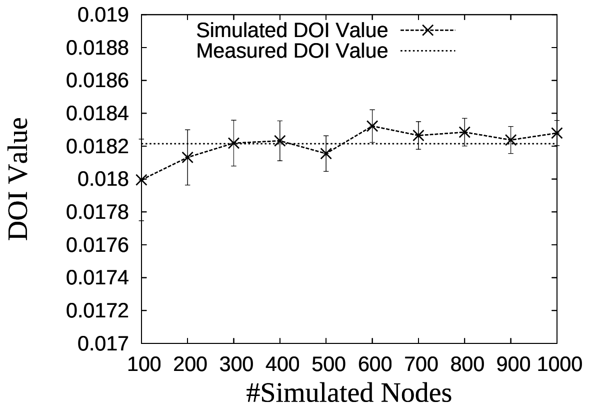


Fig. 8. Degree of Irregularity Evaluation (with 90% Confidence Intervals)

As Figure 8 illustrates, the RIM model is very accurate in simulating the degree of radio irregularity. When the simulation uses 100 nodes, the length of the 90% confidence interval is less than 0.0005, which is only 2.7% compared with the input DOI value 0.01821. In addition, according to performance evaluation (Figures 12, 18, 21, 22, 23 and 25) in later sections of this paper, 0.0005 is too small to distort the performance evaluation.

Moreover, with the increase of the sample space, the RIM model converges towards the target DOI value. For example, when simulation uses 1000 nodes, the length of the 90% confidence interval decreases to 0.00015. Compared with the input value 0.01821, it only has 0.8% variation. The RIM model also gives more accurate average DOI values with the increase in the number of simulated nodes. The average DOI is 0.01799 when 100 nodes are used, and it becomes 0.01827 when 1000 nodes are used, which is much closer to the target value 0.01821.

Accordingly, statistically speaking, the RIM model has the ability to simulate the degree of radio irregularity.

4.2.2 *Heterogeneity Property in the RIM Model.* Due to different battery status and hardware differences, the received signal strength can be different from two sending nodes of the same type in the same experimental setting. In RIM, we use the variance of signal sending power to account for such a difference. We introduce the second parameter named VSP (Variance of Sending Power), which is defined as *the maximum percentage variance of the signal sending power among different devices.* The new signal sending power is modelled by the following equation:

$$VSP\,Adjusted\,Sending\,Power \;=\; Sending\,Power \times (1 + Rand \times VSP) \quad (5)$$

In Equation 5, we assume that the variance of sending power fits the normal distribution, which is broadly used to estimate battery lifetime distribution [Battery Lifetime] and to simulate hardware differences [Devore 1982].

With the two parameters: DOI and VSP, the RIM model can be formulated as follows:

$$\begin{aligned} Received\,Signal\,Strength &= VSP\,Adjusted\,Sending\,Power \\ &\quad -DOI\,Adjusted\,Path\,Loss + Fading \end{aligned} \tag{6}$$

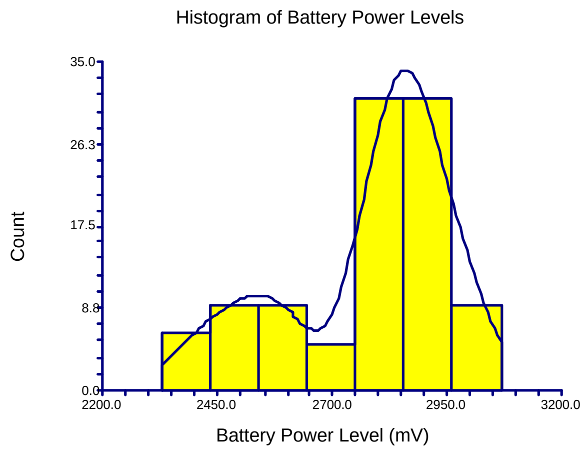


Fig. 9. Battery Power Level Snapshot in the VigilNet System

We are also aware that the default normal distribution we implemented in RIM is not a universal solution for all sensor network systems. The normal distribution may work well for initially deployed systems, which are equipped with new batteries. However, with respect to a system that has been used for a long time, the battery power level may not fit the normal distribution.

We take a snapshot of all battery power levels in the VigilNet System [He et al. 2004], and plot the distribution in Figure 9.

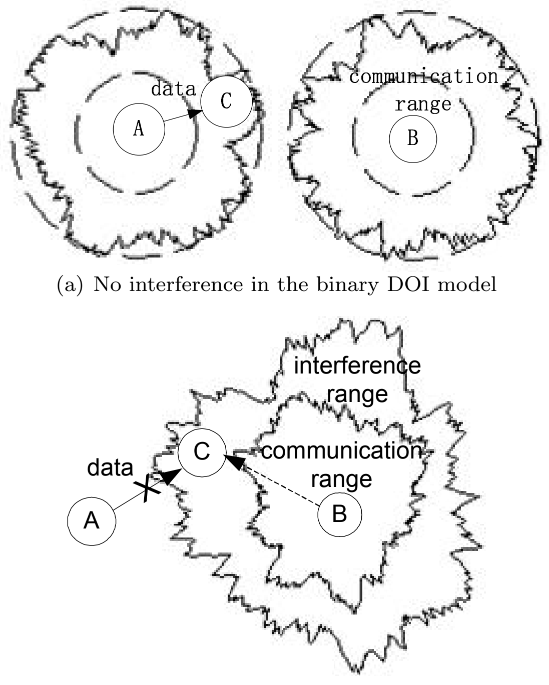
As shown in Figure 9, the battery power level does not fit a normal distribution well. This is because in the system design, all nodes are divided into two groups: sentry nodes and non-sentry nodes. Sentry nodes are supposed to work all the time and non-sentry nodes are put to sleep to save power. Non-sentry nodes are only awakened when an important event happens. Accordingly, this sentry design leads to more energy consumption for sentry nodes and less energy consumption for non-sentry nodes. To simulate this non-normal distribution, readers are encouraged to replace the normal random number generator in the RIM model with their own random number generators, to reflect the power level distribution reality in their own systems, and there is no need to modify any other code in RIM.

4.2.3 *DOI Variance in a System.* From empirical data we collected in two MICA2 systems and a MICAZ system shown in Figure 7, we observe that sensor devices in a system may have different DOI values, depending on the hardware devices used and the deployment environment. It is not convenient to measure each node's DOI value in a large scale system and assign the measured DOI values to each node in simulation. In order to reflect this fact of DOI variance among different devices in a system, we introduce the third parameter VDOI (Variance of DOI), which is defined as *the maximum percentage variance of DOI values among different devices in a system*. We assume the DOI variance in a system fits the normal distribution. So with the distribution as well as the VDOI value, each node in the system can easily get a DOI value. In performance evaluation of this paper, we first set VDOI as 0 to observe system performance with different DOI values, and then set VDOI greater than 0 to investigate performance sensitivity to different VDOI values.

4.3 Comparison with a Binary DOI Model

The RIM model is motivated by a simple binary DOI (Degree of Irregularity) model briefly mentioned in the localization work [He et al. 2003]. In the binary DOI model, the DOI parameter was originally defined as the maximum *range variation* per unit degree change in the direction of radio propagation.

The DOI model assumes an upper and lower bound on signal propagation, which are depicted as the inner and outer dashed circles in Figure 10(a). Beyond the upper bound, all nodes are out of communication range; and within the lower bound, every node is guaranteed to be within the communication range. If the distance between a pair of nodes is between these two boundaries, three scenarios are possible: 1) symmetric communication, 2) asymmetric communication, and 3) no communication.



(a) No interference in the binary DOI model

(b) Interference in the RIM model

Fig. 10. Communication Interference

The binary DOI model is a good start to model signal irregularity. However, it does not model interference in real devices well. Since the DOI model is based on an absolute communication range, it assumes that within the inner range, the signal is very strong and can always be received correctly, while beyond the outer range there is no signal at all. This binary pattern is not true in reality. For example, in Figure 10(a), the DOI model assumes that there is no interference between nodes B and C.

However in reality, there are no such clear boundaries and the communications of nodes do interfere with each other. Different from the binary DOI model, the RIM model we propose takes the radio sending energy, the energy loss, the background noise, and the interference among different communication signals into account.

The difference can be further explained with an example. In Figure 10(b), the RIM model allows node B's signal to propagate beyond its communication range to reach node C, even though it is not strong enough for node C to receive it as a valid packet. This weak signal from node B acts as one source of background noise around node C. In this case, node C may not be able to receive packets from node A, if the received signal is not stronger than the product of the Signal-Noise-Ratio (SNR) threshold and background noise level of node C.

The DOI model only models an absolute range based on the distance and determines whether one node can hear another node only by comparing the distances between these two nodes with the sender's communication range. With such a binary decision, it can't deal with interference as we mentioned earlier.

The RIM model enhances the isotropic radio model and the DOI radio model, by combing the energy models and the DOI factor together. The original DOI concept is redefined by incorporating radio energy propagation. We note that RIM is a general radio model which can default to the isotropic model when the DOI value is 0. Also, it can default to the DOI model when there is no interference among nodes.

We need to clarify that the RIM model is not proposed to simulate the exact radio pattern. Instead, it is a general radio model to simulate the degree of radio irregularity. Given measured radio patterns from a real system, the values of DOI, VSP and VDOI can be calculated and configured in the RIM model. Then the RIM model can generate a specified number of radio patterns that have the same degree of radio irregularity. For a system that adopts hardware calibration schemes [Whitehouse and Culler 2002][Hoff and Azuma 2000][Hightower et al. 2000], the values for DOI, VSP and VDOI are configured smaller, according to the measured values from sampled devices, so that the reduced radio irregularity is simulated. In a similar way, the RIM model can be configured to simulate systems that consists of hardware with different transceivers.

The RIM model is proposed to account for the three main properties of radio irregularity: anisotropy, continuous variation and heterogeneity, as we summarized in Section 3.3. Currently, the experiments we present are conducted in a parking place with MICA2 and MICAZ devices. The exact radio patterns and the degree of radio irregularity may vary in different environments. We expect that the radio is more irregular in a harsher environment, such as in a wild forest. We also expect that the three main properties of radio irregularity still hold in a harsher environment, and the radio irregularity can be simulated by choosing larger DOI values. This is subject to further confirmation with experiments in different environments.

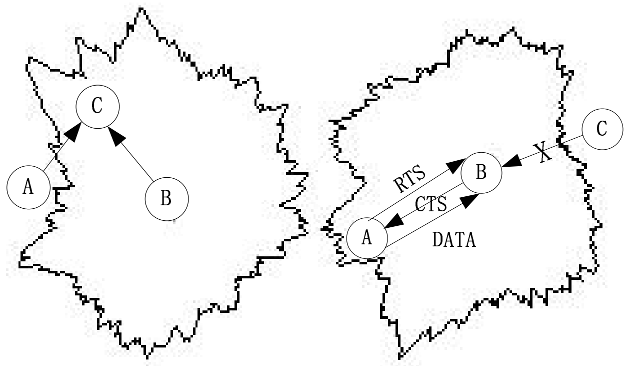
In the following sections, we use the RIM model as a simulation tool to help explore the impact of radio irregularity on MAC, routing, localization and topology control protocols.

5. IMPACT ON MAC LAYER

In this section, we first analyze how operations in the MAC layer are affected by radio irregularity. We then quantify the degree of MAC performance degradation in the presence of radio irregularity.

5.1 Logical Analysis of the Impact

Most contention-based MAC protocols are based on carrier sensing or handshaking techniques. In this section, we analyze the impact of radio irregularity from the technical point of view.



(a) Carrier Sensing (b) Handshaking

Fig. 11. Impact on MAC Protocols

(1) *Impact on Carrier Sensing:* Radio irregularity increases the chance for MAC protocols that use the carrier sensing technique to get involved in the hidden terminal problem. For example, in Figure 11(a), while node B is transmitting packets to node C, due to the irregularity, node A cannot detect the signal from node B, so node A senses a clear channel and starts to transmit packets. As a result, a collision happens at receiver C. This scenario does not occur if node B has a spherical radio range that covers node A so that A can sense node B's signal and will not send a packet to C and get corrupted. Typical protocols using the carrier sensing technique are CSMA [Kleinrock and Tobagi 1975], MACA [Karn 1990], MACAW [Bharghavan et al. 1994] and 802.11 DCF [IEEE 802.11 1999].

(2) *Impact on handshaking:* The handshaking technique is specially designed to resolve hidden and exposed terminal problems. However, they cannot resolve the hidden and exposed terminal problems due to asymmetry, which can be

Table I. Simulation Configuration One

TERRAIN	(150m X 150m)
Node Number	100
Node Placement	Uniform
Application	Many-to-one CBR Streams
Payload Size	32 Bytes
Routing Protocol	AODV, DSR, GF
MAC Protocol	CSMA, 802.11 (DCF)
Radio Layer	RADIO-ACCNOISE
Radio Model	RIM
Nominal Radio Range	40M
Radio Bandwidth	200Kb/s

produced by radio irregularity. This can be demonstrated in an example (Figure 11(b)). We assume that node A sends a RTS message to node B, and then node B responds with a CTS message to node A. Any node overhearing the CTS message is supposed to wait long enough for node A to send out the data packet. If node C can't hear the CTS message from node B while node B can hear node C, there will be a collision if node C sends data. Similar examples can be found for the exposed terminal case.

5.2 Quantitative Analysis of the Impact

We implemented the RIM model in the radio layer of GloMoSim [Zeng et al. 1998], a scalable discrete-event simulator developed by UCLA. We first describe our simulation configuration, and then evaluate the performance impact under different DOI and different VSP values, respectively.

This is not a media access control paper and we do not try to explore the impact of radio irregularity on all MAC protocols. We choose two typical MAC protocols, CSMA and 802.11 DCF, for case study, because they are popular protocols and also very typical protocols that use carrier sensing and RTS-CTS handshaking techniques. Readers can refer [Ye et al. 2002][Rajendran et al. 2003][Dam and k. Langendoen 2003][Woo and Culler 2001][Polastre et al. 2004] for more MAC protocols.

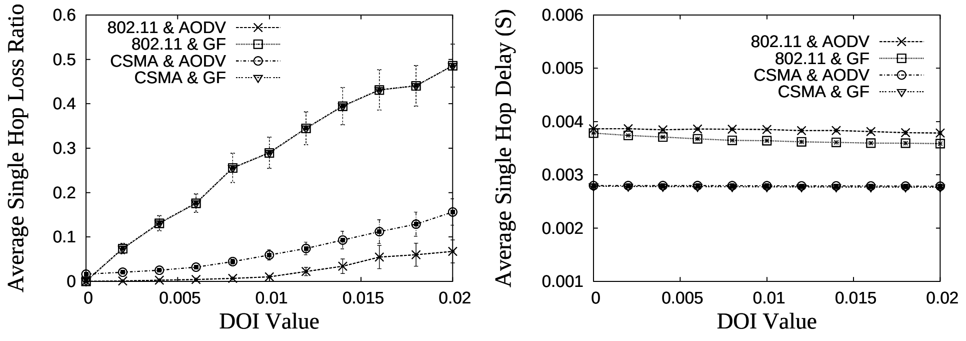
In the experiments, we use six CBR streams as the workload and set the CBR rate at a low rate, in order to isolate the effect of congestion and radio irregularity. Two metrics are used: 1) the loss ratio (number of packets lost / number of packets sent) and 2) the average single hop delay of received packets. We vary the DOI and VSP values separately in order to isolate and identify the impact individually. In each data value we present, we also give the corresponding 90% confidence interval.

In order to make our evaluation close to existing hardware proposed for use in wireless sensor network environments [CROSSBOW], we use the simulation configuration shown in Table I. In all experiments, we investigate the range of DOI values according to the experimental data obtained from MICA2 motes as shown in Figure 7.

5.2.1 *MAC Performance with Different DOI.* In this section, we set VDOI as 0, to evaluate the performance of MAC layer with different DOI values. In the next section, we investigate the performance sensitivity to DOI variance, by setting

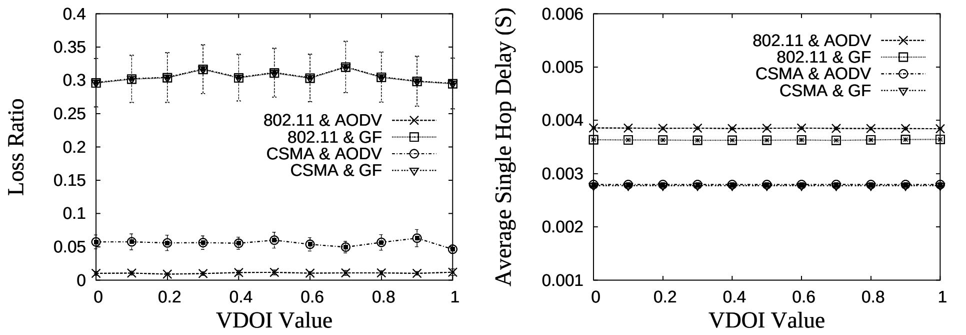
VDOI greater than 0.

In the initial setup, we use Geographic Forwarding (GF) in the routing layer and compare the MAC performance between 802.11 and CSMA. We found that the MAC loss ratio increases rapidly with the increase of DOI values (Figure 12(a)). However, 802.11 and CSMA yield roughly the same results. We realize that MAC performance can be strongly affected by routing, because an incorrect routing decision might lead to the failure at MAC layer. For instance, the routing layer designates that the MAC layer send a packet to a node that is out of reach. So we repeat the experiments with the AODV protocol as the routing layer. We find that the MAC loss ratio increases slightly with the increase of DOI values. Such a discrepancy is a strong indication that the radio irregularity has a much larger impact on routing protocols than MAC protocols. We explain this in more detail in Section 6.



(a) Loss Ratio vs. DOI (b) Average Single Hop Delay vs. DOI

Fig. 12. MAC Performance with Different DOI Values



(a) Loss Ratio vs. VDOI (b) Average Single Hop Delay vs. VDOI

Fig. 13. MAC Performance Sensitivity to Different VDOI Values

From Figure 12(b), we can see that with the increase of DOI values, the average single hop delay remains almost the same. The reason is that increasing the DOI value only increases the communication asymmetry, but not the congestion. This is also a confirmation that packet loss in Figure 12(a) is not due to congestion.

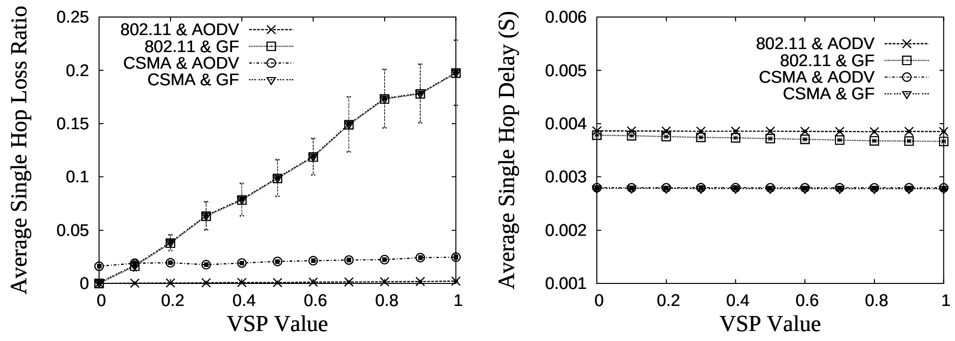
5.2.2 *Performance Sensitivity to Different VDOI.* This section is to explore whether DOI variance in a system has impact on the MAC performance. In the simulation, we set DOI as 0.01 and vary VDOI from 0 to 1, in steps of 0.1, and present the simulation results in Figure 13.

From Figure 13, we observe that when VDOI varies from 0 to 1, neither the average single hop loss ratio nor the average single hop delay varies much. The possible reason is that, statistically, while one portion of nodes have larger DOI values and hence more irregular radio, another portion of nodes will have smaller DOI values and hence less irregular radio. So their effects are cancelled by each other, and the system-wide MAC performance is not sensitive to different VDOI values.

5.2.3 *MAC Performance with Different VSP.* In this experiment, we set the DOI value to 0, which means that the radio range is isotropic. However, different VSP values make radio ranges different among nodes.

The results shown in Figure 14 are similar to the results shown in Figure 12, which we obtain by varying the DOI values. The average single hop delay remains almost the same, because the different sending powers only increase the degree of communication asymmetry, but not the congestion.

The loss ratio increases with the increase of VSP values because the irregularity results in more asymmetric links. The loss ratio when AODV is used is much lower than that when GF is used, because asymmetric links have a larger impact on GF than on AODV. This result indicates that varying the VSP values has a much larger impact on routing protocols than on MAC protocols, which is similar to the behavior we observed by varying the DOI values.



(a) Loss Ratio vs. VSP (b) Average Single Hop Delay vs. VSP

Fig. 14. MAC Performance with Different VSP Values

6. IMPACT ON ROUTING LAYER

In this section, we analyze and quantify the impact of radio irregularity on routing protocols. We first discuss three techniques that are widely used in most routing protocols: path-reversal, multi-round discovery, and neighbor discovery. Our analysis shows that both path-reversal and neighbor-discovery are greatly influenced by radio irregularity. However, the multi-round discovery technique is able to deal with radio irregularity, but with relatively high overhead. Our simulation results also show that radio irregularity has a great impact on Geographic Forwarding (GF), but a small impact on AODV and DSR.

6.1 Logical Analysis of the Impact

In this section, we study the influence of radio irregularity on path-reversal, multi-round discovery, and neighbor-discovery techniques. We also quantify this influence in two cases. In one case, path loss difference is the main reason of radio irregularity and link asymmetry, and in the second case, difference in radio sending power is the main reason.

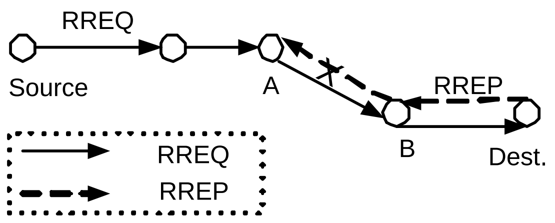


Fig. 15. Impact on Path-Reversal Technique

6.1.1 *Impact on Path-Reversal Technique.* Protocols that use path-reversal technique are built based on the assumption that if there is a path from node A to node B, there is also a reverse path from node B to node A. The path may consist of a single link or multiple links. Most on-demand routing protocols used in ad hoc networks such as AODV [Perkins and Royer 1999], DSR [Johnson and Maltz 1996], Direct Diffusion [Intanagonwiwat et al. 2000] and LAR [Ko and Vaidya 1998] depend on this technique.

Radio irregularity may result in asymmetric links and hence, it may have an adverse impact on protocols that use path-reversal techniques. For example, in Figure 15, node B can hear node A, but node A cannot hear node B. So even though there is a path from source S to destination D, we cannot assume that the reverse path from D to S exists. So during route discovery, if source S broadcasts a route request (RREQ) to discover the path to destination D, it may not be possible to deliver the reply (RREP) message to source S along the reverse path, even though node D replies to the request. In such a case, the route discovery fails.

The above analysis leads one to believe that it would be inappropriate to use any routing protocol that uses path-reversal in route discovery, such as AODV, DSR, DD and LAR, in an asymmetric environment, because they would have a very high loss ratio. However, the simulation results we present later show that AODV and DSR work reasonably well despite the asymmetric nature of communication. The reason is that in addition to path-reversal technique, these routing protocols also use the multi-round discovery, which is capable of dealing with asymmetry, but with a high overhead.

6.1.2 *Multi-Round Discovery Technique.* In AODV and DSR, the RREQ is broadcast towards the destination D. So node D receives RREQ messages from multiple paths, as shown in Figure 16. It chooses one of the many available paths to send the RREP message back to source S, according to some runtime configurable parameter, such as the RREQ arrival time, path load, or end-to-end delay of the path. If the reverse path does not exist, the RREP fails to arrive at sender S and the route discovery is repeated due to timeout. In the next attempt, thanks to the random nature of flooding, node D might receive a RREQ message from another path, which happens to be a symmetric connection.

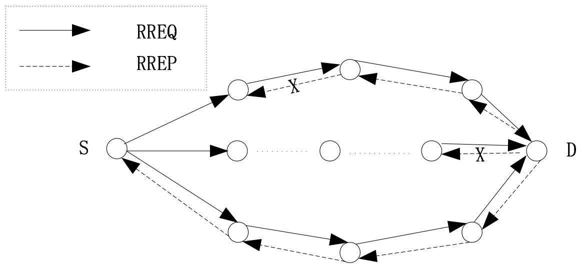


Fig. 16. Route Discovery Using Rediscovery Technique

The chance to establish a symmetric connection increases after retries. If there is no limitation on the number of retries, a symmetric path will sooner or later be discovered on the condition that such a path exists. We note that the rediscover technique provides a viable way to work around the effects of asymmetry, but with significant overhead. Also the path reinforcement scheme presented in [Intanagonwiwat et al. 2000] can reduce the impact of path-reversal technique, by continuously monitoring performances in multiple paths and reinforcing the best path.

6.1.3 *Impact on Neighbor Discovery Technique.* Many location-based routing protocols [He et al. 2003][Karp and Kung 2000][Karp 2000] use the neighbor discovery technique in order to maintain the neighborhood information. However, the neighbor discovery technique works well only if the links are symmetric. For example, in Figure 17, node A discovers its neighbors by receiving beacons. Node A might choose one of its neighbors, node B, C, or D for forwarding packets. However, if node A picks node B which is unable to hear node A, node B will never receive the packet forwarded by node A. If node A does not retry its transmission with the other neighbors, the transmission of the packet will fail. So the routing protocol based on the neighbor discovery technique is subject to failures when communication is asymmetric.

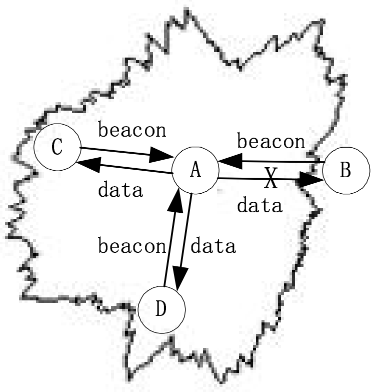


Fig. 17. Impact on Neighbor Table Technique

6.2 Quantitative Analysis of the Impact

In this section, we quantify the performance penalty of radio irregularity, through four sets of experiments. In each set, we measure four metrics: end-to-end (E2E) loss ratio, average E2E delay, number of control packets, and energy consumption.

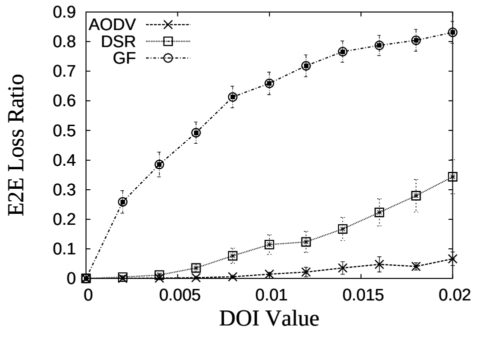
We first measure the routing performance with different DOI values, and then investigate the performance sensitivity to different VDOI values. In the third set of experiments, different VSP values are configured to explore the routing performance. In the fourth set of experiments, we take the GF routing protocol as an example, to investigate the performance changes when combinations of different DOI and VSP values are used.

Before we analyze the performance evaluation, we'd like to explain how GF [Karp 2000] works. In GF, each node beacons its ID and location periodically, so that each node can maintain up to date neighbor information. When the application requests to send a packet to a specified destination, GF compares the distance of each neighbor to the destination, and forward the data packet to the neighbor that is closest to to the destination. This routing strategy is repeated by all intermediate nodes that participate in packet forwarding, until the data packet is finally received by the destination.

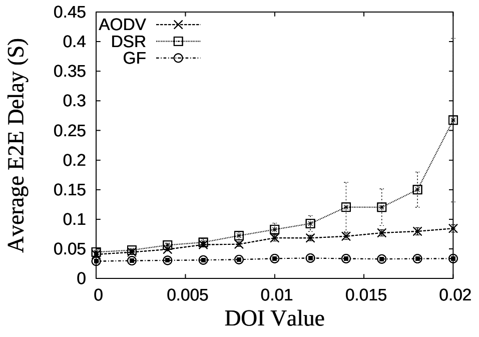
6.2.1 *Routing Performance with Different DOI.* In this section, we set VDOI to 0 to evaluate routing performance with different DOI values. In the following section, we set VDOI greater than 0 to investigate routing performance sensitivity to DOI variance.

Figure 18(a) shows that GF is greatly influenced by radio irregularity. It loses 84.5% packets when the DOI is 0.02. The reason is that according to the greedy forwarding rule, GF tends to choose a node near the border, which is more likely to have an asymmetric link with the sender. AODV and DSR perform well because they use multi-round discovery, exploring alterative paths to find a symmetric connection. However, they achieve a low loss ratio at the cost of increased overhead in control packets shown in Figure 18(c).

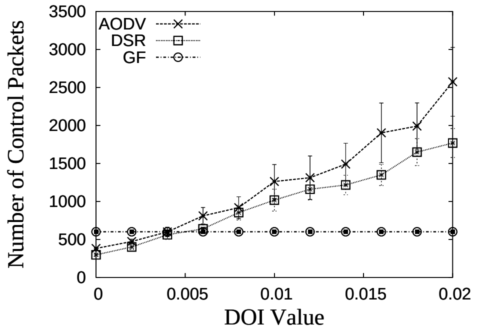
In Figure 18(b), the average E2E delay of DSR and AODV increases with the increase of DOI values. That is because more rounds of route discovery are needed



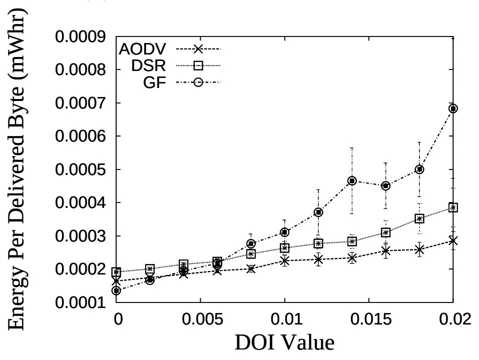
(a) E2E Loss Ratio vs. DOI



(b) Average E2E Delay vs. DOI



(c) Number of Control Packets vs. DOI



(d) Energy Consumption vs. DOI

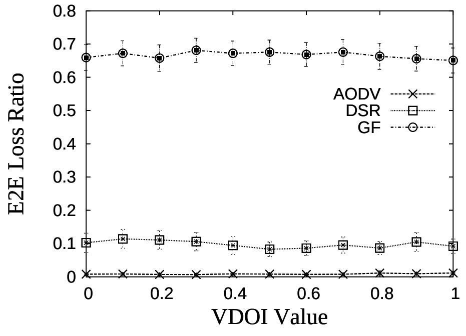
Fig. 18. Routing Performance with Different DOI Values

as the radio irregularity increases. In Figure 18(b) DSR has a higher delay than AODV, because the source routing technique in DSR adds the whole path in the header of data packets, which increases the transmission time. However, the E2E delay of GF remains the same because packets in GF either go through successfully or get dropped.

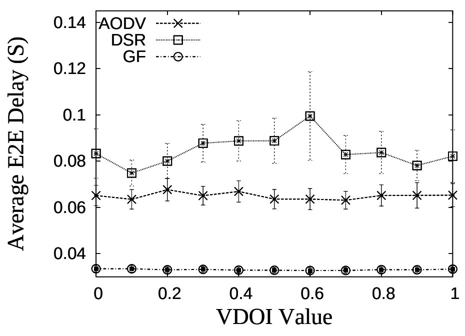
Figure 18(c) shows that while AODV and DSR need more control packets to do multi-round discovery, when the DOI value increases GF needs only a constant number of control packets for neighbor exchange.

Figure 18(d) presents the energy consumption normalized according to useful work completed. It is measured as the energy consumed for each successfully delivered end-to-end data byte. Figure 18(d) informs that AODV, DSR and GF all consume more energy to deliver a useful data byte through multiple hops, when the DOI value increases. This is because the increased radio irregularity leads to increased asymmetry links, which result in increased retransmission to deliver the same amount of useful data. Moreover, as shown in Figure 18(a), GF delivers less useful data than AODV and DSR, and DSR delivers less useful data than AODV, with the increase of DOI values. Accordingly, among the three routing protocols, GF is less energy efficient than AODV and DSR, and DSR is less energy efficient than AODV, as shown in Figure 18(d).

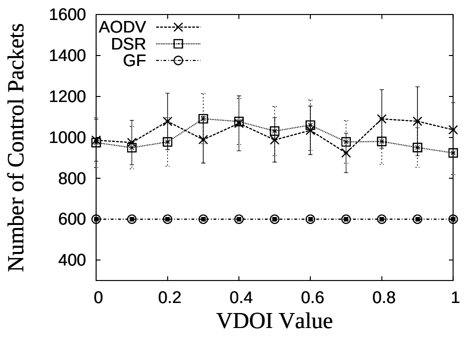
6.2.2 *Performance Sensitivity to Different VDOI.* This section is to analyze whether the DOI variance in a system has impact on routing performance. In the simulation, we set DOI as 0.01 and vary VDOI from 0 to 1, in steps of 0.1.



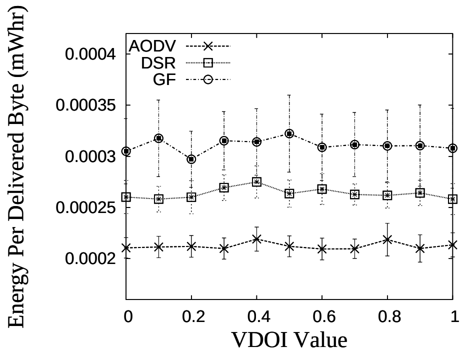
(a) E2E Loss Ratio vs. VDOI



(b) Average E2E Delay vs. VDOI



(c) Number of Control Packets vs. VDOI



(d) Energy Consumption vs. VDOI

Fig. 19. Routing Performance Sensitivity to Different VDOI Values

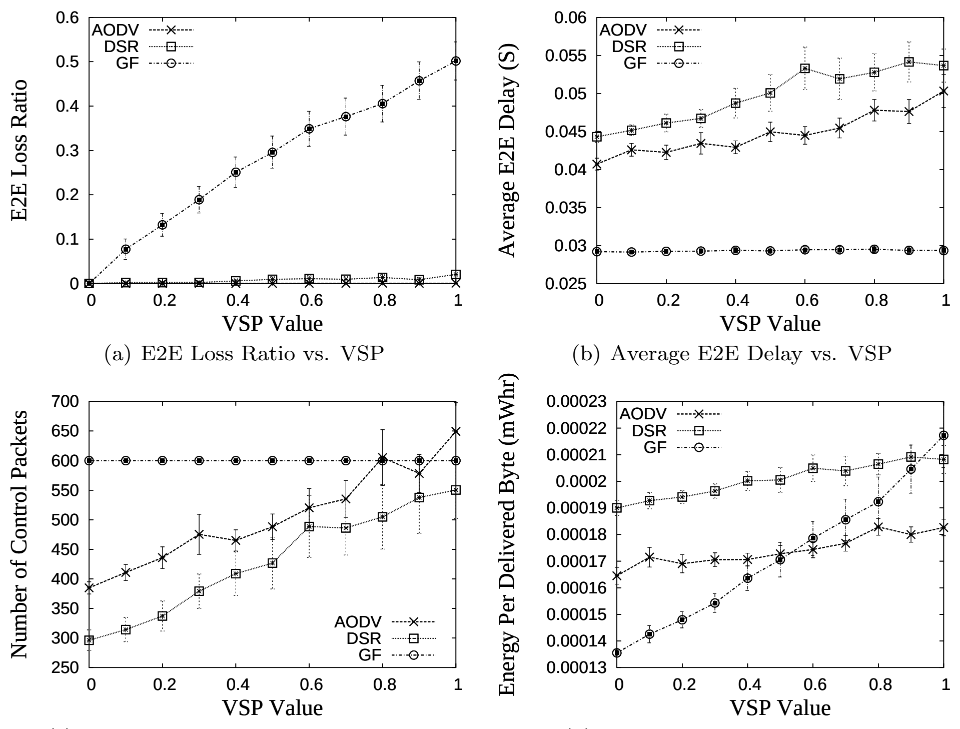
From Figure 19, we observe that when the VDOI value varies from 0 to 1, none of the four metrics, E2E loss ratio, average E2E delay, number of control packets and energy consumption per delivered data byte, shows clear performance variance statistically. This is because on one hand, some nodes get larger DOI values and have more irregular radio patterns, on the other hand, some other nodes get smaller DOI values and have less irregular radio signals. They cancel their effects with each other, and we observe no significant performance changes with different VDOI values.

6.2.3 *Routing Performance with Different VSP.* In Figure 20, the impact of radio irregularity on the routing layer is measured for different DOI values. In this section, we measure the impact of radio irregularity on the routing layer by varying the VSP values. From our results, we find that an increase of the VSP value has a similar impact on AODV, DSR and GF, as an increase of the DOI value, because both lead to a higher degree of irregularity and therefore, a higher degree of link asymmetry.

From Figure 20(a), we see that all routing protocols have higher loss ratios when the VSP value is increased, because there are more asymmetric links. GF has a much higher loss ratio than that of AODV and DSR, because GF uses neighbor discovery and tends to choose the same node near the border of the radio range as the candidate, while AODV and DSR use multi-round discovery to try different paths.

As in the case of larger DOI values, larger VSP values result in more asymmetric links, which lead to larger average E2E delays (Figure 20(b)) and higher energy consumption per delivered data byte(Figure 20(d)). However, GF does not require more beacons, so there is no increase in the control packets (Figure 20(c)) and the delay remains the same (Figure 20(a)). The energy consumption of GF for each delivered data byte increases sharply with the increase of VSP values, because its packet loss increases quickly.

To summarize, as DOI and VSP increase, radio irregularity has a greater adverse impact on the GF protocol compared to on-demand routing protocols that use multi-round discovery such as AODV and DSR.



(a) E2E Loss Ratio vs. VSP

(b) Average E2E Delay vs. VSP

(c) Number of Control Packets vs. VSP

(d) Energy Consumption vs. VSP

Fig. 20. Routing Performance with Different VSP Values

6.2.4 *Routing Performance with Different DOI-VSP Combinations.* In Section 6.2.1 and 6.2.3, we evaluate the impact of radio anisotropy and heterogeneous sending powers on routing performance, by setting different DOI and VSP values separately. In this section, we explore their composite impact on the routing protocols. We take the GF routing protocol as an example, and use combinations of different DOI and VSP values to evaluate GF's packet loss ratio.

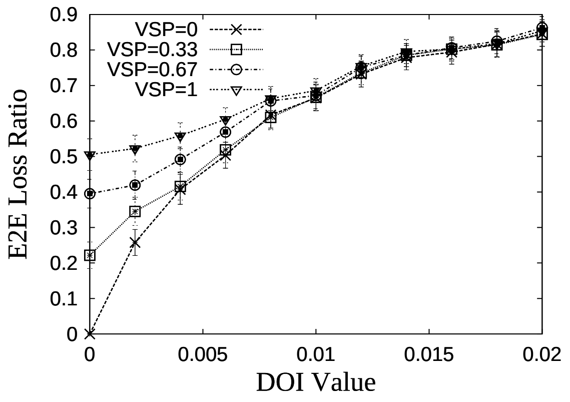


Fig. 21. GF Performance with Different DOI-VSP Combinations

As Figure 21 illustrates, for all the configured VSP values, whether it is 0 or greater than 0, a larger DOI value always leads to a larger E2E loss ratio. The routing performance decreases similarly as what we observe in Figure 18(a) when the VSP value is set to 0. On the other hand, for all the DOI values we use, a larger VSP value also leads to a larger E2E loss ratio, which is similar to what we observe in Figure 20(a) when the DOI value is set to 0. This shows that the impact of DOI and VSP do not cancel each other.

However, the impact of DOI and VSP on routing performance do not arithmetically accumulate, according to the results in Figure 21. This reflects that the asymmetric channels caused by DOI overlap with those caused by VSP in some locations of the system. The more the asymmetric channels overlap, the larger the gap between the composite E2E loss ratio and the sum of E2E loss ratios when either of them is set to 0. This is why the distances among the four curves shown in Figure 21 get closer while the DOI value increases.

7. IMPACT ON LOCALIZATION

In this section, we explore the impact of radio irregularity on localization protocols. We do not try to cover every localization protocol in detail, since that is beyond this paper's discussion. We take some popular techniques and protocols from the localization family and analyze the impact. We also present performance evaluation of the Centroid protocol, as an example of quantitative analysis.

Radio irregularity has a great impact on the localization protocols that use the Received Signal Strength Indicator (RSSI) technique, such as RADAR [Bahl and Padmanabhan 2000] and SpotOn [Hightower et al. 2000]. The RSSI technique assumes that once the distance between the transmitter and receiver is determined, the RSSI value is determined, and vice versa. However, in our experiments with

Table II. Simulation Configuration Two

TERRAIN	(150m X 150m)
Node Number	400(including anchors)
Node & Anchor Placement	Random
Routing Protocol	GF
MAC Protocol	802.11 (DCF)
Radio Layer	RADIO-ACCNOISE
Radio Model	RIM
Nominal Radio Range	65M
Radio Bandwidth	200Kb/s

MICA2 motes (Figure 2), the RSSI value varies when the receiver is put at different propagation directions from the transmitter, even though the distance between them is invariant, 10 feet in Figure 2(a) and 20 feet in Figure 2(b). Accordingly, the RSSI technique is misleading in calculating locations when the radio propagation direction is disregarded.

In DV-HOP [Niculescu and Nath 2003], the anchor nodes flood their locations throughout the whole network. Any node receiving this message records its hop-count to corresponding anchors. Then with these hop-counts to anchors, with the average distance per hop and with the anchors' locations, each node can figure out its own location. However, the radio range is not isotropic, and the communication ranges do not have an invariant value in different propagation directions. So the distance of each hop varies greatly, depending on the degree of radio irregularity. So it is misleading to calculate a node's distance to an anchor as the product of the hop-count and the average distance per hop.

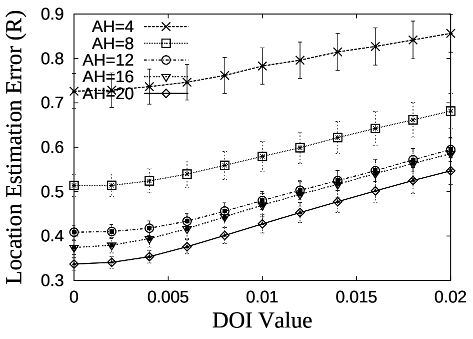
Radio irregularity has a great impact on the Centroid algorithm. In Centroid, a node's location is calculated as the geographic center of all anchors it hears. This idea does not work well because a node that can hear N anchors is not necessarily located exactly at the geographic center of the N anchors. When we consider the fact of irregular radio, the performance becomes worse, which can be observed in the simulation result illustrated in Figure 22. As before, the simulation is conducted in GloMoSim, and the simulation configuration is given in Table II.

From Figure 22(a), we observe that with the increase of DOI values, the localization error keeps increasing, with all the settings of the average Anchor Heard (AH), which is defined as the average number of anchors heard by a node and used during location estimation. For example, when DOI is 0 and AH is 20, the radio is spherical and the localization error is 33.7% of the nominal radio range. But when the DOI increases to 0.02 and AH remains 20, the radio becomes very irregular and the localization error increases to 54.7% of the nominal radio range. The decreased Centroid performance is caused by the increased radio irregularity, since larger DOI values lead to more anisotropic radio patterns.

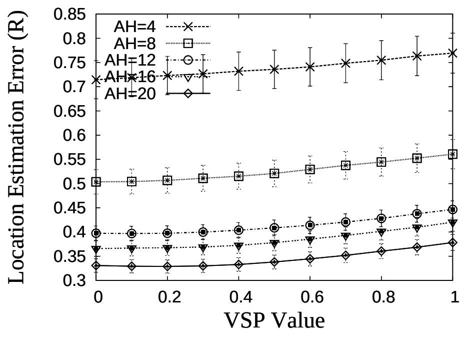
Similar experiments are repeated with different VSP values(Figure 22(b)), and we find that Centroid's localization error also increases with increasing VSP.

8. IMPACT ON TOPOLOGY CONTROL

In this section, we take GAF [Xu et al. 2001], a typical topology control protocol, as an example to study the impact of radio irregularity. In GAF, the deployment



(a) Performance with Different DOI Values



(b) Performance with Different VSP Values

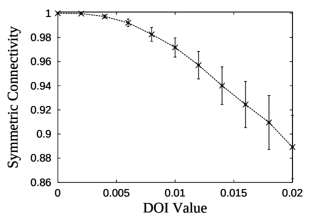
Fig. 22. Centroid's Performance with Different Radio Irreguarity

terrain is divided into virtual grids. In each grid, one node is chosen to stay awake and the others are put to sleep to save power. But at any time, the communication connectivity among adjacent grids must be maintained. In order to maintain the connectivity, the radio communication range R and the grid side length r must satisfy the following relations:

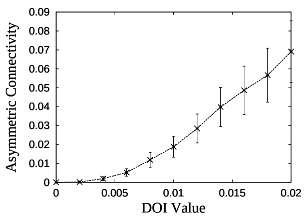
$$r \leq \frac{R}{\sqrt{5}} \tag{7}$$

Since radio is in fact irregular and the communication range is not spherical, it is hard to determine the value of the parameter R. In GAF, R is defined as the nominal radio communication range. However, using the nominal radio range R makes it impossible to guarantee the communication connectivity among adjacent grids, in the presence of anisotropic radio.

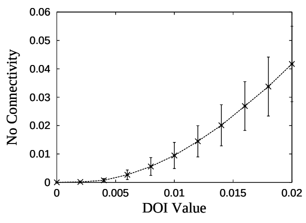
In order to investigate GAF's communication connectivity in the case of radio irregularity, we implement GAF in GloMoSim and present the simulation results in Figures 23 and 24. In the simulation, we use the configuration setting presented in Table II.



(a) Symmetric Connectivity



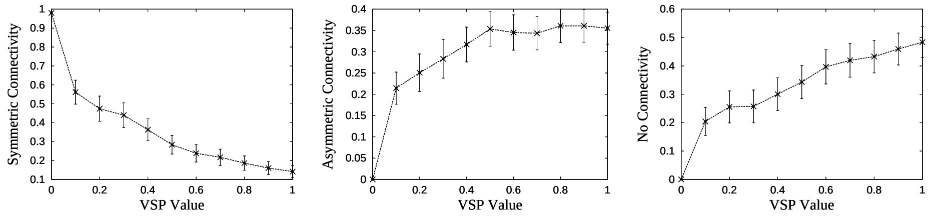
(b) Asymmetric Connectivity



(c) No Connectivity

Fig. 23. GAF's Connectivity Status with Different DOI Values

We measure the connectivity status among adjacent grids with different DOI and VSP values. From Figure 23, we observe that the percentage of symmetric connectivity decreases with the increase of DOI values. When the radio range is



(a) Symmetric Connectivity (b) Asymmetric Connectivity (c) No Connectivity

Fig. 24. GAF's Connectivity Status with Different VSP Values

spherical, i.e., DOI is 0, all connections are symmetric. But when DOI increases to 0.02 and the radio becomes irregular, the percentage of symmetric connections decreases to 89% (Figure 23(a)), and 7% of the connections become asymmetric (Figure 23(b)) and 4% of the connections totally get broken (Figure 23(c)). This is because of the radio irregularity. When the radio range becomes more and more anisotropic, the original symmetric connectivity becomes asymmetric, and the original asymmetric connections are broken in both directions.

We repeat the experiments with different VSP values, and similar results are observed in Figure 24. In Figure 24, when all nodes have the same sending power and the system is homogeneous, i.e., VSP is 0, all connections are symmetric. But when the VSP value increases to 1, only 15% of the connections are symmetric (Figure 24(a)), and there are 36% asymmetric connections (Figure 24(b)) and 49% connections are completely broken (Figure 24(c)). The reason for the decreased communication connectivity is the increasing heterogeneity in devices' sending powers, which results in greater difference in nominal radio ranges among different devices and leads to worse communication connectivity.

9. SOLUTIONS FOR RADIO IRREGULARITY

Having analyzed the causes and impact of radio irregularity, the key results can be summarized as follows:

—Radio irregularity is a common and non-negligible phenomenon in wireless systems. Link asymmetry is an upper layer phenomenon produced by irregular radio signals in the radio layer. And asymmetry links directly lead to MAC and routing failures.

—Radio irregularity has a greater impact on the routing layer than MAC layer.

—Routing protocols, such as AODV and DSR, that use multi-round discovery technique, can deal with radio irregularity, but with a high overhead.

—Routing protocols, such as geographic forwarding, which are based on neighbor discovery technique, are severely affected by radio irregularity.

—Radio irregularity results in larger localization errors and makes it harder to maintain communication connectivity.

Based on both analytical and experimental results, we present eight potential solutions to improve the protocol performance in the presence of radio irregularity.

We first describe the Symmetric Geographic Forwarding, the Asymmetry Detection Service and the Bounded Distance Forwarding solutions in detail and discuss their performance evaluation. We then follow that by briefly describing five other solutions.

9.1 Symmetric Geographic Forwarding

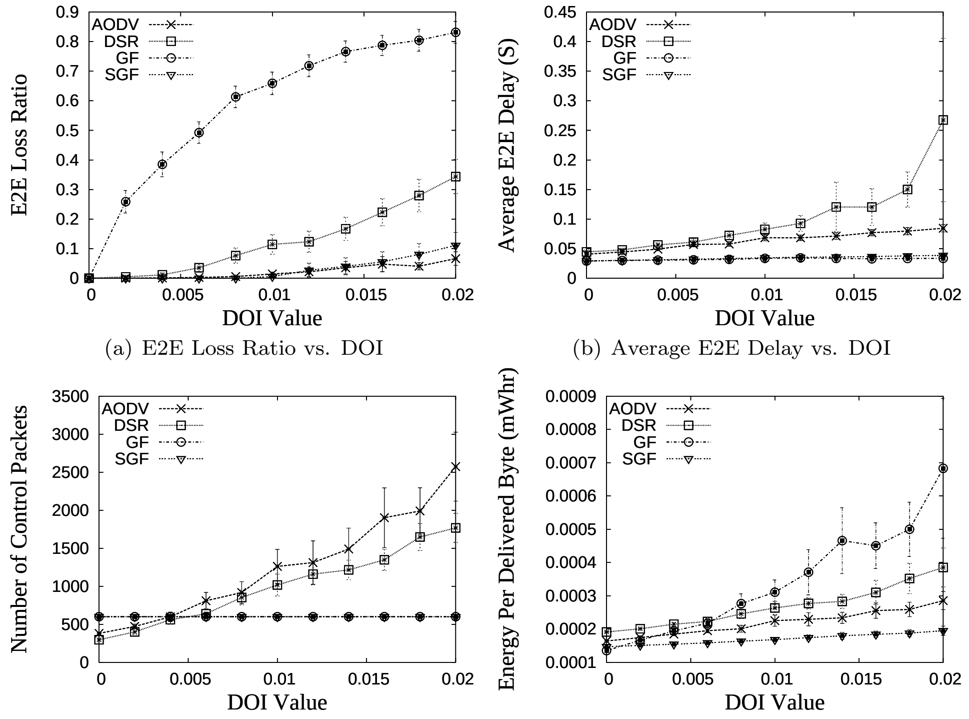
In location-based protocols, such as GF and GPSR, the beacon message only contains the node's ID and position. In our new Symmetric Geographic Forwarding (SGF) solution, we allow a node to add the IDs of all its neighbors it has discovered into the beacon message. When a node receives a beacon message, it registers the sender as its neighbor in its local neighbor table, and then checks whether its own ID is in the beacon message. If the receiver finds its own ID in the neighbor list in the beacon message, then it marks the communication link connecting it to the sender as *SYMMETRIC*. Otherwise, it marks the communication link between them as *ASYMMETRIC*. Whenever a node needs to forward a packet, it selects only those neighboring nodes with which it is connected through *SYMMETRIC* links. Here we must emphasize that when a node broadcasts a beacon message, it should add the IDs of the nodes with which it has *SYMMETRIC* connectivity as well as those nodes with which it has *ASYMMETRIC* connectivity.

We simulate SGF in GloMoSim. We find that SGF maintains most of the advantages of GF, such as scalability, and the absence of flooding. Furthermore, SGF is able to deal with asymmetry as effectively as the multi-path route discovery protocols, such as AODV and DSR, but at lower cost. The simulation setup use the same configuration as mentioned in Table I.

9.1.1 *SGF Performance with Different DOI.* In this experiment, we incrementally increase the degree of irregularity (DOI) to measure the SGF performance.

From Figure 25(a), we observe that SGF has a significantly lower loss ratio than that of GF, and performs as well as AODV. This is because it avoids forwarding data along asymmetric links. From Figure 25(b), we observe that SGF has almost the same average E2E delay as that of GF. The delay is much lower than that of ADOV and DSR. An interesting point from Figure 25(c) is that SGF consumes the same number of control packets as that of GF, and the number of control packets remains the same with the increase of DOI values. From Figure 25(d), it is observed that GF has a rapidly increased energy consumption for each successfully delivered data byte, with the increase of DOI values. But AODV, DSR and GF have comparatively slow increases of energy consumption for each delivered data byte. This is because radio irregularity has a greater impact on GF than on AODV, DSR and SGF, and GF suffers the most packet loss. DSR is less energy efficient than AODV and SGF, because it has more packet loss (Figure 25(a)) as well as more control overhead (Figure 25(c)) than AODV and SGF. On the other hand, SGF exhibits the least increased energy consumption among these four routing protocols, because its packet loss is as low as that of AODV (Figure 25(a)), and its control overhead is as low as that of GF (Figure 25(c)).

9.1.2 *SGF Performance with Different VSP.* Similar conclusions can be drawn from the results in Figure 26. Compared with GF, SGF has a much lower loss ratio,



(a) E2E Loss Ratio vs. DOI (b) Average E2E Delay vs. DOI

(c) Number of Control Packets vs. DOI (d) Energy Consumption vs. DOI

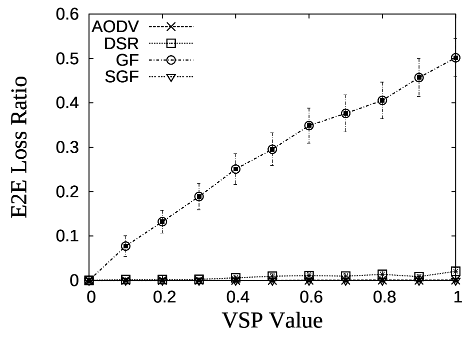
Fig. 25. SGF Performances with Different DOI Values

almost the same average E2E delay, and the same number of control packets, with an increase of the VSP value. The loss ratio of SGF is comparable to that of AODV and DSR. However, SGF has a much lower average E2E delay, a constant number of control packets, and a much lower energy consumption for each delivered data byte. SGF consumes almost constant energy with an increase of the VSP value. In contrast, GF suffers a sharply increased energy consumption for each delivered data byte, because of its rapidly increased packet loss. Plus, AODV and DSR also consume more energy for each delivered data byte compared to SGF, on account of two reasons. First, AODV and DSR need more control overhead than SGF, as shown in Figure 26(c). Second, AODV has the same level of packet loss ratio as that of SGF while DSR drops more useful data packets than SGF (Figure 26(a)).

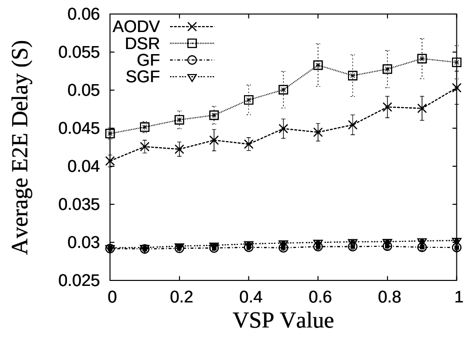
To summarize, the SGF protocol not only maintains GF's scalability, but also successfully deals with radio irregularity. Compared with AODV and DSR, it achieves similar delivery ratio in the presence of radio irregularity with a lower E2E delay, a lower number of control packets and lower energy consumption.

9.2 Asymmetry Detection Service

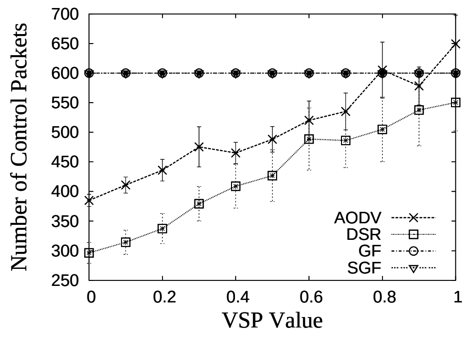
The SGF provides a basic prototype of incorporating symmetric detection into routing protocols. In a running system, more sophisticated algorithms should be



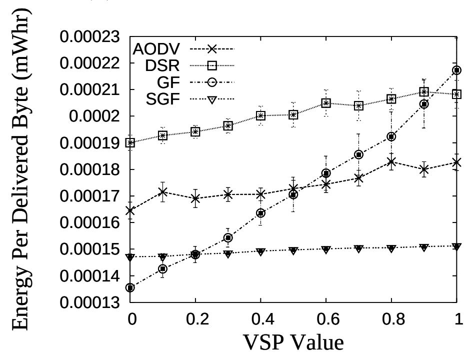
(a) E2E Loss Ratio vs. VSP



(b) Average E2E Delay vs. VSP



(c) Number of Control Packets vs. VSP



(d) Energy Consumption vs. VSP

Fig. 26. SGF Performance with Different VSP

introduced to deal with engineering issues[6]. In this section, we implement a general Asymmetry Detection Service in the VigilNet System [He et al. 2004] developed by University of Virginia.

In the Asymmetry Detection Service, the same idea is used to mark a link as *SYMMETRIC* or *ASYMMETRIC* as what is used in SGF. However, to deal with engineering issues in running systems, this marking process is repeated several times to get a statistical evaluation of a link's symmetric communication quality. Only those links that have higher symmetric communication qualities than the specified threshold are available for upper layers, and all other links are blocked from higher layer protocols.

In the VigilNet System, a communication backbone is built to relay messages back to the base station. The communication backbone is established using a classic spanning tree algorithm [Cormen et al. 2002], with the base station as the spanning tree's root. During the construction of the spanning tree, the Asymmetry Detection

[6]Engineering issues mean the system dynamics caused by the unpredictable system deployment environment, such as the changing temperature and humidity levels, the swinging trees and the jumping bugs, as well as the system dynamics caused by passing cars and human beings walking around. All these engineering issues bring communication dynamics into running system experiments.

Service is called and only symmetric links are used. We measure the performance evaluation of the Asymmetry Detection Service, by counting the percentage of nodes that are able to report back their status information successfully through the communication backbone. We conduct the experiment with 27 MICA2 devices and the result is given in Figure 27.

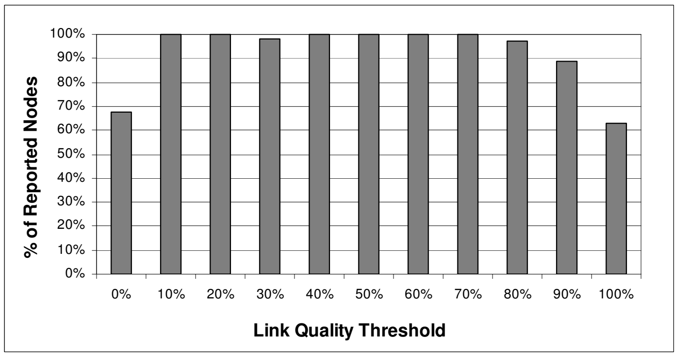


Fig. 27. Performance Evaluation of Asymmetry Detection Service

When the Asymmetry Detection Service is disabled, by setting the link quality threshold to 0 as shown in Figure 27, only 67.4% nodes are able to successfully report information back to the base station, because the communication backbone consists of a large portion of asymmetric links, and the data packets can not be correctly relayed back to the base station following the reversed path from the spanning tree's leaves to its root, the base station. However, when the Asymmetry Detection Service is used, we observe that almost all nodes are able to successfully report back to the base station. The spanning tree backbone works well when the link quality threshold is set from 10% to 70%. This performance improvement is caused by the Asymmetry Detection Service, which cuts off unidirectional links, and contributes to establishing the reliable communication backbone.

We are aware that the system still performs very well, even when the link quality threshold is set very low, as low as 10%. This is because MAC layer retransmission is used, in case of communication failures. However, the MAC layer retransmission alone can not achieve this good performance. When we disable the Asymmetry Detection Service by setting the link quality threshold to 0 and only use the MAC layer retransmission, the system performance is very poor, only 67.4% of the nodes report back to the base station.

On the other hand, when the link quality threshold keeps increasing and is close to 100%, the system performance decreases. This is because when Asymmetry Detection Service blocks all links that do not have 100% link qualities, there are not enough links available to build the communication backbone, and network partition happens.

The scheme we proposed here is related to approaches proposed in [Woo et al. 2003], in which each node snoops on the channel and eavesdrops on communications over time, to evaluate the inbound channel quality. Since routing is based on outbound (transmission) links, a separate phase is used in [Woo et al. 2003] for nodes to exchange inbound quality information to build up outbound quality information. However, our scheme is different. In our scheme, each node beacons periodically. From the received beacon packet, each node is able to locally figure out both inbound and outbound link qualities, without exchanging such quality information with neighbors. Another related solution, named blacklisting, can also be found in paper [Gnawali et al. 2004]. Moreover, readers can refer to [Seada et al. 2004][Couto et al. 2003][Yarvis et al. 2002] for more related solutions.

9.3 Bounded Distance Forwarding

Bounded Distance Forwarding restricts the distance over which a node can forward a message in a single hop. It is designed as a middleware and can act as an add-on rule to many routing protocols. It provides interfaces for configuring the bounded distance and provides services to filter out all links that are beyond the specified distance. The distance bound is configured based on the degree of radio irregularity of the real devices in a physical system.

We add the Bounded Distance Forwarding rule on the spanning tree module in a vehicle tracking system [He et al. 2004] in which we deploy 60 MICA2 motes. In the experiments, we incrementally increase the single hop forwarding bound from 8 feet to 100 feet and count the number of nodes that report their status and Figure 28 shows this data as a percentage of the total number of nodes deployed. Data points here are average values over five runs.

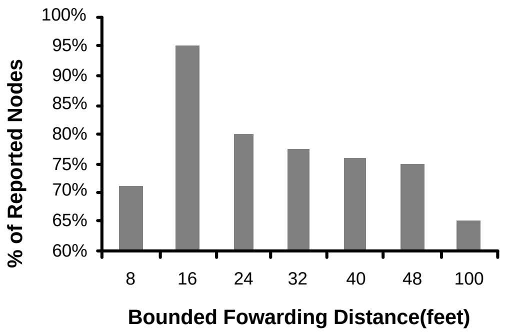


Fig. 28. Performance Evaluation of Bounded Distance Forwarding

Figure 28 indicates two interesting phenomena. First, when we use a very low forwarding bound (8 feet) to eliminate the asymmetric links, the performance, however, is not good. This is because relative node density decreases when the enforced communication range is small. Hence, the chance of a network partition increases. Moreover, a smaller forward bound per hop leads to a longer route, thus a higher chance of loss. Second, when the forwarding bound reaches larger values (16$\sim$100 feet), link asymmetry becomes the dominating factor. Figure 28 shows that when the forwarding bound is 16 feet, we receive almost every report. This bound is about half of the MICA2 radio range on the ground. Above 16 feet, performance reduces monotonically because of increase in link asymmetry.

Figure 28 shows that an effective bound, 16 feet here, exists for the application, at which the best performance is achieved. There are generally two methods to get the effective bound for a deployed system. The first method is to tune the bounded distance parameter at the initial deployment of the system, and the second method is to use a feedback control algorithm to converge the bounded distance to the effective value during the runtime of the deployed system.

9.4 OTHER SOLUTIONS

In this section, we propose five additional potential solutions to deal with radio irregularity.

—*Bidirectional Flooding:* The multi-round discovery technique can deal with radio irregularity. However, it needs multiple rounds of flooding to explore different paths, which can be very expensive. In Bidirectional Flooding, the source propagates the RREQ towards the destination through flooding. After the destination receives the RREQ, it propagates the RREP to the source through flooding, instead of using the reverse path along which it received the RREQ from the source. Multi-round discovery cannot guarantee finding symmetry connections within a bounded number of flooding stages. In contrast, bidirectional flooding completes the discovery by flooding twice.

—*Learning Function:* In an earlier section we mentioned that GF has a higher loss ratio than AODV and DSR, because GF tends to choose the same candidate near the border of its communication range to forward packets to a destination, while AODV and DSR attempt different paths due to the nature of flooding. To address this shortcoming of GF, we can enhance GF with a learning function, which allows a node to make better decisions based on previous routing failures. In the learning function, we distinguish the routing failures arising due to congestion from those that arise due to asymmetric links. This can be done with the help of the 802.11 (DCF) in the MAC layer. If a node receives the CTS, but not the ACK, then the link should be symmetric and the routing failure might be a result of congestion. Such a failure can be solved by retransmissions. However, if a node fails to receive the CTS despite several retransmissions, then the chances are that the link is asymmetric. This learning function allows a node to remember such an asymmetric link and to avoid trying it again before the topology changes.

In a real implementation of this idea, two learning functions are maintained: F_{link} and $F_{congestion}$. Whenever a packet gets lost, whether it is a CTS packet or a DATA packet, both F_{link} and $F_{congestion}$ adjust their values according to the current context. For example, if a DATA packet gets lost, $F_{congestion}$ gets a greater increase than F_{link}, because the CTS was received and there is more chance that congestion, rather than channel quality variation, causes the transmission failure. On the other hand, if the CTS packet gets lost the second time, it is more probable that the channel link quality is bad, and hence F_{link} gets more increase than $F_{congestion}$. By comparing the values of F_{link} and $F_{congestion}$, the node decides whether it is congestion or bad link quality that causes the packet loss. If $F_{congestion} > F_{link}$, the reason is congestion. So backoff and retransmission is a good choice. If $F_{congestion} < F_{link}$, the reason is the bad channel quality and rerouting is a better choice. In the case of a tie, a random decision is made between retransmission and rerouting.

To improve the accuracy of the two learning functions: F_{link} and $F_{congestion}$, the signal intensity during the carry sensing period can be monitored, together with the packet loss ratio. If both the signal intensity and the packet loss ratio increase, $F_{congestion}$ gets increased greater than F_{link}. On the other hand, if the signal intensity does not change or even decreases while at the same time the packet loss ratio increases, it is highly probable that the link quality decreases, and F_{link} gets increased by a larger amount than $F_{congestion}$.

Actually, the learning function scheme has other applications, besides the application in routing protocols. For example, ESRT [Sankarasubramaniam et al. 2003] is proposed to provide reliable event-to-sink transport service. Nodes monitor local buffer levels. If the routing buffer overflows due to excessive incoming packets, congestion is considered happened, and source nodes in the network are forced to reduce data reporting frequency. Actually, this buffer monitoring scheme does not differentiate whether the buffer overflow is due to congestion and followed by retransmission, or due to the poor link quality of data reporting paths. It is not reasonable for source nodes to reduce data reporting frequencies, if the buffer overflow is caused by the poor routing protocol that chooses poor data reporting paths. The learning function scheme can be used to differentiate these two cases, and choose to either inform source nodes to reduce data reporting frequencies or to inform the routing protocol to choose better data reporting paths.

—*RTS Broadcast:* Another solution we propose is called the RTS Broadcast, which involves both the MAC and routing layers. We first broadcast a special RTS message, which sets the destination as ANY_NODE. Any node hearing it backs off for a random amount of time and replies with a CTS message. Among all the nodes that send the CTS message, the one that is closest to the destination is chosen as the forwarding candidate. Since the RTS and CTS detect connectivity along the forward and reverse directions of a channel, forwarding packets along asymmetric channels can be avoided.

—*High Energy Asymmetry Detection:* IEEE 802.11 (DCF) uses a collision-avoidance strategy in which any node upon hearing an RTS, CTS, or DATA message defers its transmission until the data is sent out. However, a node can still interfere with the message transmission even though it is not able to hear any of the RTS, CTS and DATA messages in the presence of asymmetry. The sixth solution we propose is to send out a High Energy Asymmetry Detection (HEAD) control message which has a higher sending power than the other control messages. So more nodes will hear the high-powered signal, and prevent themselves from sending messages. The HEAD message is sent out before the RTS message. Any node other than the destination, upon hearing the HEAD message, sets its NAV to a value large enough so that data can be sent out without contention. The wait time and destination ID are included in the HEAD message. Conflicts may arise if two nodes send out the HEAD messages simultaneously. That is resolved in a manner similar to the way to resolve conflicts arising from the simultaneous transmission of two RTS messages. Hence, the transmission sequence is modified from RTS-CTS-DATA-ACK to HEAD-RTS-CTS-DATA-ACK. While the higher sending power of the HEAD message lowers the collision rate, it also introduces an extra control packet, the HEAD packet, which reduces the channel utilization and increases the NAV backoff. The tradeoff between collision rate and desired channel utilization can be balanced by choosing an appropriate value for the sending power.

—*Irregularity Insensitive Protocols:* There are two avenues for improving protocol performance in the presence of radio irregularity. The first method is to face radio irregularity and avoid getting involved into any trouble brought by radio irregularity. For example, we propose to detect asymmetry links brought by radio irregularity and try to avoid using asymmetry links. The second method is to investigate the assumptions that cause protocol performance to deteriorate in reality and then design protocols that do not make such assumptions. That is, radio irregularity can also be dealt with, by identifying protocol properties that make them particularly insensitive to radio irregularities. For example, the Cricket localization [Priyantha et al. 2000] uses a combination of RF and ultrasound technologies to location devices' locations. Cricket is insensitive to radio irregularity and avoids the problems many localization protocols get involved in because of using the received signal strength to estimate communication distances. The APIT localization protocol [He et al. 2003] is another example that avoids making the ideal radio assumption. Accordingly, irregularity insensitive protocol design is a promising avenue to address the radio irregularity as well as link asymmetry it brings.

Among the eight solutions we put forth above, the last five are still open topics and require further refinements. Extensive analysis and evaluation in the future are required to demonstrate their applicability and effectiveness.

10. CONCLUSIONS

In this paper, we confirm the existence of radio irregularity which is the main focus of several recent research papers [Ganesan et al. 2002][Woo et al. 2003][Zhao and Govindan 2003][Cerpa et al. 2003]. Our contributions are as follows:

(1) To the best of our knowledge, our work is the first to bridge the gap between isotropic radio models assumed by most simulators and the anisotropic radio properties found in reality.

After our work was first accepted in MobiSys 2004 [Zhou et al. 2004], an upper layer model [Cerpa et al. 2005] was proposed to simulate link asymmetry in the link layer, without considering the wireless communication detail in the radio layer. We compare our RIM radio model with this link layer model in APPENDIX B.

(2) We propose a novel RIM model that approximates three essential properties exhibited in radio irregularity: anisotropy, continuous variation and difference in sending power.

(3) We implement the RIM model in GloMoSim, and run a set of simulation experiments to investigate radio irregularity's impact on MAC and routing layer performance. We discover that, among the protocols we evaluate, the radio irregularity has a greater impact on the routing layer than MAC layer. We also discover that radio irregularity has a greater impact on location-based routing protocols than on-demand protocols that use multi-round discovery technique.

(4) We run a set of simulation experiments to investigate radio irregularity's impact on localization and topology control, finding that the increasing radio irregularity leads to larger localization errors, and that the communication connectivity becomes harder to maintain when the radio becomes more irregular.

(5) Finally, we present eight potential solutions. We implement SGF in GloMoSim, and implement the Asymmetry Detection Service and the Bounded Distance Forwarding methods in running systems with 27$\sim$60 MICA2 motes. From the data we collect from the simulator and the running system, we find that SGF, Asymmetry Detection Service and Bounded Distance Forwarding greatly improve system performance in the presence of radio irregularity.

The RIM model we put forth in this paper is built based on empirical data collected from MICA2 and MICAZ platforms. So to some degree, this model is self-evaluated. We also conduct preliminary evaluation of the RIM model, in Section 4.2.1, by comparing the degree of radio irregularity between the measured radio pattern from a real device and the radio patterns generated from the RIM model. We are also aware that more extensive performance comparison between the simulated results based on the RIM model and the results from real systems with MICA2 and other devices are needed, to further evaluate the precision of the RIM model. We leave this as future work.

ACKNOWLEDGEMENT

This work is supported by the National Science Foundation (grant CNS-0435060, grant CCR-0325197, and grant EN-CS-0329609) and by the DARPA IXO offices under the NEST project (grant F336615-01-C-1905). The authors would like to thank the reviewers of ACM Transactions on Sensor Networks for their valuable comments during the revision of this work.

APPENDIX A

We use the goodness-of-fit statistical testing to determine the statistical distribution of the percentage variance of the path loss (in dBm) per degree in the direction that is obtained in our experiments. We find that among different continuous distributions, the Weibull distribution [Devore 1982] has the maximum likelihood of matching our experimental data. A random variable X that has a Weibull distribution with parameters has a probability density function defined by the following equation, where a is the shape parameter and b is the scale parameter.

Table III shows the likelihood values and the parameters of the Weibull distribution that fits our experimental data. These values are computed at a 95% confidence level.

Table III. Data Fitting to the Weibull Distribution

	Likelihood	a	b
Dataset 1	48.55	1.13	0.28
Dataset 2	154.43	1.01	0.17
Dataset 3	145.25	0.86	0.18
Dataset 4	277.44	0.67	0.16
Dataset 5	204.51	0.58	0.17
Dataset 6	111.15	0.53	0.22

$$f_x = \begin{cases} (a/b^a) \times x^{a-1} \times e^{-(\frac{x}{b})^a} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \tag{8}$$

APPENDIX B

A link layer model for simulating link asymmetry is first proposed in Cerpa's technical report [Cerpa et al. 2004], and later published in IPSN 2005 [Cerpa et al. 2005]. In this model, link asymmetry is simulated without considering the lower layer wireless communications. The RIM model differs with the link layer model in that the RIM model is a radio layer model. The RIM model is proposed to simulate radio irregularity rather than link asymmetry, which happens to be one result of radio irregularity reflected in the link layer. Since the RIM model incorporates details in radio communication, it can address more issues that the simpler link layer model can not simulate. We illustrate four of them as follows:

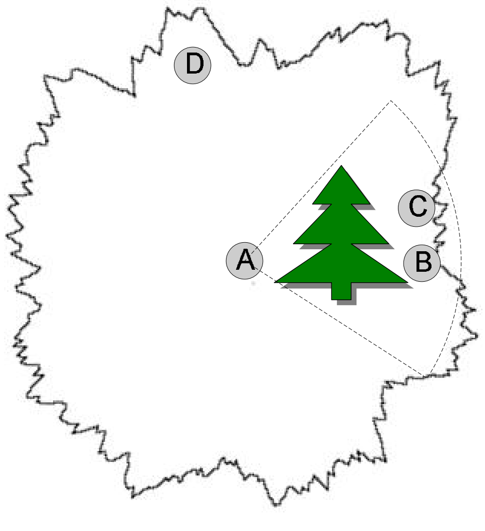


Fig. 29. Link Qualities in Adjacent Directions

First, the RIM protocol can simulate the phenomenon that links in very similar directions from the same transmitter have similar link qualities. As shown in Figure 29, there is a big tree, in the east direction of transmission node A. So the signal from A suffers more path losses, due to the tree, in the directions specified by the fan area compared with other directions. For example, when A's signal propagates to B and C, it suffers similar path losses. But the signal suffers less path loss when it propagates from A to D. Accordingly, transmitter A has similar link qualities with B and C. The link layer model does not simulate directionality of signal propagation and this phenomenon is not addressed. However, our RIM model can reflect this fact, because all the k_i values in adjacent directions are related in the sense that k_{i+1} is calculated based on k_i.

Second, the RIM model can be used to study the impact of radio irregularity on localization protocols that is sensitive to radio patterns. In section 7 of this paper, we study the impact of radio irregularity on the Centroid algorithm as an example. In a similar way, our model can be use to study the impact of radio irregularity on many other localization protocols, which use the received signal strength indicator to help location decisions. However, the link layer model that does not simulate the radio communication process can not be used for this study.

Third, the link layer model does not regenerate radio signals, so they can not really simulate radio interference, which also has a significant effect on link qualities. The RIM model works in the radio layer, which uses mature simulation techniques, such as TWO-RAY model and RICIAN mode [Shankar 2001] implemented in GloMoSim, to simulate the radio propagation and fading in a specified direction. And the DOI and VSP parameters proposed in RIM are used to account for the irregularity in different directions as well as hardware differences. Accordingly, the RIM model can also simulate radio interference, which also leads to decreased link qualities.

Fourth, with the DOI and VSP parameters, the RIM model can simulate radio irregularity due to the two root causes we found in our sensor device experiments: the path loss differences and the power heterogeneity. This offers the users the ability to configure different DOI and VSP values according to their specific hardware properties and deploy environment, to simulate the system performance within the specific hardware and environment context. However, the link layer model is not able to differentiate which of the two root causes leads to the decreased link quality and how much each of them contributes.

On one hand, we acknowledge that the link layer model has a higher abstraction and is hence smaller and light weight. On the other hand, by simulating wireless communication details, the RIM radio model is more powerful, and able to address more issues that are beyond the ability of the link layer model.

REFERENCES

Bahl, P. and Padmanabhan, V. N. 2000. RADAR: An In-Building RF-Based User Localization and Tracking System. In *IEEE InfoCom 2000*.

Battery Lifetime. Battery Technology Life Verification Test Manual. http://www.uscar.org/consortia&teams/USABC/Manuals/Technology Life Verification Test Manual - Feb 2005.pdf.

Bharghavan, V., Demers, A., Shenker, S., and Zhang, L. 1994. MACAW: A Media Access Protocol for Wireless LANs. In *ACM SigComm 1994*. 212–225.

Cerpa, A., Busek, N., and Estrin, D. 2003. SCALE: A Tool for Simple Connectivity Assessment in Lossy Environments. Tech. rep., CENS Technical Report 0021.

Cerpa, A. and Estrin, D. 2002. ASCENT: Adaptive Self-Configuring Sensor Networks Topologies. In *IEEE InfoCom 2002*.

Cerpa, A., Wong, J. L., Kuang, L., Potkonjak, M., and Estrin, D. 2004. Statistical Model of Lossy Links in Wireless Sensor Networks. Tech. rep., CENS Technical Report 0041.

Cerpa, A., Wong, J. L., Kuang, L., Potkonjak, M., and Estrin, D. 2005. Statistical Model of Lossy Links in Wireless Sensor Networks. In *ACM/IEEE IPSN'05*.

Chen, B., Jamieson, K., Balakrishnan, H., and Morris, R. 2001. SPAN: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad-Hoc Wireless Networks. In *ACM MobiCom 2001*.

ChipconCC1000. Chipcon CC1000 Low Power Radio Transceiver. http://www.chipcon.com.

ChipconCC2420. CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. http://www.chipcon.com.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. 2002. *Introduction to Algorithms*. The MIT Press.

Couto, D. S. J. D., Aguayo, D., Bicket, J., and Morris, R. 2003. A High-Throughput Path Metric for Multi-Hop Wireless Routing. In *ACM MobiCom 2003*.

CROSSBOW. XBOW Mote Specifications. http://www.xbow.com.

Dam, T. and k. Langendoen. 2003. An Adaptive Evergy-Sufficient MAC Protocol for Wireless Sensor Networks. In *ACM SenSys 2003*.

Devore, J. L. 1982. *Probability and Statistics for Engineering and the Sciences*. Brooks/Cole Publishing.

Ganesan, D., Krishnamachari, B., Woo, A., Culler, D., Estrin, D., and Wicker, S. 2002. Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks. Tech. rep., Technical Report UCLA/CSD-TR 02-0013.

Gnawali, O., Yarvis, M., Heidemann, J., and Govindan, R. 2004. Interaction of Retransmission, Blacklisting, and Routing Metrics for Reliability in Sensor Network Routing. In *IEEE SECON 2004*.

He, T., Huang, C., Blum, B. M., Stankovic, J. A., and Abdelzaher, T. F. 2003. Range-Free Localization Schemes in Large Scale Sensor Networks. In *ACM MobiCom 2003*.

He, T., Krishnamurthy, S., Stankovic, J. A., Abdelzaher, T. F., Luo, L., Stoleru, R., Yan, T., Gu, L., Hui, J., and Krogh, B. 2004. Energy-Efficient Surveillance System Using Wireless Sensor Networks. In *ACM MobiSys 2004*. 270–283.

He, T., Stankovic, J. A., Lu, C., and Abdelzaher, T. F. 2003. SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks. In *IEEE ICDCS 2003*.

Hightower, J., Boriello, G., and Want, R. 2000. SpotON: An Indoor 3D Localization Sensing Technology Based on RF Signal Strength. Tech. rep., University of Washington CSE Report 2000-02-02.

Hightower, J., Vakili, C., Borriello, G., , and Want, R. 2000. Design and Calibration of the SpotON Ad-Hoc Location Sensing System. Tech. rep., unpublished. August.

Hoff, B. and Azuma, R. 2000. Autocalibration of an Electronic Compass in an Outdoor Augmented Reality System. In *Proceedings of IEEE and ACM International Symposium on Augmented Reality 2000*. 159–164.

IEEE 802.11 1999. IEEE 802.11, part II: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification. ANSI/IEEE Std. 802.11, 1999.

IEEE 802.15.4 1999. IEEE 802.15.4, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE Std. 802.15.4, 2003.

Intanagonwiwat, C., Govindan, R., and Estrin, D. 2000. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *ACM MobiCom 2000*. 56–67.

Johnson, D. B. and Maltz, D. A. 1996. Dynamic Source Routing in Ad Hoc Wireless Networks. In Mobile Computing, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

Karn, P. 1990. MACA - A New Channel Access Method for Packet Radio. In *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*. 134–140.

Karp, B. 2000. Geographic Routing for Wireless Networks. Ph.D. thesis, Harvard University, Cambridge, MA.

Karp, B. and Kung, H. T. 2000. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *ACM MobiCom 2000*. 243–254.

Kleinrock, L. and Tobagi, F. A. 1975. Packet Switching in Radio Channels: Part I - Carrier Sense Multiple-Access Modes and their Throughput-Delay Characteristics. *IEEE Transactions on Communications COM-23*, 1400–1416.

Ko, Y.-B. and Vaidya, N. H. 1998. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *ACM MobiCom 1998*. 66–75.

N. Bulusu, J. H. and Estrin, D. 2000. GPS-Less Low Cost Outdoor Localization for Very Small Devices. *IEEE Personal Communications Magazine*, 28–34.

Niculescu, D. and Nath, B. 2003. DV Based Positioning in Ad hoc Networks. *Journal of Telecommunication Systems*.

Perkins, C. E. and Royer, E. M. 1999. Ad-hoc On-Demand Distance Vector Routing. In *The Second IEEE Workshop on Mobile Computing Systems and Applications*. 90–100.

Polastre, J., Hill, J., and Culler, D. 2004. Versatile Low Power Median Access for Wireless Sensor Networks. In *ACM SenSys 2004*.

Priyantha, N. B., Chakraborty, A., and Balakrishnan, H. 2000. The Cricket Location-Support System. In *ACM MobiCom 2000*.

Rajendran, V., Obraczka, K., and Garcia-Luna-Aceves, J. 2003. Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks. In *ACM SenSys 2003*.

RF Chamber. Turnkey Chambers. http://www.tdkrfsolutions.com/chambers.htm.

Sankarasubramaniam, Y., Ö. B. Akan, and Akyildiz, I. F. 2003. Esrt: event-to-sink reliable transport in wireless sensor networks. In *ACM MobiHoc '03*. 177–188.

Seada, K., Zuniga, M., Helmy, A., and Krishnamachari, B. 2004. Energy-Efficient Forwarding Strategies for Geographic Routing in Lossy Wireless Sensor Networks. In *ACM SenSys 2004*.

Shankar, P. M. 2001. *Introduction to Wireless Systems*. John Wiley and Sons, Inc.

Whitehouse, K. and Culler, D. 2002. Calibration as Parameter Estimation in Sensor Networks. In *ACM International Workshop on Wireless Sensor Networks and Applications*.

Woo, A. and Culler, D. 2001. A Transmission Control Scheme for Media Access in Sensor Networks. In *ACM MobiCom 2001*.

Woo, A., Tong, T., and Culler, D. 2003. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *ACM SenSys 2003*.

Xu, Y., Heidemann, J., and Estrin, D. 2001. Geography-Informed Energy Conservation for Ad Hoc Routing. In *ACM MobiCom 2001*.

Yan, T., He, T., and Stankovic, J. A. 2003. Differentiated Surveillance for Sensor Networks. In *ACM SenSys 2003*.

Yarvis, M. D., Conner, W. S., Krishnamurthy, L., Mainwaring, A., Chhabra, J., and Elliott, B. 2002. Real-World Experiences with an Interactive Ad Hoc Sensor Network. In *IWAHN 2002*.

Ye, W., Heidemann, J., and Estrin, D. 2002. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *IEEE InfoCom*. 1567–1576.

Zeng, X., Bagrodia, R., and Gerla, M. 1998. GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks. In *The 12th Workshop on Parallel and Distributed Simulations*.

Zhao, J. and Govindan, R. 2003. Understanding Packet Delivery Performance in Dense Wireless Sensor Networks. In *ACM SenSys 2003*.

Zhou, G., He, T., Krishnamurthy, S., and Stankovic, J. A. 2004. Impact of Radio Irregularity on Wireless Sensor Networks. In *ACM MobiSys 2004*. 125–138.

A Spatiotemporal Communication Protocol for Wireless Sensor Networks

Tian He, *Member, IEEE,* John A. Stankovic, *Fellow, IEEE*,
Chenyang Lu, *Member, IEEE* and Tarek F. Abdelzaher, *Member, IEEE*

Abstract— In this paper, we present a spatiotemporal communication protocol for sensor networks, called SPEED. SPEED is specifically tailored to be a localized algorithm with minimal control overhead. End-to-end soft real-time communication is achieved by maintaining a desired delivery speed across the sensor network through a novel combination of feedback control and non-deterministic geographic forward-ing. SPEED is a highly efficient and scalable protocol for sensor networks where the resources of each node are scarce. Theoretical analysis, simulation experiments and a real implementation on Berkeley motes are provided to validate the claims.

Index Terms— Wireless sensor networks, routing, real-time, spatiotemporal, networking.

1 Introduction

MANY sensor network applications, such as battle-field surveillance and earthquake response systems, are designed to interact with fast changing events in the real world. It is often necessary for the underlying communication infrastructure to meet real-time constraints [9][16][4]. In surveillance systems [8], for example, communication delays within sensing and actuating loops directly affect the quality of tracking. Our spatiotemporal communication protocol, called SPEED, is inspired by following observation: In wired networks, the end-to-end delay is independent of the physical distance between the source and destination. While in multi-hop wireless sensor networks, since communication is physically bounded, the end-to-end delay depends not only on single hop delay (time-constraints), but also on the distance a packet travels (spatial-constraints). In view of this, the key design goal of this work is to support a spatiotemporal communication service with a desired delivery *speed* across the sensor network, so that end-to-end delay is proportional to the distance between the source and destination. We deem this service as one type of soft real-time communication, because it achieves a predicable end-to-end communication delay under given distance (spatial) constraints.

The key contribution of SPEED is achieving the spatiotemporal requirements through a novel combination of a time-aware feedback control mechanism and a spatial-aware non-deterministic geographic forwarding scheme. We evaluate SPEED using GloMoSim [22]. The performance results show that SPEED 1) reduces the number of packets that miss their end-to-end deadlines, 2) reacts to transient congestion in the most stable manner, and 3) efficiently handles voids [12] with minimal control overhead. To demonstrate its applicability, we also implement SPEED on the Berkeley motes [3]. The results show that SPEED helps balance the traffic load to increase the system lifetime.

• *Tian He is with Department of Computer Science, University of Virginia,Charlottesville, 22904. E-mail: tianhe@cs.virginia.edu*
• *John A. Stankovic is with Department of Computer Science, University of Virginia, Charlottesville, 22904. E-mail: stankovic@cs.virginia.edu*
• *Chenyang Lu is with Department of Computer Science & Engineering, Washington University in St Louis. E-mail: lu@cs.wustl.edu*
• *Tarek F. Adelzaher is with Department of Computer Science, University of Virginia, Charlottesville, 22904. E-mail: zaher@cs.virginia.edu*

Manuscript received Oct.29, 2003;, revised Sept 2,2004; accepted Jan 6,2005

2 State of the Art

Several routing protocols have been developed for ad hoc wireless networks. Sensor networks can be regarded as a sub-category of such networks, but with a number of different requirements.

In sensor networks, location is more important than a specific node's ID. For example, tracking applications only care where a target is located, not the ID of the reporting node. In sensor networks, such spatial-awareness [7] is necessary to make the sensor data meaningful. Therefore, it is natural to utilize spatial-aware routing. A set of location based routing algorithms have been proposed. Finn [5] proposed a greedy geographic forwarding protocol with limited flooding to circumvent the voids inside the network. GPSR [12] by Karp and Kung use perimeter forwarding to get around voids. Similarly, GOAFR [14] combines the greedy routing with the adaptive face routing to provide an asymptotically optimal path to the destination. Geographic distance routing (GEDIR) [20] guarantees loop-free delivery in a collision-free network. LAR [13] by Young-Bae Ko improves the efficiency of the on-demand routing algorithms by restricting packet flooding in a specified "request zone". Basagni, et. al. propose a distance routing algorithm [2] for mobility (DREAM), in which each node periodically updates its location information to other nodes. An updating rate is set according to a distance effect in order to reduce the number of control packets. Recently, Huang [4] et. al. proposed Mobicast protocol extended geocast by providing just-in-time information dissemination to nodes in a mobile delivery zone.

SPEED also utilizes geographic location to make localized routing decisions. The key difference is that SPEED takes timely delivery into account and is designed to be the first spatiotemporal-aware communication protocol for sensor networks. Moreover, SPEED provides an alternative solution to handle voids other than approaches based on planar graph traversal [12] [14] and limited flooding [5].

Reactive routing algorithms such as AODV [17] and DSR [10] maintain routing information for a small subset of possible destinations, namely those currently in use. If no route is available for a new destination, a route discovery process is invoked. Route discovery broadcasts can lead to significant delays in a sensor network with a large network diameter. This limitation makes on-demand algorithms less suitable for real-time applications.

Several real-time protocols have been proposed for ad hoc and sensor networks. SWAN [1] uses feedback information from the MAC layer to regulate the transmission rate of non-real-time TCP traffic in order to sustain real-time UDP traffic. RAP [16] uses velocity monotonic scheduling to prioritize real-time traffic and enforces such prioritization through a differentiated MAC Layer. V. Kanodia etc. [11] proposed a service differentiation for delay-sensitive traffic by prioritizing 802.11. Woo and Culler [21] proposed an adaptive MAC layer rate control to achieve fairness among nodes with different distances to the base station. All of these algorithms work well by locally degrading a certain portion of the traffic. However, this kind of local MAC layer adaptation cannot handle long-term congestion where routing assistance is necessary to divert traffic away from any hotspot. SPEED provides a combination of MAC layer and network layer adaptation that effectively deals with such issues. To the best of our knowledge, no routing algorithm has been specifically designed to provide soft real-time guarantees under spatiotemporal constraints for sensor networks.

3 Design Goals

The key design goal of the SPEED algorithm is to support a spatiotemporal communication service with a desired delivery *speed* across the sensor network, so that end-to-end delay is proportional to the distance between the source and destination. It should be noted that delivery speed refers to the approaching rate along a straight line from the source toward the destination. Unless the packet is routed exactly along that straight line, delivery speed is smaller than the actual speed of the packet in the network. For example, if the packet is routed in the opposite direction from the destination, its speed is negative. Our algorithm ensures that this condition never occurs.

More specifically, SPEED satisfies the following design objectives.

Soft Real-Time: We define the soft real-time guarantee provided by SPEED as delay guarantee per unit delivery distance (speed guarantee). Under this guarantee, we can obtain a predictable end-to-end communication delay under given spatial (distance) constraints before hand. Consequently, applications can make admission control to deliver packets that are able to meet end-to-end deadlines.

Minimal State Architecture: The physical limitations of sensor networks, such as large scale, high failure rate, and constrained memory capacity necessitate a minimal state approach. SPEED only maintains immediate neighbor information. It does not require a routing table as in DSDV [18] nor per-destination states as in AODV [17]. Thus, its memory requirements are minimal.

Minimum MAC Layer Support: SPEED does not require real-time MAC support. The feedback control scheme employed in SPEED allows all existing best effort MAC layers.

QoS Routing and Congestion Management: Most reactive routing protocols can find routes that avoid network hot spots during the route acquisition phase. Such protocols work well when traffic patterns do not fluctuate during a session. However, these protocols (e.g. [10]) are less successful when congestion patterns change rapidly compared to the session lifetime. When a route becomes congested, such protocols either suffer a delay or initiate another round of route discovery. As a solution, SPEED uses a novel backpressure re-routing scheme to re-route packets around large-delay links with minimum control overhead.

Traffic Load Balancing: In sensor networks, the bandwidth and energy are scarce resources compared to a wired network. Because of this, it is valuable to utilize several simultaneous paths to carry packets from the source to the destination. SPEED uses non-deterministic forwarding to balance each flow among multiple concurrent routes.

Localized Behavior: Pure localized algorithms are those in which any action invoked by a node should not affect the system as a whole. In algorithms such as AODV, DSR and TORA, this is not the case. In these protocols, a node uses flooding to discover new paths. In sensor networks where thousands of nodes communicate with each other, broadcast storms may result in significant power consumption and possibly a network meltdown. To avoid that, all distributed operations in SPEED are localized to achieve high scalability.

Void Avoidance: In some scenarios, pure greedy geographic forwarding may fail to find a greedy path to the destination, even when one actually exists. SPEED handles the void the same way as it handles congested areas and guarantees that if there is a greedy route between the source and destination, it will discover it.

Note, while SPEED does not use routing tables, SPEED does utilize location information to carry out routing. Thus, we assume that each node is location-aware [7].

4 SPEED Protocol

SPEED maintains a desired delivery speed across sensor networks with a two-tier adaptation included for diverting traffic at the networking layer and locally regulating packets sent to the MAC layer. It consists of the following components:

- An API
- A delay estimation scheme
- A neighbor beacon exchange scheme
- A Non-deterministic Geographic Forwarding algorithm (NGF)

- A Neighborhood Feedback Loop (NFL)
- Backpressure Rerouting
- Last mile processing

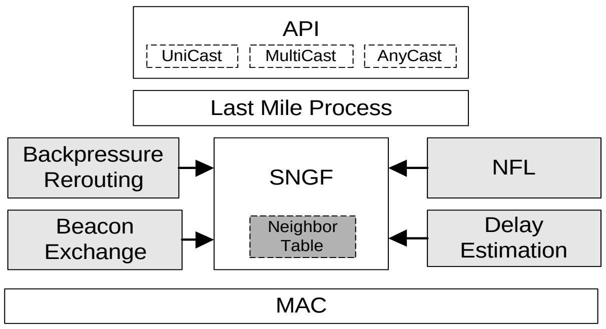


Figure 1: SPEED Protocol

As shown in Figure 1, NGF is the routing module responsible for choosing the next hop candidate to support the desired delivery speed. NFL and Backpressure Rerouting are two modules to reduce or divert traffic when congestion occurs, so that NGF has available candidates from which to choose. The last mile process is provided to support three types of real-time communication services, namely, real-time unicast, real-time area-multicast and real-time area-anycast, for sensor networks. Delay estimation is the mechanism by which a node determines whether congestion has occurred. And beacon exchange provides geographic location of the neighbors so that NGF can perform geographic based routing. The details of these components are discussed in the subsequent sections, respectively.

4.1 Application API and Packet Format

The SPEED protocol provides four application-level API calls:

- AreaMulticastSend (position, radius, packet): This service identifies a destination area by its center position and radius. It sends a copy of the packet to every node inside the specified area with a speed above a certain desired value.
- AreaAnyCastSend (position, radius, packet): This service sends a copy of the packet to at least one node inside the specified area with a speed above a certain desired value.
- UnicastSend(Global_ID, packet): In this service the node identified by Global_ID receives the packet with a speed above a certain desired value.
- SpeedReceive(): this primitive permits nodes to accept packets targeted to them.

There is a single data packet format in SPEED. It contains the following major fields:

- PacketType: the type of communication -- Area Multicast, AreaAnyCast or Unicast.
- Global_ID: only used in Unicast communication to identify a destination node.
- Destination Area: Describes a three-dimensional space with a center point and radius in which the packets are destined.
- TTL: Time To Live field is the hop limit used for last mile processing.
- Payload.

4.2 Delay Estimation

We use single hop delay as the metric to approximate the load of a node. We notice that the delays experienced by broadcast packets and unicast packets are quite different due to different handling inside the MAC layer. Unicast packet delay is more appropriate for making routing decisions. In a scarce bandwidth environment, we cannot afford to use probing packets to estimate the single hop delay. Instead, we use the data packets passing this node to perform this measurement. Delay estimation is done at the sender, which timestamps a packet ($T_{arriving}$) entering the tail of network output queue and time-stamps the packet ($T_{departure}$) when the last bit of this packet is sent out. Single trip delay equals the interval between $T_{arriving}$ and $T_{departure}$. Propagation delay is ignored. In case of transmission failures, $T_{departure}$ is set and used for calculation only at a successful transmission.

We compute the current delay estimation by combining the newly measured delay with previous delays via the exponential weighted moving average (EWMA) [15] as following:

$$Delay(k) = \alpha * Delay_{new} + (1-\alpha)*Delay(k-1)$$

We argue that this delay estimation is a better metric than average queue size for representing the congestion level of the wireless network, because the shared media nature of the wireless network allows the network to be congested even if queue sizes are small.

4.3 Neighbor Beacon Exchange

Similar to other geographic routing algorithms, every node in SPEED periodically broadcasts a beacon packet to its neighbors. This periodic beaconing is only used for exchanging location information between neighbors. We argue that the beaconing rate can be very low when nodes inside the sensor network are stationary or slow moving. Moreover, piggybacking [12] methods can also be exploited to reduce this beacon overhead.

In addition to periodic beaconing, SPEED uses an on-demand backpressure beacon, to quickly notify the upstream nodes of traffic changes inside the network. As shown in the evaluation (section 0), our on-demand beacon scheme introduces only a small overhead in exchange for a fast response to congestion.

In SPEED, each node keeps a neighbor table to store information passed by the beaconing. Each entry inside the table has the following fields: (NeighborID, Position, SendToDelay, ExpireTime). The ExpireTime is used to timeout this entry. If a neighbor entry is not refreshed after a certain timeout, it is removed from the neighbor table. SendToDelay is the delay estimation to the neighbor node identified by the NeighborID field. The details of obtaining this value have been discussed in previous section 0.

4.4 Non-deterministic Geographic Forwarding

Before elaborating on Non-deterministic Geographic Forwarding (NGF), we first introduce three definitions:

The Neighbor Set of Node *i*: NS_i is the set of nodes within the radio range of node *i*. Note, we do not assume that the radio is a perfect circle. SPEED works with irregular radio patterns.

The Forwarding Candidate Set of Node *i*: A set of nodes that belong to NS_i and are closer to the destination. Formally, $FS_i(Destination) = \{node \in NS_i \mid L - L\_next > 0\}$ where *L* is the distance from node *i* to the destination and *L_next* is the distance from the next hop forwarding candidate to the destination. These nodes are inside the cross-hatched shaded area as shown in Figure 2. We can easily obtain FS_i *(Destination)* by scanning the *NS* set of nodes once.

It is worth noticing that the membership of the neighbor set only depends on the radio range, but the membership of the forwarding set also depends on destination area.

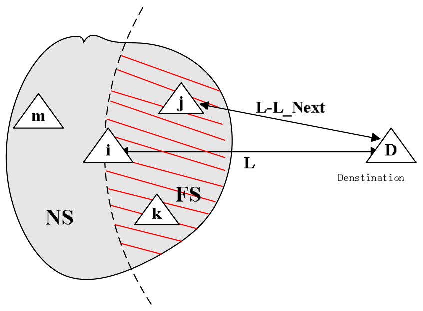


Figure 2: NS and FS definitions

Relay Speed. Relay speed is calculated by dividing the advance in distance from the next hop node j by the estimated delay to forward a packet to node j. Formally,

$$Speed_i^j(Destination) = \frac{L - L\_next}{HopDelay_i^j}$$

Since in SPEED nodes keep the Neighbor Set (NS), but do not keep a routing table or flow information, the memory requirements are only proportional to the number of neighbors.

Based on the destination of the packet and the current *FS*, the Non-deterministic Geographic Forwarding (NGF) portion of our protocol routes the packets according to the following rules:

Packets are forwarded only to the nodes that belong to the FS_i *(Destination)*. If there is no node inside the FS_i *(Destination*), packets are dropped and a backpressure beacon is issued to upstream nodes to prevent further drops (for more details see 0). To reduce the chance of such drops, we provide a lower bound of node density that can virtually eliminate these drops. The theoretical analysis on this issue is provided in section 0.

SPEED divides the neighbor nodes inside FS_i *(Destination)* into two groups. One group contains the nodes that have relay speeds larger than a certain desired speed $S_{setpoint}$, the other contains the nodes that cannot sustain such desired speed. The $S_{setpoint}$ is a system parameter that depends on the communication capability of the nodes and desired traffic workload a sensor network should support. For given bandwidth *T*, packet size *L* and Radio Range *R*, following inequality should hold:

$$0 \le S_{setpoint} \le RT/L$$

The forwarding candidate is chosen from the first group, and the neighbor node with highest relay speed has a higher probability to be chosen as the forwarding node. To trade off between load balancing and optimal delivery delay, we use following discrete exponential distribution function:

$$P(x = i) = \frac{(Speed_i)^K}{\sum_{i=0}^{N} (Speed_i)^K} \quad 1 \le i \le N$$

In this distribution function, *N* is the number of forwarding candidates inside the first group. *K* is used to trade off between load balance and optimal delivery delay. A larger *K* value leads to a shorter end-to-end delay; while a smaller *K* value achieves a better load balance.

If there are no nodes belonging to the first group, a relay ratio is calculated based on the Neighborhood Feedback Loop (NFL), which is discussed in more detail in section 0. Whether a packet drop will really happen depends on whether a randomly generated number between (0,1) is bigger than the relay ratio. In SPEED a packet is dropped only when no downstream node can guarantee the single hop speed set point $S_{setpoint}$ and dropping packets must be performed to reduce the congestion. Though one can consider buffering packets as an alternative to the dropping, however, we argue that under real-time and small memory constraints, dropping is often a better choice.

NGF provides two nice properties to help meet our design goals. First, since NGF sends packets to the downstream node capable of maintaining the desired delivery speed, soft real-time end-to-end delivery is achieved with a theoretical delay bound: *Delay Bound* = $L_{e2e}/S_{setpoint}$, where L_{e2e} is the distance between the source and destination. $S_{setpoint}$ is the uniform speed to be maintained across the sensor network. Second, NGF can balance traffic and reduce congestion by dispersing packets into a large relay area. This load balancing is valuable in a sensor network where the density of nodes is high and the communication bandwidth is scarce and shared. Load balancing also balances the power consumption inside the sensor networks to prevent some nodes from dying faster than others.

NGF provides MAC layer adaptation and reduces the congestion by locally dropping (or optionally buffering) packets. This adaptation is good enough to deal with transient overshoot inside the sensor networks. But if such congestion remains for a relatively long time, network layer adaptation is desired to redirect traffic to a less congested area, which is discuss further in section 0.

4.5 Neighborhood Feedback Loop (NFL)

The Neighborhood Feedback Loop (NFL) is the key component in maintaining the single hop relay speed. The NFL is an effective approach to maintaining system performance at a desired value. This has been shown in [19], where a low miss ratio of real-time tasks and a high utilization of the computational nodes are simultaneously achieved. Here we want to maintain a single hop relay speed above a certain value $S_{setpoint}$, a performance goal desired by the designer.

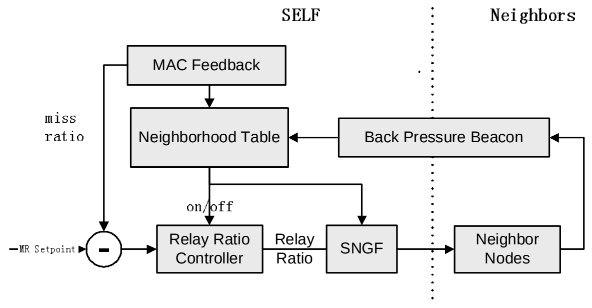


Figure 3: Neighborhood Feedback Loop (NFL)

We deem it a miss when a packet delivered to a certain neighbor node has a relay speed less than $S_{setpoint}$, or if there is a loss due to collision. The percentage of such misses is called this neighbor's miss ratio. The responsibility of the NFL is to force the miss ratios of the neighbors to converge to a set point, namely zero.

As shown in Figure 3, the MAC layer collects miss information and feeds it back to the Relay Ratio controller. The Relay Ratio controller calculates the relay ratio and feeds that into the NGF where a drop or relay action is made. The Relay Ratio controller currently implemented is a multiple inputs single output (MISO) proportional controller that takes the miss ratios of its neighbors as inputs and proportionally calculates the relay ratio as the output to the NGF. Formally it is described by the following formulas.

$$u = 1 - K\frac{\sum e_i}{N} \quad if \;\; \forall e_i > 0$$

$$u = 1 \quad if \;\; \exists e_i = 0$$

where e_i is the miss ratio of the neighbor i inside the *FS* set, N is size of the *FS* set. u is the output (relay ratio) to NGF. And K is the proportional gain.

It should be noted that the Relay Ratio controller is activated only when all nodes inside the forwarding set (*FS*) cannot maintain the desired single hop relay speed $S_{setpoint}$ and a drop is absolutely necessary to maintain the single hop delay. Such a scheme ensures that re-routing has a higher priority than dropping. In other words, SPEED does not drop a packet as long as there is another path that can meet the delay requirements.

By reducing the sending rate to the downstream nodes, the neighborhood feedback loop can maintain a single hop relay speed. However, this MAC layer adaptation can't solve the hotspot problem, if the upstream nodes, which are unaware of the congestion, keep sending packets into this area. In this case, backpressure rerouting (network layer adaptation) is necessary to reduce the traffic injected into the congested area.

4.6 Back-Pressure Rerouting

Backpressure re-routing is naturally generated from the collaboration of neighbor feedback loop (NFL) routines as well as the non-deterministic geographic forwarding (NGF). To be more explicit, we introduce this scheme with an example (Figure 4).

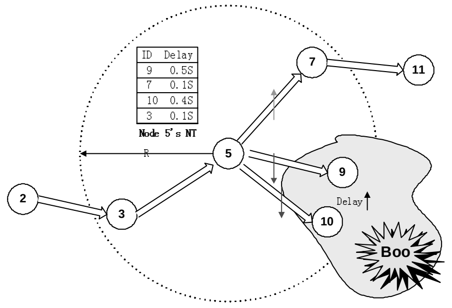


Figure 4: Backpressure rerouting case one

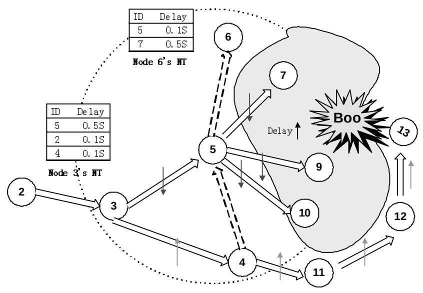


Figure 5: Backpressure rerouting case two

Suppose in the lower-right area, heavy traffic appears, which leads to a low relay speed in nodes 9 and 10. Through the MAC layer feedback, node 5 detects that nodes 9 and 10 are congested. Since NGF reduces the chance of selecting nodes 9 and 10 as forwarding candidates and routes more packets to node 7, it reduces the congestion around nodes 9 and 10. Since all neighbors of 9 and 10 react the same way as node 5, eventually nodes 9 and 10 are able to relay packets above the desired speed.

A more severe case could occur when all the forwarding neighbors of node 5 are also congested as shown in Figure 5. In this case, the neighborhood feedback loop is activated to assist backpressure re-routing. In node 5, a certain percent of packets are dropped in order to reduce the traffic injected into the congested area. At the same time, an on-demand backpressure beacon is issued by node 5 with the following fields: (ID, Destination, AvgSendToDelay)

AvgSendToDelay is the average SendToDelay of all nodes inside FS_{ID}(Destination). In our example, when the destination is node 13, AvgSendToDelay is the average delay from node 5 to nodes 7, 9 and 10.

When a neighbor receives the back-pressure beacon from node 5, it determines whether node 5 belongs to its *FS*(Destination). If node 5 does, this neighbor modifies the SendToDelay for node 5 according to the AvgSendToDelay. For example only node 3 considers node 5 as a next hop forwarding candidate to the destination where node 13 resides. If node 5 is not in the *FS*(Destination), then this neighbor ignores the backpressure beacon. This backpressure mechanism can reduce the chance of "false congestion indication", to ensure that traffic from node 4 to node 6 is not affected by the backpressure beacon.

If, unfortunately, node 3 is in the same situation as node 5, further backpressure is imposed on node 2. In the extreme case, the whole network is congested and the backpressure proceeds upstream until it reaches the source, where the source quenches the traffic flow to that destination.

Backpressure rerouting is a network layer adaptation used by SPEED to reduce the congestion inside the network. In this case no packet needs to be sacrificed. Network layer adaptation has a higher priority than MAC layer adaptation used by NGF and NFL. A drop via the feedback loop is only necessary when the situation becomes so congested and there is no alternative to maintaining a single hop speed other than dropping packets.

4.7 Void Avoidance

Greedy geographic based algorithms have many advantages over the traditional MANET routing algorithms for real-time sensor network applications. They do not suffer route discovery delay and tend to choose the shortest path to the destination. Moreover without flooding, they have relatively low control packet overhead. Unfortunately, they also have a serious drawback. In many cases, they may fail to find a path even though one does exist. To overcome this, SPEED deals with a void the same way it deals with congestion. As shown in the Figure 6, if there is no downstream node to relay packets from node 2 to node 5, node 2 sends out a backpressure beacon containing fields: (ID, Destination, ∞). The upstream node 1 that needs node 2 to relay the packets to that destination sets the SendToDelay for node 2 to infinity and stop sending packets to node 2. If node 3 does not exist, further backpressure occurs until a new route is found. It should be admitted that our scheme of void avoidance isn't guaranteed to find a path if there is one as in GPSR[12], but it is guaranteed to find a *greedy* path if one exists. To maintain real-time properties, we do not allow backtracking to violate our desired speed setpoint. However, as we can see from the evaluation section 0, such a simple scheme can significantly reduce packet loss due to voids in high-density sensor networks.

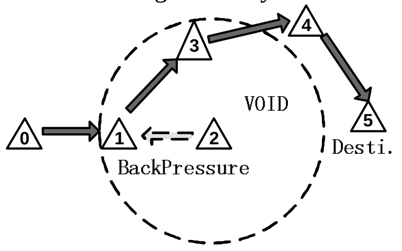


Figure 6: Void avoidance scheme

4.8 Last Mile Process

Since SPEED is targeted at sensor networks where the ID of a sensor node is not important, SPEED only cares about the location where sensor data is generated.

The last mile process is so called because only when the packet enters into the destination area will such a function be activated. The NGF module mentioned above controls all previous packet relays.

The last mile process provides two novel services that fit the scenario of sensor networks: Area-multicast and Area-anycast. The area in this case is defined by a center-point (x,y,z) and a radius, in essence a sphere. More complex area definitions can be made without jeopardizing the design of this last mile process.

Nodes can differentiate the packet type by the PacketType field mentioned in section 4.1. If it is an anycast packet, the nodes inside the destination area deliver the packet to the transport layer without relaying it onward. If it is a multicast packet, the nodes inside the destination area which first receive the packet coming from the outside of the destination area set a TTL. This allows the packet to survive within the diameter of the destination area and be broadcast within a specified radius. Other nodes inside this destination area keep a copy of the packet and re-broadcast it. The nodes that are outside the destination area just ignore it. The last mile process for unicast is nearly the same as multicast, except that only the node with a specified global_ID inside the destination area delivers the packet to the transport layer. If the location service is precise, the estimated destination area for a given node will be much smaller than single radio coverage. As a result, additional flooding overhead for the unicast packets is negligible (sometimes zero). We note that the current implementation of the last mile process is relatively straightforward. More efficient and robust techniques are desired for future research.

5 Protocol Analysis

This section provides a protocol analysis of several practical issues related to the SPEED protocol.

5.1 Impact of Node Density

One basic assumption of sensor networks is their relatively high node density. It is an interesting research issue to determine the impact of node density on routing performance. Specifically, in the geographic based algorithm, we want to find the lower bound of node density that can probabilistically guarantee no void that can prevent a greedy geographic forwarding step from happening.

In GF based algorithms, a node forwards packets to next hop nodes that are nearer to the destination. The area where such qualified nodes reside is called the *forwarding area (FA)*.

Assume the nodes are randomly distributed inside the system, the larger the size of the forwarding area, the higher is the probability that there is a candidate to be chosen. In fact, the forwarding area size is not constant; it is depends on how far away the sending node is from the destination node.

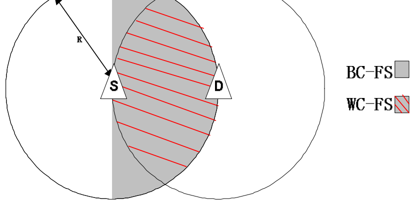


Figure 7: Forwarding Areas

As shown in Figure 7, when the destination node is infinitely far away from the sending node, the forwarding area is the largest (Best Case Forwarding Size) and when the destination node is exactly R away from the sending node, the available forwarding size is the worst case forwarding size (WCFS). For guaranteeing purposes, we only consider the worst case, even though most of the time the forwarding size is nearer to the best case. In the worst case, the forwarding size is calculated by formula (1). For the purpose of analysis, here we use R as a nominal radio radius.

$$WCFS = \left[2\cos^{-1}\frac{1}{2} - \frac{\sqrt{3}}{2} \right] R^2 \tag{1}$$

Now, we consider the worst case forwarding area. We desire to know the lower bound of node density that satisfies the following condition:

P (At least one node inside the FA) > 1-ε (2)

Equivalently,

P (No nodes inside the FA) <= ε (3)

Assuming a uniform distribution, according to (3) the following condition must hold: (the size of the area covered by the sensor network is denoted by AreaSize >> WCFS)

$$(1 - \frac{WCFS}{AreaSize})^{AreaSize \times Density} \leq \varepsilon \tag{4}$$

Since the left hand side of the inequality is a monotonically increasing function when the AreaSize increases and monotonically decreasing when node density increases, the lower bound of the node density is achieved when AreaSize is infinite:

$$\lim(1 - \frac{WCFS}{AreaSize})^{AreaSize \times Density} = e^{-WCFS \times Density} \leq \varepsilon$$

$$\text{Hence: } Density \geq \frac{-\ln \varepsilon}{WCFS} \tag{5}$$

As for the greedy geographic based routing algorithm without backpressure, we must guarantee that for every hop they can find a forwarding candidate. More formally, to guarantee:

P(successfully deliver packets to a destination through #hop greedy forwarding) >= 1- ε (6)

Assume voids follow an identical independent distribution (iid), equivalently:

$[\text{P (At least one node inside the FA)}]^{\#hop} > 1- \varepsilon$ (7)

Follows the same derivation from (2) to (5), we get the lower bound of node density:

$$Density \geq \frac{-\ln\left(1 - \sqrt[\#hop]{1-\varepsilon}\right)}{WCFS} \tag{8}$$

Figure 8 shows the lower bound of node densities that can probabilistically guarantee that there is no void that can prevent greedy routing under different ε values and lengths of the routes. For example, for a 10-hop route, it is statistically guaranteed that 99% delivery ratio in worst case if the node density inside networks is above 16 node/ nominal range.

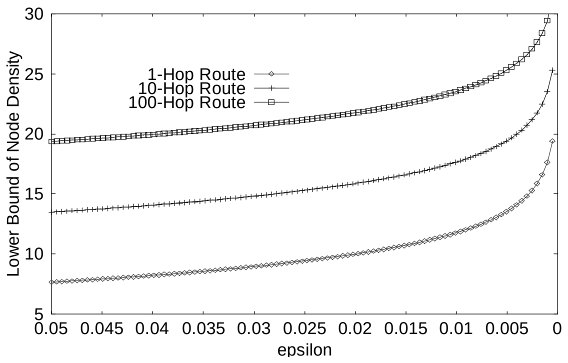


Figure 8: Lower bound of node densities

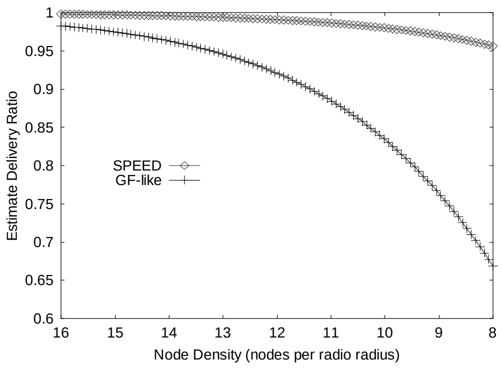


Figure 9: Estimate Deliver ratio

On the other hand, we need to guarantee there exist a greedy path to the destination for the SPEED protocol. We observe that SPEED can't enforce backpressure at the source node, where no upstream node exists. After the first hop relay, the backpressure effectively reduces packet lost due to the void at subsequent hopes as mentioned in section 0. To simplify the analysis, we only consider first hop loss due to the void. This approximation slightly overestimates the delivery ratio of SPEED and serves as an upper bound. According to inequality 7, Figure 9 plots the estimate delivery rate under different node densities.

We note that the result we obtained from the formal analysis (Figure 9) is quite similar to the results obtained through simulation (section 0); For example, both simulation and formal analysis have about 95% delivery ratio for SPEED at the density of 8 nodes per radio circle.

5.2 Analysis of Localization Impact

Theoretically, it is desirable for location-based routing algorithms to have a perfect localization service; however, in practical, in order to obtain higher location accuracy, systems have to increase the cost of the localization via sophisticated devices or additional communication overhead. Although more accurate location information is preferable, the desired level of granularity should depend on a cost/benefit analysis of the protocols that utilize this information. In this section, we investigate the impact of localization errors on the SPEED protocol. Specifically, we investigate the pseudo void problem caused by localization errors, which leads to routing failures in SPEED and other location-based routing algorithms.

Location-based routings normally follow a greedy forwarding rule, as long as there is at least a node inside forwarding area. Due to the localization error, some nodes, which are actually located inside the forwarding area of a sender, might be mistaken by the sender to be outside (Figure 10A).

More specifically, the pseudo void problem happens when all nodes inside the forwarding area get localization results that are outside of the forwarding area of the sender. We note that the forwarding area can be shifted by the localization error of the sender. Assuming that the localization error of each node follows an identical independent distribution (iid),

$$\text{P (pseudo void)} = \prod_{i=1}^{M} \text{P}\,(\mathit{Localization}\ (N_i) \notin \mathit{FA}\ (\mathit{Localization}\ (\mathit{sender}))\ |\ \text{Location}\ (N_i) \in \mathit{FA}(\mathit{Location}(\mathit{sender}))\ ) \tag{9}$$

Where *FA* is the forwarding area, N_i is an arbitrary node inside forwarding area and *M* is the number of nodes inside forwarding area.

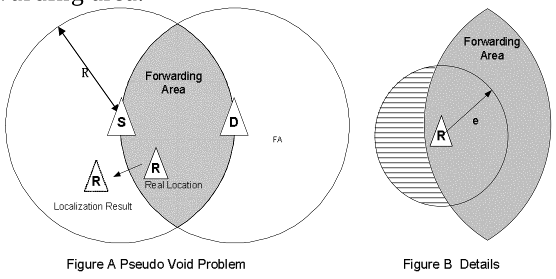


Figure 10: Pseudo Void Problem

Assume localization error is omni-directional and maximum distance between the estimated location and the real location is *e*. According to Figure 10B, P (pseudo void) equals:

$$\prod_{i=1}^{M}\left[\frac{1}{\pi e^2}\times\iint_{m^2+n^2\le e^2}\left[\iint_{FA(m,n)}\frac{Size_{dashedArea}}{\pi e^2}dxdy\right]dmd\right] \tag{10}$$

Where *FA (m,n)* denotes the forwarding area of the sender when the localization result of the sender is (m , n).

We obtain the value of equation (8) through numerical integration. We only consider the worst case in which the destination is exactly R away from sending node. The results are shown in Figure 11:

Figure 11 gives us insight on the relationship between node density (M denotes the number of nodes inside forwarding area), estimation error (in the unit of radio range) and the probability of the pseudo void problem occurring. For example in order to reduce chance of drop due to the pseudo void problem below 5%, the number of nodes inside the forwarding area should be equal to or larger than 3 when localization error is as much as half radio range. Based on the worse case forwarding area size given in equation (1), the corresponding node density should be 7.67 node/ nominal radio circle

$$\mathit{Density} \ge \frac{\pi R^2}{\mathit{WCFS}}\times 3 = 7.67\ \mathit{nodes}\ /\mathit{circle} \tag{11}$$

Figure 11 also demonstrates that when node density is sufficiently high (M = 5 & corresponding node density > 12.78), statistically the pseudo problem rarely happens (<0.2%) when the localization error below half radio range. This theoretical analysis is consistent with our simulation result in [7].

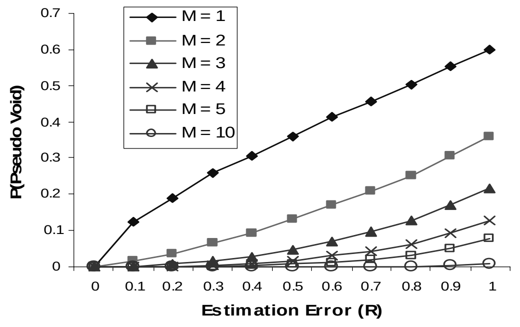


Figure 11: Probability of Pseudo Void Problem

6 Experimentation and Evaluation

We simulate SPEED on GloMoSim [22], a scalable discrete-event simulator developed by UCLA. This software provides a high fidelity simulation for wireless communication with detailed propagation, radio and MAC layers. Table 1 describes the detailed setup for our simulator. The communication parameters are mostly chosen in reference to the Berkeley Telos mote specification.

Table 1: Simulation settings

Routing	AODV, DSR, GF, GPSR, SPEED, SPEED-S, SPEED-T
MAC Layer	802.11 (Simplified DCF)
Radio Layer	RADIO-ACCNOISE
Propagation model	TWO-RAY
Bandwidth	200Kb / s
Payload size	32 Byte
TERRAIN	(200m, 200m)
Node	100 Nodes, Uniform placement
Radio Range	40m

In our evaluation, we compare the performance of seven different routing algorithms: AODV [17], DSR [10], GF [20], GPSR [12], SPEED, SPEED-S, SPEED-T. We adopt both ad hoc routing protocols (AODV and DSR) and sensor network protocols (GF, GPSR).

GF forwards a packet to the node that makes the most progress toward the destination. GPSR has identical performance as GF when network density is relatively high, however it achieves better delivery ratios in sparse networks. SPEED-S and SPEED-T are reduced versions of SPEED. SPEED-S replaces the NGF with a MAX-SPEED routing algorithm that geographically forwards the packets to nodes that can provide a max single hop relay speed. SPEED-T replaces the NGF with a MIN-DELAY routing algorithm that geographically forwards packets to nodes that have a minimum single hop delay. Both reduced versions have no backpressure rerouting mechanisms.

In our evaluation, we present the following set of results: 1) end-to-end delay under different congestion levels, 2) miss ratio, 3) control overhead, 4) communication energy consumption, and 5) packet delivery ratio under different node densities. All experiments are repeated 16 times with different random seeds and different random node topologies. We also implement SPEED on the Berkeley. The results obtained from this testbed show a load balance feature of SPEED protocol (see section 0).

6.1 Sensor Network Traffic Pattern

There are two typical traffic patterns in sensor networks: a base station pattern and a peer-to-peer pattern. The base station pattern is the most representative one inside sensor networks. For example, in surveillance systems, multiple sensors detect and report the location of an intruder to the control center. In tracking systems, a base station issues multiple tracking commands to a group of pursuers. In a different respect, the peer-to-peer pattern is usually used for data aggregation and consensus in a small area where a team of nearby motes interact with each other. The end-to-end delay in the base station pattern is the major part of delay for the sensing-actuation loop, and is therefore, the focus of our evaluation.

6.2 Congestion Avoidance

In a sensor network, where node density is high and bandwidth is scarce, traffic hot spots are easily created. In turn, such hot spots may interfere with real-time guarantees of critical traffic in the network. In SPEED, we apply a combined network and MAC layer congestion control scheme to alleviate this problem.

To test the congestion avoidance capabilities, we use a base station scenario, where 6 nodes, randomly chosen from the left side of the terrain, send periodic data to the base station at the middle of the right side of the terrain. The average hop count between the node and base station is about 8~9 hops. Each node generates 1 CBR flow with a rate of 1 packet/second. To create congestion, at time 80 seconds, we create a flow between two randomly chosen nodes in the middle of the terrain. This flow then disappears at time 150 seconds into the run. This flow introduces a step change into the system, which is an abrupt change that stress-tests SPEED's adaptation capabilities to reveal its transient-state response. In order to evaluate the congestion avoidance capability under different congestion levels, we increase the rate of this flow step by step from 0 to 100 packets/second over several simulations

Figure 12 and Figure 13 plot the end-to-end (E2E) delay for the six different routing algorithms. At each point, we average the E2E delays of all the packets from the 96 flows (16 runs with 6 flows each). The 90% confidence interval is within 2~15% of the mean, which is not plotted for the sake of legibility.

Under the no or light congestions, Figure 12 and Figure 13 show that all geographic based routing algorithms have short average end-to-end delay in comparison to AODV and DSR. There are several factors accounting for this outcome. First, the route acquisition phase in AODV and DSR leads to significant delays for the first few packets, while geographic based routing doesn't suffer from this. We argue that without an initial delay cost, geographic based routing is more suitable for real-time applications like target tracking where the base station sends the actuation commands to the sensor group, which is dynamically changing as the target moves. In such a scenario, DSR and AODV need to perform route acquisition repeatedly in order to track the target. Second, the route discovered through flooding and path reversal has relatively more hops than greedy geographic forwarding. The reason for even higher delay in AODV than DSR is that DSR implementation intensively uses a route cache to reduce route discovery and maintenance cost. As shown in Figure 13, SPEED-T has higher delay than GF, SPEED-S and SPEED, because SPEED-T only uses hop delay to make routing decision and disregards the progress each hop makes, which leads to more hops to the destination in wireless multi-hop networks. Instead, under lightly congested situation, GF, SPEED-S and SPEED tend to forward a packet at each step as close to the destination as possible, thereby reducing the number of hops and the end-to-end delay.

Under the heavy congested situations (Figure 12 and Figure 13), each routing algorithm responds differently. SPEED performs best. For example, SPEED reduces the average end-to-end delay by 30%~40% in the face of heavy congestion in comparison to the other algorithms considered. The key reasons for SPEED's better performance are 1) DSR, AODV and GF only respond to severe congestion, which leads to link failures (i.e., when multiple retransmissions fail at the MAC layer). They are insensitive to long delays as long as no link failures occur. 2) DSR, AODV and GF routing decisions are not based on the link delays, and therefore may cause congestion at a particular receiver even though it has long delays. 3) DSR and AODV flood the network to rediscover a new route when the network is already congested. 4) SPEED-T and SPEED-S do not provide traffic adaptation. When all downstream nodes are congested, SPEED-T and SPEED-S cannot reduce or redirect the traffic to uncongested routes. 5) SPEED not only locally reduces the traffic through a combination of NGF and Neighborhood Feedback loops in order to maintain the desired speed, but also diverts the traffic into a large area through its backpressure rerouting mechanism. This combination leads to lower end-to-end delay.

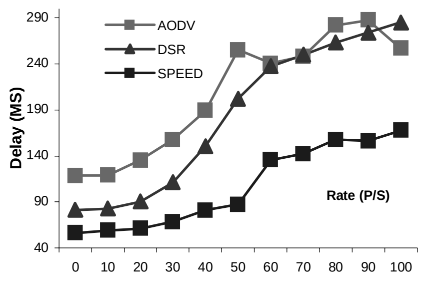


Figure 12: E2E Delay Vs. Congestion

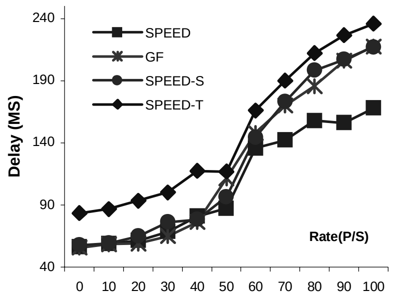


Figure 13: E2E Delay Vs. Congestion

6.3 E2E Deadline Miss Ratio

The deadline miss ratio is the most important metric in soft real-time systems. We set the desired delivery speed $S_{setpoint}$ to 1km/s, which leads to an end-to-end deadline of 200 milliseconds. In the simulation, some packets are lost due to congestion or forced-drops. We also consider this situation as a deadline miss. The results shown in Figure 14 and Figure 15 are the summary of 16 randomized runs.

AODV and DSR don't perform well in the face of congestion because both algorithms flood the network in order to discover a new path when congestion leads to link failure. This flooding just serves to increase the congestion. GF only switches the route when there are link failures caused by heavy congestion. The routing decision is based solely on distance and does not consider delay. SPEED-T only considers the single hop delay and doesn't take distance (progress) into account, which leads to a longer route. SPEED-S provides no adaptation to the congestion and cannot prevent packets from entering the congestion area. Only SPEED tries to maintain a desired delivery speed through MAC and network layer adaptations, and therefore has a much less miss ratio than other algorithms. Due to its transient behavior, SPEED still has about a 20% miss ratio when the network is heavily congested. Future work is needed to reduce the convergence time in order to improve the performance.

Comparing Figure 14 and Figure 15, we argue that purely localized algorithms without flooding outperform other algorithms when traffic congestion increases. Generally, the less state information a routing algorithm depends on, the more robust it is in the face of packet loss and congestion.

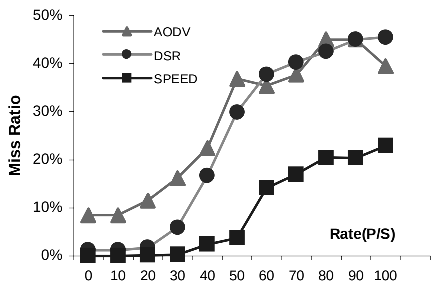


Figure 14: MissRatio Vs. Congestion

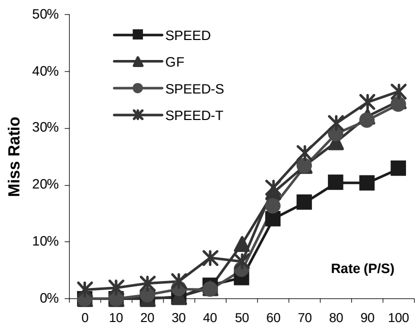


Figure 15: MissRatio Vs. Congestion

6.4 Control Packet Comparison

Except for AODV, all other routing algorithms studied use a relatively low number of control packets. Most control packets in DSR and AODV are used in route acquisition. Because AODV initiates route discovery (flooding) whenever a link breaks due to congestion, it requires a large number of control packets. DSR uses a route cache extensively, so it can do route discovery and maintenance with a much lower cost than AODV. The only control packets used in GF, SPEED-S and SPEED-T (Figure 16) are periodic beacons, whose number is constant at 750 under different congestion levels. In addition to periodic beacons, SPEED uses two types of on-demand beacons to notify neighbors of the congestion. This costs SPEED more control packets than the other three geographic based routing algorithms (Figure 16).

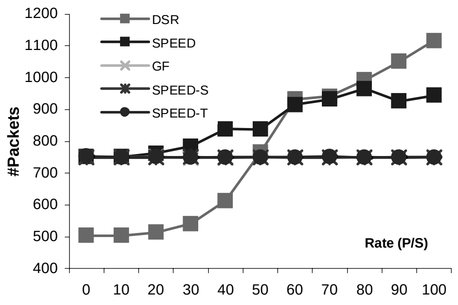


Figure 16: Control overhead comparison

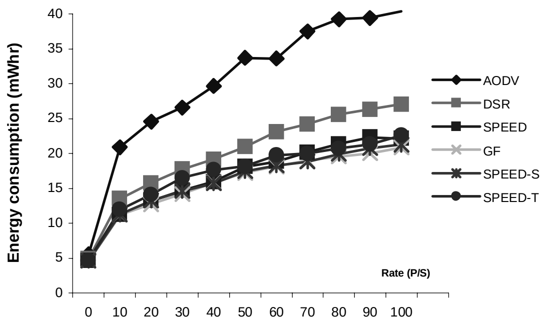


Figure 17: Transmission Energy Consumption

6.5 Energy Consumption

Under energy constraints, it is vital for sensor nodes to minimize energy consumption in radio communication to extend the lifetime of sensor networks. From the results shown in Figure 17, we argue that geographic based routing tends to reduce the number of hops in the route, thus reducing the energy consumed for transmission. AODV performs the worst as a consequence of sending out many control packets during congestion. DSR has larger average hop counts and more control packets than other geographic base routing algorithms. SPEED-T only takes delay into account, which leads to longer routes. Figure 17 shows that SPEED has nearly the same power consumption as GF and SPEED-S when the network is not congested. Under such situations, SPEED tends to choose the shortest route and does not require any on-demand beacons. Under heavy congestion, SPEED has slightly higher energy consumption than GF and SPEED-S, mainly because SPEED delivers more packets to the destination than the other protocols when heavily congested.

6.6 Node Density Impact

The typical density of a sensing-covered sensor network system [8] is about 20~30 nodes/radio range in order to provide high fidelity in localization, detection and tracking. In the previous evaluations, we use a 12 nodes/radio range as a typical setting. However, it is important to understand how SPEED performs under very low-density settings.

This experiment evaluates the end-to-end delivery ratio of all routing algorithms under different node densities. To eliminate packet loss due to congestion, we only use four flows with a rate of 0.5 packets/second, these flows go from the left side of the terrain to the base station at the right side of the terrain. To change the density of the network, instead of increasing the number of nodes in the terrain, we keep the number of nodes constant at 100, and increase the side length of the square terrain from 180 to 250 in steps of 5 meters. It is no surprise that DSR performs best in the delivery ratio, since it uses flooding to discover the route. Theoretically, DSR should have 100% delivery ratio (Figure 18) as long as the network is not partitioned. All other geographic based algorithms have 100% delivery ratio when the network has high density (>12 nodes / per radio range). However, when the network density is reduced below 9 nodes/ per radio range, GF, SPEED-S and SPEED-T degrade performance rapidly. Only SPEED manages to deliver 95% of its packets to the destination. It should be pointed out that as shown in Figure 18, GPSR [12], another well-known geographic based routing algorithm, permits backtracking and can achieve 100% delivery rate as long as the network is not partitioned. However, SPEED drops 5% of its packets, because these packets need backtracking in order reach the destination. If these packets were to backtrack, these packets would have a negative delivery speed. This is not allowed by SPEED for the sake of maintaining the real-time properties, which is not supported by GPSR. We note that GPSR defaults to the GF protocol when node density is high. The E2E deadline miss ratio of GF shown in Figure 15 is significantly higher than SPEED when the network becomes congested.

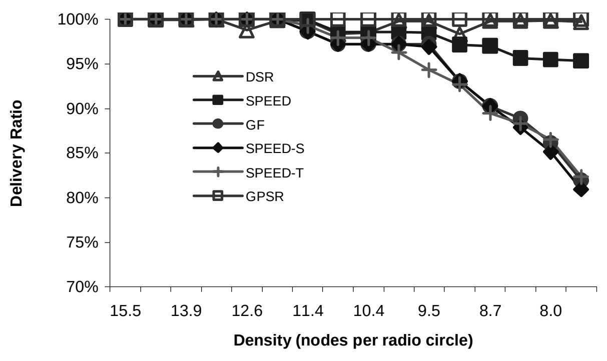


Figure 18: Deliver ratio Vs. densities

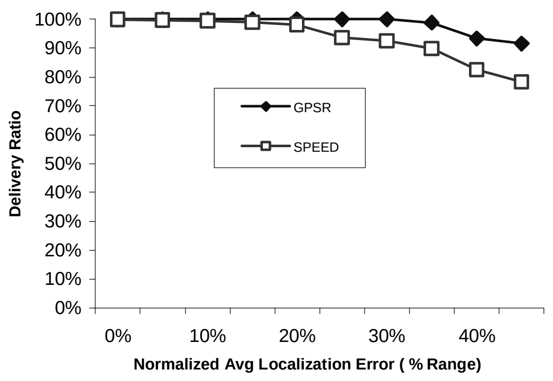


Figure 19: Delivery Ratio Vs. Loc Errors

6.7 Location Error Impact

While most work in location-based routing assumes perfect location information, the fact is that erroneous location estimates are virtually impossible to avoid. In this experiment, we investigate the tolerable localization error bound for the SPEED protocol. To prevent congestion, and therefore isolate the effects of localization error, the traffic loads are set to the rate of 1 packet/second. We increase the localization error from 0% to 50% of the radio range in steps of 5% to measure the end-to-end delivery ratio. Figure 19 shows that when the localization error is below 20% of the radio range, SPEED can achieve almost 100% delivery. We note that because the GPSR protocol allows backtracking, GPSR is more flexible in dealing with location error impact; hence, it achieves a slightly better performance in this case, as shown in Figure 19.

6.8 Implementation on Motes

We have implemented the SPEED protocol on the Berkeley motes platform with a code size of 6036 bytes (code is available at [6]). Three applications including data placement, target tracking and CBR are built on top of SPEED.

Due to the physical limitations of the motes, it is extremely difficult to perform as extensive evaluation as we did in the wireless simulator. Considering the space limitation, we only present partial results here as a study in developing a more complete solution on a mote testbed. In the experiment, we use 25 motes to form a 5 by 5 grid. To evaluate the load balancing capability of SPEED, we send a CBR flow from node 24 to node 0 which is the base station. We collect the number of packets relayed by intermediate motes (1~23) and compare this with the result obtained from the GF protocol which we also implemented on the motes.

GF relays packets via a fixed route, which leads to unbalance traffic, for example, in Figure 20, node 14 sends out 98 packets while node 13 does not sent out any packets. SPEED uses non-deterministic forwarding, which can balance energy consumption. We argue that in sensor networks, balanced energy consumption can prevent some nodes from dying faster than others, therefore increasing the network lifetime.

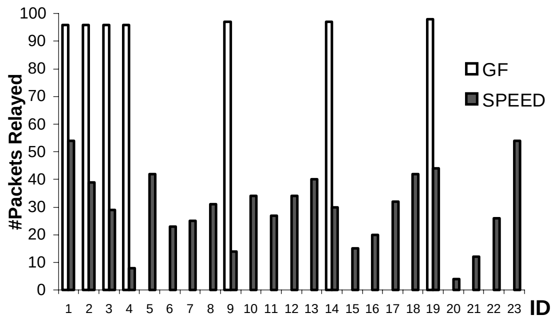


Figure 20: Traffic Balance

CONCLUSION

Many excellent protocols have been developed for ad hoc networks. However, sensor networks have additional requirements that were not specifically addressed. This paper is the first to address real-time requirements under spatial constraints by maintaining a desired delivery speed across the network through a novel combination of time-aware feedback control and spatial-aware non-deterministic geographic forwarding. The two-tier adaptation at both the MAC and network layers improves the end-to-end delay and provides good response to congestion and voids. Our simulations on GloMoSim and our implementation on Berkeley motes confirm SPEED's improved performance compared to DSR, AODV, GF, SPEED-S and SPEED-T.

ACKNOWLEDGMENT

This work was supported in part by the DAPRPA IXO offices under the NEST project (grant number F336615-01-C-1905), the MURI award N00014-01-1-0576 and NSF grant CCR-0098269.

REFERENCES

[1] G. S. Ahn, A. T. Campbell, A. Veres and L.H. Sun. SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks, In Proc. IEEE INFOCOM'2002, June 2002.

[2] S. Basagni and et. al. A Distance Routing Effect Algorithm for Mobility (DREAM). In ACM/IEEE MobiCom '98, October 1998.

[3] CrossBOW MICA2 Mote Specifications: http://www.xbow.com/.

[4] Q. Huang, C. Lu, and G. C. Roman. Spatiotemporal Multicast in Sensor Networks, In proceeding of SenSys'03, Los Angeles, CA, November 2003.

[5] G. G. Finn. Routing and Addressing Problems in Large Metropolitan-scale Internetworks. ISI/RR-87-180, USC/ISI, March 1987.

[6] T. He, L. Gu, B.Blum, Jun Xie. Nest Project Source Code http://sourceforge.net/projects/vert/

[7] T. He, C. Huang, B. M. Blum, J. A. Stankovic,and T. F. Abdelzaher, Range-Free Localization Schemes in Large Scale Sensor Networks, in MobiCom 2003, September 2003.

[8] T. He, S. Krishnamurthy, J. A. Stankovic, T. F. Abdelzaher and et.al, Energy-Efficient Surveillance System Using Wireless Sensor Networks, In MobiSys'04, Boston, MA, June 2004.

[9] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks," In proceeding of ICDCS 2003, Providence, RI, May 2003.

[10] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Mobile Computing, Chapter 5, pages 153-181, Kluwer Academic Publishers, 1996.

[11] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. W.Knightly, Distributed Multi-Hop Scheduling and Me-diumAccess with Delay and Throughput Constraints, In MobiCOM 2001, Rome, Italy, July 2001.

[12] B. Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In IEEE MobiCom, August 2000.

[13] Y.B. Ko and N. H. Vaidya. Location-Aided Routing(LAR) in Mobile Ad Hoc Networks. In IEEE MobiCom 1998, October 1998.

[14] F. Kuhn, R. Wattenhoffer and A. Zollinger. Worst-Case Optimal and Average-Case Efficient Geometric Ad-Hoc Routing, MobiHoc, June 2003.

[15] J. F. Kurose, K. W. Ross. Computer Networking A Top-Down Approach Featuring the internet. ISBN 0-201-47711-4 Addison Wesley Longman Inc.

[16] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He. RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks, In IEEE RTAS 2002, September 2002.

[17] C. E. Perkins and E. M. Royer. Ad-hoc On Demand Distance Vector Routing. In WMCSA'99, February 1999.

[18] C. E. Perkins and P. Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for Mobile Computers, in SIGCOMM Symposium on Communications Architectures and Protocols, pp. 212-225, September, 1994.

[19] J. A. Stankovic, T. He, T. F. Abdelzaher, M. Marley, G. Tao, S. Son, and C. Lu. Feedback Control Scheduling in Distributed Systems, IEEE RTSS, December 2001.

[20] I. Stojmenovic and X. Lin. GEDIR: Loop-Free Location Based Routing in Wireless Networks, IASTED Int. Conf. on Parallel and Distributed Computing and Systems, November 3-6, 1999.

[21] A. Woo and D. Culler. A Transmission Control Scheme for Media Access in Sensor Networks, In IEEE MobiCOM 2001, July 2001.

[22] X. Zeng, Rajive Bagrodia, and Mario Gerla. GloMoSim: a Library for Parallel Simulation of Large-scale Wireless Networks, In PADS '98.

Dr. Tian He received the Ph.D. degree from the University of Virginia, Charlottesville, in 2004 and the M.S. degree from the Chinese Academy of Sciences, Beijing, China, in 2000, and the B.S. degree from the Nanjing University of Science & Technologyi, Nanjing, China in 1996. He is author and co-author of 20 papers at international conference and journals. His research interests includes wireless sensor network, real-time embedded system and distributed systems. He is a member of ACM and IEEE.

Professor John A. Stankovic is the BP America Professor in the Computer Science Department at the University of Virginia. He recently served as Chair of the department, completing two terms. He is a Fellow of both the IEEE and the ACM. He also won the IEEE Real-Time Systems Technical Committee's Award for Outstanding Technical Contributions and Leadership. Professor Stankovic also serves on the Board of Directors of the Computer Research Association. Before joining the University of Virginia, Professor Stankovic taught at the University of Massachusetts where he won an outstanding scholar award. He has also held visiting positions in the Computer Science Department at Carnegie-Mellon University, at INRIA in France, and Scuola Superiore S. Anna in Pisa, Italy. He was the Editor-in-Chief for the IEEE Transactions on Distributed and Parallel Systems and is a co-editor-in-chief for the Real-Time Systems Journal. His research interests are in distributed computing, real-time systems, operating systems, and wireless sensor networks. Prof. Stankovic received his PhD from Brown University.

Professor Chenyang Lu (Member, IEEE) received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 1995, the M.S. degree from the Chinese Academy of Sciences, Beijing, China, in 1997, and the Ph.D. degree from the University of Virginia, Charlottesville, in 2001, all in computer science. He is an Assistant Professor in the Department of Computer Science and Engineering at Washington University in St. Louis. He is author and co-author of more than 35 refereed publications. His current research interests include distributed real-time embedded systems and middleware, wireless sensor networks, and adaptive QoS control. He received an NSF CAREER award in 2005.

Professor Tarek Abdelzaher received his B.Sc. and M.Sc. degrees in Electrical and Computer Engineering from Ain Shams University, Cairo, Egypt, in 1990 and 1994 respectively. He received his Ph.D. from the University of Michigan in 1999. Since 1999 he has been an Assistant Professor at the University of Virginia where he founded the Quality of Service (QoS) Laboratory. He has authored/coauthored more than 60 refereed publications. He was Guest Editor for the Journal of Computer Communications and the Journal of Real-Time Systems, and is co-editor of IEEE Distributed Systems Online. He served on numerous Program Committees, and held several steering positions including Program Chair of RTAS 2004. He is a patent holder on Adaptive Web Systems, an IEEE member, and an NSF Career award recipient. Tarek's research interests include QoS provisioning, real-time computing, operating systems, computer networking, and sensor networks.

uCast: Unified Connectionless Multicast for Energy Efficient Content Distribution in Sensor Networks

Qing Cao, Tian He, *Member, IEEE,* and Tarek Abdelzaher, *Member, IEEE*

Abstract **In this paper, we present uCast, a novel multicast protocol for energy ef cient content distribution in sensor networks. We design uCast to support a large number of multicast sessions, especially when the number of destinations in a session is small. In uCast, we do not keep any state information relevant to ongoing multicast deliveries at intermediate nodes. Rather, we directly encode the multicast information in the packet headers, and parse these headers at intermediate nodes using the scoreboard algorithm as proposed in this paper. We demonstrate that 1) uCast is powerful enough to support multiple addressing and unicast routing schemes and 2) uCast is robust, ef cient and scalable in the face of changes in network topology, such as those introduced by energy conservation protocols. We systematically evaluate the performance of uCast through simulations, compare it with other state-of-the-art protocols, and collect preliminary data from a running system based on the Berkeley motes platform.**

Index Terms **Sensor networks, multicast, content distribution.**

I. Introduction

Recent research work has observed the unique challenges of sensor networks from a variety of aspects. One challenge is that sensor nodes are severely limited by their computation, storage and communication power. In this paper, we study the implications of such limitations on multicast protocols. In particular, we address the problem of designing protocols to support a large number of small group multicasts, where the number of destinations in a single session is limited. This problem is unique, as we will show later, in that no previous protocol can be easily adapted to solve it.

There are many applications of small group multicast in sensor networks. For example, in a typical directory service, such as the protocol described in [16], each node periodically updates a small set of other nodes (named *directory servers* in [16]) with its location. Therefore, one node needs to multicast information to several destination nodes, which form a small group. Furthermore, when multiple nodes use the directory service, they will generate many small group multicast sessions. Another common example involves data-centric storage (DCS) [21], [24]. One key component of some DCS protocols is the use of Geographic Hash Tables (GHT). GHT hashes keys, usually the names of data or events, into geographic coordinates. It then stores the values at the node that is geographically nearest to the hash value of the key. Usually, the data storage protocol suggests that the key-value pairs should be stored at multiple locations for robustness. Therefore, such protocol naturally requires a small group multicast session for each storage process, and many multicast sessions for storing a large amount of data.

Qing Cao and Tarek Abdelzaher are with the Department of Computer Science, University of Illinois at Urbana Champaign, 201 North Goodwin Avenue, Urbana, IL 61801. Email: {qcao2,zaher}@cs.uiuc.edu.

Tian He is with the Department of Computer Science and Engineering, University of Minnesota, 200 Union Street SE, Minneapolis, MN 55455. Email: tianhe@cs.umn.edu.

However, providing small group multicast service in sensor networks is complicated by several unique challenges. One challenge is that sensor nodes are extremely resource constrained. Consequently, sensor networks generally employ certain energy conservation protocols, which usually allow individual nodes to switch between sleep and wake states. Therefore, the topology of sensor networks will change, which poses unique requirements on the multicast service. Consequently, any protocol that relies on certain multicast structures to keep state information, such as multicast trees, must adapt these structures to topology changes. However, so far, no previous multicast protocol has addressed this issue properly.

Second, there are many unicast routing protocols for sensor networks. In fact, there is no consensus on which one is the best, and the choice unicast protocol usually depends on the particular application. For example, when geographical location information is readily available for each node, several well-known routing protocols that take advantage of geographical information are probably the best [4], [12], [14]. On the other hand, when there is no geographical information available, protocols based on certain topology encodings are preferred [5], [20], [18]. Because of the wide range of unicast choices, it is undesirable to design multicast protocols that make assumptions regarding any particular unicast, since that will limit the applicability of the multicast service. On the other hand, it is also not smart for the multicast to provide routing service from scratch, since doing so will lead to considerable functional overlapping with the unicast protocol. No protocol so far has addressed both problems simultaneously, in fact, most protocols provide their services totally independent of available unicast protocols, or provide unicast as a special sub-service.

To address these two challenges, we present a multicast protocol that is both powerful (supporting multiple unicast routing protocols by using a *unified* interface) and robust (tolerant of topological changes by providing a *connectionless* service). We call this protocol **unified connectionless multicast**, or uCast. To the best of our knowledge, uCast is the first protocol specifically designed to support a large number of small group multicast sessions in sensor networks. We now give a more detailed explanation on the two features of uCast.

The first feature is that uCast can support multiple unicast routing protocols through a unified interface. In this sense, uCast is a modular extension to the underlying unicast layer. In fact, it can extend any unicast routing protocol as long as this unicast can export a common comparison interface, which allows a comparison operation between two nodes for the same destination node to determine which one is better, using some notion of *cost*. We implemented uCast on top of three unicast routing protocols with different addressing schemes to prove our point.

The second feature is that uCast is tolerant of topology changes caused by energy saving protocols. To achieve this, uCast does not keep any multicast-specific state at intermediate nodes. Instead, uCast dynamically decides the multicast delivery path at each intermediate node, based only on local topology information and the comparison interface as discussed earlier. Since local information is much easier to reconstruct on topological changes compared to a superimposed global multicast overlay, uCast is much more adaptable for unpredictable changes in network connectivity than previous multicast protocols.

The rest of the paper is organized as follows. We outline related work in Section II. We discuss the details of uCast in Section III. In Sections IV and V, we report simulations based and real platform based results. At last, we provide further discussions and conclusions in Section VI.

II. Related Work

A. Multicast

Multicast is a classical topic in networking. Interestingly, we find only a few multicast protocols designed for sensor networks. Some may assume that work in ad-hoc networks or even the Internet can be easily customized for sensor networks. This is not the case because of the unique challenges of sensor networks. In the following, we survey the work on multicast from three aspects, namely, multicast work for sensor networks, ad-hoc networks and the Internet.

There has been some work on multicast for sensor networks for various purposes. One category is called geocast, which considers the scenario that multicast destinations are located within a bounded geographical area. Representative work includes [13], [17]. Another multicast category is called spatiotemporal multicast, or mobicast [10]. Mobicast features a moving zone of multicast destinations. The goal is to deliver packets just in time to this zone for tracking purposes. Another category [1] studies multicast in data caching and placement. It focuses on using multicast trees for asynchronously updated data deliveries. Yet another is called TTDD [30], which is optimized for mobile sinks. It uses a grid structure, coupled with localized flooding to track mobile sinks. All these protocols do not consider the effect of topology changes introduced by energy conservation protocols, nor are they designed to handle the small group multicast scenarios. Further, none of these protocols takes into account the compatibility issue with unicast protocols. Therefore, they are usually implemented *independently* with the unicast routing protocols that are already available, or provide unicast as a special case [13], [17]. Both approaches are likely to introduce functionality redundancy, if there is already a unicast layer implemented. Since the memory size of current sensor network nodes is extremely limited ($4K$ bytes on Mica2/MicaZ), it is not smart to tolerate any functional redundancy between different routing services. From this point, these previous multicast protocols are significantly different from uCast, especially on the compatibility with multiple unicast routing protocols. Therefore, we do not compare uCast with them in this paper.

There are many multicast protocols for ad-hoc networks. Representative protocols include multicast-tree based (Multicast AODV [22]), mesh based (CAMP [7]), or group based (ODMRP [15]). However, these protocols can not be easily applied to sensor networks because they all rely on preestablished overlays. These overlays are associated with considerable signaling costs, therefore, they are usually too expensive to reconstruct in the face of frequent topology changes, such as those introduced by energy conservation protocols. Further, since they are usually designed for mobile nodes that are much more powerful than sensor nodes, such as laptops, they are usually too heavy-weighted to implement in sensor networks.

At last, we also find many multicast protocols for IP networks in the literature. Representative protocols include IGMP [6], Xcast [3], [2], [25] and DVMRP [23]. Among these protocols, Xcast [3], [2], [25] is the most relevant to our work in that similar to ours, it encodes the destination list into the packet headers. However, our work is considerably different from Xcast in two aspects. First, Xcast relies on the routing tables at intermediate routers to decide the packet flow. However, we do not assume any particular routing structure, such as the routing table. Second, Xcast can only work with a single unicast routing protocol. Therefore, if the underlying routing protocol modifies the structure of the routing table, Xcast has to be modified as well. Following this principle, it is impractical to build a single multicast layer for wireless sensor networks using Xcast. We overcome this problem by designing uCast on top of the common comparison interface exported by the underlying unicast layer. This design choice essentially decouples uCast from any underlying unicast routing details and leads to a generalized and flexible service that is significantly different from Xcast.

Based on this survey, we consider uCast as a necessary complement for previous protocols. Primarily, our work is targeted at the small group multicast scenarios. Conceptually, the application domain of uCast is shown in Figure 1[1].

As shown in Figure 1, as the number of members for a particular multicast session increases, or the traffic per session increases, the average cost per member decreases for connection based protocols, because the signaling cost becomes less significant. This implies connection based protocols are more suitable for long-term large scale multicast. However, when the number of members is small and the traffic is low, the corresponding application domain can be characterized by spontaneous, short-term content delivery requests within small groups. The expected cost per member will increase due to the signaling cost for the connection based multicast service.

[1]The curve shown is only conceptual and helps the understanding of the application domain of uCast. It does not reflect any quantitative results.

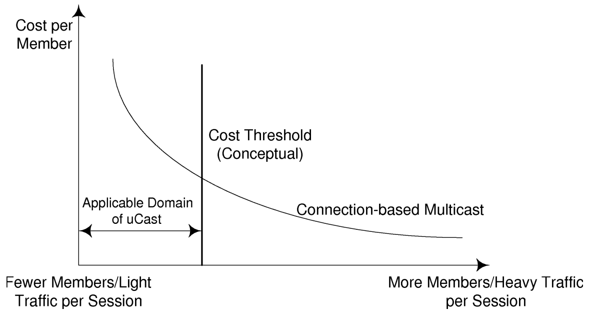

Cost per Member
Cost Threshold (Conceptual)
Applicable Domain of uCast
Connection-based Multicast
Fewer Members/Light Traffic per Session
More Members/Heavy Traffic per Session

Fig. 1. Application Domain of uCast

When the cost per member is higher than a certain threshold, uCast becomes more efficient and preferable.

B. Unicast

We briefly survey different unicast protocols in this section, and explain how uCast extends each of them. We discuss unicast protocols according to their *addressing schemes*, i.e., models that individual nodes are addressed in the routing structure. We classify current addressing schemes in sensor networks into three broad domains: identifier based, geographical location based and network encoding based.

The identifier based addressing is inherited from general ad hoc networks. In this type, nodes are addressed by their identifiers. Representative routing protocols of this type include those designed for ad-hoc networks, such as DSR [11] and AODV [22], where only node identifiers are used to find routes. One commonality of this type is that since identifiers essentially provide no information regarding topology, all protocols in this type require a flooding stage to find routes. Another commonality is that nodes need to keep routing tables, and the next hops in data deliveries are usually decided by look-up operations.

Geographical location based addressing is another scheme primarily used for sensor networks. In this scheme, individual nodes are assumed to be location aware either through GPS or some localization algorithm [8]. Geographical locations can be directly used for routing purposes, since they approximate the relative topology of sensor network nodes. Representative protocols include GFG [4]/GPSR [12], GEDIR [26], LAR [14], etc. These protocols no longer need flooding to find routes. Usually only local neighborhood information is needed to make routing decisions.

Several recently proposed protocols fall into the last category: network encoding based addressing. The key idea is to encode nodes using topology based identifiers. Such identifiers can be directly used for routing, thereby avoiding the expensive flooding process. Several network encoding schemes have been proposed, such as virtual location based geographical routing [20], relative logical coordinate based routing (LCR) [5] and graph embedding based routing (GEM) [18]. Since protocols based on network encodings do not require physical location information, they are good complements for the first two types of protocols.

```
uCast Scoreboard Algorithm

Comment: We assume the compare interface is
Compare(Node1,Node2,Destination). Each neighbor node has
a state of being selected or unselected.

Inputs: Destination node set DS, neighbor node set NS, current node S.
Outputs: The selected neighbor set SN. For each node in SN,
the algorithm outputs a subset of DS, called SD, that forms its task.

Main Algorithm:

0 Initialization of data structures.
1 For each neighbor node in NS, set it to be unselected.

Comment: First consider three special cases in steps 2,3 and 4.
2 For each node in NS, if it is in DS, set it to be selected. Remove this
  node from DS and insert it into Covered Set (CS) (destination nodes
  in CS are assumed to be covered by a certain neighbor node).
3 For each node in DS, if there is only one neighbor in NS that is closer
  than S according to the Compare Interface, remove it from DS and
  insert it into CS. Set the status of the corresponding neighbor in NS to
  be selected.
4 For each node in DS, if there is no neighbor in NS that is closer to it
  than S using the Compare Interface, insert it into LocalMaximum
  Set (LS). Remove it from DS.

5 For each selected neighbor, find all destinations for which it is closer
  compared to S. Insert these destinations into CS remove from DS.

While (DS is not empty)
{
    6 For each node in DS, find all nodes in NS that are unselected.
      Set each node with a score of 0. Assign one node one more
      score if this node is closer than S to a particular destination in
      DS based on the Compare Interface.
    7 Select the highest score among unselected nodes in NS. In case
      of tie, randomly select one or break the tie using node IDs.
      Suppose the node selected is K and set it to be selected.
    8 Among nodes in DS, find those nodes for which K is closer than
      the current node S and insert them into CS. Remove them from
      DS.

Comment: DS is empty after the loop.
Comment: Following is the optimization stage.
9  Insert all selected nodes in NS into set SN.
10 For each destination in Covered Set, choose among nodes in SN the
   best node (snode) using the Compare Interface. Add this
   destination to the corresponding SD for snode.
11 Remove those nodes in SN with an empty SD. For remaining nodes,
   form individual delivery tasks based on its SD.
12 If LS is not empty, switch to the underlying unicast protocol and
   use the corresponding local maximum handling approach to deliver
   packets to destinations in LS.
```

Fig. 2. The Scoreboard Algorithm

III. Unified Connectionless Multicast (uCast)

We first present our assumptions. As discussed earlier, we design uCast to support multiple unicast routing protocols. Therefore, we only make minimal assumptions regarding the unicast routing layer. Specifically, we do not assume any distance information, or particular configuration of the routing table. On the other hand, in order to avoid functional overlapping with the unicast layer, we introduce an interface that we expect the underlying unicast to *export*, that is, the unicast layer implements this interface and provides it to the multicast layer. We demonstrate that this is feasible in practice, and only requires minimal or no changes to the existing unicast routing protocols. As examples, we implement such an interface for three different unicast routing protocols in Section V.

The interface is defined as a pairwise comparison in the following manner:

```
Function: Compare(NODE N1,NODE N2,NODE DESC)
Return Type: NODE (N1 or N2)
```

In this interface, N1 and N2 are candidate nodes that lead to node DESC. The interface compares these two nodes, and returns the better candidate. In the following, we say the returned node is “closer” to the destination. Of course, “closer” is used only metaphorically. We do not make any assumptions on the way the comparison interface is implemented in the unicast layer.

A. uCast Design

We now describe the design of uCast. The core of the design is the *scoreboard algorithm*, which is executed at each intermediate node along the content delivery path. Once one node receives a multicast packet, it parses the destination list contained in the packet header. If this node is included in this list, it applies the scoreboard algorithm. The algorithm takes the list of destinations and all the neighbors of the current node as input, and outputs the multicast task allocation, e.g., to send the packet to destination S, the packet should be forwarded to node T, etc. Using the output, the current node generates one or more packets as required, and forwards these packets to the next-hop neighbors. This process continues until all destinations receive the multicast packet.

So how does the algorithm work internally? The detailed pseudocode of the scoreboard algorithm is shown in Figure 2. As the first step, the algorithm considers the destinations one by one. For each destination, it applies the comparison interface to determine which neighbors are “closer” than the current node. Those neighbors are said to *cover* this destination and get one score point. When all destinations are considered, the neighbor node that has the highest score is chosen as a forwarding candidate. When multiple neighbor nodes share the same score, the algorithm breaks the tie either by randomly choosing one node or by using node ID. Next, the algorithm records and removes the neighbor with the highest score, as well as those destinations that have been covered by this node, from the next round of comparisons. This comparison-select-removal process continues until all destinations are covered. To this point, the algorithm has finished the preliminary neighbor sifting and ends up with a set of candidate neighbors, called the *forwarding candidate set*.

Next, the algorithm begins to further optimize the candidate set. For each destination, it compares every pair of nodes in the forwarding candidate set to determine the closest node. This node is assigned the corresponding destination node. Note that this node may not be the one that initially *covered* the destination. After this step, some nodes in the forwarding set may not be assigned any destination. Therefore, they are removed from the forwarding candidate set. The remaining nodes form the optimized candidate set, and each node in this set gets a list of assigned destinations. The resulting set and the destination assignment constitute the final output of the algorithm.

B. Design Discussion

We now discuss several tradeoffs in the algorithm. First, we discuss the performance implications of the scoreboard algorithm. Since it is greedy by nature, it remains unclear how *optimal* it is. We provide an analysis on this topic. Second, uCast uses packet headers to enumerate destinations. Therefore, there is a limit on the maximal number of destinations that one packet can address. We describe several possible solutions to this problem and discuss their effects on our protocol.

1) Analysis on the Greedy Neighbor Selection: In this section, we show by simulations that our scoreboard algorithm is very efficient at minimizing the number of branches in the multicast tree, hence reducing its cost. Recall that we always select the node with the highest score in the neighbor table until all destinations are covered. We now show that this approach is approximately as good as finding a minimal cover of destinations at each hop. Since choosing the minimal cover is the well-known set cover (SC) problem which is NP-Complete, solutions to it are not scalable with the neighbor table size. General greedy selection approaches for SC problems guarantee an approximation ratio of $1 + ln(maximalsubsetsize)$ [19] (here, approximation ratio refers to the ratio between the size of the subset selected by the greedy algorithm to the size of the subset selected by the locally optimal minimal cover algorithm). In practice, we show that scoreboard algorithm is much closer to the optimal case than guaranteed by the general approximation bound.

Note that, however, although we use the minimal cover technique as the comparison baseline, this technique is not globally optimal. In fact, finding the global optimal tree is another NP-complete problem, the Steiner tree generation in graph theory. Because of the large number of nodes, the globally optimal tree structure can not be generated in a reasonable period of time. There are, of course, various heuristic techniques to construct approximate Steiner trees, however, constructing these trees is not practical in real implementations either because this process requires global topology information. In real sensor networks, each node only has local topology information. Therefore, we compare our scoreboard algorithm with the minimal set cover algorithm because both of them only require local topology information.

In simulations, we deploy nodes with a communication range of $50m$ in a region of $500m \times 500m$. We place the source node at $(250, 250)$ and multicast packets to six nodes located at the boundary of the region within a maximum angle of sixty degrees. The packets need to be relayed at least six hops, thereby ensuring that different neighbor selection approaches will have an effect. The density of the network increases from 18 to 26 nodes per communication range. We deliberately choose a relatively high density so that the size of neighbor tables is considerably large, thereby emphasizing the effects of different neighbor selection strategies. Each scenario is tested 100 rounds. We ensure that exactly the same topology is replayed for the minimal cover selection and the greedy selection (the scoreboard algorithm), respectively. The results are shown in Figure 3.

In this figure, we use the average number of packets sent in one round, plotted on the Y axis, to compare the performances of different neighbor selection strategies. We observe that the difference between choosing the minimal cover and the scoreboard can be neglected. In fact, since the minimal cover neighbor selection at each hop does not necessarily lead to a better global tree structure than the scoreboard algorithm, we find the scoreboard algorithm sometimes performs even better. Therefore, we conclude through these simulations that the scoreboard algorithm is at least as good as the minimal cover neighbor selection strategy.

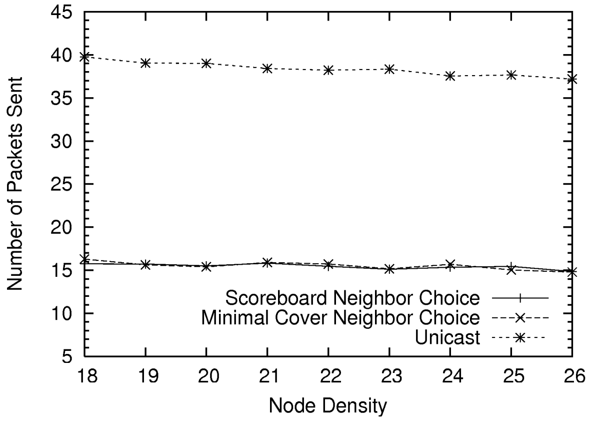


Fig. 3. Algorithm Optimality Analysis

A further implication of this neighbor selection strategy is that when the destinations are clustered, usually only one node or two will be selected as the next hop, since they are expected to get the most scores. Therefore, uCast has the tendency to minimize the branches, thereby controlling the potential congestions between nearby data streams. Further, since uCast does not incur state reconstruction overhead when the topology changes, it further decreases congestions between control and data packets. Above all, although we generally consider congestion control as orthogonal to the purpose of uCast in this paper, the design of uCast has a natural advantage in the aspect of congestion control.

2) Discussion on the effects of Destination Encoding: In our design, we encoded all destinations into packet headers. This design choice poses a limit on the maximal number of destinations a single packet can address. In this section, we discuss three possible tradeoffs to mitigate the scalability problem introduced by this design choice.

First, as the radio on sensor nodes becomes more powerful (for example, from CC1000 on Mica nodes to CC2420 on MicaZ nodes), it is likely that nodes will send longer packets in the next-generation sensor networks. Further, new sensors such as video cameras naturally require long packets to transmit images. In such cases, it will not be a problem to encode all destinations into the packet headers. Second, instead of enumerating all destinations, we can compress the destination list before storing them. By nature, this approach is to exchange computation time for storage space. In case that the space limit is severe, the designer may switch to this approach for a better performance. At last, nodes can employ in-network aggregation techniques to further reduce the effects of destination enumeration. More specifically, after one node sends out a packet containing all destinations, it can then follow up with a train of pure data packets which do not contain any destination information. To achieve this, certain synchronization and retransmission mechanisms may be employed to guarantee correctness. This train of packets that share the same destination list can be viewed as a single large packet at the receiver side.

Above all, we believe the designer can use various tradeoffs to alleviate the header problem. Further, as emphasized earlier, uCast is designed primarily for small group multicast. Under such scenarios, we expect the number of destinations should be considerably small, therefore, encoding all destinations into packet headers will not be a problem.

IV. Performance Evaluation of uCast

We now present the performance evaluation of uCast. We are primarily interested in three aspects: energy efficiency, its interaction with energy conservation protocols, and its integration with different unicast routing protocols. We observe that the performance of uCast is considerably affected by the positions of the destination nodes, that is, how clustered the destinations are and how far away they are from the source node will significantly affect the multicast performance. Therefore, we first present a parameterized destination placement model to control the above attributes. We then evaluate the performance of uCast using this model.

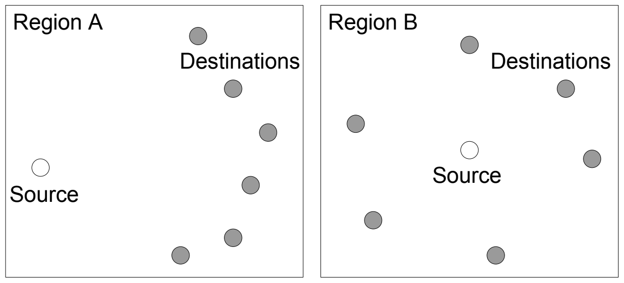


Fig. 4. Different Deployment Impact

To demonstrate the performance superiority of uCast, we compare it with connection-based protocols. The baselines include Shortest Path Tree (SPT), Greedy Incremental Tree (GIT), and plain unicast. In SPT, we assume that the source node sends packets along the shortest paths to all destinations and aggregates common paths to form a tree structure. We select SPT because it is the backbone tree structure used in several representative connection based multicast protocols [22]. GIT is another selected baseline. The construction process of GIT is centralized and requires full knowledge of the topology. It proceeds as follows. First, we connect the source node with the nearest destination via a shortest path. This path forms a partially completed tree structure.

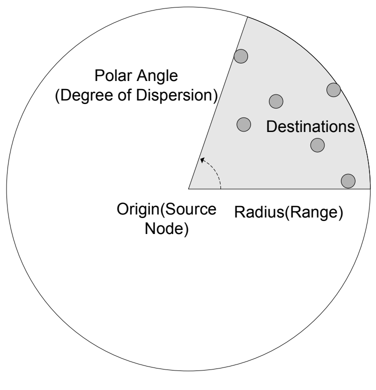


Fig. 5. Destination Placement Model

Then, we find the nearest destination node to the existing tree, and connect this node to the closest node in the structure. We iteratively find the next nearest node in the remaining destinations and connect it, until all destinations are connected. Clearly, each step requires global topology information, and the construction process is quite computationally intensive. Therefore, GIT is not applicable for sensor networks. However, previous literature has pointed out that a GIT tree is usually very compact, implying that if we deliver packets along such a tree, we may distribute data in fewer hops compared to other tree structures. Therefore, we use the GIT structure as a best-case baseline for comparison.

A. The Destination Placement Model

We describe the destination placement model in this section. We first give an intuitive explanation on why this model is important for the performance evaluation of uCast. Consider the two scenarios in Figure 4. Intuitively, using multicast in region A saves more energy than in region B compared with using unicast, because the destinations are more clustered in region A. Our model is designed to characterize such differences. It presents four parameters of destination placement that have effects on the performance of multicast. These parameters model a minimal pie-shaped region that contains the destinations and the source, as shown in Figure 5. The parameters are the angle of dispersion (AOD); the radius, which corresponds to the farthest distance one destination can be positioned from the source node; the density, i.e., the number of nodes within a communication range; and the number of destination nodes. We note that if we set AOD as 2π and the range large enough, our model defaults to a random placement model. In the following simulations, once the polar angle is set, the distances of nodes from the source conform to a uniform distribution.

Unless otherwise stated, the default parameters are as follows. The communication range is $50m$, the area $500m \times 500m$, the density 20 nodes per communication range, AOD 90 degrees, the number of destinations 10, and the radius of pie shape area $250m$. A total of 636 nodes are deployed by default. The data rate is 6 packets per minute, except Section IV-C, where multiple data rates are tested. In Section IV-B, we simulated for 100 packets (about 16 minutes). In Section IV-C, we simulated for 120 minutes. We selected different time lengths because the evaluation purposes are different. We assume that each node has the same transmission power level. The simulations are done in the Glomosim [28] environment.

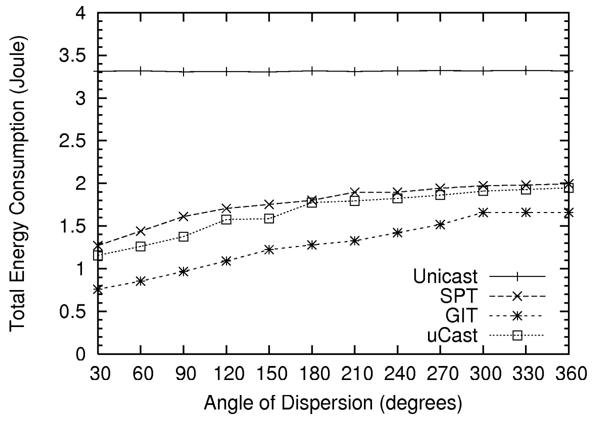


Fig. 6. Impact of AOD on Energy Consumption

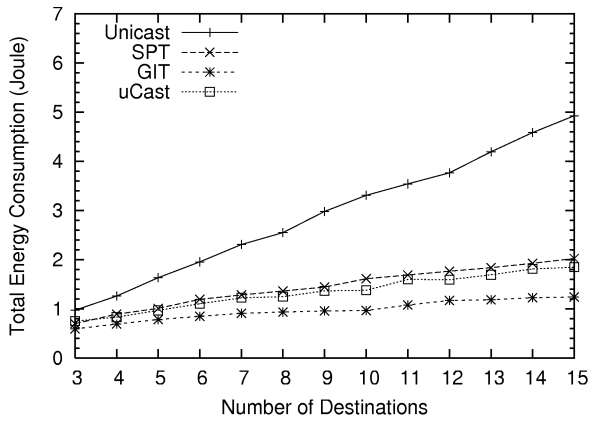


Fig. 7. Impact of Number of Destinations on Energy Consumption

B. Energy Efficiency

In this section, we compare the energy efficiency aspect of uCast with other multicast protocols. To accurately estimate energy consumptions, we use realistic parameters of MicaZ nodes (one of the most advanced sensor network nodes

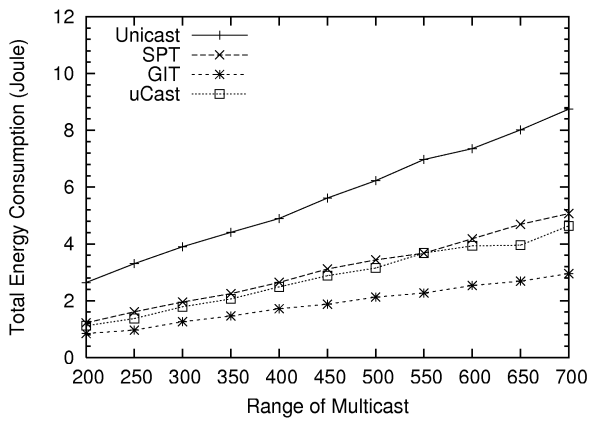


Fig. 8. Impact of Range on Energy Consumption

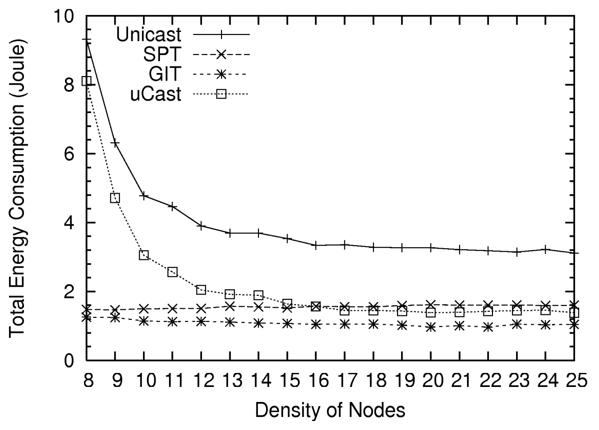


Fig. 9. Impact of Density on Energy Consumption

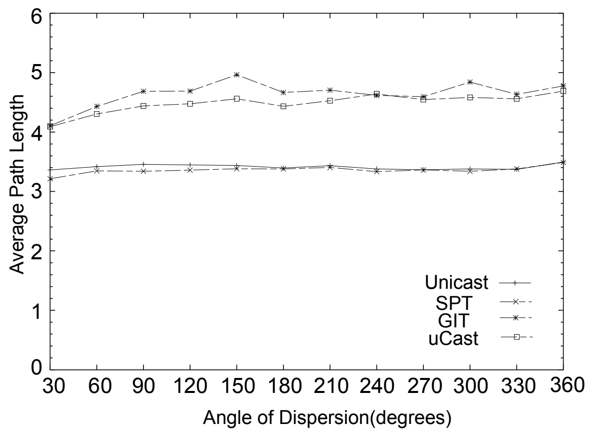


Fig. 10. Impact of AOD on Path Length

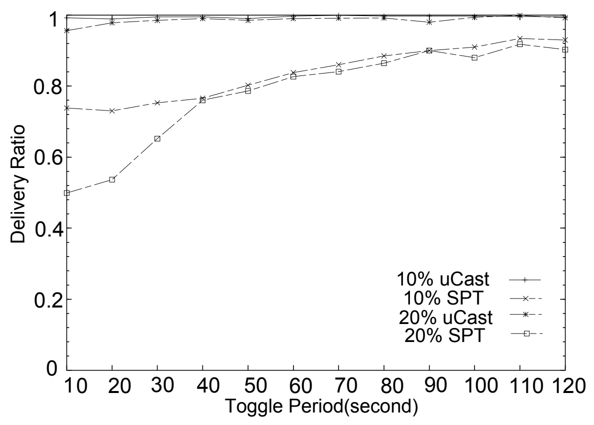


Fig. 12. Impact of Toggle Period on Delivery Ratio

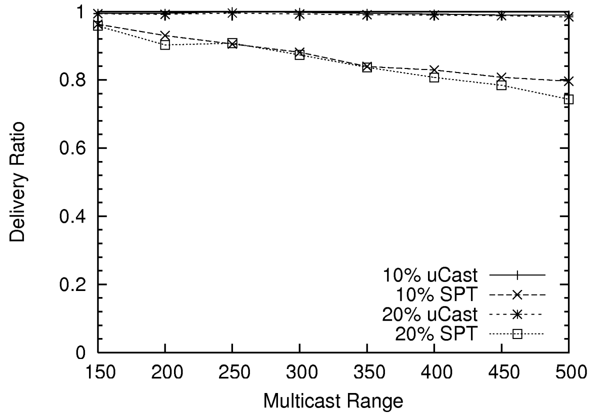


Fig. 13. Impact of Scale on Delivery Ratio

currently available) in energy consumption simulations. More specifically, energy consumption comes from both sending and receiving packets. According to the data sheet of the CC2420 radio on MicaZ [27], sending and receiving have current levels of $17.4mA$ and $18.8mA$, respectively. The voltage supply is assumed to be $3V$, and the data rate is $250kbps$. Packets are assumed to have a payload of 20 bytes, and each destination requires 4 bytes in the header. We do not consider the signaling cost of connection based protocols, since the impact of this cost depends on how the specific protocol is implemented and how frequently the topology changes. The key metric we use is the total energy consumption, in joules, for sending 100 packets to all destinations from the source.

We begin with static network topology. Observe that this is not the scenario uCast is optimized for. The main advantage of uCast lies in its robustness to topological dynamics. Hence, our objective of using static topology is to show that we do not

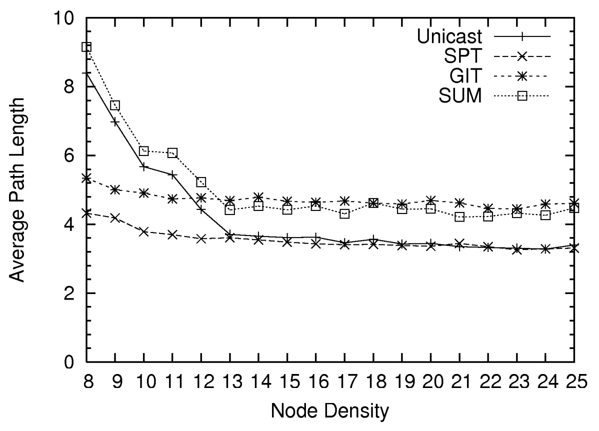


Fig. 11. Impact of Density on Path Length

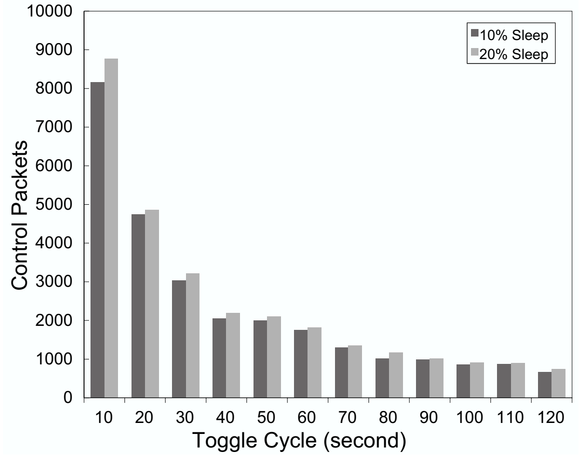


Fig. 14. Control Packets of SPT multicast with range of 250m

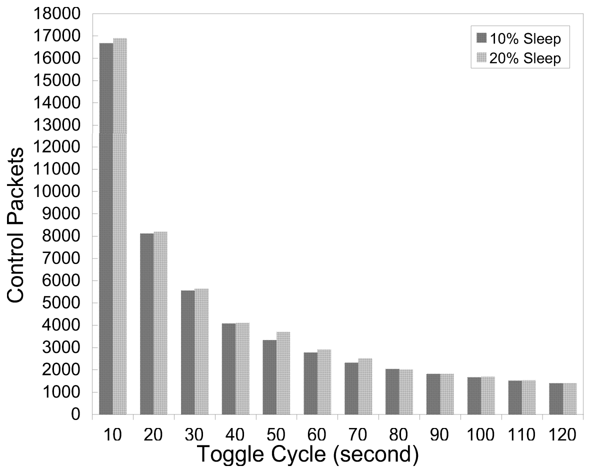


Fig. 15. Control Packets of SPT multicast with range of 500m

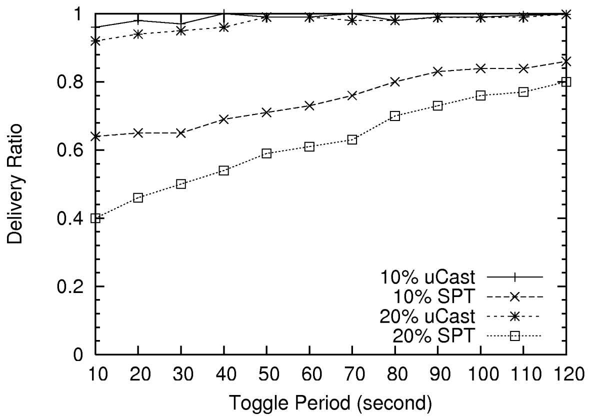


Fig. 16. Impact of Toggle Period on Delivery Ratio with a Higher Data Rate

degrade the performance by removing multicast state when the network is static. Later, we shall present the key advantages of uCast by considering topology changes.

In the following simulations, uCast is integrated with geographical forwarding, a commonly employed unicast protocol in sensor networks. The common comparison interface is implemented by simply returning the node that is geographically nearer to the destination. When a local minimum is reached, uCast leverages the GPSR [12] traversing technique to handle nodes in the *LocalMinimum* set. Since there are no state transitions in this experiment, no routing layer route repairs are needed.

Figures 6 through 9 show the impact of the four destination placement parameters on multicast performances. Based on these results, we have several observations. First, observe that uCast performs better than SPT in these figures, except Figure 9, where the traversing technique of GPSR significantly increases the path length. Also observe that as we expected, GIT performs better than uCast. We note that the prohibitive construction cost of GIT makes it unsuitable for sensor networks and hence it is not a contender in practice.

Figure 9 is especially interesting. In this case, both uCast and unicast increasingly turn to the GPSR traversing technique to deliver packets around voids, which degrades their performances. Considering that practical sensor networks are usually deployed with a sufficiently high density to ensure coverage, topology voids are not common. Furthermore, the designer may decide to incorporate adaptive features into multicast, where the applications have the option to switch from uCast to SPT when the density becomes extremely low. Therefore, we conclude that our stateless multicast generally does not incur a performance penalty compared to stateful approaches even when the network is static.

Figure 10 and Figure 11 show the comparison results of the average path length. Due to the effect of path aggregation, we observe that uCast and GIT deliver packets along longer routes compared with SPT and unicast. This is intuitive, since SPT and unicast typically find near-optimal paths. The increase in the path length means that uCast may have a slightly higher end-to-end delay. Since the main constraint in sensor networks is the limited energy supply, we believe that increasing path lengths to save total energy consumption is an acceptable compromise. An operator would welcome a slightly longer latency for each packet in exchange for a significantly extended network lifetime.

C. Impact of Topological Changes

In this section, we evaluate uCast in the presence of topological changes. Such changes are introduced by energy saving protocols that turn nodes into and out of sleep states. We expect that in this case, the advantages of uCast should dominate.

We use three parameters of energy conserving protocols to evaluate the multicast performance:

- Toggle Cycle: Toggle Cycle is the time interval between consecutive transitions into the sleep state by individual nodes. This parameter reflects the frequency at which the state information kept by intermediate nodes becomes invalid. As the frequency goes higher, the performance of state based multicast protocols should drop accordingly.
- Scale: Scale refers to the size of the multicast area. As the size scales up, the impact of topological changes becomes more significant and the reconstruction cost goes higher. As a result, we expect that the performance of state based multicast protocols will drop with a larger scale.
- Packet Delivery Rate: Another parameter that we change is the packet delivery rate. We use two such rates in our experiments, 6 packets per minute and 12 packets per minute, respectively. We do not choose higher rates because we observed a higher level of radio congestion. This congestion comes from, for example, neighboring nodes send out packets simultaneously in a multicast session. Since congestion leads to packet retransmissions and possible loss of packets when topology changes, this adds additional penalty to both state based protocols and stateless protocols. It is, in practice, hard to quantify this penalty and make sure that it will not skew the performance comparison results. To achieve accurate and unbiased comparisons between uCast and state based protocols such as SPT, we decide to remove the effect of congestion altogether, by deliberately choosing relatively low packet delivery rates in this experiment.

In the simulation setting, we place the source node at $(0, 0)$, and let it periodically deliver packets to ten destinations with an AOD of 90 degrees. The total simulated time is 120 minutes. Other settings are set as the default.

The energy conservation model we use is random sleep scheduling. For example, in Figure 12, 10% sleep scheduling with a 10 seconds toggle period means that one node sleeps for one second in every ten seconds. Each node has the same toggle period. We assume that there is no coordination between nodes, since this is the model that can be most easily implemented in sensor networks. It is also the foundation of a variety of other more complex sleep scheduling protocols [9], [29].

We compare uCast with SPT in this section. We don't include GIT because it is computed in a centralized manner and it has a prohibitively high computational cost in the presence of topological changes. Therefore, it is not a contender in practice.

Figure 12 and 13 show the performance evaluation results. These two experiments are carried out using a data rate of 6 packets per minute. The comparison results demonstrate the superiority of stateless multicast in the presence of node state transitions. More specifically, we have the following observations.

First, as the toggle periods become shorter, we observe that the delivery ratio for SPT multicast degrades considerably. For example, when nodes use a toggle cycle of 10 seconds and sleep 20% of the time, only around half of all packets successfully arrive at the destinations using the SPT tree for multicast. On the other hand, we observe that uCast achieves a delivery ratio of around 96%, enough for common multicast purposes. We attribute the superior performance of uCast to its statelessness.

Second, we observe from Figure 13 that connection based multicast protocols exhibit less scalability compared with uCast. This is quite intuitive in that as the multicast range scales up, it is more likely for one node on the tree to enter sleeping state for energy conservation purposes. Therefore, there is a higher probability for one packet delivery session to encounter a state loss.

One tentative solution to fix the state loss problem for state based protocols is to let the last node that has successfully received the packet to locally reconstruct the SPT, once it detects the next hop has entered sleeping mode. This approach will guarantee that the SPT achieves a 100% delivery ratio. However, this approach is quite expensive. We implemented this tentative patch for SPT and record how many control packets are sent out to reconstruct the SPT structure, and plot the results with two different multicast ranges, $250m$ and $500m$, respectively. The results are shown in Figure 14 and 15.

As shown in Figures 14 and 15, even with only 100 packets sent from the source, there are usually thousands of control packets required to locally rebuild the tree. The reason is that when a state transition occurs for a node that was initially in the SPT tree structure, it can no longer forward packets from its upstream nodes. Therefore, the upstream node must initiate a flooding process to try to locate the next downstream node available. In our simulation, we find that this upstream nodes usually needs to flood packets to two-hop neighbors, while in rare cases, three-hop neighbors are needed. Therefore, even if the tree structure is only partially broken, the flooding process generates a considerable amount of traffic. Of course, other modification possibilities also exist, such as enforcing that nodes should not go to sleep when they are in the multicast sessions. However, doing so requires non-trivial modifications to the existing energy saving protocols, and may significantly affect the overall system performance. Therefore, further exploration of these optimization possibilities is outside the scope of this paper. On the other hand, uCast has significantly less overhead, because it does not need any control packets to handle individual node state transitions. We acknowledge that uCast does have additional overhead in the form of destination lists in the packet headers. This overhead, however, is usually quite small when only a few destinations need to be enumerated. Furthermore, we shall consider this overhead in the experiments based on a realistic testbed in Section V, and show that this overhead does not have a significant effect on the efficiency of uCast.

Figure 16 studies the effect of the increased data rate, in which we change the data rate to 12 packets per minute. We can observe a slight decrease in the delivery ratio, compared with Figure 12. As expected, the advantages of uCast still dominate.

D. Integration between uCast and Unicast Protocols

Another goal of uCast is to integrate with different unicast protocols. We implemented uCast on top of three unicast protocols: geographical forwarding, logical coordinate based routing and graph embedding based routing. In each of them, we made no changes to the existing unicast protocols other than extending them to provide the common comparison interface. In this section, we first describe how we implemented the common comparison interface, followed by performance comparisons based on simulations.

For the geographical forwarding routing protocol, we implemented the comparison interface based on physical distance comparisons. More specifically, the interface simply returns the node that is nearer to the destination. The second routing protocol we extend is the logical coordinate based routing protocol(LCR) [5]. LCR uses hop counts to a few landmarks from each node as its logical coordinate vector. Based on these vectors, LCR also provides a definition of logical distances. In the comparison interface, we simply compare the logical distances from nodes $N1$ and $N2$ to node $DEST$, and the node with a smaller distance is returned by the interface.

The way we implemented the compare interface in Graph Embedding based Routing (GEM) [18] is a little more complex. In GEM, one node is chosen as the root. GEM then constructs a tree structure and assigns a (level,angle) combination to each node based on its topological position. The assigned combination forms a unique identifier for each node. GEM then delivers packets using this tree structure based on considerations of both the level and the angle of each node. Interestingly, GEM has no definition of distance, therefore, we used both the level and the angle information to implement the comparison interface. More specifically, when comparing two nodes, we followed the same procedure as the routing process in GEM: if one node is the parent or the offspring of the destination node in the tree structure, and if the other node is not, then the parent/offspring node is returned by the interface; if both nodes are parent/offspring nodes, then the node with a level nearer to the destination is returned; if both nodes are not parent/offspring nodes, then the node with a nearer angle range is returned. Theoretically this approach guarantees 100% delivery ratio if all nodes in the same level are perfectly aligned.

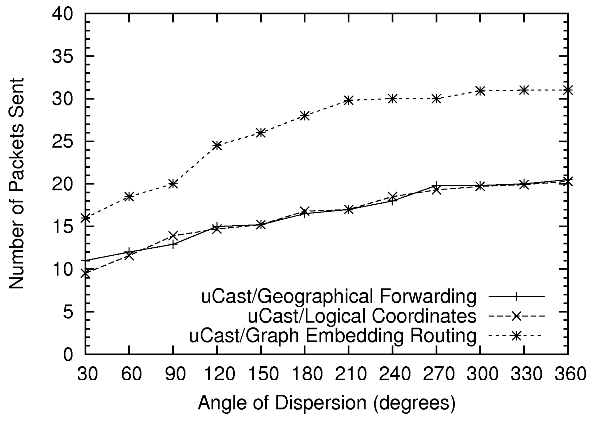


Fig. 17. Impact of Addressing Schemes on Traffic

Figure 17 shows the performance evaluation results of running uCast on the three aforementioned unicast protocols. We observe that both geographical forwarding based and logical coordinates based integrations appear quite similar in their performances. However, uCast based on GEM shows quite different performance characteristics. We attribute such differences to the more convoluted delivery paths in GEM, which increase path lengths considerably. Another way to explain the differences is that both logical coordinates and physical coordinates are based on Cartesian-like coordinate frameworks, which are considerably different from GEM, whose identifiers are more like polar coordinates.

We didn't implement uCast on identifer based unicast routing protocols like DSR. In such protocols, a look-up operation is used to return the next node on the path to the destinations. The implementation of the compare interface is therefore very straightforward: it simply returns the next hop node for a given destination from the look-up table, and this node will always percolate to the top and be chosen as the best candidate node.

V. Implementation on Sensor Platform

To investigate the applicability of the uCast protocol in a running system, we implemented the uCast protocol on the MICA2 platform. The code size is 992 bytes. As shown in Figure 18, we bridge the uCast protocol with the underlying unicast routing protocol (Geographic Forwarding protocol) through the uCast2uniCast interface (the NesC definition of this interface is shown in Figure 19). In this interface definition, the $compare()$ command is the mandatory part of

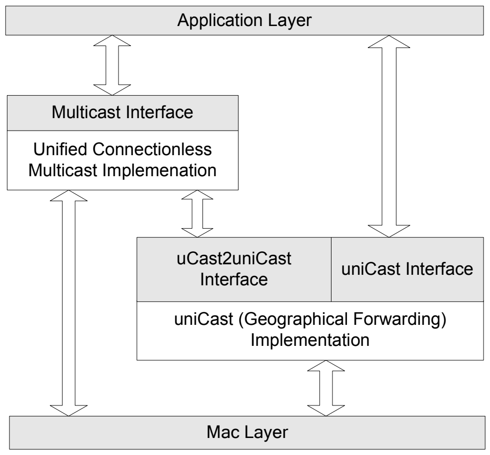

Application Layer
Multicast Interface
Unified Connectionless Multicast Implemenation
uCast2uniCast Interface
uniCast Interface
uniCast (Geographical Forwarding) Implementation
Mac Layer

Fig. 18. Implementation of uCast protocol

```
interface uCast2uniCast
{
   /* For comparing distances of two neighboring nodes */
  command result_t compare(uint16_t ID1, uint16_t ID2, uint16_t destination);
  /* Optional: get neighbor information */
  command result_t getNeighborTable(uint16_t *IDs, uint8_t *count, uint8_t  MAX_COUNT);
}
```

Fig. 19. uCast interface in NesC definition

the interface for node comparison. The $getNeighborTable()$ command is optionally provided for the purpose of neighbor table manipulations.

In the experiment, we used a testbed of 25 MICA2 motes (5 by 5). Figure 20 shows the experiment setting and data delivery traces. We tested three set of experiments, with 3 or 5 destination nodes selected in each set. We plot the multicast traces and unicast traces in this figure. All data are gathered from real tests. For multicast traces, we use *forking points* to represent the positions where data are sent to multiple receivers.

We now compare the energy consumption aspect between uCast and unicast. We use the following parameters: each node sends out packets at a current level of $21.5mA$; each packet maintains a current level of $7.4mA$ in receiving mood; the bandwidth is $19.2Kbps$; each node has a $3V$ voltage; each packet contains a 12 bytes payload. We then calculate energy

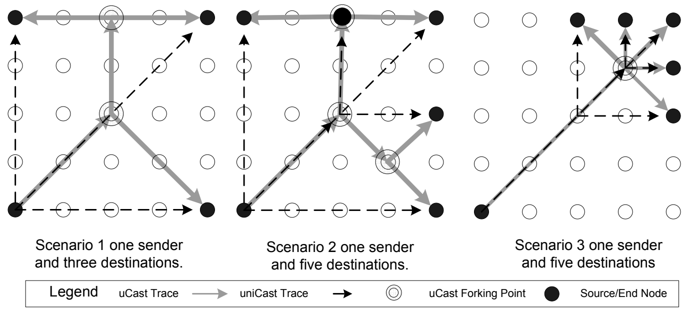
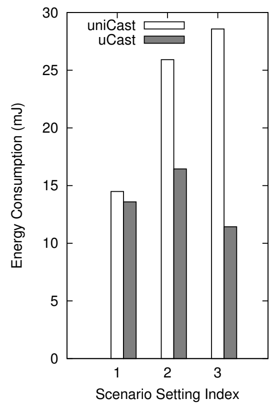

Scenario 1 one sender and three destinations.
Scenario 2 one sender and five destinations.
Scenario 3 one sender and five destinations
Legend
uCast Trace
uniCast Trace
uCast Forking Point
Source/End Node

Fig. 20. The prototype experiment traces



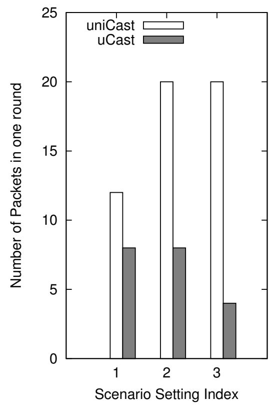


Fig. 21. The Performance Comparison

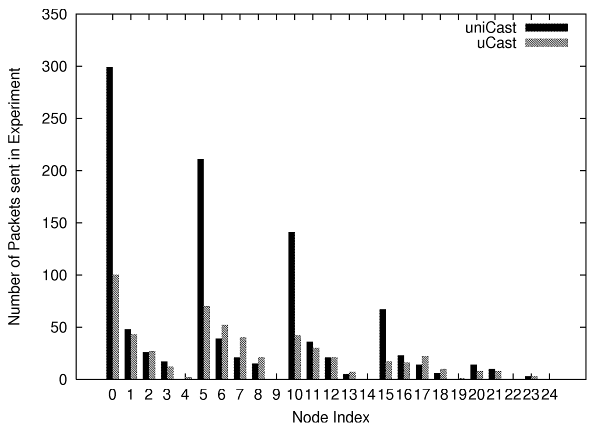


Fig. 22. Experimental comparison between uCast and uniCast

consumptions of different data distribution approaches using these parameters and plot the results in Figure 21. Observe that in these three settings, uCast significantly decreases energy consumption compared to unicast. Furthermore, if we compare Scenario 1 and Scenario 2, we notice that as the number of destinations increases, uCast saves more energy. The same observation also holds when the destinations become more clustered, as from Scenario 2 to Scenario 3. These observations are consistent with our analysis in Section IV.

At last, we study a more generalized scenario. In this experiment, the source node 0 delivers data packets periodically to three randomly selected destinations. We compare the total number of packet transmissions in uCast to unicast. Again, we use geographic forwarding as the basic unicast routing protocol. A total of 300 packets are delivered and the recorded data load for each node is plotted in Figure 22. We observe a considerably reduced data load for uCast, in fact, uCast reduces the total number of data transmissions by 45.7% compared to unicast. Therefore, we conclude that uCast exhibits a quite satisfactory energy efficiency for content delivery in realistic experiments.

VI. Conclusions

In this paper, we presented uCast, a unified connectionless multicast protocol for sensor networks. The design of uCast is motivated by the problem that state based protocols can not adapt efficiently to the network dynamics introduced by energy conservation protocols. We designed and implemented uCast on top of three different unicast routing protocols to show that it is generic. Several conclusions are drawn from our evaluation and comparisons. First, uCast is generally as efficient as connection based multicast protocols, even when the network is static. Second, the connectionless nature of uCast makes it more robust in the face of network dynamics. Finally, uCast can be easily implemented on different unicast routing protocols. The implementation of uCast on a real testbed also supports our conclusions.

References

[1] S. Bhattacharya, H. Kim, S. Prabh, and T. Abdelzaher. Energy-conserving data placement and asynchronous multicast in wireless sensor networks. In *ACM MobiSys*, 2003.

[2] R. Boivie, N. Feldman, Y. Imai, W. Livens, D. Ooms, O. Paridaens, and E. Muramoto. Internet draft draft-ooms-xcast-basic-spec-06.txt.

[3] Rick Boivie, Nancy Feldman, and Christopher Metz. On the wire - small group multicast: A new solution for multicasting on the internet. volume 4, pages 75–79, 2000.

[4] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *ACM/Baltzer Wireless Networks*, 2001.

[5] Q. Cao and T. Abdelzaher. A scalable logical coordinates framework for routing in wireless sensor networks. In *IEEE RTSS*, 2004.

[6] W. Fenner. Internet group management protocol, version 2. In *IETF RFC 2236*, November 1997.

[7] J. Garcia-Luna-Aceves and E. L. Madruga. The core-assisted mesh protocol. *IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks*, August 1998.

[8] T. He, C. D. Huang, B. Blum, J. A. Stankovic, and T. Abdelzaher. Range-free localization and its impact on large scale sensor networks. In *ACM MobiCom*, September 2003.

[9] T. He, S. Krishnamurthy, L. Luo, T. Yan, B. Krogh, L. Gu, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, and J. Hui. Vigilnet: An integrated sensor network system for energy-efficient surveillance. In *ACM Transaction on Sensor Networks*, 2006.

[10] Q. F. Huang, C. Y. Lu, and C. C. Roman. Spatio-temporal multicast in sensor networks. In *ACM Sensys*, November 2003.

[11] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing chapter 5, pages 195–206*, 1996.

[12] B. Karp and H. T. Kung. Greedy perimeter stateless routing for wireless networks. In *ACM MobiCom*, August 2000.

[13] Y. Ko and N. Vaidya. Geocasting in mobile ad hoc networks: Location-based multicast algorithms. In *IEEE WMCSA*, February 1999.

[14] Y. B. Ko and N. H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. In *MobiCom*, 1998.

[15] S. J. Lee, M. Gerla, and C. C. Chiang. On-demand multicast routing protocol. In *Proceedings of IEEE WCNC99*, September 1999.

[16] J. Y. Li, J. Jonnotti, D. D. Couto, D. R. Karger, and R. Morris. A scalable location service for geographic ad hoc routing. In *ACM Mobicom*, 2000.

[17] J. C. Navas and T. Imielinski. Geocast, geographic addressing and routing. In *MobiCom*, 1997.

[18] J. Newsome and D. Song. Gem: Graph embedding for routing and data-centric storage in sensor networks without geographic information. In *SenSys*, 2003.

[19] V. T. Paschos. A survey of approximately optimal solutions to some covering and packing problems. *ACM Computing Surveys*, June 1997.

[20] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *MobiCom*, 2003.

[21] S. Ratnasamy and et al. Data-centric storage in sensornets with ght, a geographic hash table. In *Mobile Networks and Applications (MONET), Special Issue on Wireless Sensor Networks*, 2003.

[22] E. Royer and C. E. Perkins. Multicast operation of ad-hoc, on-demand distance vector routing protocol. In *ACM/IEEE MobiCom*, 1999.

[23] S.Deering and D. Cheriton. Multicast routing in datagram internetworks and extended lans. In *ACM Transactions on Computer Systems*, 1990.

[24] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin. Data-centric storage in sensornets. In *ACM SIGCOMM HotNets*, 2002.

[25] M. Shin, K.S. Park Y.J. Kim, and S.H. Kim. Explicit multicast extension (xcast+) for efficient multicast packet delivery. In *ETRI Journal, vol.23, no.4, pp.202-204*, 2001.
[26] I. Stojmenovic and X. Lin. Gedir: Loop-free location based routing in wireless networks. In *International Conference on Parallel and Distributed Computing and Systems*, Nov 1999.
[27] the CC2420 datasheet from Chipcon Company. http://www.chipcon.com.
[28] the Glomosim Project. http://pcl.cs.ucla.edu/projects/glomosim/.
[29] T. Yan, T. He, and J. Stankovic. Differentiated surveillance for sensor networks. In *ACM Sensys*, 2003.
[30] F. Ye, H.Y. Luo, J. Cheng, S.W. Lu, and L.X. Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *ACM MOBICOM*, 2002.

Qing Cao is currently a Ph.D. student in Department of Computer Science at University of Illinois, Urbana-Champaign. He received the M.S. degree from the University of Virginia, Virginia, U.S. in 2004, and the B.S. degree from Fudan University, Shanghai, China in 2002, both in computer science. His research interests include wireless sensor networks, real-time embedded systems and distributed systems.

Tian He is currently an assistant professor in Department of Computer Science and Engineering at University of Minnesota-Twin City. He received the Ph.D. degree under Professor John A. Stankovic from the University of Virginia, Virginia in 2004, and the M.S. degree from the Institute of Computing Technology, Chinese Academy of Sciences, China, in 2000, and the B.S. degree from the Nanjing University of Science and Technology, Nanjing, China in 1996. Dr. He is author and co-author of more than 30 refereed publications at international conferences and journals. His research interests include wireless sensor networks, real-time embedded systems and distributed systems. He is a member of ACM and IEEE.

Tarek Abdelzaher received his B.Sc. and M.Sc. degrees in Electrical and Computer Engineering from Ain Shams University, Cairo, Egypt, in 1990 and 1994 respectively. He received his Ph.D. from the University of Michigan in 1999 on Quality of Service Adaptation in Real-Time Systems. He has been an Assistant Professor at the University of Virginia until his promotion with tenure in 2005, where he founded the Software Predictability Group. He is currently an Associate Professor at the Department of Computer Science, the University of Illinois at Urbana Champaign. He has authored/coauthored three book chapters and more than 60 refereed publications in leading conferences and journals in several fields including real-time computing, distributed systems, sensor networks, and control. He is an Associate Editor of the IEEE Transactions on Mobile Computing, the Journal of Real-Time Systems, the International Journal of Embedded Systems and the Ad Hoc Networks Journal, as well as Editor of ACM SIGBED Review. He was Guest Editor for the Journal of Computer Communications and the Journal of Real-Time Systems, and is Co-Editor of IEEE Distributed Systems Online. He served on numerous technical program committees in real-time computing, networking, quality of service, distributed systems, sensor networks, multimedia, and mobile computing, among others. He also held several conference organization positions including Program Chair of RTAS 2004, Demo Chair of Mobisys 2005, Poster Chair of ICDCS 2003, Sensor Networks Vice Chair of RTSS 2005, and General Chair of RTAS 2005. Abdelzaher's research interests lie broadly in understanding and controlling the temporal properties of software systems in the face of increasing complexity, distribution, and degree of embedding in an external physical environment. Tarek Abdelzaher is a member of IEEE and ACM.

VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance

Tian He, Sudha Krishnamurthy, John A. Stankovic, Tarek Abdelzaher, Liqian Luo, Radu Stoleru, Ting Yan, Lin Gu, Gang Zhou
Department of Computer Science
University of Virginia, Charlottesville, VA 22903
{tianhe, skrish, stankvoic, zaher, ll4p, rs6bd, ty4k, lg6e, gz5d}@cs.virginia.edu
and Jonathan Hui, Bruce Krogh
Department of Electrical and Computer Engineering
Carnegie-Mellon University, Pittsburgh, PA 15213
jwhui@cs.berkeley.edu, krogh@ece.cmu.edu

This paper describes one of the major efforts in the sensor network community to build an integrated sensor network system for surveillance missions. The focus of this effort is to acquire and verify information about enemy capabilities and positions of hostile targets. Such missions often involve a high element of risk for human personnel and require a high degree of stealthiness. Hence, the ability to deploy unmanned surveillance missions, by using wireless sensor networks, is of great practical importance for the military. Because of the energy constraints of sensor devices, such systems necessitate an energy-aware design to ensure the longevity of surveillance missions. Solutions proposed recently for this type of system show promising results through simulations. However, the simplified assumptions they make about the system in the simulator often do not hold well in practice and energy consumption is narrowly accounted for within a single protocol. In this paper, we describe the design and implementation of a complete running system, called VigilNet, for energy-efficient surveillance. The VigilNet allows a group of cooperating sensor devices to detect and track the positions of moving vehicles in an energy-efficient and stealthy manner. We evaluate middleware and system performance extensively on a network of 70 MICA2 motes. Our results show that our surveillance strategy is adaptable and achieves a significant extension of network lifetime. Finally, we share lessons learned in building such an integrated sensor system.

Categories and Subject Descriptors: C.2.1 [**Computer Communication Networks**]: Network Architecture and Design

General Terms: Design, Performance, Experimentation, Measurement

Additional Key Words and Phrases: Sensor networks, Energy conservation, Tracking, Wireless

1. MOTIVATION

One of the key advantages of wireless sensor networks (WSN) is their ability to bridge the gap between the physical and logical worlds, by gathering certain useful information from the physical world and communicating that information to more powerful logical devices that can process it. If the ability of the WSN is suitably harnessed, it is envisioned that WSNs can reduce or eliminate the need for human involvement in information gathering in certain civilian and military applications. In the near future, sensor devices will be produced in large quantities at a very low cost and densely deployed to improve robustness and reliability. They can be miniaturized into a cubic millimeter package (e.g., smart dust [Kahn et al. 1999]) in order to be stealthy in a hostile environment. Cost and size considerations imply that the resources available to individual nodes are severely limited. We believe, however, that limited processor bandwidth and memory are temporary constraints in sensor networks. They will disappear with fast developing fabrication techniques. The energy constraints on the other hand are more fundamental. According to R.A. Powers [Powers 1995], battery capacity only doubles in 35 years. Energy constraints are unlikely to be solved in the near future with the slow progress in battery capacity and energy scavenging. Moreover, the untended nature of sensor nodes and the hazardous sensing environment preclude manual battery replacement. For these reasons, energy awareness becomes the key research challenge for sensor network protocol design. Several researchers have addressed energy conservation recently. Most of them focus on particular protocols and investigate whether their energy conservation goal can be achieved. To the best of our knowledge, none of them investigate energy-conservation for a running system as whole. Normally they evaluate their approach through simulations. Simulation approaches tend to make simplified assumptions that often do not hold well in practice and they are subject to incompleteness. For example, in [Yan et al. 2003; Wang et al. 2003; Ye et al. 2003], several sensing coverage schemes are proposed for energy conservation. None of them consider energy consumption in activities other than sensing.

In this paper, we describe our effort that involves system design and implementation of VigilNet on a MICA2 platform with 70 MICA2 motes. The primary goal of the VigilNet is to support the ability to track the position of moving targets in an energy-efficient and stealthy manner. Our experimental results show that the probability of false alarms observed reaches zero when aggregation is achieved among more than 3 member motes. The experimental results we obtained also show that with 5% of deployed motes serving as sentries and the non-sentries operating at a 4% duty cycle, our algorithm extends the lifetime of a sensor network by up to 900%.

The main contributions of this paper are: 1) the design and implementation of an integrated system with energy-awareness as the main design principle across a whole set of middleware services, 2) mechanisms for dynamic control, which allow tradeoffs between energy-efficiency and system performance by adjusting the sensitivity of the system, and 3) a physical implementation and extensive field evaluation that reveal the practical issues that are hard to capture in simulation.

The remainder of this paper is organized as follows. Section 2 describes the requirements of a typical ground surveillance application. In Section 3, we describe the system setup and hardware components. In Section 4, we provide an overview of VigilNet design. In Section 5, we elaborate on the individual components of the system. In Section 6, we discuss the VigilNet implementation issues. We present experimental results in Section 7, and summarize the lessons learned from our experience in Section 8. We present related work in Section 9, Finally we conclude in Section 10 and discuss some future work in Section 11.

2. APPLICATION REQUIREMENTS

The VigilNet design is motivated by the requirements of a typical ground surveillance application. The general objective of such an application is to alert the military command and control unit in advance to the occurrence of events of interest in hostile regions. The event of interest for our work is the presence of moving vehicles in the deployed region. The deployed sensor devices must have the ability to detect and track vehicles in the region of interest. Successful detection and tracking requires that the application obtain the current position of a vehicle with acceptable precision and confidence. When the information is obtained, it has to be reported to a remote base station within an acceptable latency. Several application requirements must be satisfied to make this system useful in practice:

—**Longevity:** The mission of a surveillance application typically lasts from a few days to several months. Due to the confidential nature of the mission and the inaccessibility of the hostile territory, it may not be possible to manually replenish the energy of the power-constrained sensor devices during the course of the mission. Hence, the application requires energy-aware schemes that can extend the lifetime of the sensor devices, so that they remain available for the duration of the mission.

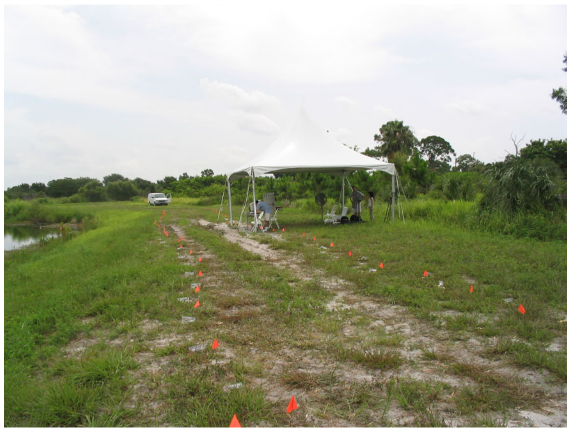
—**Adjustable Sensitivity:** The system should have an adjustable sensitivity to accommodate different kinds of environments and security requirements. In critical missions, a high degree of sensitivity is desired to capture all potential targets even at the expense of possible false alarms. In other case, we want to decrease the sensitivity of the system, maintaining a low probability of false alarms in order to avoid inappropriate actions and unnecessary power dissipation.

—**Stealthiness:** It is crucial for military surveillance systems to have a very low possibility of being detected and intercepted. Miniaturization makes sensor devices hard to detect physically; however, RF signals can be easily intercepted if sensor devices actively communicate during the surveillance stage. A zero communication exposure is desired in the absence of significant events.

—**Effectiveness:** The precision in the location estimate, and the latency in reporting an event are the metrics that determine the effectiveness of a surveillance system. Accuracy and latency are normally considered important metrics of tracking performance. However, the requirement of these two metrics can actually be slightly relaxed in many tracking applications. For example, it may be acceptable to obtain location estimation within several feet and receive a detection report within several seconds.

3. SYSTEM DESCRIPTION AND REQUIREMENTS

Figure 1 shows the deployment of our VigilNet surveillance system. We deployed 70 tiny sensor devices, called MICA2 motes [Horton et al. 2002], along a 280 feet long perimeter in a grassy field that would typically represent a critical choke point or passageway to be monitored. Each of the motes is equipped with a 433 MHz Chipcon radio with 255 selectable transmission power settings. While this radio is sufficient to allow the motes deployed in the field to communicate with each other, it is not capable of long-range ($>$ 1000 ft) communication when put on the



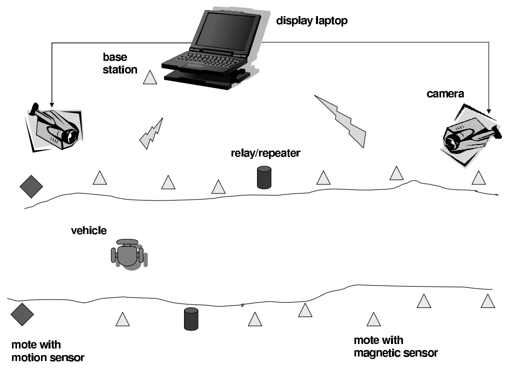


Fig. 1. Sensor Network Deployment

ground. Therefore, in a real system where the command and control units may be deployed several thousands of feet away from the sensor field, devices capable of long-range communication, such as replay, are deployed as gateways to assist the sensors to relay back information from the motes in the field to the base station. In this prototypical deployment, we use a mote as the base station that is attached to a portable device, such as a laptop. The portable device is the destination of the surveillance information and is mainly used for visualization in our prototype system. The camera devices shown in Figure 1 are controlled by the laptop to provide the next level of surveillance information, when triggered by the sensor field.

Each mote is equipped with a sensor board that has magnetic, acoustic, motion and photo sensors on it. While the different sensors make it possible for a mote to detect different kinds of targets, only the magnetic sensors are relevant to the application described in this paper. We use the HMC1002 dual-axis magnetometers from Honeywell [Honeywell]. These magnetic sensors detect the magnetic field generated by the movement of vehicles and magnetic objects. They have an omni-

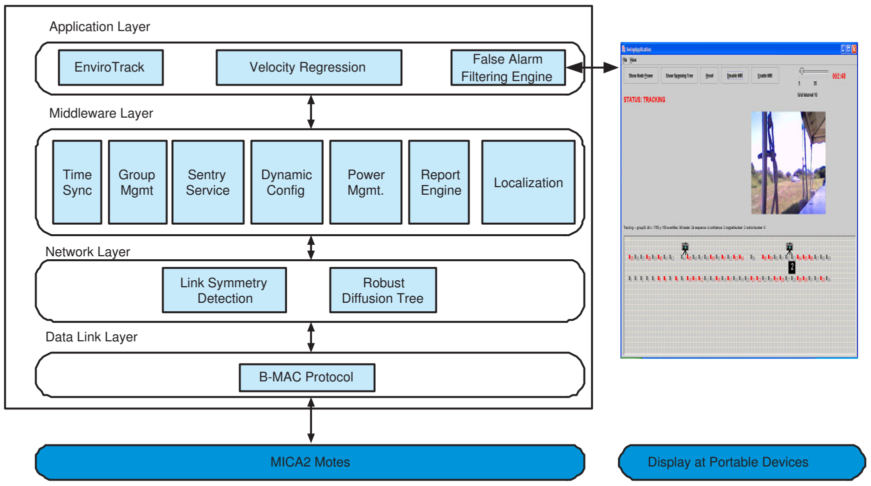


Fig. 2. VigilNet System Overview

directional field of view and are therefore less sensitive to orientation. They have a resolution of 27 μGauss and their sensing range varies with the size of the magnetic object they are sensing. From our experiments, we found that these sensors can sense a small magnet at a distance of approximately 1 ft and slowly moving passenger vehicles at a distance of approximately 8-10 ft.

4. VIGILNET SYSTEM OVERVIEW

The key contribution of this work is the design and implementation of a integrated wireless sensor network system that enables energy-efficient tracking and detection of events. Such a system is useful for surveillance applications, such as the one outlined in Section 2. The system we have designed is organized into a layered architecture comprised of higher-level services and lower-level components, as shown in Figure 2. It is implemented on top of TinyOS [Hill et al. 2000]. We first provide an overview of the different software components we have designed and then follow that with a detailed discussion of the role played by those components in the context of our tracking and surveillance application.

Time synchronization, localization, and routing comprise the lower-level components and form the basis for implementing the higher-level services, such as aggregation and power management. Time synchronization and localization are important for a surveillance application because the collaborative detection and tracking process relies on the spatio-temporal correlation between the tracking reports sent by multiple motes. The time synchronization module is responsible for synchronizing the local clocks of the motes with the clock of the base station. The localization module is responsible for ensuring that each mote is aware of its location. In our prototype system, we design and implement the walking GPS solution [Stoleru et al. 2004], which assigns motes their location at the time they are deployed. Once the technique is mature enough, this static configuration can be replaced with dynamic localization schemes such as in [He et al. 2003].

The routing component establishes routes through which the motes exchange information with each other and the base station.

Power management and collaborative detection are the two key higher-level services provided by VigilNet. The sentry service component is responsible for power management, while the group management component is responsible for collaborative detection and tracking of events. The sentry service conserves energy of the sensor network by selecting a subset of motes, which we define as *sentries*, to monitor events. The remaining motes are allowed to remain in a low-power state until an event occurs. When an event occurs, the sentries awaken the other motes in the region and the group management component dynamically organizes the motes into groups in order to enable collaborative tracking. Together, these two components are responsible for energy-efficient event tracking.

All the deployed motes are programmed to run the distributed application. VigilNet supports the ability to reprogram the motes dynamically with new configuration parameters such as sensitivity. This eliminates the need to download the application code on all the motes each time the configuration is modified. We have a display module for portable devices (Figure 2)which is not part of the software that runs on each mote. We use it primarily for visualization and debugging purposes. Optionally, the display software also has the logic to filter out any residual false alarms that have not been filtered out in the network. We now elaborate on how the individual components of the system shown in Figure 2 interact with each other in the context of a typical tracking application. In particular, we discuss the design decisions that make the target system energy-efficient and illustrate trade-offs between performance and energy-awareness.

5. TIME-DRIVEN SYSTEM DESIGN

In VigilNet, the MICA2 motes prepare for tracking by going through an initialization process. This process is used to synchronize the motes, set up communication routes, and configure the system with the correct control parameters. The initialization process proceeds in a sequence of phases and the transition between phases is time-driven, as shown in Figure 3. Phases I through IV comprise the initialization process which normally takes about 2 minutes. At the end of phase IV, the motes begin the power management and tracking activity. After performing this activity for a certain duration of time (e.g., one day), they begin a new system cycle. The duration of each phase is a control parameter that can be dynamically configured by the base station. Our multi-phase cyclic process satisfies following design objectives:

—First, it eliminates interference between operations. The constrained bandwidth in MICA2 doesn't allow a high concurrency in communication. If all operations run simultaneously, the traffic will severely interfere with each other.

—Second, we can confine the exposure of sensor activity within a short period time during the initialization phase (phase I to IV). As a result, the system can achieve zero exposure (complete stealthiness) during surveillance when no significant event happens.

—Third, a new system cycle is a natural way to allow the rotation of sentry responsibility among motes in order to achieve uniform energy dissipation across

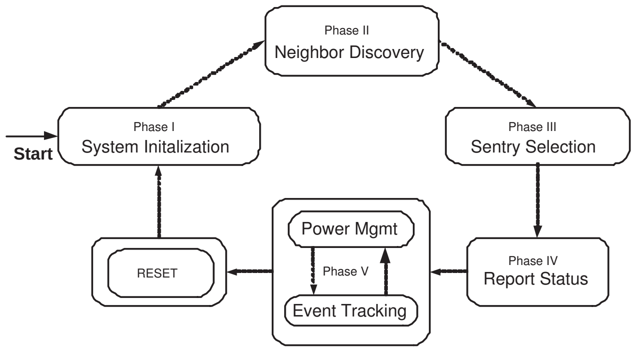


Fig. 3. Time Driven System Transition

the network.

—Last, the cycling introduces system-wide soft-states. It allows the motes to periodically synchronize their clocks to avoid significant clock drifts over time. In addition, since mote failures and new deployment may occur anytime during a cycle, a new system cycle gives the remaining motes an opportunity to repair routes and discover new neighbors.

We now discuss the activities occurring during each phase of the system cycle in more detail.

5.1 Phase I: Basic Initialization

We observe that three functions in our system need system-wide broadcast: time synchronization, network backbone creation and system-wide reconfiguration. These functions can be isolated into three different modules that perform separately. However, such a design would not be bandwidth and energy efficient due to the multiple flooding phases required. Instead, we use a unique application-specific design to perform these operations simultaneously in one flooding operation to reduce overhead as described in following sections.

5.1.1 *Time Synchronization.* System initialization begins with time synchronization. Several schemes proposed recently are able to achieve a high synchronization precision, however they do not match well with VigilNet requirements. GPS-based schemes typically achieve persistent synchronization with a precision of about 200 ns. However, GPS devices are expensive and bulky. The reference broadcast scheme (RBS) proposed in [Elson and Romer 2002] maintains information relating the phase and frequency of each pair of clocks in the neighborhood of a node. The relation is then used to perform time conversion when comparing the timestamps of two different nodes. While RBS achieves a precision of about 1 μs, the message overhead in maintaining the neighborhood information is high and may not be energy-efficient in large systems.

We argue that fine-grained clock synchronization achieved by costly periodic beacon exchanges may not be suitable for the energy-constrained surveillance system. Moreover continuous adjustment through beaconing in these solutions [Elson and Romer 2002] defeats our purpose of stealthiness. In our system, we value energy-efficiency and stealthiness above high synchronization precision. Therefore, we use a lightweight scheme that synchronizes the motes only during the initialization phase, using a synchronization beacon broadcast by the base station at the beginning of each initialization cycle. Since the underlying MAC layer provided by TinyOS does not guarantee reliable delivery, the base station retransmits the synchronization beacon multiple times. The synchronization beacons are propagated across the network through limited flooding with timestamp values reassigned at intermediate motes immediately prior to the transmission of the timestamp. This eliminates the uncertainty in MAC contention delay. Receivers take the timestamp from the beacon plus a fix hardware delay as their own local time. To satisfy the stealthiness requirement, we confine time synchronization within the initialization phase. The timer drift accumulated overtime is rectified by a new system cycle (i.e., a repeated initialization phase).

5.1.2 *Diffusion Tree Creation.* While the primary purpose of the synchronization message is to coordinate the clocks of the motes, it also serves as an exploratory message for motes to set up reverse routes to the base station, like the technique used by directed diffusion [Intanagonwiwat et al. 2000]. The route that is set up during the propagation of the time synchronization message is essentially a diffusion tree rooted at the base station. The decision to use a diffusion tree is made based on several observations. 1) Sent along with the time synchronization operation, it is nearly free of cost in communication and code memory. 2) It allows any leaf motes to go to sleep without disrupting communication of other motes.

We encounter two practical issues when implementing the diffusion tree algorithm on the MICA2 platform.

—**Mote Failures:** The failure of a MICA2 mote can disable a subtree below it. Initially, we attempted to add failure detection to the MAC layer to quickly identify link failures and choose alterative routes. Soon, we discovered that link layer reliability in such a bandwidth constrained platform is too heavyweight and the effective data rate is reduced by nearly 50%. With such an observation, we introduce soft-state into the diffusion tree. The diffusion tree is refreshed per system cycle to prune failed links and discover new routes. After this modification, no bandwidth penalty is experienced during data communication.

—**Asymmetric Links:** Low power radio components, such as Chipcon CC1000 used by MICA2, exhibit very irregular/anisotropic communication patterns [Zhou et al. 2004], especially when sensor nodes are placed on the ground. If motes choose their parents without considering the distance separating them, it results in asymmetric links which leads to different reception rates along different directions between the same pair of motes. This asymmetry can be solved by link layer handshaking; however we discovered that it is very expensive. Our solution to this issue is called Link Symmetry Detection (LSD). The purpose of LSD is to reduce the impact of radio irregularity on upper layer protocols. The main idea of the link symmetry detection is to build a symmetry overlay on top of the anisotropic radio layer, so that those protocols whose correctness depends on link symmetry can be used without modification. Symmetry detection is done by local beaconing. A sending node adds the IDs of all its neighbors it has discovered into the beacon. When a node receives a beacon, it registers the sender into its local neighbor table, and then checks whether its own ID is in the beacon message or not. If it is, it labels this communication link to the sender as *SYMMETRIC*. Otherwise, it labels the communication link between them as *ASYMMETRIC*. This labeling process is repeated several times to get a statistical evaluation of a link's symmetric communication quality. Only those links that have higher symmetric communication qualities than the specified threshold are available for upper layers, and all other links are blocked from higher layer protocols. We evaluate our solution in Section 7.3

5.1.3 *Dynamic Reconfiguration.* The capability of dynamic reconfiguration facilitates re-tasking of sensor networks for future changes of mission requirements. Currently, this capability makes our work in system tuning and debugging much easier. When we deployed 70 motes on the field for the first time, it took us an hour to collect the motes and reprogram them manually, before the reconfiguration capability was added into the system. Now we can reconfigure the network within 1 minute. VigilNet supports reconfiguration with the help of the time synchronization message. The base station piggybacks the values of the control parameters in the synchronization message and motes adopt the new values when they accept the synchronization message. Such a strategy is energy-efficient, because it comes along with time synchronization beacons, obviating the need to send separate messages to reset parameters on the motes. Examples of control parameters that can be dynamically reconfigured include the duration of each phase shown in Figure 3, the duration for which a mote remains asleep and awake when power management is enabled, the sampling rate and the degree of in-network aggregation. This reconfiguration capability enables us to dynamically trade off between the energy-awareness and tracking performance as we show later in this paper.

5.1.4 *Localization.* Due to inherent irregularity in radio propagation and limited effective ranges in distance measurements through acoustic/ultrasound, little progress has been made in sensor network localization over a large area. As the first step, we design and implement a walking GPS solution [Stoleru et al. 2004] based on the fact that currently sensor nodes are deployed manually in the field. In this solution, the deployer (either person or vehicle) carries a GPS device that periodically broadcasts its location. The sensor nodes being deployed, infer their position from the location broadcast by the GPS device. This solution enabled us to push all complexity derived from the interaction with the GPS device to a single node, the GPS Mote, and to significantly reduce the size of the code and data memory used on the sensor node. Through this decoupling, a single GPS Mote is sufficient for the localization of an entire sensor network, and the costs are thus reduced. We built a prototype, called the GPS Mote assembly, that can be worn during the deployment. This prototype consists of a GPS device mounted on top of a bicycle helmet. The GPS device is connected through and RS232 cable to the GPS Mote that is attached with a velcro to a wristband. Figure 4 illustrates the prototype. We will evaluate our localization solution in Section 7.2.

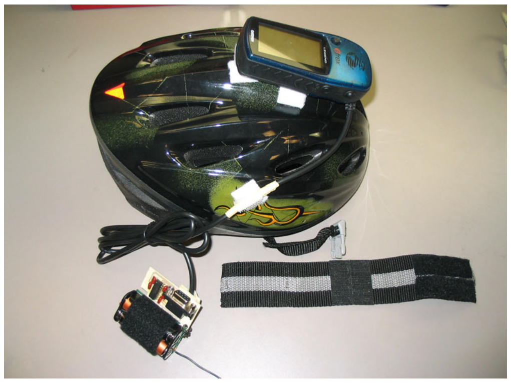


Fig. 4. GPS Mote Assembly

5.2 Phase II: Neighbor Discovery

After the basic initialization phase, the motes make a transition to a neighbor discovery phase. Motes notify their neighbors by locally broadcasting `HELLO` messages. In the `HELLO` message, a sender sends its identifier, its status indicating whether it is a sentry or not, the number of sentries that are currently covering it and its location. The sender also identifies the sentry mote it reports to, if it is covered by at least one sentry. This local information is used to build a neighborhood table at each mote, and forms the basis for sentry selection in Phase III.

5.3 Phase III: Sentry Selection

In our sentry selection scheme, the decision to become a sentry is made locally by each mote, using the information gathered from its neighbors (the neighbor discovery goes through Phase II and III).

A mote decides to become a sentry if any one of the following conditions holds. 1) it is one of the internal nodes of the diffusion tree, or 2) it discovers that none of its neighbors either is a sentry or is covered by a sentry. When a mote decides to become a sentry, it advertises its intent. Three practical issues need to be solved to make this scheme work in a running system:

—**Race Conditions:** Contention occurs when multiple motes in the same neighborhood decide to become sentries at the same time. In order to reduce the collision probability, each mote uses a random backoff delay to transmit a `SENTRY DECLARE` message. If a mote receives a `SENTRY DECLARE` message from one of its neighbors during the backoff period, it updates its neighborhood table and cancels any pending outgoing `SENTRY DECLARE` messages. It then re-evaluates its decision to become a sentry based on the updated neighborhood information. If the mote finds that it is still necessary for it to become a sentry, it repeats the sentry declaration process described above.

—**Energy Balancing and Efficiency:** We set the backoff delay of a mote inversely proportional to its residual energy. Thus, a mote with higher residual energy has a greater likelihood of being selected as a sentry, thereby balancing the energy dissipation uniformly across the network. The backoff delay of a mote

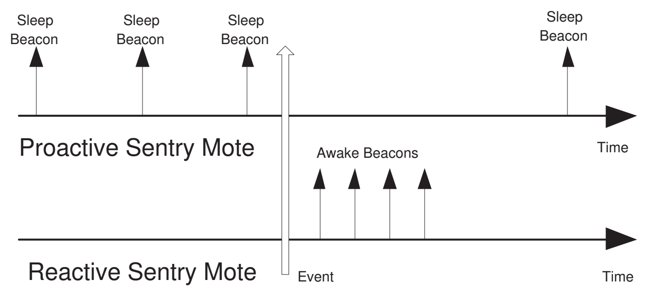


Fig. 5. Two Power Management Schemes

is also inversely proportional to the number of neighbors that are not covered by a sentry. Thus, motes in regions where there is insufficient sensing coverage are favored for being selected as sentries. The key feature of this sentry selection algorithm is that it provides an adaptive, self-configuring technique for choosing the sentries purely based on local information. However, the lack of global knowledge may result in a non-optimal number of sentries.

—**Sensing Coverage:** Surveillance addresses the sensing coverage problem of every physical point in the terrain, instead of communication coverage as in LEACH [Heinzelman et al. 2000b] and SPAN [Chen et al. 2001]. Since the sensing range of our Honeywell magnetometer [Honeywell] is much smaller than the Chipcon radio range, we need to use a smaller transmission power setting to send out `SENTRY DECLARE` messages in order to ensure sensing coverage. The power setting is chosen in such a way that there is at least one sentry within each sensing range. Unlike [Yan et al. 2003; Wang et al. 2003], this unique design enables us to provide sensing coverage without the requirement of localization. More details can be found in the evaluation Section 7.1.

5.4 Phase IV: Status Report

After the routing backbone is finalized, all the motes use the backbone to report their status to the base station in Phase IV. The base station forwards those reports to the display module, which can then be used to visualize the network topology, residual energy distribution and sentry distribution and detect any failed motes. Since the sole purpose of Phase IV is for visualization and debugging, it is optional.

5.5 Phase V-A: Power Management

The selection of sentries sets the stage for the power management phase. In this phase, the non-sentry motes alternate between sleep and wakeup states. A mote in the sleep state conserves power by disabling all processing, including those that are related to communication and sensing. We have proposed and implemented two different schemes to control the sleep-wakeup cycle. Now we discuss the pros and cons of these two schemes to clarify some practical issues

In the first implementation, which we call **proactive** control (Figure 5), the sentry mote sends out sleep beacons periodically. A non-sentry mote stays awake until it receives a beacon from its sentry mote, signaling the non-sentry mote to sleep for a certain duration of time. Upon receiving the sleep beacon, the non-sentry mote makes a transition to the sleep state and remains in that state for the specified amount of time. It wakes up when the timer expires and repeats the process by waiting for the next sleep beacon. Since neighboring non-sentry motes are likely to receive the same sleep beacon, their sleep-wakeup cycle proceeds in a lock-step fashion. The regular synchronization of the non-sentry motes with their respective sentries is beneficial in two ways. First, it allows multiple motes to receive the same beacon, and obviates the need to send out individual sleep beacons to put each non-sentry mote to sleep. This reduces the message overhead. Second, since motes in a neighborhood are all awake at the same time, the correlated sleep-wakeup cycle helps improve the tracking efficiency.

The second implementation to control the sleep-wakeup cycle is called the **reactive** control (Figure 5). In this scheme, the sentries are not required to send out explicit beacons to put the non-sentry motes to sleep. Instead, the transition between sleep and wakeup states is timer-driven. Each non-sentry mote remains awake for *awakeDuration* amount of time and then sleeps for *sleepDuration* amount of time. A non-sentry mote breaks out of its cycle and remains awake for a longer duration only when receiving an awake beacon from a sentry mote.

The reactive scheme is more stealthy compared to the proactive scheme, because no unnecessary beacons are sent unless an event occurs. Hence, the reactive approach is more appropriate for a surveillance application. However, one practical issue needs to be solved in the reactive scheme; since the non-sentries do not periodically synchronize their clocks with the clocks of their sentries, the clocks of the non-sentry motes may drift in course of time. Consequently, neighboring non-sentry motes may no longer have a sleep-wakeup cycle that is strictly in lock-step. As a result, a sentry no longer knows for certain which of its neighbors are awake. It has to retransmit the awake beacon multiple times in order to awaken non-sentries when an event occurs (Figure 5). We compare the message overhead between the proactive and reactive schemes in Section 7.6.2.

5.6 Phase V-B: Event Tracking and Reporting

After the sentry backbone has been created and power management is enabled, the motes are ready for tracking. Tracking and power management are toggle-states in phase V. When an event happens, motes wakeup and start tracking, when event disappears, motes toggle back to power management states.

A simple way to track events is by allowing each mote that has sensed an event to report its location and other relevant information about the event to the base station. The base station can then filter out the false alarms and infer the location of the event from the genuine reports. The advantage of this approach is that it allows all of the complex processing of the sensor readings to be deferred to the more powerful base station. However, the main drawback is that, if the motes are densely deployed, multiple motes may sense the event at the same time and send their individual reports to the base station. This results in higher traffic and wasteful expenditure of energy which can be reduced by aggregating multiple reports about the same event and sending a digest, instead of the individual reports to the base station. Previous in-network aggregation techniques fuse the data at the source through cluster headers [Heinzelman et al. 2000b] and/or along the route back to the sink [Bhattacharya et al. 2003][He et al. 2004][Intanagonwiwat et al. 2000][Madden et al. 2002]. In addition, Zhao [Zhao et al. 2002] propose a optimal sensor selection approach to aggregate the fidelity of detections while eliminating redundant communication.

The system we have designed also performs in-network aggregation by organizing the motes into groups. However, different from previous schemes, the groups in our work are more dynamic in the sense that they are formed in response to an external event and migrate when an event moves. A group represents an event uniquely and exists only as long as the event is in the scope of the sensor field. The design of our group management and tracking component is described in [Blum et al. 2003]. We review its key features here for completeness. It should be noted that the work reported in this paper is the first real implementation of the aforementioned design.

Each mote is programmed to detect an event by its sensory signature. This signature is a condition on the output of a filter that processes the raw sensory measurements (and removes noise). When the indicated condition is detected by a set of nearby motes, the group management component reacts by creating a group. All motes that detect the same event join the same group. The main contribution of the group management component, described in [Blum et al. 2003], is to establish a unique one-one mapping between a group and a physical event as well as to maintain the membership of the group as the event moves through the environment. It is assumed that different events are far enough apart that membership of motes to the corresponding groups can be decided without ambiguity based on spatial adjacency to one of the events.

Each group is represented by a *leader* to the external world. Group members (who by definition can sense the tracked event) periodically report to the group leader. The leader records each report keeping only the most recent one from each member. Reports that are older than a certain threshold are purged. We define the confidence level of event detection as the number of distinct motes that have reported the event in the last t_r units of time. When the confidence level of detecting an event is at least as high as the threshold required by the application, called the *degree of aggregation* (DOA), the leader sends a digest of the reports to the base station. The confidence threshold can be tuned to manipulate the sensitivity of the system. A low threshold increases sensitivity at the expense of possible false alarms. A high threshold could result in missing some smaller targets. The effect of manipulating the degree of aggregation is explored experimentally in Section 7.4.2.

5.7 Velocity Estimation

In addition to provide traces of the targets, VigilNet also estimates the velocity of targets. Velocity estimation is rather straight forward if detections are reported in order and there is no false alarms. Unfortunately in practice, both conditions do not hold well. To reduce the impact of such disturbance, we use least-square estimation to obtain velocity of the targets and use spatiotemporal relationship between consecutive reports to filter out false alarms. Specifically, each report includes a tuple $(timestamp, x, y)$. The “timestamp” shows the time when a group lead sends the report, and “x” and “y” shows the triangulated location reported for the target. When the number of reports in a group accumulates over a threshold, the velocity of the target is calculated by a least-square estimation. The x-component

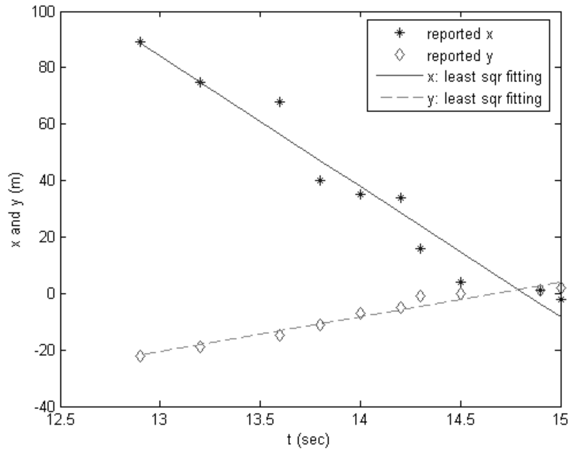


Fig. 6. Velocity Estimation

and y-component of the velocity are calculated separately according to Equation 1 (the number of reports for the velocity calculation is an adjustable parameter).

$$
\begin{aligned}
Vel_x &= \frac{\sum_{i=0}^{N-1} (x_i - \bar{x})(t_i - \bar{t})}{\sum_{i=0}^{N-1} (t_i - \bar{t})^2} \quad where \quad \bar{x} = \frac{\sum_{i=0}^{N-1} x_i}{N} \\
Vel_y &= \frac{\sum_{i=0}^{N-1} (y_i - \bar{y})(t_i - \bar{t})}{\sum_{i=0}^{N-1} (t_i - \bar{t})^2} \quad where \quad \bar{y} = \frac{\sum_{i=0}^{N-1} y_i}{N}
\end{aligned} \tag{1}
$$

In Equation 1, (t_i, x_i, y_i) $i = 0, ..., N-1$ are the latest reports from the same group. Figure 6 shows the least square fitting of the x-component and y-component of the reported locations, and the slopes of the two fitting lines are the x-component and y-component of the calculated velocity. This data are obtained from one of field test.

Once the velocity is known, we can filter out false alarms, if a report contains an unreachable position, given difference in time stamp since the last valid report. We evaluate the performance of velocity estimation further in Section 7.5.

6. IMPLEMENTATION

The architecture described in Section 4 was built on top of TinyOS [Hill et al. 2000]. TinyOS is an event driven computation model, written in NesC [Gay et al. 2000] specifically for the motes platform. TinyOS provides a set of essential components such as hardware drivers, scheduler and basic communication protocols. These components provide low level support for application modules, which are also written in NesC. NesC is a C-like language that enables the programmers to define the function of components and the relations (dependencies) among them. Components from TinyOS and user applications are processed by the NesC compiler into a running executable, which runs (in our case) on the MICA2 mote platform. MICA2 is the third generation mote built for wireless sensor networks [CrossBow]. Besides normal computation and communication capabilities, MICA2 motes have (i) selectable transmission power settings (255 levels) which enable us to dynamically adjust the communication range, (ii) a power control function with up to six sleep modes provided by the ATmega128 Microcontroller, and (iii) a wireless reprogramming capability that eliminates the need for manual code downloads. The first two functions are utilized extensively by our protocols. The last facilitates deployment. In particular, we use a lower communication power setting during neighbor discovery for diffusion tree creation. This ensures that when the diffusion tree is created and communication power is subsequently increased, all found edges along the tree are quite reliable. In contrast, running diffusion tree creation at the normal power setting could result in unreliable or asymmetric edges between some nodes. This choice would ultimately reduce performance.

The implementation of VigilNet on the MICA2 motes was driven by several requirements that arise from platform limitations. Namely:

—**Energy Efficiency:** MICA2 operates on a pair of batteries that approximately supply 2200 mAh at 3V. It consumes 20mA if running a magnetic sensing application continuously which leads to a lifetime of 5 days.

—**Bandwidth Efficiency:** The Chipcon radio on MICA2 provides an effective data rate of 12.4kbps, which equals a maximum packet rate of 43 pkts/sec. Our experiments show that a mote barely reaches 20 pkts/sec when it is exposed to channel contention.

—**Simplicity:** Our system requires many essential functions shown in Figure 7 to make target tracking efficient, while the whole system must fit in 4K data memory and 128K code memory. This necessitates a simple, yet effective, design for the MICA2 platform.

—**Flexibility:** Our prototype system spans 280 feet and comprises 70 motes. Once deployed, motes can not be easily collected. Dynamic configuration is desirable for fast performance tuning and debugging.

6.1 Software Architecture

The architecture of VigilNet, written in NesC, is shown in Figure 7. The whole system occupies 39,496 bytes of code memory and 3,725 bytes of data memory. We divide system components into four major groups; initialization, tracking, power management, and general utilities. Initialization components are responsible for basic infrastructure establishment. Tracking components support the event tracking functions. The SentryPM module performs power management which puts motes to sleep as described earlier, when no significant events are detected. We also use some utilities to facilitate downloading, debugging, tuning and statistical logging. We provide a *backbone* module which is in charge of time-driven transitions between phases. We also use this module to pass state information among other modules to reduce the dependency among components.

In implementing the above architecture, several system challenges were met, primarily due to lack of common operating system support which TinyOS doesn't

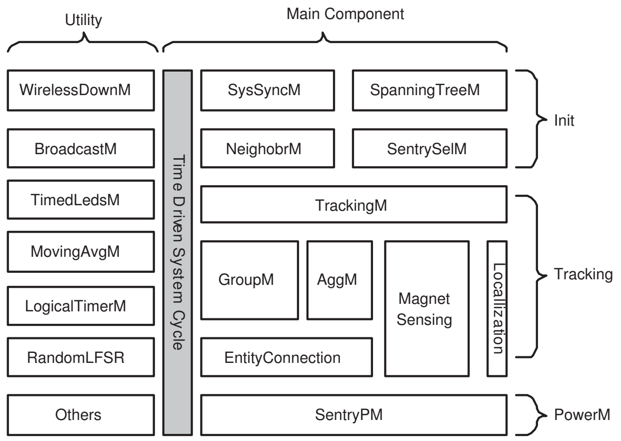


Fig. 7. System Architecture in NesC

have. Some of the most important issues were the following:

Concurrency Control: TinyOS provides minimal support for concurrency control. The latest NesC compiler detects potential data races and give warnings at compile-time, however, it still requires the programmer to deal with it. Data races can be avoided by atomic sections or tasks. An atomic section is implemented through disabling and enabling interrupts. This requires the critical section to be very short. Otherwise, the system will become unresponsive. For example, if the soft timer cannot get updated by clock interrupts, time drift will happen. A better approach is to put all operations that access shared data into a task context. This guarantees sequential access to the data. However, the current task model doesn't allow parameter passing. The solution to this limitation is to put parameters into shared variables accessible by all tasks and use atomic sections to protect the read and write operation on these variables.

Packet Scheduling: For now, the TinyOS communication module doesn't provide a buffering mechanism. It is often the case that multiple components send out packets concurrently. All but one operation fails due to the mutual exclusion mechanism described above, used in the lower layer. The current solution we used is to provide application layer buffering. We reinitiate the transmission with linear backoff when contention happens.

Aggregation: The TinyOS communication module has a relatively high overhead. The packet header is 7 bytes (MAC header+ CRC) and the preamble overhead is 20 bytes in MICA2. For a default payload size of 29 bytes, the overhead to send a single packet is 48%! This limitation motivates us to use aggregation techniques. We use piggybacking whenever possible to increase the effective data rate. For instance, we piggyback system-wide parameters in time synchronization messages and piggyback sentry declaration information in neighbor beaconing. A more advanced aggregation technique such as in [He et al. 2004] is desired to efficiently use bandwidth.

Hardware Limitations: In general, the MICA2 platform is effective in supporting our system. However, in some cases, we have to modify our design to

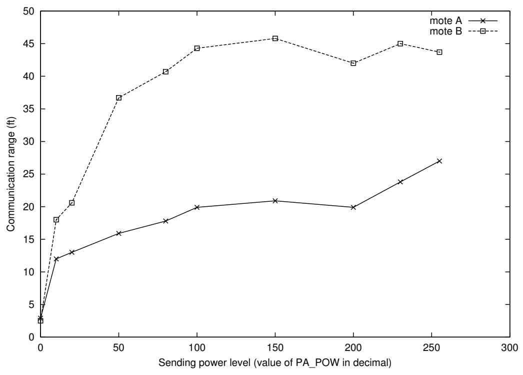


Fig. 8. Impact of Sending Power on RF Range

accommodate the limitations on hardware. First, the MICA2 mote has no circuit support for remote passsive wakeup [Gu and Stankovic 2004]. The current snooze implementation relies on a timer interrupt. This increases the chance of false negatives when the sleep duration of non-sentries is relatively long. Second, while the operating frequency of the Chipcon radio is selectable, external hardware attached to the chip can only support one frequency. This prevents us from designing a better collision avoidance algorithm to improve radio performance.

Due to space limitations, here we only give a snapshot of the issues we encountered during the implementation. In general, we feel that platform-specific system designs are necessary to improve the performance.

7. PERFORMANCE EVALUATION

We now present experimental results that evaluate the performance of the physical system described in the previous section. We obtained most of the experimental results through an actual deployment of MICA2 motes in a grassy field, using the setup described in Section 3. However, for some experiments which require a long duration of time, we can not afford to deploy the system unattended due to security issues. Instead we conduct this type of experiments with a smaller number motes in controlled environments. In addition, simulations are also used to reveal the tradeoff between different design decisions.

We classify the experiments into three broad categories. The first set of experiments evaluate the basic capabilities of VigilNet such as the MICA2 radio in different environments, performance of walking GPS localization and symmetry detection. The second set of experiments evaluate the performance of the tracking component. Finally, we evaluate the sentry service and the power management features of our system.

7.1 Evaluation of Capability of MICA2 Radio

The communication range of a MICA2 mote depends on several factors, such as the length of the antenna, the transmission power, the elevation above the ground, and the non-line-of-sight effects from objects in the surroundings (e.g., grass, trees,

Table I. Impact of Antenna Lengths on RF Range

Antenna	Power level = 50	Power level= 255
17.3 cm	37 ft	43 ft
34.6 cm	59 ft	> 84 ft

Table II. Impact of Elevations on RF Range

Elevation	0 ft	0.5 ft	1 ft
Mote A	27 ft	30 ft	> 84 ft
Mote B	43 ft	> 84 ft	> 84 ft

buildings, people, cars). Although the absolute values may vary in different environments, we can still draw some general observations about the MICA2 platform:

—We measure a set of MICA2 communication ranges under different sending power settings with two senders and one receiver. Results shown in Figure 8, indicate that 1) the communication range nonlinearly increases as the sending power increases. It increases more slowly when the power setting is large. 2) Asymmetry in communication range is more than what we expect, and it might primarily come from the differences in hardware calibration.

—We measure MICA2 communication ranges under different antenna lengths and different elevations above the ground. As expected, Table I indicates that longer antennas can significantly increase communication range in MICA2. Table II shows that the high elevation reduces floor attenuation, and hence increases RF range.

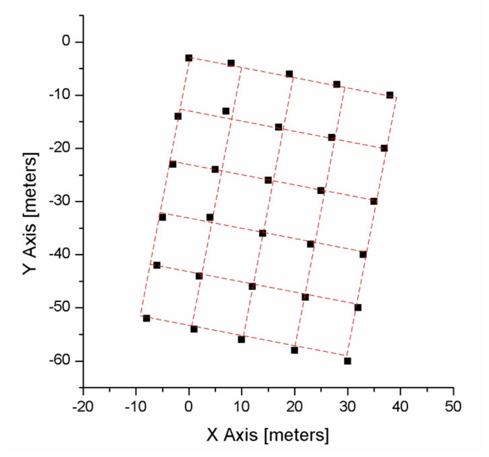
7.2 Evaluation of Walking GPS Localization

VigilNet use walking GPS as a practical solution for manually deployed sensor networks. This solution is evaluated in an open grass field. We marked a 6x5 grid with 10 meters grid side length on the ground and we deployed the sensor motes in this grid. We note that a grid is used to only to facilitate evaluation. In actual deployment, geometric layout of individual sensors doesn't affect the performance.

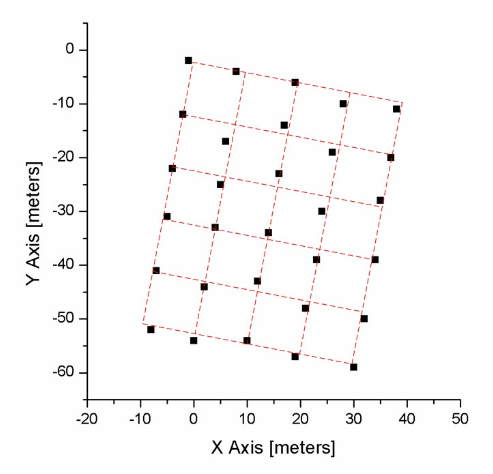
We evaluate walking GPS localization under two different deployment methods. In the first method, each mote is turned on right before being deployed; In the second method, each mote is powered on all the time. The experimental results for both deployment methods are shown in Figure 9.

The average localization error obtained from fitting a grid to the experimental data is 0.8 ±0.5 meters for the first deployment method and 1.5 ±0.8 meters for the second deployment. The less accurate location estimation in the second deployment is mainly because of the imprecise inference of the exact moment a sensor node was placed on the ground.

Since radio range is for MICA2 on the ground is about 10 meters, this absolute error equals about 10-15% normalized localization error. Studies in [He et al. 2003] demonstrates that such localization accuracy is sufficient for routing, sensing and tracking operations.



(a) First Deployment Method



(b) Second Deployment Method

Fig. 9. Performance of Walking GPS Localization

7.3 Evaluation of Symmetry Detection

As mentioned in Section 5.1.2, system routing infrastructure is built on top of a symmetry overlay on top of the anisotropic radio layer. During the construction of the diffusion tree, the symmetry detection blocks all the asymmetric links. In this experiment, we evaluate performance of the symmetry detection service, by counting the percentage of nodes that are able to report back their status information successfully. We conduct the experiment with 27 MICA2 motes and the result is given in Figure 10.

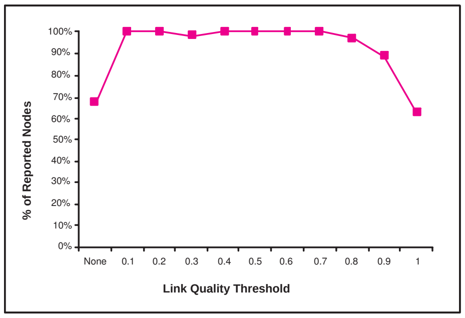


Fig. 10. Performance Evaluation of Asymmetry Detection Service

When the symmetry detection is disabled, which allows upper layer protocols use any link available, only 67.4% nodes are able to successfully report information because diffusion-like protocols need symmetric reverse path back to the base. If a node chooses a parent it actually can not reach, the routing failure would happen. However, when the symmetry detection is used, we observe that all nodes are able to successfully report back to the base station, when we choose the link quality threshold between 10% and 70%. This performance improvement is attributed to the symmetry detection, which prevents a node from choosing unidirectional links.

As shown in Figure 10, system performs very well, even when the link quality threshold is set very low, as low as 10%. We attribute this to retransmission supported in our system, in case of communication failures. However, we also note that retransmission alone can not achieve this good performance. Once symmetry detection is disabled, even with retransmission, only 67.4% nodes report back.

On the other hand, when the link quality threshold keeps increasing and is close to 100%, the system performance decreases. This is because symmetry detection uses neighbor exchange to estimate the link quality. Link quality can be affected not only by anisotropic radio patterns, but also by congestion. It is possible that a certain link is symmetric, however, can not reach 100% link quality due to transient congestion. If we cut all non-perfect links, it is possible that a node can not find any reverse path back to the base, which leads to poor delivery performance shown in Figure 10.

7.4 Evaluation of In-Network Aggregation

In this experimental setup, we deployed 70 MICA2 motes along two sides of a road at a distance of 7-8 ft from each other. They were deployed densely in order to improve the data aggregation among motes.

Our goal is to track a car being driven along the stretch of road and study the impact of system parameters on the tracking performance. One key parameter is the degree of aggregation (DOA). This parameter decides the sensitivity of the surveillance system and is used to trade off between energy-awareness and surveillance performance. It is defined in our system as the minimum number of reports about

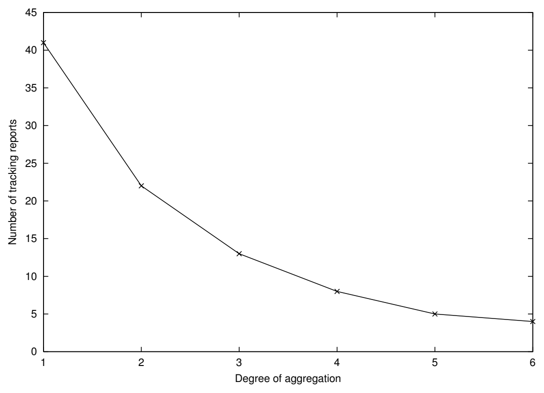


Fig. 11. Impact of DOA on the Message Overhead

an event that a leader of a group waits to receive from its group members, before reporting the event's location to the base station. In our implementation, the value of the DOA is dynamically configurable from the base station. We were interested in studying the impact of the degree of aggregation on the following metrics:

—the number of tracking reports (Figure 11),

—the number of false alarms generated (Figure 12), and

—the latency in reporting an event (Figure 13).

7.4.1 *Impact of Aggregation on Transmission Overhead.* In our tracking experiments we drove a car at a speed varying between 5-10 mph. We varied the degree of aggregation from 1 to 6 and repeated the tracking experiment for each value of DOA ten times. Figure 11 shows how the number of the tracking reports received by the base station varies with the DOA. From the figure, we see that when the value of DOA increases from 1 to 2, the number of tracking reports reduces by almost 50%. As the value of DOA increases even further, we observe that there is a steady drop in the number of tracking reports generated. These results verify the fact that the in-network aggregation, resulting from organizing the sensor motes into groups, significantly reduces the message overhead during tracking, and hence leads to much less energy consumption in data transmission.

7.4.2 *Impact of Aggregation on False Alarms.* Our next experimental result shows how the degree of in-network aggregation affects the false alarms generated when tracking an event. False alarms are normally caused by events such as burst distortions of readings due to power state transitions and incorrect readings from faulty sensors. Since a simulation-based approach normally assumes that sensors behave according to their specifications, such phenomena are usually not investigated in simulation. We classify false alarms into *false positives* and *false negatives.* A false positive occurs when a group of motes report the presence of the moving car in their neighborhood, when in reality, the car is not in their vicinity. A false negative occurs if the base station does not receive any reports of the car, although in reality, there is a car moving though the sensor field. In other words, if the car never appears on the display as it moves from one end of the sensor field to the

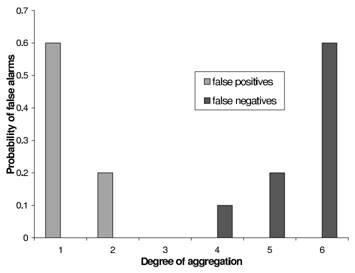


Fig. 12. Impact of DOA on False Alarms

other, we treat it as a false negative. It is important to emphasize that we do not consider a delayed report as a false negative.

We determined the probability of false alarms for each value of DOA by counting the number of false positives and false negatives we observed on the display during a set of 10 tracking rounds. Figure 12 shows how the probability of false positives and the probability of false negatives are each affected by the degree of aggregation. From Figure 12 we see that as the value of DOA increases from 1 to 6, the probability of false positives drops from 0.6 to 0, while the probability of false negatives increases from 0 to 0.6. These results can be explained as follows.

When the DOA = 1, the leader of a group reports the event to the base station, as soon as at least one member of the group detects the event. In an ideal scenario in which the sensing is perfect, even a single sensor reading should generate a high level of confidence. However, in practice, the sensor boards are sometimes inaccurate. This could result in an event being reported when it is not actually present. Hence, a single sensor reading may not be very reliable. One way to improve the reliability of event detection is to increase the redundancy, by either waiting for multiple reports from the same sensor mote (temporal redundancy), or by waiting for reports from multiple neighboring sensor motes (spatial redundancy). We chose to experiment with the latter option because we assumed that the faults in the sensor boards are independently distributed. Therefore, the probability that multiple neighboring sensor motes are simultaneously in error is lower than the probability that a single sensor mote is in error. From Figure 12, we see that our assumption is validated. The figure shows that if the leader waits until at least 3 different sensor motes have detected the event, before reporting the event to the base station, the number of false positives drops to 0. However, if the sensing range and the density of deployment is not sufficiently high, it is harder to achieve a higher degree of aggregation. This results either in more false negatives, as shown in Figure 12, or in higher reporting latency as shown in the next section.

7.4.3 *Impact of Aggregation on Tracking Latency.* Figure 13 shows how the reporting latency increases with the degree of aggregation for a car moving at 5 mph through a sensor field where the motes are deployed 7-8 ft apart. We define the reporting latency as the time elapsed from the instant at which the car enters the sensor field until the instant at which the base station receives the first *genuine*

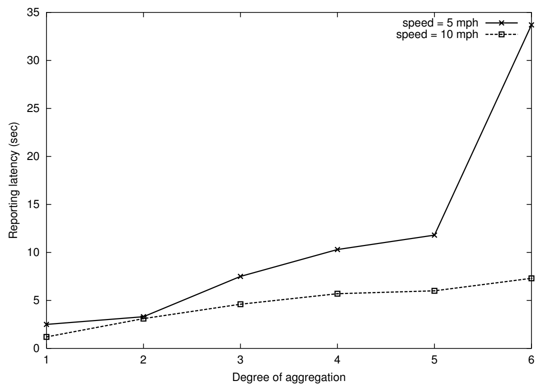


Fig. 13. Impact of DOA on Reporting Latency

report about the location of the car. In addition to the density, the increase in the latency and false negatives depends on the sleep cycle of the sensor motes and the speed of the moving vehicle. To our surprise, we found that we were able to reduce the latency and false negatives for higher degree of aggregation (DOA ≥ 4), by increasing the speed of the vehicle from about 5 mph to about 10 mph (Figure 13). However, increasing the speed beyond that value resulted in more false negatives. The reason is that when motes are some distance apart, a higher speed allows the vehicle to be in the sensing range of more motes during a period of time t_r. Hence, the vehicle can be detected even at a higher degree of aggregation. However, the sensors have a non-negligible reaction time, which further increases if the motes are sleeping. Hence, if the speed is increased beyond a certain threshold, the vehicle may move past the sensing range of the motes before they have a chance to react. That could result in more false negatives.

We must emphasize that the performance numbers we have presented above exhibit some degree of variance across different experimental runs and in different environments. Therefore, instead of using the above experimental results to deduce absolute performance numbers, we use them to draw some general conclusions about choosing the degree of in-network aggregation. First, a higher DOA certainly helps reduce the message overhead and the number of false positives. However, if the density with which the motes are deployed is not sufficiently high, a higher degree of aggregation may adversely affect the tracking performance. This effect is more pronounced in the case of slow-moving events. Even if the motes are densely packed and the events are fast-moving, it is harder to achieve a high degree of aggregation, if the motes sleep for a long duration and their sleep-wakeup cycles are not in lock-step. Thus, we see that the degree of aggregation represents a tradeoff between different parameters. The recommendation we follow based on our results is to choose a value of DOA that is large enough to maintain the probability of false negatives within a certain threshold. Our experiments show that a value of 2 or 3 for the degree of in-network aggregation is reasonable for MICA2 platform. If this value is not large enough to maintain the false positives within the desired threshold, then we recommend using a second tier of false alarm processing at the base station.

The above discussion motivates us to develop an analytical model in the future that captures the tradeoff between the key parameters, such as the degree of aggregation, density of node deployment, sleep duration, and the maximum probability of false alarms that a user can tolerate. Such a model can then be used to choose the appropriate degree of aggregation, when the values of the other parameters are known. Such a model is also valuable in estimating the probability of false alarms that a user can expect for a specific design and configuration.

7.5 Evaluation of Velocity Estimation

To measure the velocity of the targets, we place 70 motes in two lines with 35 motes in each line. We drive the car in the middle of the road. Actual velocities are obtained from speedometer of the car. Table III presents the experimental results we obtained. we found out that our system has about 5 $\sim$ 10% error in speed estimation and a detection delay under the sentry service below 3 seconds.

Table III. Velocity Estimation

DETECTION DELAY (S)	REPORTED VELOCITY (MPH)	ACTUAL VELOCITY (MPH)
1.7	11.1	10 ± 1
2.6	18.5	20 ± 1
1.9	23.0	20 ± 1
2.6	12.7	12 ± 1
0.9	22.1	20 ± 1

7.6 Evaluation of Sentry Service

In this section, we analyze the key features of the sentry service component. We first analyze power buget of the system, point out the importance of the sentry service, then we discuss about the stealthiness of the power management scheme, and then assess the extension in lifetime achieved for different sentry distributions and for different periods of the sleep-wakeup cycle of the non-sentries.

7.6.1 *Power Budget for Surveillance System .* One of misconceptions about sensor networks is that communication consumes most energy. It is true that transient power draw in the radio module is larger than that of microcontroller and sensing modules, however, in many applications, communication is intermittent (e.g., once per 10 minutes). As a result, average power draw in communication over time is very small. As shown in Figure 14, the predominant power draw lies in the surveillance operation. This indicates a fact: the most effective method to save energy is turning off as many redundant nodes as possible. This warrants our design of a sentry service.

7.6.2 *Stealthiness of Power Management Component.* In Section 5.5, we compared and contrasted the proactive and reactive schemes for controlling the sleep-wakeup cycle of the non-sentry motes when power management is enabled. The

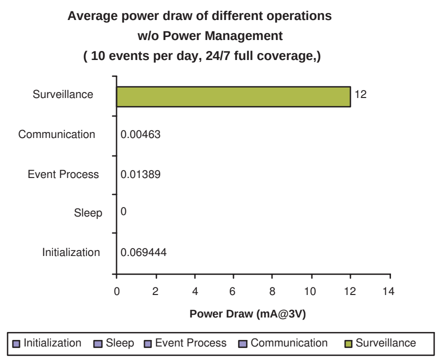


Fig. 14. Power Draw of Different Operations

proactive scheme provides better responsiveness when an event occurs, at the cost of transmitting more messages in the absence of an event. In contrast, the reactive scheme provides better stealthiness during the idle periods, at the cost of retransmitting multiple messages in order to awaken the non-sentries when an event occurs. A sentry chooses the interval between successive retransmissions in such a way that the beacon transmission coincides with the wakeup period of the neighboring non-sentry motes. We use the following equation to control the number of retransmissions of the awake beacon (n_r).

$$n_r = \frac{sleepDuration + awakeDuration}{awakeDuration + 1} \tag{2}$$

A larger value of *awakeDuration* results in fewer retransmissions of the awake beacon when a sentry detects an event. However, if the motes are awake longer, more energy is consumed and therefore, the lifetime of the sensor network reduces.

In order to compare the message overhead between the reactive and proactive schemes, we implemented both the schemes and conducted experiments using the TOSSIM simulator[Levis et al. 2003], a simulator that actually runs our system and TinyOS code. We simulated a simple scenario in which a tank moved across a sensor field in which 10 motes capable of magnetic sensing were deployed. The duration of each simulation run was 600 seconds. The *awakeDuration* of the motes was fixed at 2 seconds for each run. Figure 15 compares the number of messages sent out by the proactive and reactive schemes during the tracking phase when power management is enabled.

Figure 15 shows that the number of power management messages in the reactive scheme increases from 2 to 11 as the sleep duration increases from 2 seconds to 20 seconds. This is justified by Equation 2, which indicates that a longer sleep duration requires more retransmissions of the awake beacon, in order to ensure that one of the beacons is received by the non-sentry motes. In contrast, the message overhead in the case of the proactive scheme reduces as the sleep duration increases. This is because the periodicity with which a sentry sends out the sleep beacon is equal to *sleepDuration* + *awakeDuration*. As the sleep duration increases, the sleep beacons are sent out less frequently, thereby reducing the message overhead.

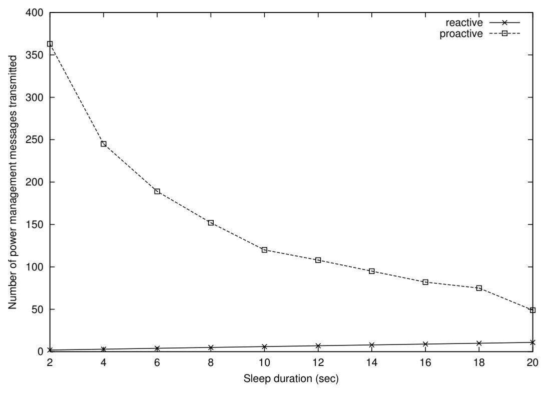


Fig. 15. Power Management Message Overhead

The results in Figure 15 also show that the message overhead due to power management is significantly lower in the reactive scheme compared to its proactive counterpart. This suggests that the reactive scheme is more stealthy compared to the proactive scheme. While this is true for the 2 second awake period we have chosen, it may not be true for smaller values of *awakeDuration*. In our experiment, we chose a relatively high value of 2 seconds for *awakeDuration*, in order to compensate for the high rate of drift in the software timers in the current TinyOS implementation. If the timer drift is smaller in future implementations of TinyOS, we would choose a smaller awake duration for the motes, so that the overall energy consumption of the network can be reduced. However, a smaller value of *awakeDuration* would increase the message overhead for the reactive scheme. We have currently adopted the reactive scheme for our surveillance application, because it provides better stealthiness for the duration of the sleep-wakeup cycle we have chosen. However, an investigation into a hybrid scheme that combines the advantages of both the proactive and reactive schemes would be worthwhile to pursue as future work. In addition, the hardware solution mentioned in [Gu and Stankovic 2004] might also be an alternative strategy for aggressive energy conservation.

7.6.3 *Power Savings.* One of the main goals of the sentry service module is the extension of the lifetime of the sensor network. The sentry service extends the lifetime by conserving the energy consumption of the motes when the network is idle. Non-sentry motes alternate between sleep and wakeup states, and in Section 7.6.2, we justified our choice of a timer-driven, reactive approach to control the sleep-wakeup cycle. When a mote is in the sleep state, its radio is turned off, all of its I/O ports are configured appropriately to minimize the current consumption, the ADC module is turned off to disable any sampling, and the controller is placed in a power-save state. When the sleep timer expires, the controller is awakened by a timer interrupt, and all of the modules resume activity. The extent to which our power management approach increases the lifetime of a mote depends on the fraction of time the mote spends in the sleep state. We now use the current consumed in the sleep and wakeup states using the above power management scheme to predict how the expected lifetime of a sensor network varies with the fraction of

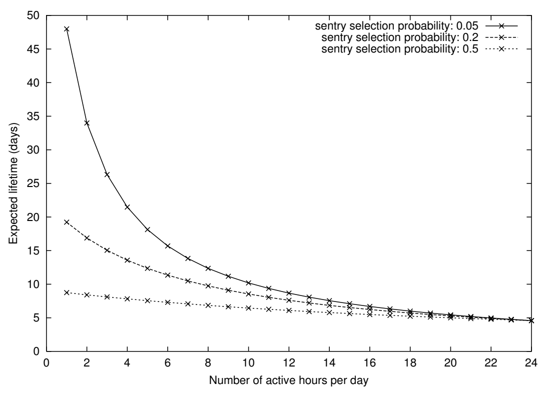


Fig. 16. Expected Lifetime of a Sensor Network Using Sentry-based Power Management

sentries selected.

A MICA2 mote is powered by a pair of AA batteries, supplying a combined voltage of 3V. Assuming that a pair of batteries will supply 2200 mAh at 3V [Mainwaring et al. 2002], we can estimate the lifetime of a mote, if we know the current consumed in the sleep and wakeup states and the duty cycle of the mote. The duty cycle of a mote is the number of hours per day it remains awake polling for events. Based on our measurements, we found that a MICA2 mote equipped with a magnetic sensor board and running our sentry-based power management software consumes 20 mA in the wakeup state. The wakeup current includes the current consumed by the magnetometer to sample at a rate of 10 samples per second. On the other hand, we measured the sleep current of the mote to vary between 50 μA to 130 μA, which results in a 99% reduction in the current consumption. We use a sleep current of 130 μA for the discussion in this section.

From the above data, we can determine the lifetime of a sensor network that uses our sentry-based power management scheme. The lifetime of a sensor network depends on the fraction of sentries selected and the fraction of time the non-sentry motes remain awake. Let $P(s)$ denote the probability that a mote is selected as a sentry, and $P(a)$ denote the probability that a non-sentry mote is awake. The total current (C) consumed by a mote in the baseline case, when there are no events in the network, is given by Equation 3. The lifetime of the motes, L, is the ratio of the battery capacity to the total current consumed. Assuming a battery capacity of 2200 mAh, the lifetime of the motes in hours is simply $2200/C$.

$$C = P(s) * 20 + (1 - P(s)) * (P(a) * 20 + (1 - P(a)) * 0.13) \tag{3}$$

Figure 16 uses the above equation to predict the expected lifetime of the motes for different percentages of their duty cycle. The actual values of $P(s)$ and $P(a)$ are measured from the our prototype system. A mote that is always asleep is expected to survive for 2 years, whereas a mote that is always awake (i.e. always remains a sentry), can survive only up to 5 days. The exponential curves show that the lifetime greatly improves when the duty cycle is low. For example, when the probability that a mote is selected as a sentry is 0.5, and its duty cycle is reduced from 24 hours per day to one hour per day, its lifetime extends by nearly 100%. The

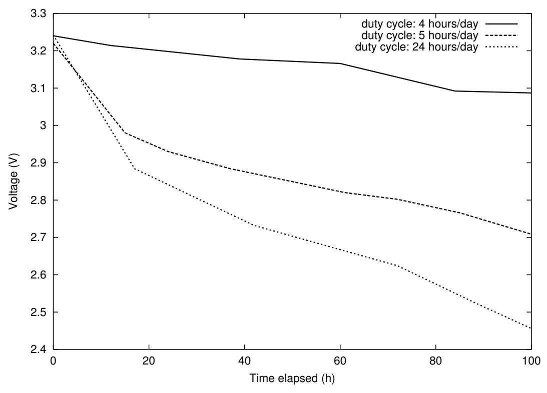


Fig. 17. Impact of Sleep Duration on Power Consumption

graphs also show that the lifetime improves significantly as the number of sentries is reduced. For example, when the probability that a mote is selected as a sentry is reduced to 0.05, and its duty cycle is reduced to 4%, its lifetime extends by nearly 900%. The probability of selecting a mote as a sentry involves a tradeoff between the sensing coverage that can be achieved and the required network lifetime. A higher probability results in more sentries and provides better sensing coverage. However, it also reduces the lifetime of the network, as Figure 16 shows. In order to reduce the number of sentries without adversely affecting the sensing coverage, we can either choose magnetometers with a higher sensing range or increase the density with which the motes are deployed. For example, in our experiments we found that when the motes were placed at a distance of 8 ft from each other, the probability that a mote was selected as a sentry was nearly about 40%. However, in a more dense deployment in which the motes were placed within a few inches from each other, the probability of selecting a mote as a sentry dropped to about 20%. The reason is that a dense deployment results in a larger number of neighbors for each mote. Therefore, a single sentry is able to cover more neighbors, and that gives fewer motes a chance to elect themselves as a sentry.

In addition to predicting the lifetime of the network using a simple model, we also conducted experiments to compare the rate at which energy is dissipated for different duty cycles in an actual deployment. In each of our experiments we deployed 6 motes, all equipped with magnetic sensor boards, inside an office building. Sentry rotation occurred once every 4 hours. Since there is no direct way to measure the energy consumed by the motes, we used the voltage drop across the batteries supplying power to the motes as an indirect way to measure the energy dissipation. We measured the voltage for each mote at regular intervals over a period of 100 hours and found that the voltage drop was reasonably uniform across the motes. Figure 17 shows the voltage drop during the observation period for one of the 6 motes for different values of duty cycles. From the figure, we see that the battery voltage for a mote does not drop uniformly with time. One of the reasons for the non-uniform energy dissipation is the periodic rotation of the sentry responsibility. The voltage drop of a mote is higher during an interval in which it is serving as a sentry than when it is serving as a non-sentry because the periodic sampling operation performed by a sentry consumes significant energy. The results also confirm that a higher duty cycle results in a higher energy dissipation. We see that when the mote is always awake, it loses most of its capacity within 100 hours (about 4 days). This reasonably matches with the results in Figure 16, which predicted that a mote operating 100% of the time will last only 5 days.

The experimental results we obtained are promising in that they show that the sentry-based power management algorithm is adaptive and that it is successful in extending the lifetime of the sensor network. While our current sentry selection algorithm does not choose the minimal number of sentries, by knowing the lifetime of the mission in advance, we can choose the density of deployment and the duty cycle in such a way that the lifetime requirement can be met.

8. LESSONS LEARNED

The work described in this paper is our experience in building a complete system for using wireless sensor networks for a practical application and evaluating it through an actual deployment of motes. This practical experience has been valuable, because it has taught us that some of the simplified assumptions made about the hardware platform and operating system in much current research do not hold well in practice. The lessons we learned have greatly impacted some of the design choices we had to make in building our system.

(1) **Application-specific Reliability :** We found that the packet loss in the MICA2 platform can be as large as 20%. A well-known approach to counter message loss is to retransmit the message multiple times, in order to improve the probability of delivery. Such retransmissions can be initiated either in the lower layers of the protocol stack or at the application layer. Since retransmitting a message consumes significant energy, it is important that the messages are retransmitted selectively, based on application-specific knowledge. For instance, applications that transmit ephemeral sensor readings, such as the instantaneous temperature, may not require reliability. Lower layers, such as the MAC layer, often lack domain-specific knowledge. So implementing reliability guarantees in the lower layers makes it harder to provide application-specific reliability. Hence, for a system that strives to achieve energy efficiency, providing reliability guarantees at the application layer is a better option.

(2) **False Alarm Reduction:** We found that our sensors generated false alarms at a non-negligible rate. This introduces unnecessary energy consumption and inappropriate actions. False alarms we experienced can be categorized into two major types: Transient and persistent false alarms. A simple exponential weighted moving average (EWMA) on the mote is sufficient to deal with transient false alarms such as the burst distortion of sensing readings. However, if the false alarms are persistent due to errors in the sensor device, more advance techniques are desired. In VigilNet, we successfully eliminated individual persistent false alarms by utilizing in-network aggregation with a relatively high DOA value. In the worst case, when multiple persistent false alarms are generated simultaneously, we are able to filter out such false alarms by analyzing spatial-temporal correlations among the consecutive reports at the base sta-

tion. In addition, we implement a faulty node detection algorithm to shutdown misbehavior nodes automatically.

(3) **Race Conditions Reduction:** Race conditions are another example of a phenomenon that is often ignored in simulation-based approaches, but must be addressed when building the running system. For example, contention occurs not only when different motes try to transmit simultaneously, but also when different software components on the same mote initiate transmissions simultaneously through split-phase operations. Due to the limited support from TinyOS, the latter can lead to race conditions. Race conditions can be avoided, if the OS can support synchronized processing, based on semaphores, in order to coordinate the shared resources among the contending modules. While TinyOS supports concurrency control through atomic sections and tasks, it is more flexible and efficient to use application level synchronization such as packet scheduling mentioned in Section 6.1 to coordinate the operations.

(4) **Asymmetry Reduction:** Another issue we had to address was to account for the effect of asymmetric channels which is largely ignored in simulation approaches. Communication in low power devices, such as the motes, is largely asymmetric [Zhou et al. 2004] due to differences in hardware, signal attenuation, and residual battery capacity. In practice, we were able to reduce the effect of asymmetric channels by symmetry detection technique mentioned in Section 5.1.2.

(5) **Software Calibration:** In a simulation-based approach, it is common for sensor devices of the same type to generate the same readings under identical conditions. However, in practice, the same type of sensors are capable of generating quite different sensor readings under identical conditions. Such a phenomenon may occur because of differences in the way the devices are manufactured, and it is often hard to accurately capture those differences in a simulator. We found that the impact of such heterogeneity is significant in the MICA2 platform, such as shown in Figure 8. The variance in the sensor readings can be accounted for at the very outset through software calibration of the sensors. And continuous calibration is also needed to adapt to the changing environment over time.

(6) **Other Lessons:** The drift in the software timers in TinyOS presents another practical issue, especially when motes transit into sleep state. In order to compensate for the drift in the soft timers, we need to increase the duration for which a mote remains awake, and design appropriate strategies to control the sleep-wakeup cycle, as described in Section 7.6.2. Another practical challenge we faced was the lack of appropriate tools for debugging a network of motes. We utilize the dynamic configuration method mentioned in 5.1.3 and overhearing tools to facilitate our work. However, more sophisticated debugging and configuration tools will greatly ease the burden on the programmer in the future. We acknowledge that our design choices sometimes are restricted by limited hardware and operation system support. It is desirable to have new features such as interruptible snoozing, alti-alias filter for sensing, a more reliable RF module and process management, so that we can improve our design and implementation in the future.

9. RELATED WORK

Energy efficiency has drawn a lot of attention at various aspects of sensor network research. At hardware level, sensor nodes [CrossBow] provide multiple sleep modes to allow users to tailor the power consumption to the application requirements. It is now possible to do fine-grained control over individual modules. They can be turned on/off on demand with little overhead and a low switch time. MAC layer protocols take advantage of overhearing to allow nodes to sleep while they are not transmitting or receiving messages [Guo et al. 2001; Heinzelman et al. 2000a], or to reduce receiver-side power consumption by sending a long preamble packet [Polastre and Culler 2004]. At the network layer, methods are proposed to balance power through the distribution of messages among various paths from source to destination, such as [He et al. 2003], or to use efficient cache schemes to balance the energy cost between data query and dissemination [Bhattacharya et al. 2003]. Data aggregation techniques are used in [He et al. 2004; Krishnamachari et al. 2002] to reduce energy consumption by aggregating multiple reports about the same event. Topology control maintains the network connectivity, while allowing some of nodes go to sleep [Xu et al. 2001]. Some protocols form static groups and rotate leadership responsibilities allowing non-leader nodes to sleep and conserve their energy [Chen et al. 2001]. Sensing coverage protocols such [Yan et al. 2003; Tian and Georganas 2003; Ye et al. 2003] achieve energy saving through different node duty cycle scheduling algorithms.

Target tracking is another research area closed related to our work. Zhang et.al. [Zhang and Cao 2004] propose a tree-based algorithm to facilitate collaborative tracking of moving targets. Patterm et. al.[Pattem et al. 2003] investigate the tradeoff between energy and tracking quality by selectively activating sensor nodes along predicated path. Aslam [Aslam and et. al. 2003] propose a particle filtering style tracking algorithm using binary sensors which can detect whether an object is approaching or not. All these research provide nice properties on improving the tracking performance in one aspect or another, however these approaches mainly focus on simulation without real implementation. Brook et. al. [Brooks et al. 2002] implement a distributed tracking system based on extended Kalman filter techniques. Based on a novel information-driven approach, Feng et. al. [Zhao et al. 2002; Liu et al. 2003] build a tracking system with distributed Bayesian estimation, given previous estimation (belief) and new sensor inputs.

The difference of our work from aforementioned approaches is that instead of designing individual protocols, we are aiming at building a depolyable surveillance system which incorporates a whole set of middleware services. This requires us to choose the right combination of sensor network techniques, reconcile the conflicting design goals among different protocols, and propose new techniques that are compatible with current solutions in the context of target surveillance and tracking.

10. CONCLUSIONS

Research in wireless sensor networks has been very active. Most of the published work studies an individual protocol and performs evaluations via simulations. In contrast, in VigilNet, we implement an entire integrated suite of protocols and application modules and evaluate the performance extensively on a system composed of 70 MICA2 motes in a realistic outdoor setting. Empirical results identify the capability of the MICA2 radio, localization and routing performance, the value of in-network aggregation, false alarm processing and application layer tracking latency, and the value of power management. Design decisions and how those decisions were influenced by the empirical data were described. Key lessons learned were also itemized. From our experience in building and analyzing this system it is clear that key realistic hardware, software and environmental issues must not be ignored in developing usable solutions. This includes realism of sensor performance, asymmetries in communication, false alarms, and race conditions.

11. FUTURE WORK

System design and engineering are two of the keys to bring sensor network paradigm into reality. The system described in this paper is still an ongoing project. Many outstanding design issues are yet to be resolved. We are currently investigating 1) target classification under constraint resources through collaborative data fusion, 2) the possibility to design a more aggressive power management strategy with passive wake-up capabilities [Gu and Stankovic 2004], 3) approaches to build extremely robust routing infrastructure, which can survive under hostile environments and 4) a scalable architecture up to thousands of nodes while maintaining operational performance requirements.

REFERENCES

Aslam, J. and et. al. 2003. Tracking a Moving Object with a Binary Sensor Network. In *First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*.

Bhattacharya, S., Kim, H., Prabh, S., and Abdelzaher, T. 2003. Energy-Conserving Data Placement and Asynchronous Multicast in Wireless Sensor Networks. In *The First International Conference on Mobile Systems, Applications, and Services (MobiSys)*.

Blum, B. M., Nagaraddi, P., Wood, A., Abdelzaher, T. F., Son, S., and Stankovic, J. A. 2003. An Entity Maintenance and Connection Service for Sensor Networks. In *The First Intl. Conference on Mobile Systems, Applications, and Services (MobiSys)*.

Brooks, R. R., Ramanathan, P., and Sayeed, A. 2002. Distributed target tracking and classsification in sensor networks. *Proceedings of the IEEE*.

Chen, B., Jamieson, K., Balakrishnan, H., and Morris, R. 2001. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *6th ACM MOBICOM Conference*.

CrossBow. *Mica2 data sheet*. CrossBow. Available at `http://www.xbow.com/Products/Product_pdf_files/MICA%20data%20sheet.pdf`.

Elson, J. and Romer, K. 2002. Wireless Sensor Networks: A New Regime for Time Synchronization. In *Proc. of the Workshop on Hot Topics in Networks (HotNets)*.

Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E., and Culler, D. 2000. The nesC Language: A Holistic Approach to Networked Embedded Systems. In *Proceedings of Programming Language Design and Implementation (PLDI) 2003*.

Gu, L. and Stankovic, J. A. 2004. Radio-Triggered Wake-Up Capability for Sensor Networks. In *Proceedings of RTAS*.

Guo, C., Zhong, L. C., and Rabaey, J. M. 2001. Low power distributed mac for ad hoc sensor radio networks. In *IEEE GlobeCom*.

He, T., Blum, B. M., Stankovic, J. A., and Abdelzaher, T. F. 2004. AIDA: Adaptive Application Independent Data Aggregation in Wireless Sensor Networks. *ACM Transactions on Embedded Computing System, Special issue on Dynamically Adaptable Embedded Systems*.

He, T., Huang, C., Blum, B. M., Stankovic, J. A., and Abdelzaher, T. 2003. Range-Free Localization Schemes in Large-Scale Sensor Networks. In *Proc. of the Intl. Conference on Mobile Computing and Networking (MOBICOM)*.

He, T., Stankovic, J., Lu, C., and Abdelzaher, T. 2003. SPEED: A Stateless Protocol for Real-Time Communication in Ad Hoc Sensor Networks. In *Proc. of International Conference on Distributed Computing Systems (ICDCS)*.

Heinzelman, W., Chandrakasan, A., and Balakrishnan, H. 2000a. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS*.

Heinzelman, W. R., Chandrakasan, A., and Balakrishnan, H. 2000b. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proc. of the Intl. Conference on System Sciences*.

Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D. E., and Pister, K. S. J. 2000. System Architecture Directions for Networked Sensors. In *Proc. of Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 93–104.

Honeywell. *1- and 2-Axis Magnetic Sensors*. Honeywell. Available at `www.ssec.honeywell.com/magnetic/datasheets/hmc1001-2_1021-2.pdf`.

Horton, M., Culler, D. E., Pister, K., Hill, J., Szewczyk, R., and Woo, A. 2002. MICA: The Commercialization of Microsensor Motes. *Sensors Online*. www.sensorsmag.com/articles/0402/40.

Intanagonwiwat, C., Govindan, R., and Estrin, D. 2000. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *the Sixth Annual International Conference on Mobile Computing and Networks*.

Kahn, J. M., Katz, R. H., and Pister, K. S. J. 1999. Next Century Challenges: Mobile Networking for Smart Dust. In *Proc. of Intl. Conference on Mobile Computing and Networking (MOBICOM)*.

Krishnamachari, B., Estrin, D., and Wicker, S. 2002. Impact of Data Aggregation in Wireless Sensor Networks. In *Proc. of Intl. Workshop on Distributed Event-Based Systems*.

Levis, P., Lee, N., Welsh, M., and Culler, D. 2003. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*.

Liu, J., Reich, J., and Zhao, F. 2003. Collaborative In-Network Processing for Target Tracking. *J. on Applied Signal Processing*.

Madden, S., Franklin, M., Hellerstein, J., and Hong, W. 2002. TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks. In *Operating Systems Design and Implementation*.

Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D. E., and Anderson, J. 2002. Wireless Sensor Networks for Habitat Monitoring. In *Proc. of the ACM Workshop on Sensor Networks and Application (WSNA)*.

Pattem, S., Poduri, S., and Krishnamachari, B. 2003. Energy-quality tradeoffs for target tracking in wireless sensor networks. In *The 2nd International Workshop on Information Processing in Sensor Networks (IPSN'03)*. Palo Alto, California, 32–46.

Polastre, J. and Culler, D. 2004. Versatile Low Power Media Access for Wireless Sensor Networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*.

Powers, R. 1995. Batteries for Low Power Electronics. *In Proceedings of the IEEE*, 687–693.

Stoleru, R., He, T., and Stankovic, J. A. 2004. Walking GPS: A Practical Solution for Localization in Manually Deployed Wireless Sensor Networks. In *1st IEEE Workshop on Embedded Networked Sensors EmNetS-I*.

Tian, D. and Georganas, N. 2003. A Node Scheduling Scheme for Energy Conservation in Large Wireless Sensor Networks. *Wireless Communications and Mobile Computing Journal*.

Wang, X., Xing, G., Zhang, Y., Lu, C., Pless, R., and Gill, C. 2003. Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks. In *First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*.

Xu, Y., Heidemann, J., and Estrin, D. 2001. Geography-informed energy conservation for ad hoc routing. In *MobiCom*.

Yan, T., He, T., and Stankovic, J. 2003. Differentiated Surveillance Service for Sensor Networks. In *First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*.

Ye, F., Zhong, G., Lu, S., and Zhang, L. 2003. PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks. In *Proc. of International Conference on Distributed Computing Systems (ICDCS)*.

Zhang, W. and Cao, G. 2004. Optimizing tree reconfiguration for mobile target tracking in sensor networks. In *IEEE INFOCOM*.

Zhao, F., Shin, J., and Reich, J. 2002. Information-Driven Dynamic Sensor Collaboration for Tracking Applications. *IEEE Signal Processing Magazine*.

Zhou, G., He, T., and Stankovic, J. A. 2004. Impact of Radio Irregularity on Wireless Sensor Networks. In *The Second International Conference on Mobile Systems, Applications, and Services (MobiSys)*.

AIDA: Adaptive Application Independent Data Aggregation in Wireless Sensor Networks[1]

Tian He Brian M. Blum John A Stankovic Tarek Abdelzaher
Department of Computer Science
University of Virginia
{Tianhe,Bmb5v,Stankovic,Zaher@cs.virginia.edu

Abstract

Sensor networks, a novel paradigm in distributed wireless communication technology, have been proposed for use in various applications including military surveillance and environmental monitoring. These systems could deploy heterogeneous collections of sensors capable of observing and reporting on various dynamic properties of their surroundings in a time sensitive manner. Such systems suffer bandwidth, energy, and throughput constraints that limit the quantity of information transferred from end to end. These factors coupled with unpredictable traffic patterns and dynamic network topologies make the task of designing optimal protocols for such networks difficult. Mechanisms to perform data centric aggregation utilizing application specific knowledge provide a means to augmenting throughput, but have limitations due to their lack of adaptation and reliance on application specific decisions. We therefore propose a novel aggregation scheme that adaptively performs application independent data aggregation in a time sensitive manner. Our work isolates aggregation decisions into a module that resides between the network and the data link layer and does not require any modifications to the currently existing MAC and network layer protocols. We take advantage of queuing delay and the broadcast nature of wireless communication to concatenate network units into an aggregate using a novel adaptive feedback scheme to schedule the delivery of this aggregate to the MAC layer for transmission. In our evaluation we show that end-to-end transmission delay is reduced by as much as 80% under heavy traffic loads. Additionally, we show as much as a 50% reduction in transmission energy consumption with an overall negative header overhead. Theoretical analysis, simulation, and a test-bed implementation on Berkeley's MICA motes are provided to validate our claims.

1. Introduction

Wireless Sensor Networks have emerged as a new information-gathering paradigm based on the collaborative effort of a large number of sensing nodes. In such networks, nodes deployed in a remote environment must self-configure without any *a priori* information about the network topology or global view. Nodes will act in response to environmental events and relay collected and possibly aggregated information through the formed multi-hop wireless network in accordance with desired system functionality. The inherently dynamic and distributed behavior of these networks, coupled with inherent physical limitations such as small instruction and data memory, constrained energy resources, short communication radii, and a low bandwidth medium in which to communicate, make developing communication protocols difficult.

Research on hardware for such devices has taken place at Berkeley [14][32][34] and various other research institutions [26] throughout the world. Using such hardware as a basis for development, the software architecture and communication stack residing on these devices are built taking into consideration the prolific research in the areas of ad-hoc networking [10][15][17][20], data aggregation [16][21][28], cluster formation [27], distributed services [22], group formation [6], channel contention [3][5][7][19], and power conservation [4][12]. Work targeted to these devices include research in query processing (e.g. TinyDB [25]), and aggregation (e.g. TAG [24]). Work on the utility of such innovative technologies has unearthed potential applications including, event tracking [1], environmental monitoring, disaster relief, and search and rescue.

In this work, we address the problems of low bandwidth and energy limitations inherent to sensor devices. These networks' ever-changing and unpredictable state demands a self-configuring, adaptive solution. We

[1] This work was supported in part by NSF grant CCR-0098269, the MURI award N00014-01-1-0576 from ONR, and the DAPRPA IXO offices under the NEST project (grant number F336615-01-C-1905).

develop a novel adaptive application independent data aggregation (AIDA) component that fits seamlessly into current sensor network communication stack. Our goal is to maximize utilization of the communication channel (single frequency) with energy savings coming as an ancillary benefit. With significant costs incurred from channel contention, packet header overhead, and data padding for fixed sized packets, this work abates such costs by employing varying degrees of data aggregation at forwarding nodes in accordance with current local traffic patterns.

Data aggregation techniques have been extensively investigated in recent literature. Our work, as a novel data aggregation approach, distinguishes itself from current state of the art solutions in three respects. First, prior Application Dependent Data Aggregation (ADDA shown in Figure 1b) relies on application layer information and must have a bi-directional interface, and therefore dependence with, the data centric routing protocol implemented. AIDA isolates aggregation decisions from application specifics by performing adaptive aggregation in an intermediate layer that resides between the traditional data-link and network layer protocols (Figure 1.a). This component is generalized enough to be utilized over a wide range of applications (data types) without incurring the costs of rewriting components to support application-specific logics. Second, no prior work in data aggregation adapts itself to the traffic situation in a time sensitive manner. AIDA takes the timely delivery of messages as well as protocol overhead into account to adaptively adjust aggregation strategies in accordance with assessed traffic conditions and expected sensor network requirements. Simulation results show that AIDA can adapt to varying traffic situations and dramatically reduce network congestion and transmission energy consumption**.** Third, previous data aggregation schemes (e.g., data centric routing [16]) perform in-network processing to reduce the amount of application data transmitted. These in-network processes (e.g. averaging) can achieve higher degrees of aggregation; however data are less available to the application (e.g. standard deviation of the data set can not obtained from the average). In contrast, AIDA performs lossless aggregation allowing the upper layer to decide whether information compression is appropriate at the time. *Very important*, our design enables AIDA to remain complementary to other data aggregation strategies (Figure 1.c) while providing significant timesaving benefits in the lower layers of the communication stack.

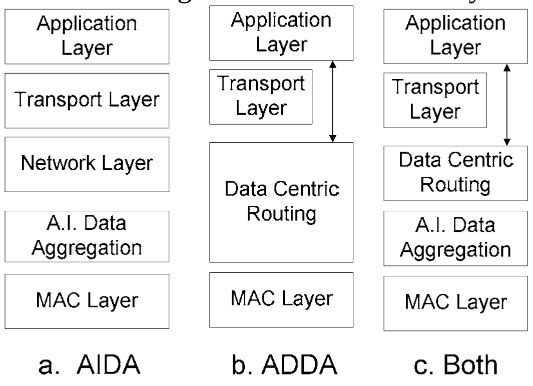


Figure 1: Architectural Designs

This paper attempts to address the aforementioned problems through a novel adaptive time sensitive data aggregation component. As an introduction to sensor networks, and to provide a more in depth discussion of the type of research taking place within this field, we begin section 2 with a discussion of related and ongoing work. Section 3 addresses the need for adaptation, data aggregation, and real-time data delivery. Section 4 then presents specific details about our protocol. Sections 5 and 6 describe our simulation environment, the type of experiments run, and a discussion of the results we obtain in both simulation and in the Berkeley MICA test-bed. Finally, we conclude in Section 7.

2. Leveraging Previous Work

Efforts to maximize channel utilization have been spread across various layers of the sensor network communication stack. Starting at the MAC layer, these include attempts to minimize collisions through contention-based mechanisms designed for a lossy wireless medium. Such work includes 802.11 [3], MACA [19], MACAW [5], FAMA[7], S-MAC[31], and Multi-Hop Scheduling [18], to name a few. All of these solutions reside within the data-link layer of the communication stack and, therefore, can coexist with the higher layer aggregation component we provide.

Similar to the data link layer the network layer, and more specifically the routing component, has brought about significant efforts to avoid congestion and maximize use of the communication medium. Such schemes include distributing the traffic load to route around congestion [10] and using a minimal hop path to reduce the total number of transmissions [29]. Beyond the routing layer the communication stack in sensor networks becomes more amorphous. Clustering [27], group formation [6], and other higher layer hierarchical components serve to combine node responsibilities and come to consensus on what data to send. Often such information is application specific and must rely on a general understanding of exactly what the network is tasked to do. Additionally, the hierarchical and grouping components often utilize various forms of data aggregation through consensus algorithms or other forms of local processing.

Basic schemes [16] for the aggregation of data include the Center at the Nearest Source (CNS), where data is aggregated at the source nearest to the destination; Shortest Path Trees (SPT), where data is sent along the shortest path from source to sink and aggregated at common intermediate hops along the way; and Greedy Incremental Trees (GIT), which builds an aggregation tree sequentially to merge paths and provide more aggregation opportunities.

Expressing queries [25] and utilizing those queries for data aggregation [24] present opportunities for in network data aggregation. An extremely popular data aggregation scheme for sensor networks, Directed Diffusion [15][11], is a data-centric architecture where named (application specific) data gets propagated along paths back to the requestor. Effective paths are reinforced as they are used to optimize communication from point to point. Specifically designed for sensor networks, Directed Diffusion aggregates data along these reinforced paths to reduce the quantity of data transmitted across the network. Similarly Data Placement [28] is designed for applications where multiple sinks coexist and use in-network caching to update and distribute data to leaf nodes at the minimally requested rate. LEACH [12] is a high layer protocol that provides clustering and local processing to aggregate sensor data and reduce global communication. Many other data aggregation schemes exist that also provide network, transport, and application level mechanisms taking advantage of application specific knowledge about the data in question. All of these schemes reside either at or above the network layer and are orthogonal and can coexist with our work.

Aggregation scheme comparison studies have demonstrated the effect of network parameters and the utility of aggregation mechanisms in a wide variety of applications [16][21]. These studies discuss potential savings that aggregation can provide and are noted to explicate the potential for such work to improve network throughput.

To date, very few sensor network papers have addressed the need for incorporating adaptive behavior into their protocols. Sensor networks exhibit complex distributed behavior rendering static pre-configuration utterly useless as network traffic, often initiated by environmental events of interest, transitions from one extreme to another. Several protocols have taken a first stab at addressing the need for adaptive behavior in such dynamic networks. RAP [23] and SPEED [10] utilize locally available information to adjust priority levels or make more informed routing decisions in response to network congestion and changing traffic patterns. SPIN [13] makes adaptive decisions to participate in data dissemination based on current energy levels and the cost of communication. In [32], A. Woo uses adaptive rate control at the data-link layer to fine tune contention parameters in response to local traffic conditions. GAF [30] monitors network connectivity and turns nodes on/off to adapt network density for energy-conservation. While many more examples of online adaptation exist, these solutions provide relevant examples of how adaptation is beneficial in dynamic and unpredictable sensor networks and serve as a starting point to introduce adaptive behavior into these complex systems.

In addition to maximizing channel utilization and adapting to dynamic network conditions, energy conservation has become a central focus in sensor network research. Similar to data aggregation, work in energy conservation for sensor networks has been considered at various levels of the communication stack. Aside from minimizing power consumption at the hardware level [26], MAC layer protocols developed for energy savings mostly take advantage of overhearing and scheduling to allow nodes to sleep while they are not transmitting or receiving messages [8][12][29]. At the network and routing layers, schemes work to minimize power along the transmission path [28], set routes according to the energy remaining at nodes along that path

[33], and use mechanisms to save power through the distribution of messages among various paths from source to destination [10]. Finally, higher layer protocols that often incorporate routing semantics exist to form groups and rotate leadership responsibilities allowing non-leader nodes to sleep and conserve their energy [4]. Again all of these protocols involve layered decisions that should adhere to strict modular programming interfaces allowing our work to coexist with them.

3. Analysis of the Problem

Various studies of throughput and channel utilization for wireless ad hoc networks have identified the limits of sensor networks due to asymmetric channels, multi-hop interference, high traffic density, and unpredictable communication patterns. To minimize such problems, mechanisms for contention have been introduced to notify neighbors of a node's intention to send a message. While such mechanisms have proven effective in minimizing collisions and, therefore, make better use of the channel, the overhead involved in sending control messages remains significant. Aside from control overhead incurred during handshake, additional idle time is spent listening to the channel and backing off to determine when it is appropriate to initiate channel contention. Such properties create ample opportunity for improvement.

If it is possible to reduce the number of control messages sent while still distributing information about a node's communication intentions, it would save significant time and energy by reducing the total number of messages and time spent contending for the channel. One mechanism for achieving such a feat is through application dependent data aggregation (ADDA). The merging of data that maintain common properties (semantics) and are destined for the same node has been a common approach to reducing traffic. While such mechanisms have proven effective in reducing traffic and easing congestion, several issues that limit the extent to which they are evolvable provide us with insight into developing an application independent aggregation (AIDA) mechanism.

- Due to the nature of application specific aggregation, such mechanisms require the appropriate naming of data and require that lower level protocols performing such aggregation have knowledge and logic to support these naming semantics. As a result, in an application specific aggregation scheme, the logic of the components will need to be changed every time the operation or task changes. For example, different aggregation logic may be needed for mapping, counting, averaging, standard deviation, etc. The more operations the applications have the more specific the aggregation logic needs to be, leading to time consuming modifications and a cumbersome design. AIDA seeks a solution without such cross-layer dependencies in order to be utilized over a wide range of data types and applications without incurring the costs of rewriting components. This reduction of inter-layer dependencies leads to a lower cost to system evolvability.

- Pervious aggregation schemes combine application specific data through consensus algorithms, averaging functions, or by some other mathematical manipulation of data, resulting in a loss of information. Because such schemes bind algorithms to the application and make it difficult to control the degree of information loss we seek a solution that performs lossless aggregation in a more general context.

- The sensor networks we envision will be multi-purpose systems. These systems should therefore support aggregation across different data types. An ADDA scheme will be limited and somewhat ineffectual as it is hard to aggregate temperature readings with light readings in an application specific way. We desire a solution that allows us to aggregate traffic originating at various application protocols without any knowledge of the application that generated this data.

- To properly aggregate named data from a common source, one must associate both location and time to that data to ensure that information is not lost or inappropriately merged. For example, reports on temperature from the northeast corner of a network should not be combined with temperature reports from the southwest corner just because they share a common type. Any aggregation performed must therefore be time and direction sensitive to ensure that data received at the requester remains meaningful.

- Current aggregation schemes assume that more aggregation is always better. As sensor network traffic changes, there exist times when varying degrees of aggregation are necessary to optimize communication and augment throughput. However at other times aggregation simply acts to delay data transmission. AIDA utilizes feedback control based on network traffic conditions when making aggregation decisions to adaptively optimize bandwidth while minimizing system energy consumption, which is underexploited by pervious schemes [16][21][24][25][28]

Application dependent data aggregation (ADDA) schemes have proven to be effective solutions for sensor networks. Given the research issues underexploited by such schemes, we seek a value-added solution that adapts to changing network conditions, improves the networks use of bandwidth, is simple and fast, has limited overhead, performs aggregation without loss of information, and considers the timeliness of end-to-end traffic. In addition, we require a solution that performs aggregation transparent to other components. This will allow AIDA to work with, or exist independently of, other communication protocols so that AIDA can leverage the performance and maintain the benefit inherent to existing ADDA schemes.

4. Protocol Design

Our solution is an aggregation layer module that resides between the data-link and networking layer to aggregate packets through network unit concatenation. The aggregation component combines network units into a single outgoing AIDA payload to reduce the overhead incurred during channel contention and acknowledgment. No semantics of the data in the network units are used. Aggregation decisions are made in accordance with an adaptive feedback-based packet-scheduling scheme that dynamically controls the degree of aggregation in accordance with changing traffic conditions.

4.1. AIDA Architecture Design

The basic design of AIDA is shown in Figure 2. We separate AIDA functionality into two components. One is the functional unit that aggregates and de-aggregates network packets (units). The other is the AIDA Aggregation Control Unit, employed to adaptively control timer settings and fine-tune the desired degree of aggregation.

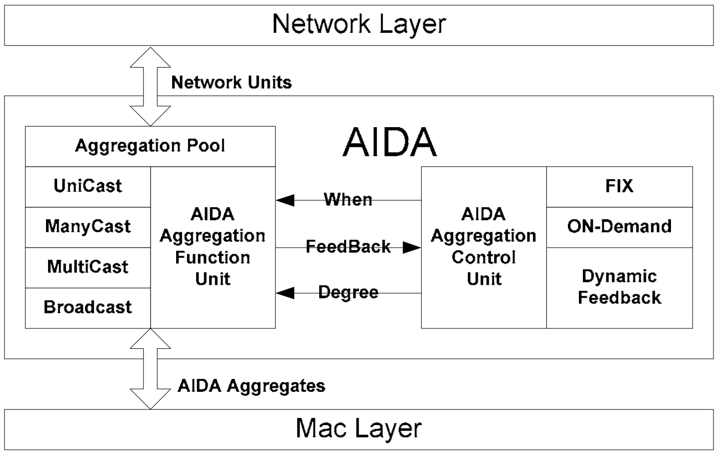


Figure 2: AIDA Components

The protocol works as follows: Packets from the network layer are placed into an aggregation pool. According to the number of packets to be concatenated in one aggregate and the next-hop destinations of those packets, AIDA's Aggregation Function Unit chooses one of four AIDA packet formats (Described in depth in section 4.3) to build an aggregate and passes this aggregate down to the MAC layer for transmission. The decision of how many packets to aggregate and when to invoke such aggregation is left up to the AIDA Aggregation Control Unit, a feedback based adaptive component which makes on-line decisions based on local current network conditions.

Similar to outgoing traffic, incoming traffic is received at the MAC layer and passed up to AIDA. Within AIDA the incoming aggregates are re-fragmented into their original network units of which each piece of the aggregate is passed up to the network layer for re-routing or application de-multiplexing and delivery. Although we acknowledge that many aggregates may be bound for the same ultimate destination (it could be more efficient not to de-aggregate and re-aggregate at every intermediate node), we perform such de-aggregation to ensure the modularity of layers and allow the networking component to determinate routes independently for each network unit.

The aggregation of multiple network units into a single AIDA aggregate for transmission reduces the overhead of channel contention (wait/backoff) and the transmission overhead of control packets (such as RTS/CTS/ACK in 802.11 [3], RTS/CTS in MACAW [5], ACK in regular reliable MAC) so that these costs are incurred once per aggregate. By increasing the number of network units combined into a single AIDA aggregate (referred to as the degree of aggregation [DOA]), we are able to save [DOA – 1] * [contention time] msec on each transmission.

While the aforementioned AIDA function unit is straightforward, it is an intricate research problem to design an adaptive AIDA control unit to set appropriate timing and DOA parameters online. As we show in our evaluation section, different control schemes do have a huge impact on system performance. More detail on these control schemes are provided and discussed in section 4.2.

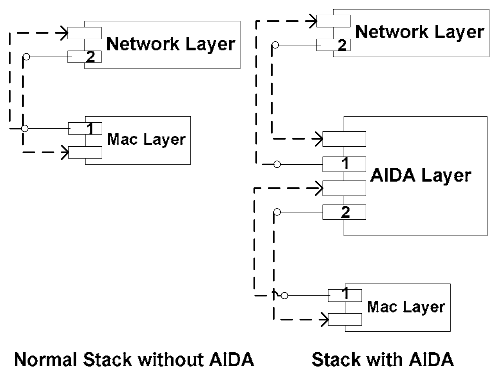


Figure 3: AIDA Implementation Design

To keep AIDA transparent from other protocol layers, we use a *delegation* approach to intercept all function calls between the MAC and Network layer. The networking component assumes it is talking directly to the MAC layer and vice versa. Using this method, our data aggregation layer imitates the interfaces exposed by both the MAC and Networking layer. The stack resulting from this technique appears in Figure 3.

4.2. Aggregation Schemes in AIDA control Unit

To better understand the effect of aggregation and our success in building an adaptive solution, we design, implement, test, and compare several versions of AIDA. Versions of our architecture include the FIX, On-Demand and Dynamic Feedback schemes. These schemes range from aggregation decisions based on static thresholds to our ultimate solution that incorporates a dynamic online feedback control mechanism into our protocol. A baseline without aggregation is also provided for comparison. Details of these implementations are provided in this section.

4.2.1. No Aggregation

With no aggregation (the baseline scheme), we simply employ the normal network stack without modification passing packets directly from the network protocol to the MAC protocol and vice versa.

4.2.2. Fixed Scheme

In the fixed scheme (FIX), AIDA aggregates a fixed number of network units into each AIDA payload (DOA = N_{fixed}). When this fixed number of network units has been aggregated, the AIDA payload is passed down to the MAC layer for transmission. To ensure that network units don't wait an indefinite amount of time before being sent, we also incorporate a timeout value (T_{fixed}) into this scheme to ensure that aggregation is performed, regardless of the number of network units, within some time threshold. The design of the FIX scheme is shown in Figure 4.

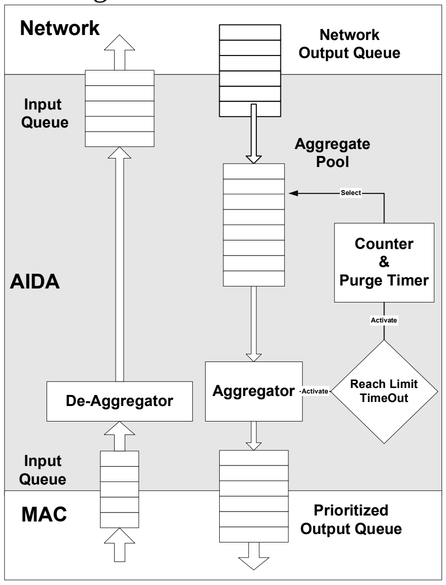


Figure 4: AIDA FIX scheme

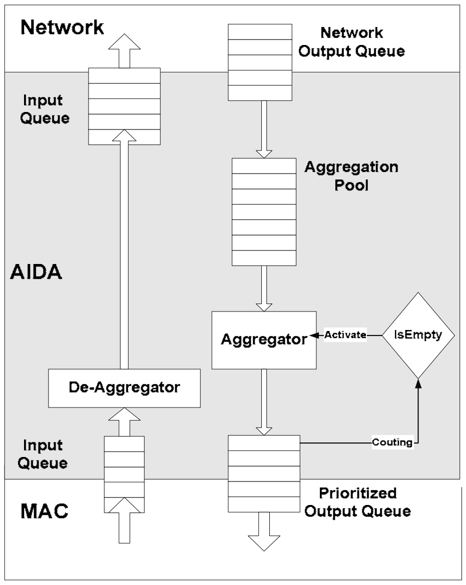


Figure 5: AIDA On-Demand scheme

4.2.3. On-Demand Scheme

To prevent unnecessary per hop delay, our On-Demand scheme monitors the AIDA output queue to ensure that there is always an AIDA payload resident for MAC layer dequeing and transmission. When the MAC is available for transmission, no network units will be held back by the AIDA layer in an attempt to achieve a higher DOA (unless the maximum MAC unit size is reached). AIDA layer data aggregation only takes place when time is available (the outbound message queue has built up or the medium is busy preventing the MAC layer from accessing the channel). This scheme provides virtually transparent aggregation without incurring message delay costs. The inner works of the On-Demand scheme is shown in Figure 5. It is worth noting that the On-Demand Scheme is a reactive solution, where passive measures allow the DOA to dynamically change with varying traffic patterns. When there is little traffic, the outbound message queue rarely builds up and no aggregation is performed. As traffic increases, the length of the outbound message queue increases resulting in a proportional increase in the DOA.

As shown in Figure 5, the On-Demand scheme only requires simple monitoring logic to test whether the outbound queue is empty or not. This simplicity of code is preferable for a constrained sensor node. It should be noted that by aggregating a train of network units with one MAC header per aggregate, ON-Demand scheme can reduce the header overhead than the scheme that plainly flushes all packets out in the queue.

4.2.4. Dynamic Feedback Scheme (DYN)

Our ultimate solution, the Dynamic Feedback scheme (DYN), implements a combination of on-demand and fixed aggregation where the DOA threshold (N_{DYN}) is adjusted dynamically. As shown in Figure 6, the scheme works by monitoring the AIDA output queue to determine its availability while also collecting data on the queuing delay imposed on AIDA payloads awaiting transmission. Using this information and operating under the basic premise of control theory, our aggregation mechanism dynamically adjusts the degree of aggregation (DOA=N_{DYN}) to converge MAC delay to a certain set point. This scheme begins with N_{DYN} set to one. In the case of low network traffic, DYN will default to the On-Demand mechanism delivering packets to the MAC transmission queue as soon as they are ready. As network traffic builds up and the contention delays transmission, our feedback loop adjusts our admission threshold (N_{DYN}) to allow a greater degree of aggregation prior to sending.

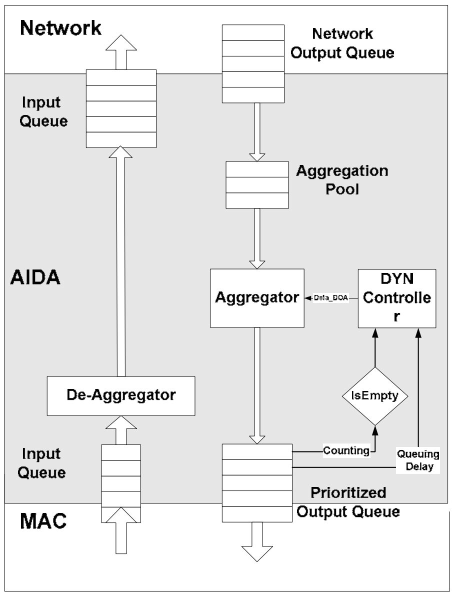


Figure 6: AIDA Dynamic Feedback scheme

Feedback Control Design:

Intuitively, an algorithm based on heuristics rather than theoretical foundations can be used to adjust the DOA values to affect the MAC layer delay a packet experiences. When the MAC delay increases, the DOA threshold increases to lower the feeding rate to the MAC layer. As a result, fewer nodes participate in channel contention leading to a lower MAC delay. However, since heuristic feedback control lacks knowledge of system dynamics, it is subject to over or under reaction and cannot adapt to the system well. This warrants the development of an analytical model to reveal the dynamics between DOA values and the MAC layer. Such a model serves as a guide for developing an appropriate feedback controller.

It is common practice to use a time slotted approach (e.g. in ALHOA and CSMA) to analyze the performance of contention-based protocols and establish a system model. Here while our approach does not assume a slotted MAC, we adopt this analysis technique to simplify problem formulation. The modeling process goes as following:

A general form for calculating MAC delay can be defined as

$$D_{mac}(k) = D_{minimum} + \#collisions(k) * D_{reslove} \qquad (1)$$

where $D_{mac}(k)$ is the MAC delay packets experience during time period [$k,k+1$], $D_{minimum}$ is the MAC delay when no collision is experienced, and it is the performance set point that control loop wants to achieve. *#collisions(k)* is the number of collisions a successful transmission will encounter at time interval [$k,k+1$], and $D_{reslove}$ is the collision delay plus the time to resolve a single collision, also considered to be a constant. It should be noted that (1) establishes the model for the MAC layer. The wait delay to build an AIDA packet is traffic-dependent and should not be considered in MAC modeling process.

Assume at a certain time interval *N(k)* packets from different sensor nodes are ready for transmission. Statistically, AIDA will pass down only an average of *N(k)/DOA(k)* packets to actively compete for the channel. *DOA(k)* here is the average DOA values of the all nodes who compete for the channel. We denote the probability of a packet being transmitted at this time period by the symbol τ. This τ value is a function of the type of MAC protocol. An outgoing packet encounters a collision when it overlaps with the transmission of at least one other packet from the remaining *N(k)/DOA(k)*-1 packets. Accordingly, the average collision probability P can be calculated as

$$p = 1-(1-\tau)^{N(k)/DOA(k)-1} \quad N/DOA \geq 1 \tag{2}$$

Naturally, the average number of transmissions required for each successful transmission is

$$E(\#collsions+1) = \frac{1}{(1-p)} \tag{3}$$

Substituting (2) into (3) gives us the expected number of collisions each successful transmission will encounter.

$$E(\#collsions) = \frac{1}{(1-\tau)^{N(k)/DOA(k)-1}} - 1 \quad N/DOA \geq 1 \tag{4}$$

Combining (1) and (4) then gives us the approximate correlation between the DOA values and the MAC layer delay

$$D_{mac}(k) = \left[D_{minimum} - D_{reslove}\right] + D_{reslove}(1-\tau)^{1-\frac{N(k)}{DOA(k)}} \tag{5}$$

Since D_{minium}, $D_{reslove}$ and τ are independent of *DOA* values, we calculate the differential of equation (5) and get the small-signal model for the system:

$$\begin{cases} D_{mac}(k+1) = D_{mac}(k) + \dfrac{\lambda_1}{DOA(k)^2}\lambda_2^{\frac{1}{DOA(k)}}\Delta DOA(k) \\ DOA(k+1) = DOA(k) + \Delta DOA(k) \end{cases} \tag{6}$$

$$where \quad \lambda_1 = D_{reslove} * (1-\tau)N(k)(Ln(1-\tau), \quad \lambda_2 = (1-\tau)^{-N(k)}$$

Because λ_1 and λ_2 are independent of the DOA, they can be considered constant in the vicinity of a small signal control model. Note that the goal of this approximate model (6) is not used to precisely calculate MAC delay under different DOA settings, but is used to design our controller. A tailored model can be established by deriving on the values of λ_1 and λ_2 based on particular properties of the chosen MAC protocol. However, for the sake of MAC-independence, we design a general form for our controller in accordance with equation (6) as follows.

$$\Delta DOA(k) = G(k) * e(k) \tag{7}$$

$$where \quad G(k) = P_{DOA} * DOA(k)^2 \quad e(k) = (D_{mac}(k) - D_{\min imum})$$

In equation (7), P_{DOA} is an implementation parameter to set the gain between the changes of DOA and the error in MAC delay control. Thus AIDA is essentially modeled as a first-order system and therefore the gain *G(k)* in equation (7) does not need to be constant for stability analysis, as long as G(k) is bounded. The pictorial notation of this control loop is shown in Figure 7.

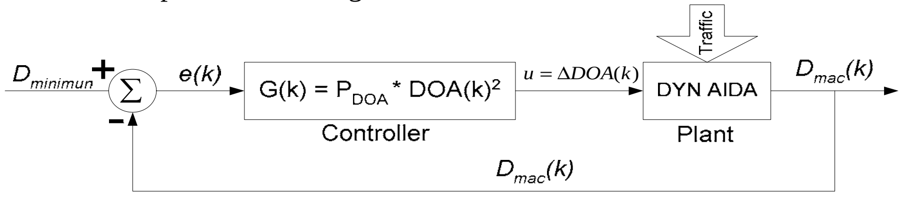


Figure 7: The control loop for the DYN AIDA scheme

As we will show in the evaluation, the current adaptive controller works best under a wide range of traffic scenarios under investigation. However, we acknowledge that the modeling portion of our work has room for improvement to precisely reflect the nonlinear behavior of the MAC contention.

4.3. AIDA Function Unit

The AIDA Aggregation Function Unit (Figure 2) is responsible for the aggregation and de-aggregations of network units. This component builds four different types of aggregates, namely *Unicast*, *Manycast*, *Multicast* and *Broadcast*, in accordance with the set AIDA parameters and current state of the module.

- If there is only one network unit ready when the AIDA Control Unit is ready to aggregate (e.g. a time out occurs), the AIDA Function Unit will use Unicast to send the waiting unit out to the specified neighboring node. In this case, no aggregation is performed.
- If all network units to be aggregated are targeting the same next-hop node, AIDA sends out an aggregate using Manycast with the target specified.
- When network units to be aggregated have different next-hop addresses, the slightly more complex Multicast type is used to take advantage of the broadcast nature of wireless communication. In this case, AIDA merges network units, regardless of which neighbor each network unit targets, into a single aggregate and uses the MAC broadcast address as the destination. Every neighbor of the sending node will receive and de-aggregate this Multicast packet to determine whether or not a portion of the aggregated payload was destined to it.
- Finally, the Broadcast type of AIDA is used in the case where all aggregated network units are Broadcast messages.

Although a single packet format (Multicast) is logically enough to support all of the aforementioned scenarios, we argue that tailored packet formats for each scenario can reduce the AIDA header size and save bandwidth. These savings are beneficial in a resource constrained sensor network justifying the small amount of complexity added through AIDA typing.

4.4. Packet Format Details

Like most communication stack layers, AIDA adds meta-information to a packet in the form of a header. This header defines the aggregation format used for later de-aggregation, de-multiplexing, and seamless delivery to the appropriate network layer protocol. This header is placed in front of all aggregated network units and is included in the AIDA data units passed down to the MAC layer for transmission. Upon delivery at a node, the AIDA header can then be used to validate the specific aggregation mechanism used (in the case where multiple aggregation options are provided), assess the structure of the AIDA payload for de-aggregation, and potentially break apart, de-multiplex, and deliver each network unit to the appropriate network layer module.

It should be noted that by aggregating the network payloads, AIDA reduces the number of packets sent at the MAC layer, thus actually reduce overall header cost. The general form of the AIDA header is provided in Figure 8. Some fields inside this general form are *not* used for certain AIDA payload types.

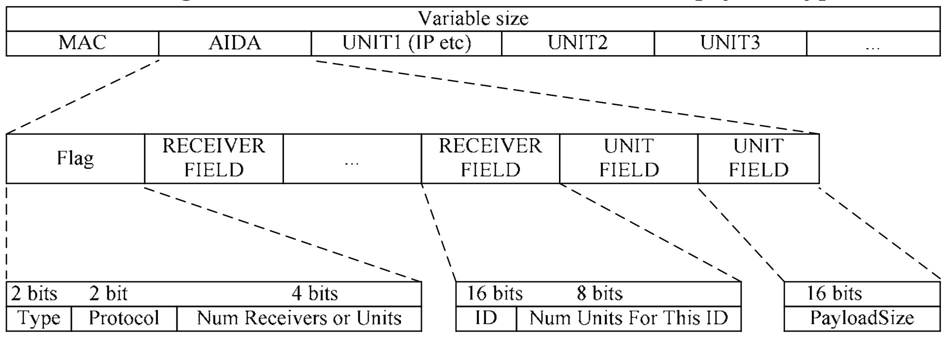


Figure 8: AIDA General Header format

4.4.1. FLAG for All Types

The first component of the AIDA header is an eight bit (1 byte) flag specifying information relevant to all aggregated network units. The Flag is composed of a Type field (2 bits), a protocol field (2 bits), and the number of Next Headers (4 bits).

- **Type Field**: The Type bits are used to specify whether the AIDA packet should be treated as a *Unicast*, *Manycast*, *Multicast*, or *Broadcast*.
- **Protocol Field:** The Protocol field (2 bits) of the AIDA Flag denotes to which network layer AIDA should de-multiplex network units.
- **Num Receiver/Units**: This filed (4 bits) denotes how many headers follow. For *Unicast*, *Manycast* and *Broadcast* traffic, this field is set to the number of network units inside this aggregate. For *Multicast* traffic this field will contain the number of neighbors receiving portions of this aggregate.

4.4.2. Receiver Field for Multicast Type

The Receiver Field is *only* used by Multicast AIDA packets. Each field contains an ID specifying the intended recipient followed by the number of network units contained in this aggregate that are destined for the specified neighbor. In the case of Unicast, Manycast or Broadcast AIDA payloads, there is *no* need to differentiate between receiving nodes so this field is not used.

- **ID Filed:** The ID field (2 bytes) contains a locally unique identifier of the node receiving a specified number of network units.
- **Num Units For this ID Field:** This field is an 8 bit (1 byte) field that identifies the number of aggregate network units that are destined for the neighbor specified in the ID Field.

4.4.3. Unit Field

UNIT filed is used during de-aggregation for delimiting the boundaries between network units. It consists of a 16 bit (2 byte) field that specifies the size of each network units. In the Unicast case there is no boundary to be identified, so the UNIT field is not used.

4.5. AIDA Header Overhead Analysis

First, it should be reminded that though AIDA introduces a new header, it actually reduces overall header overhead by aggregating several network units into one MAC payload; For example, in 802.11 the MAC header length is 28 bytes. To send out N network units without AIDA, the total header overhead would be 28*N bytes. Using AIDA we reduce the total header overhead to 28+AIDAHeaderSize bytes. As long as the value of N (the DOA) is greater than 1, AIDA effectively reduces the total packet overhead incurred during transmission.

It is simple to assess the overhead incurred during the aggregation of network units according to the description in section 4.4. For comparison, the packet structure with and without AIDA is shown in Figure 9.

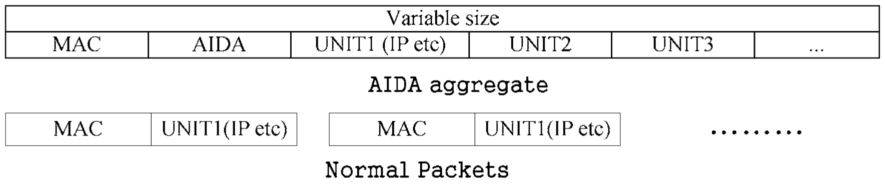


Figure 9: Format Comparison

- Unicast only uses the Flag field and therefore incurs a single byte of overhead.
- Besides the 1 byte flag, Manycast and Broadcast packets need to delimitate the boundaries of multiple network units, thus incurring an average of (2+1/N) bytes overhead per network unit (where N is the number of network units aggregated into an AIDA payload).

- Because multiple next-hop node addresses need to be differentiated, Multicast payloads have a slightly larger overhead on the average (2+1/N+3/M) bytes per network unit (where N is the same as before and M is the average number of network units for each next-hop node),

4.6. AIDA Savings Analysis

Adding header information to any transmission will intuitively increase transmission time for a single packet. We therefore only see savings in per transmission overhead costs when aggregating multiple upper layer payloads into a single transmission. By analyzing our AIDA header structure, we can see that savings differ for Unicast, Manycast, and Multicast transmissions. To better understand the potential benefits of aggregation, and to compare different levels of aggregation under different traffic patterns, we provide a theoretical analysis to assess overhead with respect to transmission time. The analysis presented assumes optimal aggregation to the specified DOA without incurring any additional cost waiting for network layer payloads. We also assess savings without considering collisions and backoff, two factors that will ultimately increase the utility of AIDA.

The cost of packet transmission in the simple single sender, single receiver scenario with no channel contention and an arbitrary MAC layer is the time consumed by the MAC acquiring and setting up each transmission plus the time for sending the message, all multiplied by the number of individual transmissions. To maintain MAC layer independence, we simply assign the variable *M*, to the time (in msec) for performing MAC layer transmission preparation. For an 802.11 like MAC, this cost includes the channel sense, RTS, CTS, ACK, and intermittent wait times between control packets. For network units of size *S* transmitted at *R* bytes/second, the AIDA header overhead is *O* (in bytes), and *DOA* is the number of packets aggregated. The cost C_{AIDA} (in msec) can be calculated from equation (8):

$$C_{AIDA} = M + (S * DOA + O) * R \tag{8}$$

In contrast, the cost of sending DOA number of packets without the aggregation scheme C_{None} is

$$C_{None} = (M + S * R) * DOA \tag{9}$$

Hence, the percentage saving in cost is calculated as following:

$$SavingPercentage = \left(\frac{C_{None} - C_{AIDA}}{C_{None}}\right) = \left[1 - \frac{S * R}{M + S * R}\right] - \frac{M + O * R}{M + S * R} * \frac{1}{DOA} \tag{10}$$

From equation 10, we can see that the saving increases as the DOA increases when the cost at the MAC layer (M) is non-negligible. To demonstrate the utility of AIDA, we graph theoretical savings for our scheme under an 802.11 like MAC contention scheme for a 200 Kbps channel. The AIDA payload is passed down to a simplified 802.11 MAC that performs idle listening, RTS/CTS handshaking, and follows up each DATA packet with an acknowledgment. The control packet size for our theoretical MAC is 11 bytes. Contention also includes 5 msec's of idle listening and the DIFS and SIFS intervals are chosen at 10 and 5 msec's respectively in accordance with the current MICA specifications. We graph variable size network units to better understand the effect of packet size on potential savings.

Figure 10 demonstrates theoretical time savings as a percentage of the total time it would take to send the number of packets without AIDA. These savings are calculated by comparing the time to send a single AIDA aggregate, consisting of [DOA] network units with one MAC header, versus the time to send [DOA] separate packets without any AIDA header information or data aggregation performed. From this chart we can see that as the degree of aggregation increases, the percentage of savings in time increases drastically. We also note that as payload size increases, the relative time saving decreases. This occurs when data transmission time becomes a larger percentage of the total transmission time. Finally, we note that when AIDA fails to perform any aggregation as shown in Figure 10 when DOA = 1, the cost incurred is a single byte of data, which amounts to virtually no increase in transmission time.

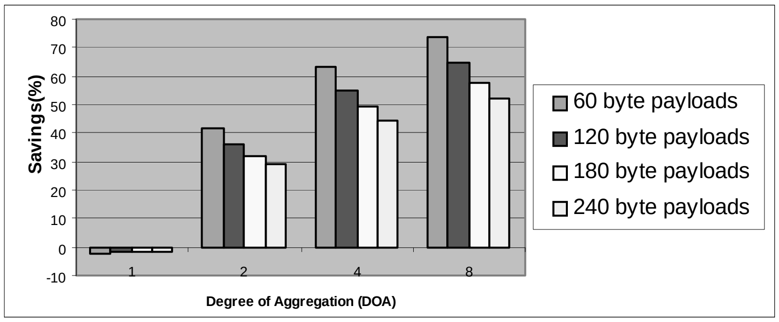


Figure 10: AIDA Theoretical Savings

5. Evaluation

We simulate AIDA in GloMoSim, a scalable discrete-event simulator developed at UCLA. This software provides a high fidelity simulation for wireless communication with detailed propagation, radio, MAC, and network layer components. Table 1 describes the detailed setup for our simulator. For our experiments the communication parameters are mostly chosen in accordance with Berkeley MICA mote specifications [34], the popular hardware platform on which sensor network research systems are currently deployed for testing. The current version of the MICA motes supports a 40kbps transmission rate and the next generation is expected to provide higher than 1Mbps rates. Based on these considerations, we choose 40 ~ 200Kb/s as the effective bandwidth for our evaluation (default 200Kbps unless otherwise specified). Finally, we choose 802.11 as our MAC layer protocol, which has been implemented in a scaled down version on the MICA platform.

Routing	GF
MAC Layer	Simplified 802.11 DCF
Radio Layer	RADIO-ACCNOISE
Propagation model	TWO-RAY
Bandwidth	40 ~ 200Kb/s
Payload size	32 Byte
TERRAIN	(200m, 200m)
Number of Motes	100
Node placement	Uniform
Radio Range	40m

Table 1. Simulation settings

Since our work is the first we know of concerning data aggregation without utilizing application information, we evaluate our work based on different aggregation schemes we provide and a normal stack without aggregation support. In this evaluation we compare the performance of four schemes: No-aggregation, FIX, On-Demand, and DYN as previously defined. We show that DYN feedback is the best solution with better performance under all traffic scenarios tested.

In our evaluation, we analyze the following set of metrics: end-to-end delay, energy consumption, MAC control packets, degree of aggregation (DOA) and AIDA control overhead. These metrics are investigated under three sets of typical traffic patterns with a total of 72 different traffic loads, which allow us to access AIDA's adaptation capability under a wide range of traffic situations. Each plotted data point is the average of 10 runs generated from different random seed values. This ensured that 95% confidence intervals for our data are within 2~5% of obtained means. For legibility reasons we do not plot these confidence intervals in this paper. Full experimental data can be obtained from the authors upon request.

5.1. Work load Settings

We expect typical communication patterns inside a sensor network to be established based on request and retrieval semantics for data delivery between sensor nodes and a querying entity. One-to-one, many-to-one and many-to-many communication patterns are representative workloads in sensor networks. One-to-one communication happens when one sentry node detects some activity that needs to be reported to a remote entity. Alternatively, a quering entity will require periodic reports from the whole sensor area, which take the form of many-to-one communication. It is more common that multiple applications run simultaneously and the traffic flows interleave with each other, which is a many-to-many cross-traffic pattern.

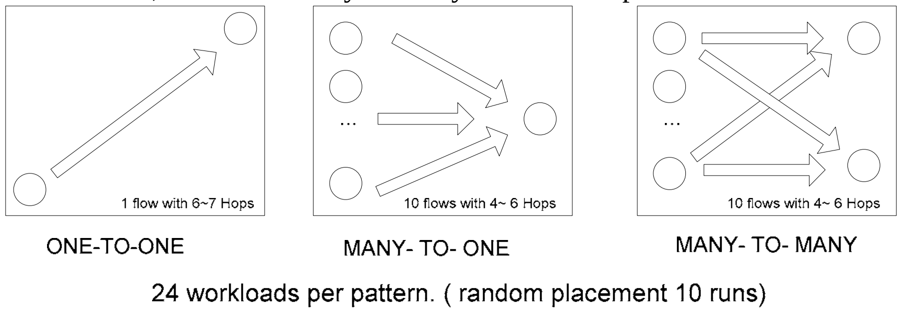


Figure 11: Traffic Load Settings

In our evaluation we focus on the aforementioned three representative communication patterns (Figure 11). To test the one-to-one scenario, we have a single node randomly placed on the left lower corner of our terrain send out a single CBR flow to the right upper corner of the terrain where the average route is approximately 6~7 hops. In the many-to-one scenario, 10 nodes on the left side of the terrain send out 10 CBR flows to the center-right side of the terrain where we place a single querying node. In many-to-many scenario, 5 nodes on the left side of the terrain send out 10 CBR flows (2 flows for each node) to the two querying nodes at the upper and lower right corner of the terrain, respectively. The sending rate of each CBR flow is incrementally increased to test the performance of AIDA under different traffic loads.

5.2. End-To-End Delay

5.2.1. End-to-end delay under different schemes

A major goal of the AIDA protocol is to achieve energy savings without jeopardizing end-to-end delay. AIDA not only doesn't add to the end-to-end delay, but in the presence of high degrees of aggregation, actually decreases end-to-end delay by reducing the number of control packets used at the MAC layer.

Figure 12, Figure 13 and Figure 14 graph end-to-end delay as a function of traffic loads under three traffic scenarios. These graphs show that end-to-end delay for CBR without performing aggregation increases dramatically as the overall traffic increases gradually. This is the typical case for multi-hop wireless networks where channel contention is much higher than in a single hop wireless LAN, As shown in figures, when traffic is low (e.g. below 3 packets/per flow in Figure 13), all schemes except the FIX have very short end-to-end delay (abut 70~100ms). The reason for additional delay in the FIX scheme is because the FIX scheme holds packets despite an available channel in order to obtain its specified degree of aggregation. The lower the sending rate is, the longer the FIX scheme needs to wait. In contrast, the On-Demand and DYN schemes send out packets whenever possible, eliminating any additional end-to-end delay. On-Demand scheme performs well because of its reactive adaptive mechanism. The DYN scheme performs the best in all scenarios because it dynamically adjusts the required DOA according to the MAC delay that the outgoing packets experience. In heavy traffic, it is beneficial to reduce number of node competing for the channel by reducing sending rate. In the presence of extremely heavy traffic, we show that DYN scheme is capable of reducing the end-to-end delay by as much as 80%, compared to non-aggregation case, when flow rate at 8.5 packets/second per flow (see Figure 14).

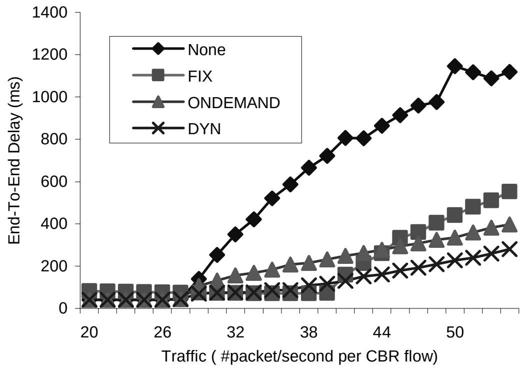


Figure 12: Avg E2E delay (one-to-one 200Kbps)

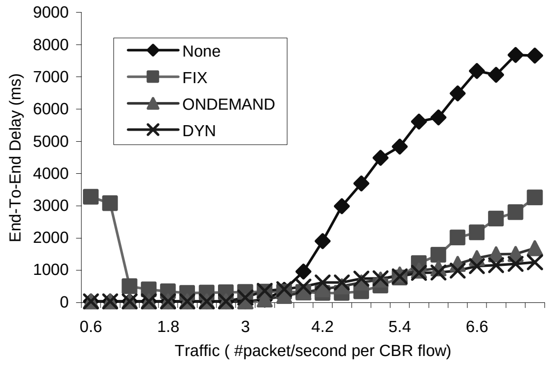


Figure 13: Avg E2E delay (many-to-one 200 Kbps)

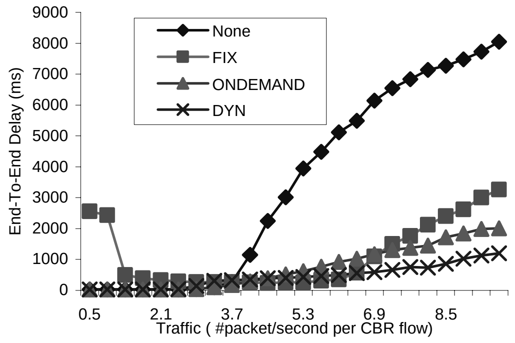


Figure 14: Avg E2E delay (many-to-many 200Kps)

5.2.2. End-to-end delay under different available bandwidth settings

In this experiment, we investigate the end-to-end delay under the different bandwidth settings. The workloads are chosen differently for each bandwidth setting in order to compare the performance of each scheme under from underutilized to saturated traffic situations.

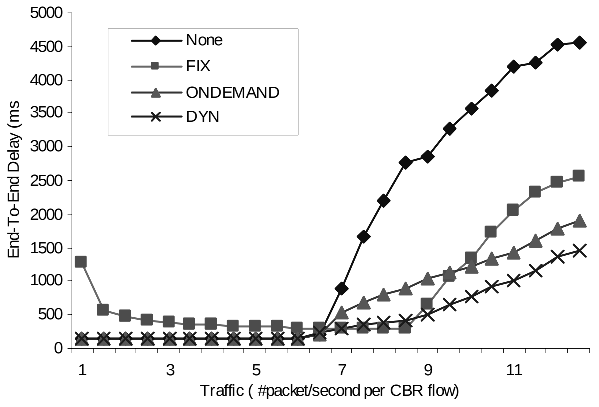


Figure 15: E2E delay (one-to-one under 40Kps)

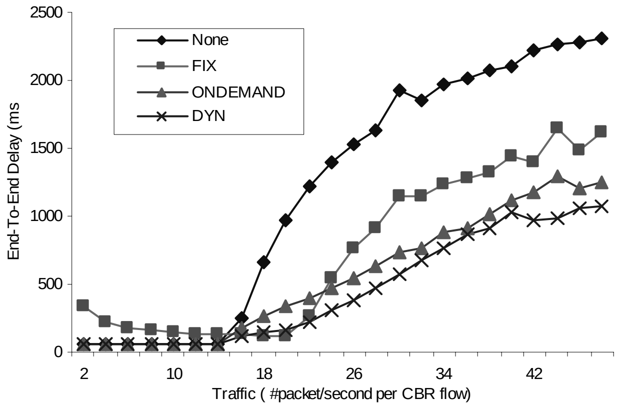
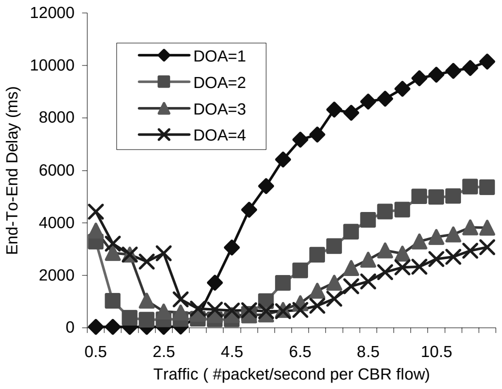


Figure 16: E2E delay (one-to-one under 100Kps)

The Figure 12, Figure 15 and Figure 16 demonstrate that DYN scheme out performances other schemes regardless the available bandwidth settings. This is mainly because that DYN can more effectively aggregate and schedule the packets according to the feedback of the currently traffic situations than other schemes. Base on such an investigation, we conclude that the improvement made by DYN scheme over other schemes is orthogonal to the available bandwidth setting, though the absolute performance gain may vary.

5.2.3. End-to-end delay under different DOA setting for the FIX scheme

In this experiment, we measure end-to-end delay for various traffic loads under different DOA settings in the FIX scheme. Figure 17 reveals the disadvantage of the FIX scheme and explains why dynamic adaptability is desired for such system. From Figure 17, we can see that there is no single DOA value that works well for every traffic pattern.



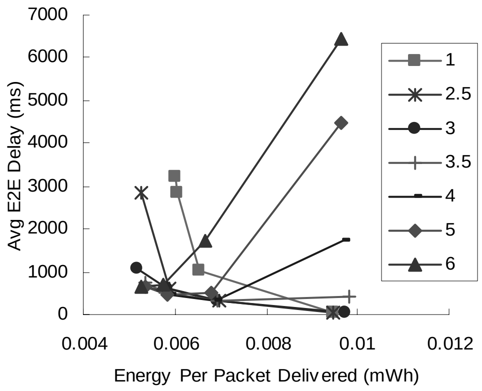


Figure 17: Avg E2E delay (many-to-one) Figure 18 E2E Delay vs. Energy (many to one)

On one hand, a high DOA value in the FIX scheme doesn't perform well under low traffic loads. For example, when the DOA is higher than 1, additional delay is incurred when the traffic load is 0.5 packets/second per flow or lower. The higher DOA settings tend to reduce congestion, but increase delay in the AIDA component for packets waiting to be sent. On the other hand, low DOA value settings don't perform well under heavy traffic. For example, shown in Figure 17, the FIX scheme with DOA = 1 has nearly double the end-to-end delay as that with DOA=2 when the traffic is about 10 packet/second per flow or higher.

In addition, Figure 18 demonstrates the performance penalty due to the lack of adaptability in the FIX scheme. We plot the relationship between average end-to-end delay and average energy consumption per packet delivered under different CBR rates form one to six packets/second. Under the light traffic (e.g. one packets/per second per CBR), the FIX scheme needs to hold back packets in order to reduce energy consumption. Under heavy traffic, (e.g. six packets/per second per CBR), the FIX scheme would cause an increase in both delay and energy consumption by choosing a fixed DOA value that doesn't reflect the traffic load.

The FIX scheme is insensitive to the traffic situations. To optimize for both light and heavy traffic, online adaptation is provided in On-Demand and DYN schemes, which can passively and proactively change the DOA value in accordance with these traffic patterns, respectively. Therefore, they exhibit a better overall performance as shown in Figure 12, Figure 13 and Figure 14.

5.3. Energy Consumption

In this section, Energy consumption, in transmission energy, is adopted as another revealing metric to evaluate the AIDA performance. Since transmission energy increases proportionally with the number of bits sent out, it can adequately summarize and reflect the performance of other related metrics such as total header overhead, number of collision, total number of bit transmitted bytes.

5.3.1. Energy consumption under different schemes

With limited power resources, it is vital for sensor nodes to minimize energy consumption during radio communication to extend the lifetime of the sensor network. AIDA achieves such energy savings via several approaches. First, AIDA reduces MAC channel contention costs by distributing these costs across multiple network units. Second, by using less MAC control packets, AIDA dampens congestion and reduces the number of collisions resulting in fewer retransmissions. Finally, networking protocols designed for sensor networks usually adopt fixed packet sizes (e.g. TinyOS networking [14]), which leads to unnecessary padding costs. In our simulation with variable size support, AIDA takes advantage of the first two approaches.

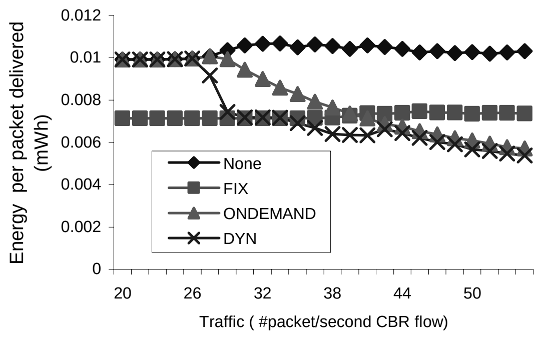


Figure 19: Energy per unit delivered (one-to-one).

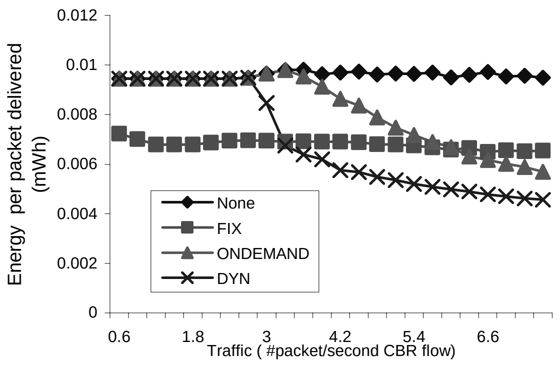


Figure 20: Energy per unit delivered (many-to-one)

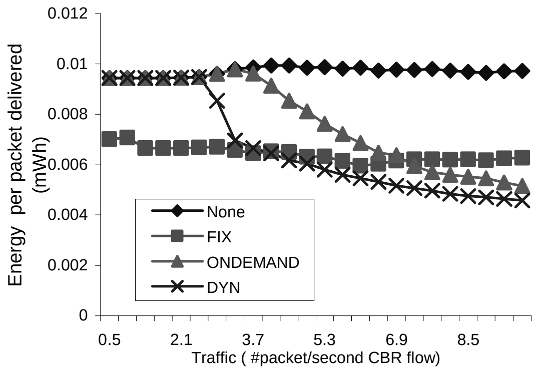


Figure 21: Energy per unit delivered (many-to-many)

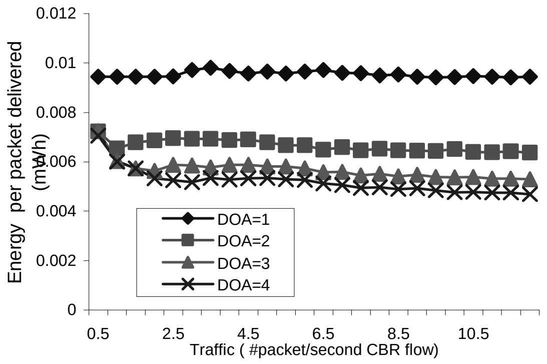


Figure 22: Energy per unit delivered (FIX scheme)

In this experiment, we measure average transmission energy per delivered packet under 24 increasing traffic loads for three traffic patterns. In, Figure 19, Figure 20 and Figure 21, our energy metrics show that the scheme without AIDA (None) demonstrates the worst performance. For example, None consumes double the energy as the DYN scheme when traffic load is about 6 packets/second per flow in Figure 21. The FIX scheme always aggregates 2 packets before sending which leads to nearly constant energy saving in both the low and high traffic situation. However, in the FIX scheme, the DOA values are set and congestion levels are not taken into account resulting in worse performance than in DYN and On-Demand schemes under heavy traffic conditions. For example, shown in Figure 21, in DYN scheme, nodes consumes about 20% less energy per packet delivered as in the FIX scheme, when traffic load is about 8 packets/second per flow.

5.3.2. Energy consumption under different DOA for the FIX scheme

Figure 22 shows energy consumption per packet delivered for varying DOA's under the FIX scheme. This graph shows that for the FIX scheme, AIDA can achieve a higher percentage of energy savings by using higher DOA values. However, as we have shown in section 5.2, a higher DOA leads to additional delay when the network is lightly loaded, therefore taking end-to-end delay into account, it is not always beneficial to increase the DOA value.

5.4. MAC control packets

Even though our AIDA design is independent of any MAC layer protocol, it can reduce MAC overhead by sending longer, but less numerous payloads to the MAC layer for transmission. This reduces the number of channel access operations performed by the MAC. This section identifies the savings incurred through AIDA aggregation at the MAC layer. The data collected here are for the 802.11 MAC protocol although we would expect very similar results from other MAC protocols.

Figure 23, Figure 24 and Figure 25 graph the number of control packets sent over various traffic loads. As shown in these graphs, the FIX scheme reduces the number of MAC control packets by approximately 50%

when the DOA parameter is set to 2. On-Demand and DYN vary their DOA and therefore incrementally reduce MAC overhead as network congestion levels increase. For example shown in Figure 25, when per flow rate exceeds 9 packets/second, DYN only used about 20% of the control packets compared to the none-aggregation case. This dramatically reduces congestion and energy consumption as shown in other evaluations.

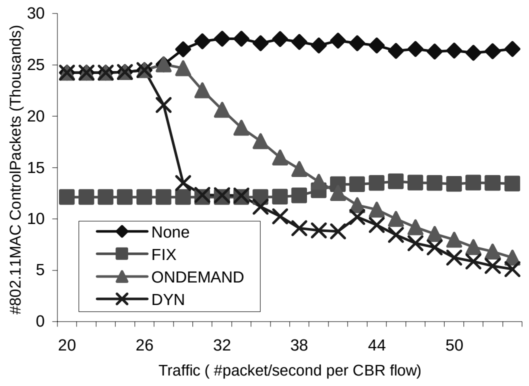


Figure 23: MAC control Packets (one-to-one)

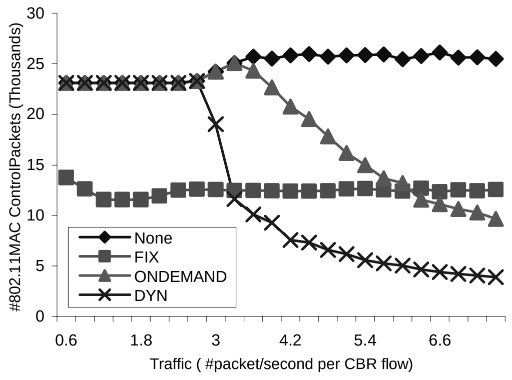


Figure 24: MAC control Packets (many-to-one)

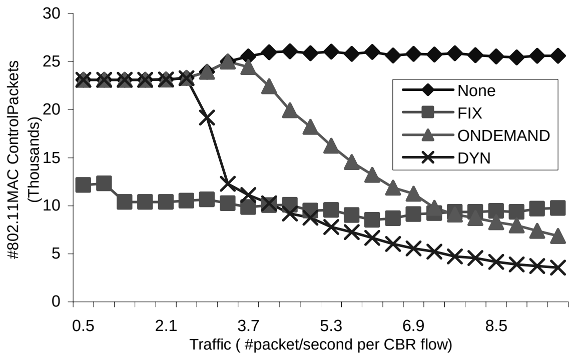


Figure 25: MAC control Packets (many-to-many)

5.5. Degree of Aggregation

As seen in the context of reducing the MAC overhead, the degree of aggregation is a major indicator reflecting AIDA's ability to achieve energy savings and congestion dampening. Without aggregation, the DOA always equals one (e.g. None case in Figure 26). In the FIX scheme where DOA is set to 2, we can see that a constant value for the degree of aggregation is achieved. In the On-Demand scheme, the DOA naturally follows traffic congestion levels. In DYN, the DOA is controlled by a feedback loop embedded inside AIDA.

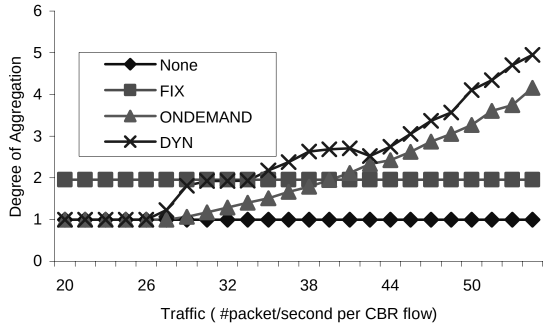


Figure 26: DOA (one-to-one)

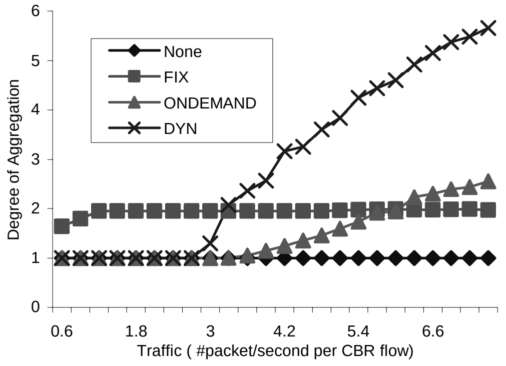


Figure 27: DOA (many-to-one)

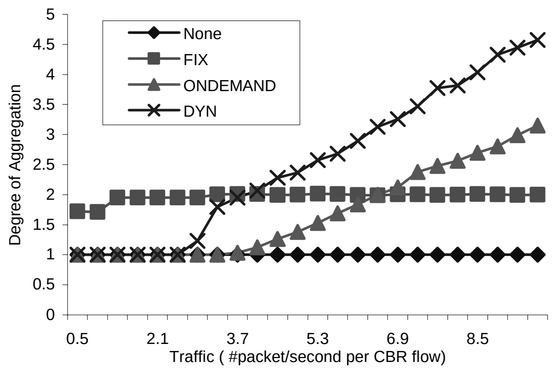


Figure 28: DOA (many-to-many)

Figure 26 , Figure 27 and Figure 28 graph the achieved DOA under various traffic conditions for the tested schemes. Figure 26 shows how DYN has roughly the same DOA value as the On-Demand scheme in the one-to-one pattern situation. However, in the more congested situations (Figure 27 and Figure 28), DYN achieves a higher DOA value than On-Demand resulting in more savings on channel bandwidth and energy consumption.

5.6. AIDA overhead

As shown in AIDA Header Overhead Analysis (section 4.5), AIDA's header overhead is about 3 bytes for Multicast packets, 2 bytes for Manycast, and 1 byte for Unicast and Broadcast per network unit. Figure 29, Figure 30 and Figure 31 graph per packet AIDA overhead under various traffic loads. As shown in Figure 30, under many-to-one conditions, the FIX scheme will send out only Manycast packets with its DOA value set to 2. This leads to an average of 2 bytes of AIDA header overhead. When the flow rate is very low (shown by the first two values for the FIX scheme in Figure 30), the FIX scheme times out before it can reach its aggregation level of 2. When this happens the FIX scheme sends Unicast packets resulting in a smaller average AIDA overhead per network unit.

In one-to-one and many-to-one traffic patterns, AIDA uses Unicast when the network is not congested in order to avoid additional delay and Manycast when congestion is apparent. This is shown in Figure 29 and Figure 30 as congestion levels increase and the overhead approaches 2 bytes per header. In one-to-one and many-to-one traffic patterns, no multicast packets are sent out, explaining why AIDA overhead never exceeds 2 bytes per network unit.

On the contrary, in many-to-many situations, AIDA takes advantage of the broadcast nature of wireless networks, uses multicast packets to address multiple next-hop nodes in a single aggregation, which require 3 bytes overhead for each multicast packet. This is shown in Figure 31 where AIDA overhead is somewhere between 2 and 3 bytes for the FIX scheme.

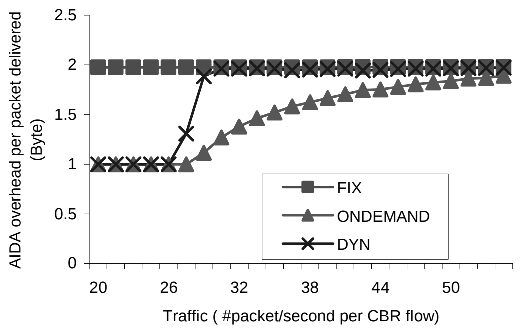


Figure 29: AIDA overhead (one-to-one)

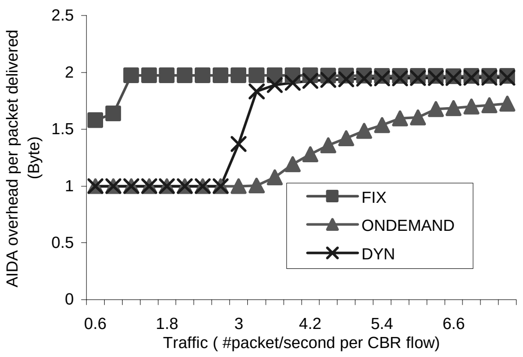


Figure 30: Aida overhead (many-to-one)

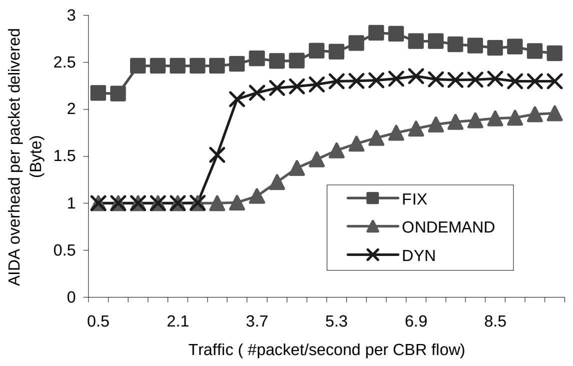


Figure 31: Aida overhead (many-to-many)

5.7. Comparisons and Summary

In summary, the FIX scheme does not take congestion into account and is not adaptable to changing traffic loads. There is no single DOA value that works well for every traffic pattern. The feedback information utilized in the ON-DEMAND scheme is essential binary: either the MAC component is busy or free. This only provides limited information to the controller. In comparison, DYN obtains delay information that directly reflects the current traffic situation resulting in a better control model and, therefore, better performance.

6. Implementation on the Berkeley Mote Test Bed

We have implemented the AIDA protocol on the Berkeley motes platform with a code size of 3,840 bytes (code is available at [9]). Three applications including data placement [28], target tracking [6], and CBR are built and tested on top of AIDA. Due to the physical limitation on the motes, it is extremely difficult to perform as extensive evaluation as we did in the wireless simulator. As a result, we only present partial results here as a study to better understand the effect of aggregation in developing a more complete adaptive solution. More detailed evaluation on upgraded versions of motes is left as future work.

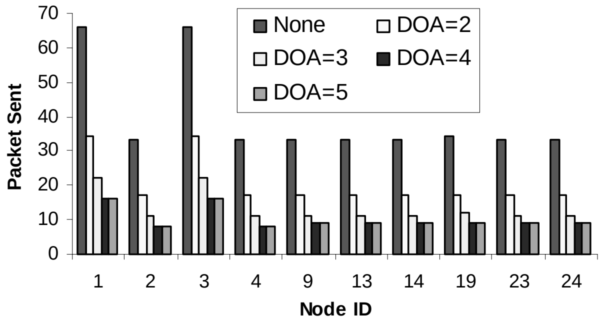


Figure 32: Packets Sent Under different DOA

In the experiment we use 25 motes to form a 5 by 5 grid. To evaluate the aggregation performance of AIDA we send three CBR flows (5 bytes payload) from node 24 to node 0 (the requesting node). The experiment collects the number of packets relayed by intermediate motes (1~23) and compares this with the results obtained from a basic GF [20] protocol without AIDA. In some embedded designs, fixed packet sizes are supported for the sake of simplicity making padding costs large when sensor data payloads are small. AIDA takes advantage of this and aggregates multiple payloads into one packet to minimize padding costs. The savings achieved by AIDA are shown in Figure 32 graphing the number of packets sent at intermediate nodes under various DOA settings. We demonstrate that the transmission cost (packets sent) is reduced as the DOA value increases. For example, when the DOA value is 2, node 1 sends out nearly half as many packets as it did without aggregation. It is worth noting that with a fixed size packet, when the DOA reaches a certain value AIDA comes to a point where it cannot compact any more network units into the AIDA aggregate. For our experiment and payload size this occurred when the DOA was 5. The latest version of TinyOS [14] supports variable packet size during transmission. Under this, AIDA can achieve higher DOA values.

7. Conclusion

In this paper we introduce AIDA, an adaptive application independent data aggregation mechanism for sensor networks. AIDA performs lossless aggregation by concatenating network units into larger payloads that are sent to the MAC layer for transmission. Due to the highly dynamic and unpredictable nature of wireless communication in sensor networks, a novel feedback-based scheduling scheme is proposed to dynamically adapt to changing traffic patterns and congestion levels. By isolating our work in a layer that sits between the networking and data-link components of the communication stack, AIDA is able to perform such aggregation without incurring the costs of rewriting components to upper or lower layer protocols. Moreover, very significantly, AIDA is a value-added compatible solution that can complement and augment the gain of application specific data aggregation (ADDA) schemes.

In our experiments we evaluate the performance gain achieved by AIDA. We show that by adaptively configuring our aggregation parameter (DOA), AIDA only introduces a small header overhead (around 2 bytes per network unit / negative overall header overhead) while reducing end-to-end delay by as much as 80% and transmission energy by 30~50% in heavy traffic conditions. As shown in our evaluation, AIDA running in the DYN (fully adaptive) scheme provides the best overall solution. The DYN feedback control loop dynamically tunes our DOA threshold and sending rate to optimize aggregation performance under varying traffic conditions by monitoring queuing delay to perform data aggregation without sacrificing end-to-end delay. The MAC control overhead is also reduced to allow for more efficient channel scheduling.

A physical implementation of AIDA on the Berkeley test bed provides initial evidence of the savings obtainable by an application independent aggregation scheme and pave the path for future implementations of our adaptive control based protocol.

8. References

[1] T. Abdelazaher, B. Blum, et. al. "EnviroTrack: An Environmental Programming Model for Tracking Applications in Distributed Sensor Networks". Technical Report CS-2003-02, University of Virginia 2003.

[2] M. Adamou, S. Khanna, I. Lee, I. Shin, S. Zhou, "Fair Real-time Traffic Scheduling over A Wireless LAN", In *Proceedings of the 22nd IEEE RTSS 2001*, London, UK, December 3-6, 2001

[3] ANSI/IEEE, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," ANSI/IEEE Std 802.11, 1999 Edition.

[4] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, Robert Morris "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks". In *Proc. of the 6th ACM MOBICOM Conf.,* Rome, Italy, July 2001.

[5] Vaduvur Bharghavan, Alan Demers, Scott Shenker, and Lixia Zhang. "MACAW: A Media Access Protocol For Wireless LAN's". In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 212-225, August 1994.

[6] Brian Blum, Prashant Nagaraddi, Anthony Wood, Tarek Abdelzaher, Sang Son, John Stankovic, "An Entity Maintenance and Connection Service for Sensor Networks," The *First International Conference on Mobile Systems, Applications, and Services* (*MobiSys*), San Francisco, CA, May 2003

[7] C. Fullmer and J.J. Garcia-Luna-Aceves, "Floor Acquisition Multiple Access (FAMA) for packet radio networks", *Computer Communication Review, vol. 25, (no. 4)*, ACM, Oct. 1995.

[8] C. Guo, L. C. Zhong and J. M. Rabaey, "Low Power Distributed MAC for Ad Hoc Sensor Radio Networks", In *Proceedings of IEEE GlobeCom 2001*, San Antonio, November 25-29, 2001

[9] T. He, L. Gu, B.Blum, Jun Xie "Nest Project Source Code Base" *at http://sourceforge.net/projects/vert/* Univeristy of Virginia

[10] Tian He, John A. Stankovic, Chenyang Lu, and Tarek F. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks", In *International Conference on Distributed Computing Systems (ICDCS 2003)*, Providence, RI, May 2003.

[11] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. "Building Efficient Wireless Sensor Networks with Low-Level Naming," In *Proceedings of the Symposium on Operating Systems Principles*, Lake Louise, Banff, Canada, October, 2001.

[12] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, “Energy-Efficient Communication Protocol for Wireless Microsensor Networks”, In *HICSS '00*, January 2000.

[13] W.R. Heinzelman, J. Kulik and H. Balakrishnan, “Adaptive protocols for information dissemination in wireless sensor networks”, In *MOBICOM, 1999*, Seattle, 174-185.

[14] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, “ System architecture directions for network sensors,” in *ASPLOS 2000*.

[15] C. Intanagonwiwat, R. Govindan and D. Estrin, “Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks,” the Sixth Annual International Conference on Mobile Computing and Networks (MobiCOM 2000), August 2000, Boston, Massachusetts.

[16] C. Intanagonwiwat, D. Estrin, R. Govindan, and J Heidemann. “Impact of Network Density on Data Aggregation in Wireless Sensor Networks”, In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, Vienna, Austria, IEEE. July, 2002.

[17] David B. Johnson and David A. Maltz. “Dynamic Source Routing in Ad Hoc Wireless Networks”. In *Mobile Computing, Chapter 5, pages 153-181*, Kluwer Academic Publishers, 1996.

[18] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. W.Knightly, “Distributed Multi-Hop Scheduling and Medium Access with Delay and Throughput Constraints”, In *IEEE MobiCOM 2001*, Rome, Italy, July 2001.

[19] Phil Karn. “MACA – A New Channel Access Method for Packet Radio”. In *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, pages 134-140, September 1990.

[20] Karp, B., “Geographic Routing for Wireless Networks”, Ph.D. Dissertation, Harvard University, Cambridge, MA, October, 2000.

[21] B. Krishnamachari, Deborah Estrin, and Stephen Wicker. “Impact of data aggregation in wireless sensor networks”. In *International Workshop on Distributed Event-Based Systems*, Vienna, Austria, July 2002.

[22] Alvin Lim, "Distributed Services for Information Dissemination in Self-Organizing Sensor Networks," the *Special Issue on Distributed Sensor Networks for Real-Time Systems with Adaptive Reconfiguration*, Journal of Franklin Institute, 2001.

[23] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He, “RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks”, In *IEEE RTAS 2002*, San Jose, CA, September 2002

[24] S. R. Madden, M. J. Hellerstein, and W. Hong. “TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks”, In *Proceedings of the ACM Symposium on Operating System Design and Implementation* (OSDI), Dec. 2002.

[25] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. “The Design of an Acquisitional Query Processor for Sensor Networks.”, SIGMOD, June 2003

[26] Rex Min, Manish Bhardwaj, Seong-Hwan Cho, Amit Sinha, Eugene Shih, Alice Wang, and Anantha Chandrakasan, “An Architecture for a Power-Aware Distributed Microsensor Node”, *2000 IEEE Workshop on Signal Processing Systems (SiPS '00)*, October 2000

[27] Radhika Nagpal and Daniel Coore, “An Algorithm for Group Formation in an Amorphous Computer”, In *Proceedings of the 10th International Conference on Parallel and Distributed Computing Systems* (PDCS'98), Nevada, Oct 1998.

[28] Sagnik Bhattacharya, Hyung Kim, Shashi Prabh, Tarek Abdelzaher , “Energy-Conserving Data Placement and Asynchronous Multicast in Wireless Sensor Networks,” *The First International Conference on Mobile Systems, Applications, and Services (MobiSys)*, San Francisco, CA, May 2003.

[29] H.Takagi and L.Kleinrock. “Optimal Transmission Ranges For Randomly Distributed Packet Radio Terminals”. *IEEE Trans. on Communication*, 32(3):246-257, March 1984

[30] Y. Xu, J. Heidemann, and D. Estrin, “Geography-informed Energy Conservation for Ad Hoc Routing,” *ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2001)*, Rome, Italy, July 2001.

[31] Wei Ye, John Heidemann and Deborah Estrin, “An Energy-Efficient MAC Protocol for Wireless Sensor Networks”. In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies* (INFOCOM 2002), New York, NY, USA, June, 2002.

[32] A. Woo and D. Culler. “A Transmission Control Scheme for Media Access in Sensor Networks,” *ACM/IEEE International Conference on Mobile Computing and Networks (MobiCOM 2001)*, Rome, Italy, July 2001.

[33] Yuan Xue, Baochun Li. , “A Location-aided Power-aware Routing Protocol in Mobile Ad Hoc Networks”, in *Proceedings of IEEE Globecom 2001*, Vol. 5, pp. 2837-2841, San Antonio, Texas, November 25-29, 2001

[34] http://www.xbow.com/Products/Product_pdf_files/MICA%20data%20sheet.pdf

Range-Free Localization and Its Impact on Large Scale Sensor Networks

Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, Tarek Abdelzaher

ABSTRACT

With the proliferation of location dependent applications in sensor networks, location awareness becomes an essential capability of sensor nodes. Because coarse accuracy is sufficient for most sensor network applications, solutions in range-free localization are being pursued as a cost-effective alternative to more expensive range-based approaches. In this paper, we present APIT, a novel localization algorithm that is range-free. We show that our APIT scheme performs best when an irregular radio pattern and random node placement are considered, and low communication overhead is desired. We compare our work via extensive simulation, with three state-of-the-art range-free localization schemes to identify the preferable system configurations of each. In addition, we provide insight into the impact of localization accuracy on various location dependent applications and suggestions on improving their performance in the presence of such inaccuracy.

1. INTRODUCTION

Sensor networks have been proposed for various applications including search and rescue, disaster relief, target tracking, and smart environments. The inherent characteristics of these sensor networks make a node's location an important part of their state. For such networks, location is being used to identify the location at which sensor readings originate, (for example, identifying a target's position during tracking, providing the location of an earthquake survivor buried underneath rubble). It is also used in communication protocols that route to geographical areas instead of IDs ([18][19][21][37]), and in other location based services, such as sensing coverage [38] and location directory service [22]. In addition to the applications and protocols mentioned, continued research in WSNs will serve to invent and identify many additional protocols and applications, many of which will likely depend on location aware sensing devices.

Many localization algorithms for sensor networks have been proposed to provide per-node location information. With regard to the mechanisms used for estimating location, we divide these localization protocols into two categories: *range-based* and *range-free*. The former is defined by protocols that use absolute point-to-point distance estimates (range) or angle estimates for calculating location. The latter makes no assumption about the availability or validity of such information. Because of the hardware limitations of WSN devices, solutions in range-free localization are being pursued as a cost-effective alternative to more expensive range-based approaches.

This paper makes three major contributions to the localization problem in WSNs. First, we propose a novel range-free algorithm, called APIT, with enhanced performance under realistic system configurations. Second, though many different protocols [4][24][28] have been proposed to solve the localization problem in a range-free context, no prior work has been done to compare them in realistic settings. This paper is the first to provide a realistic and detailed quantitative comparison of existing range-free algorithms to determine the system configurations under which each is optimized. We perform such a study to serve as a guide for future research. Third, no attempt has previously been made to broadly study the impact of location error on various location-dependent applications and protocols. This paper provides insight into the effect of localization accuracy on applications and suggestions on how to improve their performance in the presence of such inaccuracy.

The remainder of the paper is organized as follows: Section 2 discusses previous work in localization for sensor networks. Section 3 describes APIT. Section 4 gives brief descriptions of three other state-of-the-art range-free protocols to which we compare our work. Section 5 describes our simulation. Section 6 follows with a detailed performance comparison of the four range-free localization algorithms described. Section 7 further investigates the impact of localization error on various location-dependent applications and protocols such as routing and target tracking. Finally, we discuss future work in Section 8 and conclude in Section 9.

2. STATE OF THE ART

Many existing systems and protocols attempt to solve the problem of determining a node's location within its environment. The approaches taken to solve this localization problem differ in the assumptions that they make about their respective network and device capabilities. These include assumptions about device hardware, signal propagation models, timing and energy requirements, network makeup (homogeneous vs. heterogeneous), the nature of the environment (indoor vs. outdoor), node or beacon density, time synchronization of devices, communication costs, error requirements, and device mobility. In this section, we discuss prior work in localization with regard to these characteristics. We divide our discussion into two subsections where we present both range-based and range-free solutions.

2.1 Range-Based Localization Schemes

Time of Arrival (TOA) technology is commonly used as a means of obtaining range information via signal propagation time. The most basic localization system to use TOA techniques is GPS [35]. GPS systems require expensive and energy-consuming electronics to precisely synchronize with a satellite's clock. With hardware limitations and the inherent energy constraints of sensor network devices, GPS and other TOA technology present a costly solution for localization in wireless sensor networks.

The Time Difference of Arrival (TDOA) technique for **ranging** (estimating the distance between two communicating nodes) has been widely proposed as a necessary ingredient in localization solutions for wireless sensor networks. While many infrastructure-based systems have been proposed that use TDOA [1][13][30], additional work such as AHLos ([32][33]) has employed such technology in infrastructure-free sensor networks. Like TOA technology, TDOA also relies on extensive hardware that is expensive and energy consuming, making it less suitable for low-power sensor network devices. In addition, TDOA techniques using ultrasound require dense deployment (numerous anchors distributed uniformly) as ultrasound signals usually only propagate 20-30 feet.

To augment and complement TDOA and TOA technologies, an Angle of Arrival (AOA) technique has been proposed that allows nodes to estimate and map relative angles between neighbors [29]. Similar to TOA and TDOA, AOA estimates require additional hardware too expensive to be used in large scale sensor networks.

Received Signal Strength Indicator (RSSI) technology such as RADAR [1] and SpotOn [17] has been proposed for hardware-constrained systems. In RSSI techniques, either theoretical or empirical models are used to translate signal strength into distance estimates. For RF systems [1][17], problems such as multi-path fading, background interference, and irregular signal propagation characteristics (shown in an empirical study of this technology [11]) make range estimates inaccurate. Work to mitigate such errors such as robust range estimation ([12]), two-phase refinement positioning ([31], [33]), and parameter calibration ([36]) have been proposed to take advantage of averaging, smoothing, and alternate hybrid techniques to reduce error to within some acceptable limit. While solutions based on RSSI have demonstrated efficacy in simulation and in a controlled laboratory environment, the premise that distance can be determined based on signal strength, propagation patterns, and fading models remains questionable, creating a demand for alternate localization solutions that work independent of this assumption.

2.2 Range-Free Localization Schemes

In sensor networks and other distributed systems, errors can often be masked through fault tolerance, redundancy, aggregation, or by other means. Depending on the behavior and requirements of protocols using location information, varying granularities of error may be appropriate from system to system. Acknowledging that the cost of hardware required by range-based solutions may be inappropriate in relation to the required location precision, researchers have sought alternate range-free solutions to the localization problem in sensor networks. These range-free solutions use only regular radio modules as basics for localization; hence, they do not incur any additional hardware cost.

In [4], a heterogeneous network containing powerful nodes with established location information is considered. In this work, anchors beacon their position to neighbors that keep an account of all received beacons. Using this proximity information, a simple centroid model is applied to estimate the listening nodes' location. We refer to this protocol as the *Centroid algorithm.*

An alternate solution, *DV-HOP* [28] assumes a heterogeneous network consisting of sensing nodes and anchors. Instead of single hop broadcasts, anchors flood their location throughout the network maintaining a running hop-count at each node along the way. Nodes calculate their position based on the received anchor locations, the hop-count from the corresponding anchor, and the average-distance per hop; a value obtained through anchor communication. Like DV-Hop, an *Amorphous Positioning* algorithm proposed in [24] uses offline hop-distance estimations, improving location estimates through neighbor information exchange.

These range-free techniques are described in more depth in section 4, and are used in our analysis for comparison with our work.

3. APIT LOCALIZATION SCHEME

In this section, we describe our novel area-based range-free localization scheme, which we call APIT. APIT requires a heterogeneous network of sensing devices where a small percentage of these devices (percentages vary depending on network and node density) are equipped with high-powered transmitters and location information obtained via GPS or some other mechanism. We refer to these location-equipped devices as **anchors**. Using beacons from these anchors, APIT employs a novel *area-based* approach to perform location estimation by isolating the environment into triangular regions between beaconing nodes (Figure 1). A node's presence inside or outside of these triangular regions allows a node to narrow down the area in which it can potentially reside. By utilizing combinations of anchor positions, the diameter of the estimated area in which a node resides can be reduced, to provide a good location estimate.

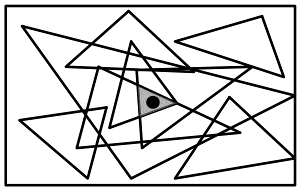


Figure 1: Area-based APIT Algorithm Overview

3.1 Main Algorithm

The theoretical method used to narrow down the possible area in which a target node resides is called the Point-In-Triangulation Test (PIT). In this test, a node chooses three anchors from all audible anchors (anchors from which a beacon was received) and tests whether it is inside the triangle formed by connecting these three anchors. APIT repeats this PIT test with different audible anchor combinations until all combinations are exhausted or the required accuracy is achieved. At this point, APIT calculates the center of gravity (COG) of the intersection of all of the triangles in which a node resides to determine its estimated position.

The APIT algorithm can be broken down into four steps: 1) Beacon exchange, 2) PIT Testing, 3) APIT aggregation and 4) COG calculation. These steps are performed at individual nodes in a purely distributed fashion. Before providing a detailed description of each of these steps, we first present the basic pseudo code for our algorithm:

```
Receive location beacons $(X_i,Y_i)$ from N anchors.
InsideSet = $\Phi$ // the set of triangles in which I reside
For (each triangle $T_i \in \binom{N}{3}$ triangles) {
    If (Point-In-Triangle-Test ($T_i$) == TRUE)
            InsideSet = InsideSet $\cup$ { $T_i$ }
    If( accuracy(InsideSet) > enough ) break;
}
/* Center of gravity (COG ) calculation */
Estimated Position = COG ( $\cap T_i \in$ InsideSet);
```

We note that the size of *InsideSet* grows cubically with the number of anchor beacons heard. For example, with 30 audible beacons in a sensor network of 1,500 nodes, the radio region will be divided by 4,060 triangles into small pieces. If the PIT tests render correct inside/outside decisions, each decision will narrow down the area in which a target node can possibly reside, making the final error small. In the next two sections, we describe the perfect PIT test and discuss the infeasibility of performing this test in a WSN. We then introduce a practical approximation to this perfect PIT test, applicable to our work.

3.2 Perfect PIT Test

In this section, we provide a perfect, albeit theoretical, solution to the following problem: For three given anchors: $A(a_x,a_y)$, $B(b_x,b_y)$, $C(c_x,c_y)$, determine whether a point *M* with an unknown position is inside triangle ΔABC or not.

Propositions I: *If M is inside triangle ΔABC, when M is shifted in any direction, the new position must be nearer to (further from) at least one anchor A, B or C.* (Figure 2A)

Proposition II: *If M is outside triangle ΔABC, when M is shifted, there must exist a direction in which the position of M is further from or closer to all three anchors A, B and C.* (Figure 2B).

Propositions I and II are intuitively correct (the formal proofs are in [14]). Accordingly, the Perfect PIT test methodology derived from propositions I and II is as follows:

Perfect P.I.T Test Theory: *If there exists a direction such that a point adjacent to M is further/closer to points A, B, and C simultaneously, then M is outside of ΔABC. Otherwise, M is inside ΔABC.*

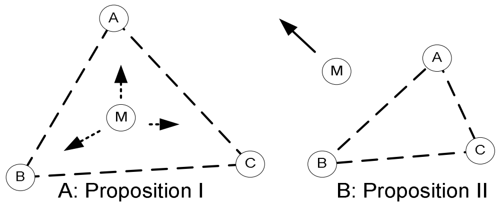


Figure 2: Propositions I and II

The Perfect P.I.T test is guaranteed to be correct in deciding whether a point M is inside triangle ΔABC. However, there are two major issues when performing this in a WSN:

- How does a node recognize directions of departure from an anchor without moving?
- How to exhaustively test all possible directions in which node M might depart/approach vertexes A, B, C simultaneously?

We address these issues in the next section.

3.3 Approximation of the Perfect PIT Test

The Perfect P.I.T. test is infeasible in practice; however, we can still obtain a very high level of accuracy by an approximation method introduced in this section.

3.3.1 Departure Test

In previous work [1][17], researchers have assumed a circular, or otherwise well-defined, mathematical or empirical model such as a log-normal attenuation model for radio propagation characteristics that describes the relationship between the signal strength degradation and the distance a radio signal travels. However, according to a recent empirical study by D. Ganesan at UCLA [11], this assumption does not hold well in practice. In our work, we make a much weaker assumption about radio propagation characteristics. We assume that in a certain propagation direction, defined to be within a narrow angle from the sending anchor (Figure 3), the received signal strength is monotonically decreasing in an environment without obstacles. This simply says that in a given direction, the further away a node is from the anchor, the weaker the received signal strength will be. Through signal strength comparisons between neighboring nodes, this assumption allows a node to determine whether a neighboring node is closer to a given anchor.

Departure Test Definition: *Test whether M is further away from anchor A than N.*

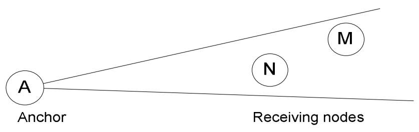


Figure 3: Departure Test

In addition to gathering evidence drawn from prior empirical studies of WSNs [11], we checked the validity of our assumption on Berkeley's MICA mote testbed in an obstruction free laboratory environment. In this experiment, we incrementally increased the distance between sending (anchor) and receiving motes. Figure 4 shows the measured signal strength of 40 beacons from a single anchor at varying distances.

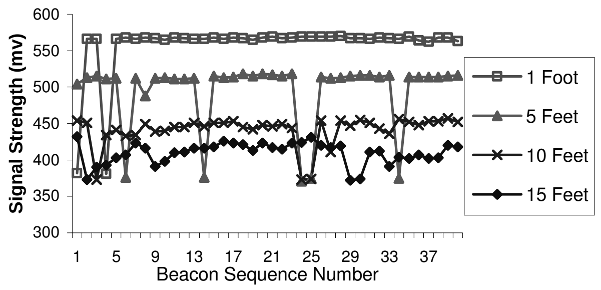


Figure 4: Signal Strength at Different Distances

We conclude from Figure 4 that our assumption of monotonically decreasing signal strength in a given direction is usually valid. For example, the signal strength readings shown in Figure 4 are usually about 560 mv at one-foot, and about 510 mv at five-feet. However, we note that there are various points on the graph where this signal strength property is violated due to burst disturbance effects. Two approaches to minimize the effect of such disturbances include taking a running average of the signal strength over time and using our robust aggregation, a technique discussed further in section 3.4.

It should be noted that our scheme does not make any assumptions about the correlation between *absolute* distance and signal strength; hence, we consider our scheme a range-free solution. More importantly, though we use radio signal comparisons throughout the paper, our scheme can actually work with any system, so long as it can support a form of the departure test. For example by using the hop counts.

3.3.2 Approximate PIT Test

To perform PIT testing in sensor networks without requiring that nodes move, we define an Approximate PIT Test (APIT) that takes advantage of the relatively high node density of these networks (usually with connectivity above 6). The basic idea behind the APIT test is to use neighbor information, exchanged via beaconing, to emulate the node movement in the Perfect PIT test. The APIT test is formally described below.

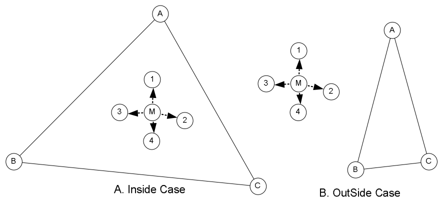


Figure 5: Approximate P.I.T Test

Approximate P.I.T Test: *If no neighbor of M is further from/closer to all three anchors A, B and C simultaneously, M assumes that it is inside triangle ∆ABC. Otherwise, M assumes it resides outside this triangle.*

We further explain the APIT test through an example. Figure 5A presents a scenario where none of M's neighbors, 1, 2, 3 or 4, is further from/closer to all three anchors A, B and C simultaneously. In this scenario, M will assume that it is inside the triangle ∆ABC according to the definition. The other scenario is shown in Figure 5B, where neighbor 3 will report to node M that it is further away from A, B, and C than M. This allows M to assume it resides outside of triangle ∆ABC.

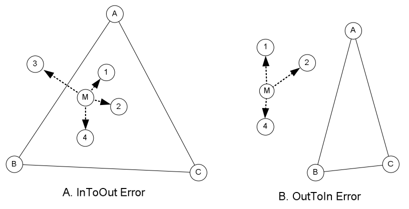


Figure 6: Error Scenarios for the APIT Test.

Because APIT can only evaluate a finite number of directions (the number of neighbors), APIT can make an incorrect decision. The two scenarios where incorrect decisions are made are depicted in Figure 6. In Figure 6A, we show what we deem *InToOut* error, where the node is inside the triangle, but concludes based on the APIT test that it is outside the triangle. This can happen when M is near the edge of the triangle, while some of M's neighbors (3 in this case) are outside the triangle and further from all points ABC, in relation to node M. As a result, M mistakenly thinks it is outside of triangle ABC due to this **edge effect**. On the other hand, the **irregular placement** of neighbors can result in *OutToIn* error. Figure 6B depicts a scenario where M is outside of triangle ABC and none of its neighbors is further from/closer to all three anchors, A, B and C, simultaneously. This makes M mistakenly assume it is inside triangle ABC.

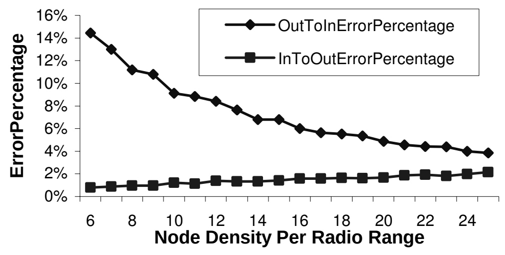


Figure 7: APIT Error under Varying Node Densities

Fortunately, from experimentation, we find that the percentage of APIT tests exhibiting such an error is relatively small (14% in the worst case). Figure 7 demonstrates this error percentage as a function of node density. When node density increases, APIT can evaluate more directions, considerably reducing *OutToInError* (Figure 6B). On the other hand, *InToOutError* will slightly increase due to the increased chance of **edge effects**.

3.4 APIT Aggregation

Once the individual APIT tests finish, APIT aggregates the results (inside/outside decisions among which some may be incorrect) through a grid SCAN algorithm (Figure 8). In this algorithm, a grid array is used to represent the maximum area in which a node will likely reside. In our experiments, the length of a grid side is set to 0.1R, to guarantee that estimation accuracy is not noticeably compromised.

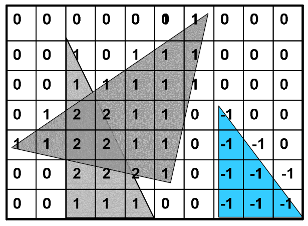


Figure 8: SCAN Approach

For each APIT **inside decision** (a decision where the APIT test determines the node is inside a particular region) the values of the grid regions over which the corresponding triangle resides are incremented. For an **outside decision,** the grid area is similarly decremented. Once all triangular regions are computed, the resulting information is used to find the maximum overlapping area (e.g. the grid area with value 2 in Figure 8), which is then used to calculate the center of gravity for position estimation.

The pseudo code for APIT aggregation is as follows:

```
For (each triangle $T_i \in \binom{N}{3}$ triangles) {
    If (APIT($T_i$) == Out ) AddNegativeTriangle($T_i$);
    If (APIT($T_i$) == In ) AddPositiveTriangle($T_i$);
};
Find the area with Max values;
```

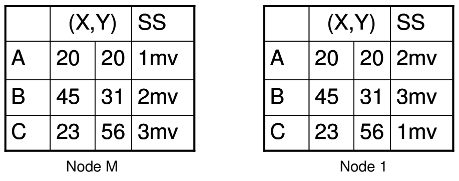
APIT aggregation is a robust approach that can mask errors in individual APIT tests. As we know from Figure 7, the majority (more than 85% in the worst case) of APIT tests are correct. With limited error, the correct decisions build up on the grid and the small number of errors only serves as a slight disturbance to the final estimation.

If the maximum range of an anchor node is known, we can filter out the grid points, which are out of range of any anchors heard by this node before we run SCAN algorithm. This leads to better localization accuracy and less memory requirement.

3.5 A Walk through the APIT Algorithm

In this section, we present an example to further explain our APIT algorithm.

1. Having received beacons from anchors A, B, and C, each node maintains a table (Anchor ID, Location, Signal Strength) for each anchor heard (Figure 9).



	(X,Y)		SS
A	20	20	1mv
B	45	31	2mv
C	23	56	3mv

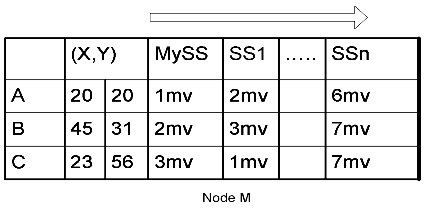
Node M

	(X,Y)		SS
A	20	20	2mv
B	45	31	3mv
C	23	56	1mv

Node 1

Figure 9: Table of heard Anchors

2. Each node beacons once to exchange anchor tables with its neighbors. These tables are merged at every node to maintain neighborhood state (Figure 10).



	(X,Y)		MySS	SS1	…..	SSn
A	20	20	1mv	2mv		6mv
B	45	31	2mv	3mv		7mv
C	23	56	3mv	1mv		7mv

Node M

Figure 10: Combined Table

3. APIT runs on every column of the node's table to determine whether a neighboring node exists that has consistently larger/smaller signal strengths from the three anchors A, B and C[1]. If such a neighbor is found, M assumes that it is outside triangle ABC. If no such neighbor is found, M assumes it is inside this region.
4. Each node repeats step 3 for varying combinations of three anchors. (Note: we only demonstrate 1 combination of three anchors in this example).
5. The algorithm described in Section 3.4 is then used to determine the area with maximum overlap.
6. Finally, the center of gravity of this area is used as the final location estimation.

3.6 APIT Performance Analysis

We consider a static senor network with N anchors and M nodes. Since APIT requires each anchor and node to broadcast once, the communication overhead of our APIT algorithm is $N+M$ under collision-free situation. We have proven (see authors for proof) that if a target node can receive beacons from K anchors, the maximum number of polygons partitioned by these anchors can be achieved by placing all anchors on a convex curve. This anchor placement creates $(K-1)(K-2)/2 + K(K-1)(K-2)(K-3)/24$ partitions. Assuming the nominal anchor radio range is R, the average size of each partition is then:

$$\frac{\pi R^2}{(K-1)(K-2)/2 + K(K-1)(K-2)(K-3)/24}$$

It should be noted that the above formula only indirectly reflects the upper bound performance of the Perfect PIT test. APIT has less accuracy due to approximation as we will show in our evaluations.

By using our SCAN algorithm during APIT aggregation, we bound the computational complexity of the APIT algorithm by O(L) (L is the number of APIT tests and each test only requires several comparisons). If we use a geometric algorithm to perform APIT aggregation precisely, the computational complexity will be O(L^2). In order to perform SCAN algorithm, each node keeps a bitmaps (Figure 8)

In a mobile sensor network, periodic beaconing is a straightforward solution to maintain the current anchor and node positions. A more sophisticated method to minimize localization cost under such a network is left as future work.

3.7 Key Observations

We note several key observations here to justify the use of our APIT algorithm in sensor networks.

- Redundancy and high node density are the key positive characteristics of sensor networks over traditional ad hoc networks. By exploiting this redundancy, aggregated decisions can provide good accuracy during location estimation, regardless of the fact that information obtained by an individual test is coarse and error prone.
- In order to obtain high redundancy without increasing deployment costs, we can use a single moving anchor that sends out beacons at different locations to localize all nodes inside a sensor network.

[1] No P.I.T. test is performed when neighboring nodes do not share three common anchor points.

4. RANGE-FREE SCHEMES

In this section, we briefly describe the key features of three state-of-the-art range-free localization algorithms studied in our simulation. These algorithms are implemented in accordance with the published design; with the exception of a few enhancements, made to ensure that our comparison is as fair as possible. The protocols discussed include:

- **Centroid Scheme** [4] by N.Bulusu and J. Heidemann
- **DV-Hop Scheme** [28] by D.Niculescu and B. Nath
- **Amorphous Scheme** [24] [25] by R. Nagpal

In addition to the aforementioned range-free algorithms, we implement an oracle version of APIT that uses the Perfect PIT Test defined in Section 4.2. For completeness, we provide brief descriptions of these algorithms. More details can be found in [4], [24], and [28].

4.1 Centroid Localization

N. Bulusu and J. Heidemann [4] proposed a range-free, proximity-based, coarse grained localization algorithm, that uses anchor beacons, containing location information (X_i,Y_i), to estimate node position. After receiving these beacons, a node estimates its location using the following centroid formula:

$$(X_{est}, Y_{est}) = \left(\frac{X_1 + \cdots + X_N}{N}, \frac{Y_1 + \cdots + Y_N}{N} \right)$$

The distinguished advantage of this Centroid localization scheme is its simplicity and ease of implementation. In a later publication [5], N. Bulusu augments her work by suggesting a novel density adaptive algorithm (HEAP) for placing additional anchors to reduce estimation error. Because HEAP requires additional data dissemination and incremental beacon deployment, while other schemes under consideration only use ad hoc deployment, we do not include this later work in our simulations.

4.2 DV-Hop localization

DV-Hop localization is proposed by D. Niculescu and B. Nath in the Navigate project [27]. DV-Hop localization uses a mechanism that is similar to classical distance vector routing. In this work, one anchor broadcasts a beacon to be flooded throughout the network containing the anchors location with a hop-count parameter initialized to one. Each receiving node maintains the minimum counter value per anchor of all beacons it receives and ignores those beacons with higher hop-count values. Beacons are flooded outward with hop-count values incremented at every intermediate hop. Through this mechanism, all nodes in the network (including other anchors) get the shortest distance, in hops, to every anchor. The hop count for a single anchor A, generated by simulation, is shown in Figure 11.

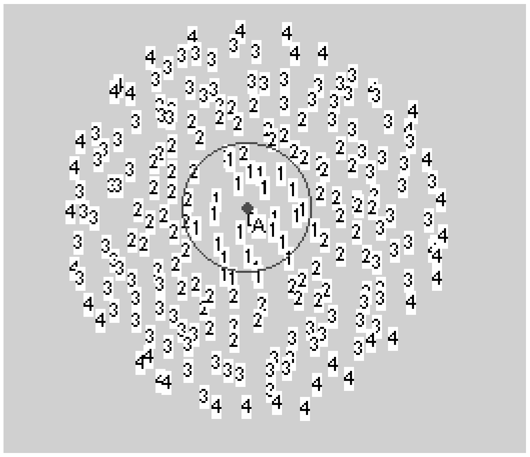


Figure 11: Anchor Beacon Propagation Phase

In order to convert hop count into physical distance, the system estimates the average distance per hop without range-based techniques. Anchors perform this task by obtaining location and hop count information for all other anchors inside the network. The average single hop distance is then estimated by anchor *i* using the following formula:

$$HopSize_i = \frac{\sum \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum h_j}$$

In this formula, (x_j,y_j) is the location of anchor *j*, and h_j is the distance, in hops, from anchor *j* to anchor *i*. Once calculated, anchors propagate the estimated HopSize information out to the nearby nodes.

Once a node can calculate the distance estimation to more than 3 anchors in the plane, it uses triangulation (multilateration) to estimate its location. Theoretically, if errors exist in the distance estimation, the more anchors a node can hear the more precise localization will be.

4.3 Amorphous localization

The Amorphous Localization algorithm [24][25], proposed independently from DV-Hop, uses a similar algorithm for estimating position. First, like DV-Hop, each node obtains the hop distance to distributed anchors through beacon propagation.

Once anchor estimates are collected, the hop distance estimation is obtained through local averaging. Each node collects neighboring nodes' hop distance estimates and computes an average of all its neighbors' values. Half of the radio range is then deducted from this average to compensate for error caused by low resolution.

The Amorphous Localization algorithm takes a different approach from the DV-Hop algorithm to estimate the average distance of a single hop. This work assumes that the density of the network, n_{local}, is known *a priori,* so that it can calculate HopSize offline in accordance with the Kleinrock and Silvester formula [20]:

$$HopSize = r(1 + e^{-n_{local}} - \int_{-1}^{1} e^{-\frac{n_{local}}{\pi}\left(\arccos t - t\sqrt{1-t^2}\right)} dt)$$

Finally, after obtaining the estimated distances to three anchors, triangulation is used to estimate a node's location.

4.3.1 Amorphous Localization Enhancement[2]

By using only three anchors, Nagpal suggests in [24] a critical minimum average neighborhood size of 15, imposed to obtain good accuracy. As shown in the APIT algorithm, increasing estimation redundancy reduces estimation error. We, therefore, argue that the same design philosophy can be applied to [24]. By increasing the number of anchors used in their estimation, we can effectively reduce the critical minimum average neighborhood requirement from 15 nodes per communication area, to 6, under uniform node placement (Figure 12) without reducing estimation accuracy (this number would be 8 for random node placement).

This enhancement uses work done by Jan Beutel [2] in the Picoradio Project at UC Berkeley. A minimum mean square error (MMSE) algorithm triangulates node positions based on the locations of multiple anchors (in this case more than 3), and associates distances between each anchor and the target node.

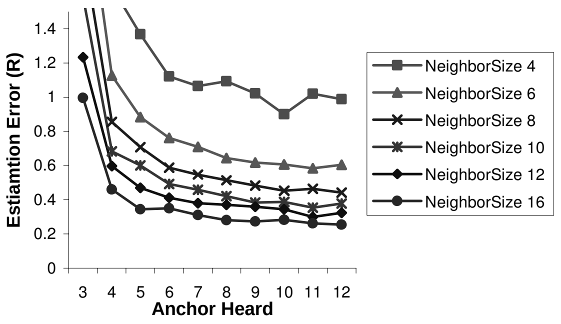


Figure 12: Phase Transition in the DV-Based Algorithm

Using this enhancement, we show that the Amorphous algorithm can actually work in a sparsely connected network. Increasing the number of anchors participating in multilateration can dramatically reduce the required level of network connectivity. In Figure 12, we see that when 3 anchors are used, the estimation error (normalized to units of node radio range R) is large, regardless of the level of connectivity. By increasing the number of anchors to 5, we obtain better precision than that with 3 anchors, when the levels of connectivity as low as 6.

More importantly, Figure 12 shows two kinds of phase transitions that occur. First, when the neighbor size exceeds 8, increasing the number of anchors participating in multilateration brings down the estimation error below half of the radio range, a bound tolerated by the applications we studied in section 7. Second, the estimation accuracy increases dramatically as the number of anchors heard increases to 6. However, after that, continuing to increase the number of anchors heard only slightly increases precision. In accordance with Figure 12, for DV-based algorithms, in order to confine the average estimation error to reside within half of the radio range, we suggest that both the neighborhood size, and the number of anchors used in multilateration, remain about 8~10. We argue that it is not quite cost-effective to further increase node density or the number of anchors used in multilateration for better accuracy after these phase transition points.

[2] A recent publication [25] in ISPN'03 by Nagpal etc. makes a similar enhancement to the one we propose here.

4.4 Perfect PIT algorithm

As previously mentioned, the precision of our APIT algorithm is highly dependent on the correctness of the APIT Test. To obtain boundary conditions for a best estimate in our localization scheme, we simulate a perfect PIT algorithm that utilizes an oracle. This oracle can guarantee correctness when determining whether a node resides within the triangular region created by the three anchors. We use this as a precise bound on our APIT algorithm

5. SIMULATION SETTINGS

This section describes the simulation settings we use in our evaluation.

5.1 Radio Model

Some previous work in localization assumes that a perfect circular radio model exists. As stated before, empirical studies [11] on real testbeds have shown that this assumption is invalid for WSNs. To ensure that our evaluation is as true to reality as possible, we use a more general radio model in our evaluation. Specifically, we assume a model with an upper and lower bound on signal propagation (Figure 13). Beyond the upper bound, all nodes are out of communication range; and within the lower bound, every node is guaranteed to be within communication range. If the distance between a pair of nodes is between these two boundaries, three scenarios are possible: 1) symmetric communication. 2) uni-directional asymmetric communication, and 3) no communication.

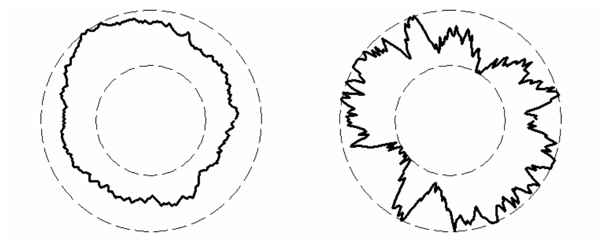


Figure 13: Irregular Radio Pattern

The parameter DOI is used to denote the irregularity of the radio pattern. It is defined as the maximum radio range variation per unit degree change in the direction of radio propagation. When the DOI is set to zero, there is no range variation, resulting in a perfectly circular radio model. To get a better idea of how this DOI parameter affects signal propagation characteristics, Figure 13 shows the radio patterns generated in simulation with DOI values set to 0.05 and 0.2 respectively. To investigate how well our model resembles the reality in sensor motes. We measure the communication range of a MICA mote as the receiver direction varies from 0 degrees to 360 degrees. The two communication ranges are got when received signal strength threshold is set to -55.5 dBm and -59 dBm, respectively. The radio patterns are shown in Figure 14. These patterns give us the measured DOI values of 0.12 and 0.09 for two received signal thresholds, respectively.

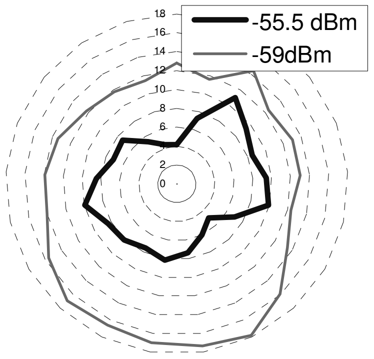


Figure 14: Radio Pattern from MICA2

5.2 Placement Model

In our simulations, nodes and anchors are distributed in a rectangular terrain in accordance with predefined densities. Two common placement strategies are investigated, namely random and uniform.

- Random placement: it distributes all nodes and anchors randomly throughout the terrain.
- Uniform placement: the terrain is partitioned into grids and nodes and anchors are evenly divided amongst these grids (random distribution inside each grid).

5.3 System Parameters

In our experiments, we study several system-wide parameters that we feel directly affect estimation error in range-free localization algorithms. A description of these parameters follows:

- Node Density (ND): Average number of nodes per node radio area.
- Anchors Heard (AH): Average number of Anchors heard by a node and used during estimation.
- Anchor to Node Range Ratio (ANR): The average distance an anchor beacon travels divided by the average distance a regular node signal travels. When this value equals one, the anchor and nodes have the same average radio range. The larger this value, the fewer anchors required to maintain a desired AH value.
- Anchor Percentage (AP): The number of anchors divided by the total number of nodes (1000~3000 nodes). This value can be derived from the three parameters described above using the formula: $AP=AH/(AH+ND*ANR^2)$.
- Degree of Irregularity (DOI): DOI is defined in section 5.1 as an indicator of radio pattern irregularity.
- GPS Error: In reality, GPS equipped anchors will render imprecise readings. In our evaluation, this parameter is defined as the maximum possible distance from the real anchor position to the GPS estimated anchor position in units of node radio range (R).
- Placement: Random and Uniform node/anchor placements are investigated in the evaluation.

In the evaluation, all distances including error estimation are normalized to units of node radio range (R) to ensure generally applicable results.

5.4 A Note about Comparisons

The range-free localization algorithms studied in this paper share a common set of system parameters, and most of them are defined in a consistent way across the algorithms we analyze. However, due to different anchor beacon propagation methods utilized in different algorithms, the Anchor to Node Range Ratio (ANR) parameter varies between algorithms. In the Centroid and APIT algorithms, direct communication between anchors and **target nodes** (nodes attempting to determine their location) is used. In this case, ANR is set to the physical radio range ratio between anchor and target nodes. In the Amorphous and DV-Hop algorithms studied, the physical radio range of anchors is the same as that of target nodes, and the ANR is set to the distance an anchor beacon can propagate in units of node radio range (R). In our evaluation, we indicate any performance implications that result from this implementation difference.

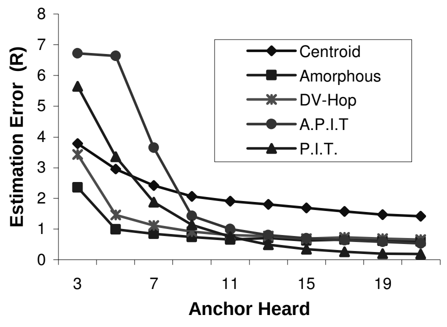
6. EVALUATION

This section provides a detailed quantitative analysis comparing the performance of the range-free localization algorithms described in Sections 3 and 4. The obvious metric for comparison when evaluating localization schemes is location estimation error. We have conducted a variety of experiments to cover a wide range of system configurations including varying 1) anchor density, 2) target node density, 3) radio range ratio (ANR), 4) radio propagation patterns, and 5) GPS error. Because communication can have a significant impact on sensor network systems with low bandwidth, we also use communication overhead, in terms of number of beacons exchanged, as a telling secondary metric to evaluate the cost and performance of the localization schemes studied.

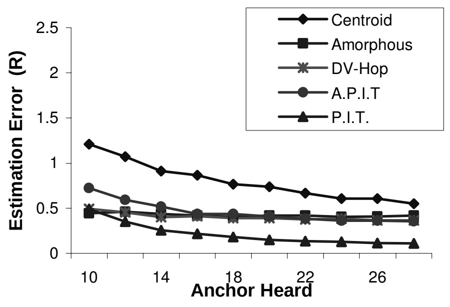
Outside of studying the effect of certain parameters on localization error, we use default values of AH=16, ND=8, and ANR=10 (Anchor Percentage = 2%) in most of our experiments. These settings are in line with our expectation of future sensor network technology and facilitate comparisons between figures. In all of our graphs, each data point represents the average value of 600 trials with different random seeds and the 90% confidence intervals for the data are within 5~10% of the mean shown. We note that for legibility reasons, we do not plot these confidence intervals in this paper. Full experimental data can be obtained from the authors upon request.

6.1 Localization Error when Varying AH

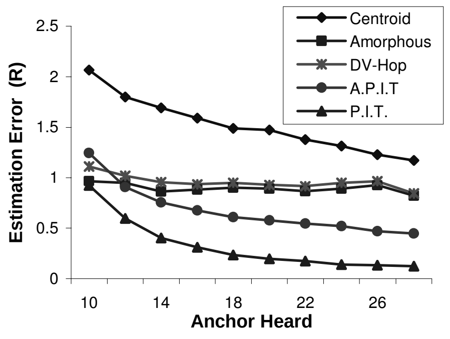
In this experiment, we analyze the effect of varying the number of anchors heard (AH) at a node to determine its effect on localization error.



A. AH=3~21, DOI=0, ANR = 10, ND = 8, Random



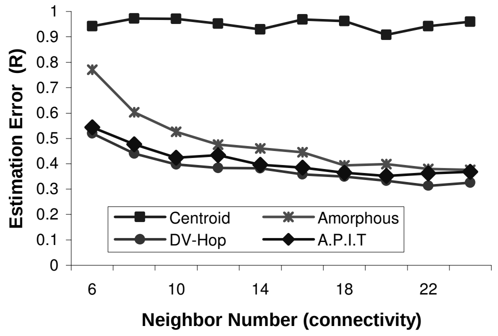
B. AH=10~28, DOI=0, ANR = 10, ND = 8, Uniform



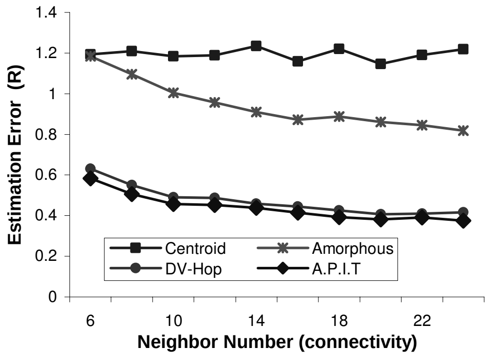
C. AH=10~28, DOI=0, ANR = 10, ND = 8, Random

Figure 15: Error Varying AH

Figure 15A shows that the overall estimation error decreases as the number of anchors heard increases. However, it is important to note that different algorithms transition at different points in the graph. For example, the Amorphous and DV-Hop schemes improve rapidly when AH is below 7, and are nearly insensitive to the addition of anchors above 7. In contrast, the precision of APIT and the Centroid localization scheme constantly improve as AH is increased (Figure 15B and Figure 15C). Our APIT algorithm performs worse than the Centroid algorithm when AH is below 8 due to the fact that the diameter of the divided area is not small enough. This effect is significantly reduced by increasing AH values. For larger AH values, APIT consistently outperforms the Centroid scheme. Figure 15B extends AH to higher values in order to show estimation error below 0.6 R. We note that our APIT algorithm requires only 12 anchors to reach the 0.6R level while the Centroid scheme requires 24. Finally, Figure 15C presents the same experimental results for random node placement. By comparing graphs B (uniform placement) and C (random placement), we show that the DV-Based algorithm is more sensitive to irregular node placement than both APIT and the Centroid scheme. This is mainly due to the fact that *HopSize* estimation in the DV-Hop and Amorphous schemes, is less precise in non-isotropic deployment.



A. DOI=0.1, ANR = 10, AH=16, Uniform



B.DOI=0.2, ANR = 10, AH=16, Uniform

Figure 16: Error Varying ND

6.2 Localization Error when Varying ND

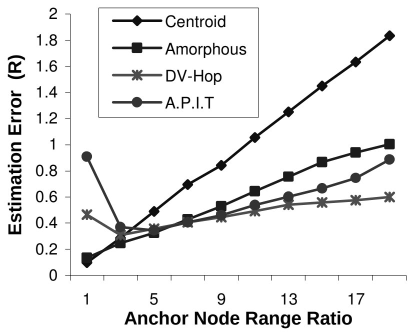
Figure 16 explores the effect of node density (ND) on the localization estimation accuracy. For all but the Centroid algorithm, localization error decreases as the number of neighbors increases. Since there is no interaction between nodes in the Centroid algorithm, we see nearly constant results while varying ND. However, due to its relatively simple design, the Centroid localization scheme does not perform as well as the others.

Because the offline estimation of *HopSize* in the Amorphous algorithm has large error when the node density is small, the estimation error is large when the node density is below 10. APIT and DV-Hop however, are robust to varying ND, and produce good results as long as the neighbor density remains above 6. By comparing Figure 16A (DOI=0.1) and Figure 16B (DOI=0.2), we show that the DV-Based algorithms, especially the Amorphous algorithm, are more sensitive to irregular radio patterns than the APIT scheme. This is mainly due to the fact that *HopSize* estimation in the previous schemes is less precise in the presence of irregular radio patterns. However, it should be noted that DV-Hop abates this error by online estimation.

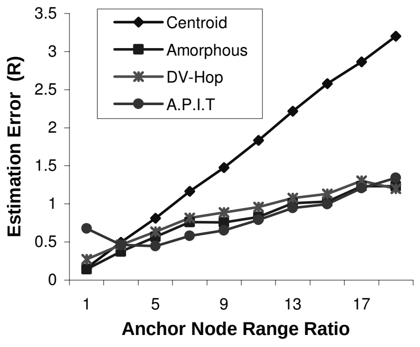
6.3 Localization Error when Varying ANR

Section 6.1 demonstrated that a large number of anchors are desired for good estimation results. The cost of having such a large percentage of anchors can be ameliorated by increasing the anchor radio range to which beacons travel. This happens because larger beacon propagation distances mean less anchors required to achieve the same AH value. For example, if an algorithm requires AH equal to the neighborhood node density (ND), we need 50% of the nodes to be anchors when the ANR equals one. By increasing the ANR by a factor of 10, we can reduce the required anchor percentage to only 1%.

The implication of this solution, as shown in Figure 17, is that estimation error increases as ANR increases. This occurs because larger beacon propagation distances result in larger accumulated error. We note from Figure 17 that while all algorithms possess this relationship, the estimation error of the Centroid algorithm increases more significantly with increased ANR, in comparison to the other three algorithms. However, we also note that when the ANR is smaller than 3, APIT has a large InToOutErrorRatio due to the edge effect (described in Section 3.3.2). In this system configuration, a Centroid algorithm has its advantages.



A. ND = 8, AD=16, DOI = 0.1, Uniform



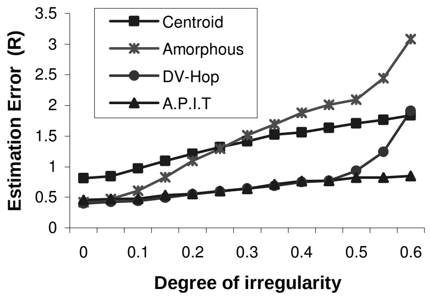
B. ND = 8, AD=16, DOI = 0.1, Random

Figure 17: Error under Different ANR

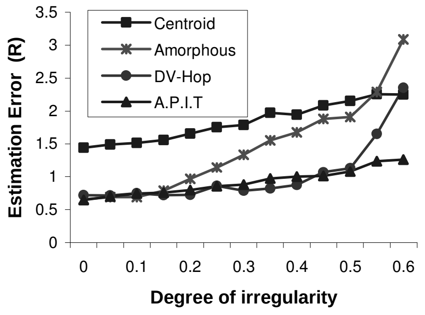
From an alternate perspective, we show that we can increase accuracy by using a smaller ANR. For example, the estimation error, shown in previous sections, can be reduced by about 30~50% when we use an ANR value of 5 instead of 10. However, this will increase the anchor percentage (AP) from 2% to 8%, requiring that more anchors be deployed.

6.4 Localization Error when Varying DOI

In this experiment, we investigate the impact of irregular radio patterns on the precision of localization estimation. It is intuitive that irregular radio patterns can affect the network topologies resulting in irregular hop count distributions in the Amorphous and DV-Hop algorithms. The HopSize formula, used in the Amorphous algorithm, assumes that radio patterns are perfectly circular. We can see, in Figure 17, how this inaccurate estimate directly contributes to localization error as the DOI increases. In contrast, the DV-Hop scheme estimates HopSize using online information exchanged between anchors. This results in much better performance than the Amorphous algorithm, even though they are both DV-Based algorithms. Because the Centroid and APIT algorithms do not depend on hop-count and HopSize estimations, and because the effect of DOI is abated by the aggregation of beaconed information, these algorithms are more robust than the Amorphous algorithm.



A. ANR = 10, ND = 8, AH=16, Uniform

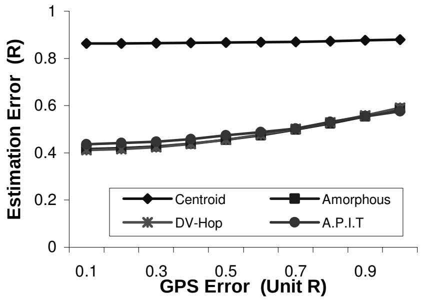


B. ANR = 10, ND = 8, AH=16, Random

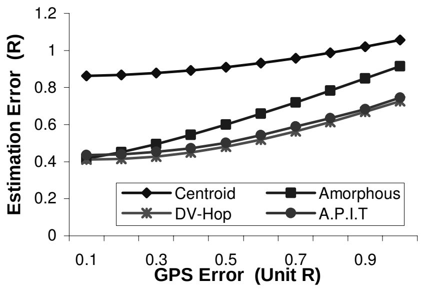
Figure 18: Error under Varying DOI

6.5 Localization Error when Varying GPS Error

In other experiments, we consider the distinct possibility that the GPS or an alternative system, which provides anchor nodes with location information, is error prone. Figure 19A and B demonstrate how initial location error at anchors directly affects the error of the range-free localization protocols studied. In general, in all four schemes GPS error is abated considerably by utilizing location information from multiple anchors. In the random error case (Figure 19A), we assume GPS error is isotropic; that is, the estimation error can occur in any direction. In this situation, the error impact of GPS is very small. We also see (Figure 19B) that when GPS error is biased (skewed in a particular direction) due to non-random factors, the estimation error of all schemes increases at a much slower rate than GPS error due to aggregation.

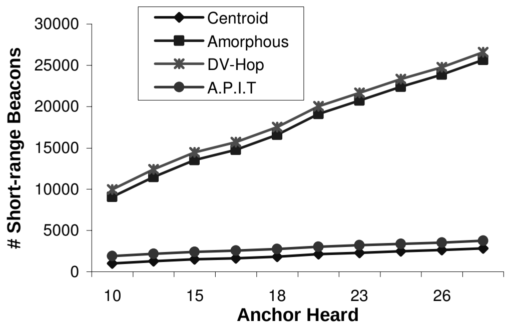


A.ANR = 10, ND = 8, AH=16, Uniform, Random Error



B. ANR = 10, ND = 8, AH=16, Uniform, Bias Error

Figure 19: Error under Different GPS Error



ANR=10, ND = 8, DOI = 0.1, Uniform

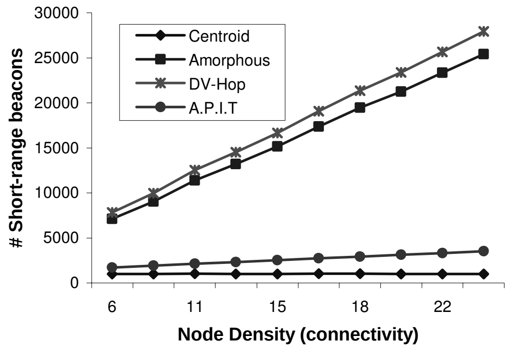
Figure 20: Communication Overhead for Varied AH

6.6 Communication Overhead for Varied AH

Figure 20 shows the results of experiments that test the communication overhead with regard to AH. It is important to note that the Centroid and APIT schemes use *long-range* anchor beacons, while the Amorphous and DV-hop algorithms use *short-range* beacons. Considering that energy consumption quadratically increases with increased beacon range, in Figure 20 we equate one long-range beacon to ANR^2 short-range beacons. This means that one long-range beacon sent out by APIT is counted as 100 short-range beacons when ANR = 10. Figure 20 shows that without flood-based beacon propagation, the Centroid and APIT algorithms use much fewer beacons than DV-based algorithms. For example, the APIT algorithm uses only about 10% of the beacons that the DV-Hop scheme uses when AH is set to 16.

Figure 20 also shows that APIT requires more beacons than the Centroid algorithm because of the neighborhood information exchange. In addition, DV-Hop requires more beacons than the Amorphous algorithm because of additional online HopSize estimation requirements.

It should be noted that the evaluation of communication overhead here assumes a collision-free environment. If taking the collision into account, we expect that Amorphous and DV-hop algorithms introduce even more control overhead because of the flooding required by those two schemes.



ANR=10, AH = 16, DOI = 0.1, Uniform

Figure 21: Overhead for Varied Node Density

6.7 Communication Overhead for Varied ND

Figure 21 demonstrates the effect of neighborhood density on required communication for localization. We can see from this graph that because there is no interaction between nodes in the Centroid scheme, the overhead stays constant. Communication overhead in our APIT scheme does increase with increased node density; however, it does so at a much lower rate than the DV-based schemes.

Drawing conclusions from Figure 20 and Figure 21, we argue that as far as the communication overhead is concerned, the DV-Hop and Amorphous schemes are less suitable solutions for sensor networks with limited bandwidth when compared to the APIT and Centroid schemes. This is due to the large number of beacons required in these schemes.

6.8 Computational Overhead

The predominant concerns about sensor network protocols are the communication and power consumption overhead. However, it is desirable to evaluate the computational overhead of each algorithm. The major complexity of APIT algorithm is from the intersection of overlapped triangles. This has been discussed in Section 3.6. DV-Hop and Amorphous localization use multilateration to estimate nodes' locations. Their overheads are relatively smaller. Centroid algorithm only uses a simple averaging function, thus it has the smallest computation overhead.

6.9 Evaluation Summary

In addition to the experiments previously discussed, we have conducted a variety of experiments to cover a varying range of system configurations. These experiments help us better understand the situations where the different localization schemes considered are more or less appropriate than one another.

Table 1 Performance and requirements summary

	Centroid	**DVHop**	**Amorp.**	**APIT**
Accuracy	Fair	Good	Good	Good
NodeDensity	>0	>8	>8	>6
AnchorHeard	>10	>8	>8	>10
ANR	>0	>0	>0	>3
DOI	Good	Good	Fair	Good
GPSError	Good	Good	Fair	Good
Overhead	Smallest	Largest	Large	Small

Table 1 provides an overview of our results, and it can be used as a design guide for applying range-free schemes in WSN systems. This table shows that no single algorithm works best under all scenarios, and that each localization algorithm has preferable system configurations. Though the Centroid scheme has the largest estimation error, its performance remains independent of node density and it boasts the smallest communication overhead and simplicity of implementation. Although DV-Hop requires more communication beacons to perform online estimation, it is notably more robust than the Amorphous algorithm in HopSize estimation. Finally, our APIT algorithm trumps the other algorithms when an irregular radio pattern and random node placement are considered, and low communication overhead is desired. However, we acknowledge that APIT has more demanding requirements for both ANR values and the number of anchors used in localization.

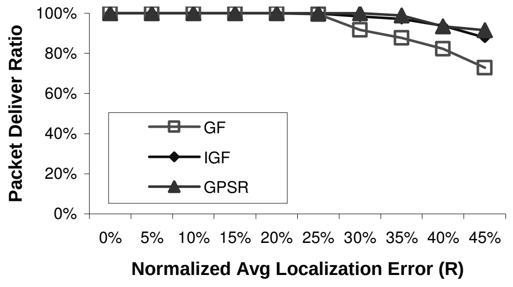
7. LOCALIZATION ERROR IMPACT

In localization for WSNs, achieving better results (usually with regard to location accuracy) requires increasing the relative cost of the localization scheme via additional hardware, communication overhead, or the imposition of constraints and system requirements. Although more accurate location information is preferable, the desired level of granularity should depend on a cost/benefit analysis of the protocols that utilize this information. In this section, we investigate four types of location dependent applications, namely, 1) location-based routing, 2) target estimation, 3) target tracking and 4) sensing coverage. Based on the results, we conclude that except the routing in sparse networks, range-free localization schemes are able to support these sensor network applications sufficiently with only slight performance degradation.

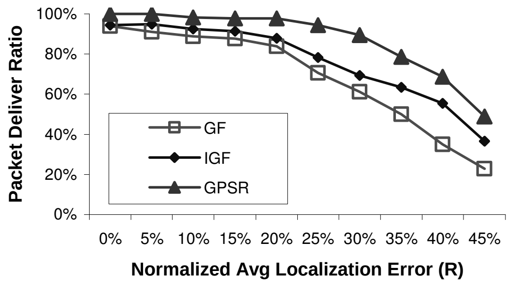
7.1 Routing Performance

A localization service is critical for location-based routing protocols such as GF [26], GPSR [19], IGF [15], LAR [21] and GAF [37]. In these protocols, individual nodes make routing decisions based on knowledge of their geographic location. While most work in location-based routing assumes perfect location information, the fact is that erroneous location estimates are virtually impossible to avoid. Problems arise as error in the location service can influence location-based routing to choose the best next hop (the neighbor closest to the destination), or can make a node inadvertently think that the packet could not be routed because no neighbors are closer to the final destination.

To investigate the impact of localization error on routing, we studied three routing protocols GF [26], GPSR [19] and IGF [15] under the low traffic network conditions so that network congestion does not influence our results. In the experiments, localization errors are uniformly distributed in [0, 2×Avg Localization Error], and the localization errors are normalized to units of node radio range (R) to ensure generally applicable results.

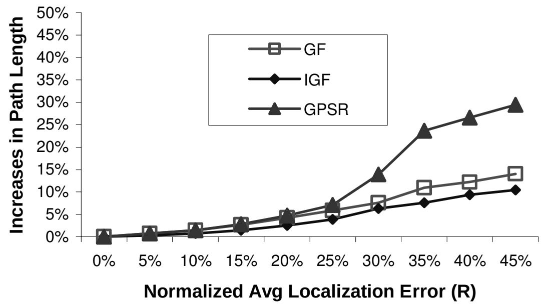


A: High-density scenario (22 node/radio range)

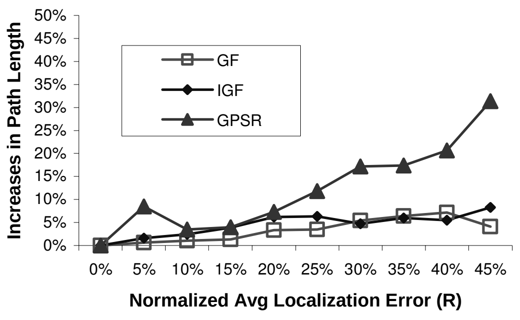


B: Low density scenario (8 node/ radio range)

Figure 22: Delivery ratio with varied localization errors



A: High-density scenario (22 node/radio range)



B: Low density scenario (8 node/ radio range)

Figure 23: Increase in Path length overhead with different localization errors

We investigate both high-density (22 nodes per radio range) and low-density scenario (8 nodes per radio range).

In the experiment, we increase the average localization error from 0% to 50% of the radio range in steps of 5% to measure the end-to-end delivery ratios and the percentage of increase in path length due to the localization errors. Two observations can be made about the impact of localization error to routing algorithms. First, when node density is high, the localization impact is relatively small. For example, all algorithms achieve 100% delivery ratio with moderate localization error (< 25%). GPSR and IGF achieves 95% delivery ratio under about half radio range error. However, when node density is small, location-based routings suffer a lot. For example, though GPSR can deal with void, however it can only delivery 50% packets when the localization error is about half radio range. This phenomenon suggests that ID-based routing should be used in sparse sensor network, if we cannot obtain very precise localization results.

Second, the path length increases moderately with the increase of the localization errors. The localization impact to the path length of GPSR is higher than other protocols, because GPSR uses the perimeter forwarding if the greedy forwarding fail due to the localization errors. We also note that the path length overhead due not affect much by the network node density, a fact that was not true for packet delivery ratio metric.

7.2 Target Estimation Performance

Many of the most frequently proposed applications for WSNs utilize target position estimations for tracking, search and rescue, or other means. In these proposed applications, when a target is identified, some combination of the nodes that sensed that target report their location to a centralized node (leader or base station). This node then performs aggregation on the received data to estimate the actual location of the target. Because target information could be used for locating survivors during a disaster, or identifying an enemy's position for strategic planning, the accuracy of this estimation is crucial to the application that uses it. Note that nodes normally have different sensing ranges with different sensing devices, and sensing ranges are normally different from communication radio range. For general applicability, in our following experiments, we use sensing density (the number of nodes per sensing range) as one of system parameters.

Intuitively an increase in localization error will directly lead to target estimation error. To better understand the degree to which this error will propagate to other protocols, we investigate average estimation error under different node densities for varying degrees of location error. For these experiments, we implement a target estimation algorithm described in [3]: the average *x* and *y* coordinates of all reporting nodes[3] are taken as the target location estimation. The results of various experiments are depicted in Figure 24. This graph shows that target estimation error due to location error is dampened during the aggregation process. Aside from showing varying degrees of estimation error with respect to node location error, Figure 24 also shows that the absolute target estimation error decreases with increased node density. For example, when localization error is equal to 0.5R, and sensing density reaches 16 nodes per radio range, the estimation error is only about 65% as large as when the node density is 8. Based on Figure 24, we suggest

[3] Nodes report when they sense the event of interest in the environment.

that localization error impact can be ameliorated through aggregation with higher degree of aggregation.

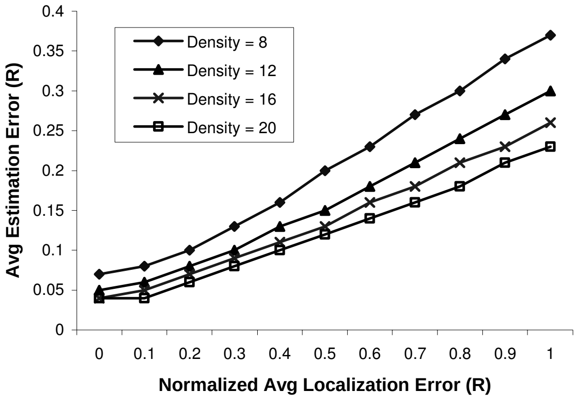


Figure 24: Target estimation error with different localization errors under varying node density

7.3 Object Tracking Performance

We evaluate the performance of tracking application that uses estimation in context. In this experiment, a mobile evader randomly walks around the specified terrain while a pursuer attempts to catch it. Initially the evader and pursuer are placed at the left top and left bottom corner respectively. The evader chooses a direction to go across the terrain at a constant speed. After simulation starts, the pursuer is informed of the current location of the evader periodically via sensing nodes in the terrain that detect the evader, coordinate to estimate the targets position with regard to their own positions, and periodically report this result to the mobile pursuer. When receiving a report, the pursuer readjusts its direction in an attempt to intercept the evader. When the pursuer comes within the sensing radius of the evader, the evader is considered caught and the simulation ends. For this experiment, we compare the average tracking time (the time from pursuer take-off to when the evader is caught) under different localization errors, to the tracking time in the case of no localization error. Figure 25 shows normalized tracking time in relation to the localization error for various pursuer speeds.

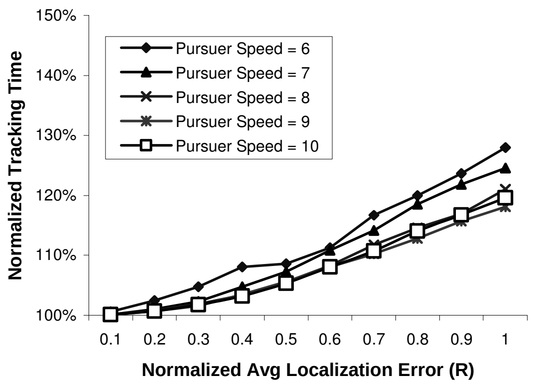


Figure 25: Normalized tracking time with different pursuer speeds varying localization error. Terrain size: 1400mx1400m, Radio range: 100m, Sensing range: 50m, Density: 4 nodes per circle, Evader speed: 5 m/second

Figure 25 shows that tracking time increases slowly as the localization error increases. For example, when the average localization error is as large as 0.8R, and the pursuer speed is 6 meters per second, the pursuer requires only 20% more time in comparison to the ideal situation in which no localization error exists. This overhead can be further reduced to 10%, by increasing the pursuer's speed to 10 meters per second.

7.4 Sensing Coverage Performance

Energy efficient sensing coverage is critical for many important WSNs applications such like military surveillance. A recent work sensing coverage [38] proposed a scheme that attempts to minimize energy consumption of sensing coverage services. The basic idea of this scheme is when multiple sensor nodes cover a geographic location, idally only one of the nodes is turned on at any time so that energy consumption is minimized. For each geographic location in the sensing area, all the sensors that can cover the location jointly determine their schedules of being turned on and off. Hence, energy consumption is minimized while sensing coverage is not compromised. In the absence of localization error, this scheme can guarantee 100% sensing coverage when no sensing void (location that can not be covered by any node) exists.

To investigate the impact of localization errors on the performance of this scheme, we implemented aforementioned algorithm and studied its sensing coverage ratio and energy consumption under different sensing densities and localization errors. Localization errors conceivably have negative impact on sensing coverage ratio because sensor nodes can falsely claim being able to cover a location based on inaccurate information of their locations and hence make the location left uncovered at some time. It is the case, however simulation results in Figure 26 indicate that such an effect is very small. Sensing coverage decreases slowly with increased localization errors across all sensing density values in our experiments. This mainly is because with uniform distributed localization error, the effects of under-cover and over-cover counteract each other, hence reduce the chance of uncovered points.

In terms of energy consumption, the impact of localization errors is also small. Figure 27 plots the energy consumption under various localization errors and sensing densities. We can see that average energy consumption increases only 1.7% when localization error is as large as one radio range (statistically insignificant), compared with the no error case. This is because all nodes are scheduled according to error locations, the length of resulting schedules are irrelevant to the actual locations of the nodes.

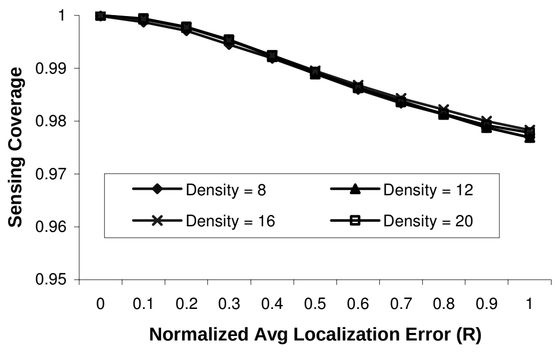


Figure 26: Local Error Impact on Sensing Coverage

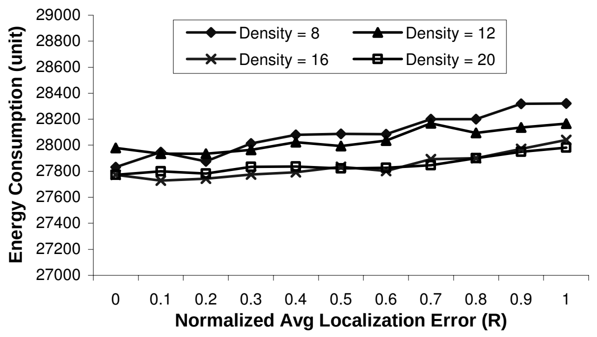


Figure 27: Local Error Impact on Energy Consumption

8. FUTURE WORK

It is well known that range-free localizations are subject to the effect of irregular radio patterns. We have done extensive evaluation of range-free protocols in simulation under a realistic radio model. However, we acknowledge that such a model can only serve as an approximation to real situation. Due to the lack of long-range anchor nodes and other constraints, we are not be able to evaluation all aforementioned protocols in a running system. We left this as future work.

9. CONCLUSION

Given the inherent constraints of the sensor devices envisioned and the estimation accuracy desired by location-dependent applications, range-free localization schemes are regarded as a cost-effective and sufficient solution for localization in sensor networks. From our comparison study, we identify preferable system configurations of four different recently proposed range-free localization schemes as a design guideline for further research. In particular, an APIT scheme, proposed in this paper, performs best when irregular radio patterns and random node placement are considered, and low communication overhead is desired. Moreover, we provide insight on how localization error affects a variety of location-dependent applications. These results show that the accuracy provided by the range-free schemes considered is sufficient to support various applications in sensor networks with only slight performance degradation.

REFERENCES

[1] P. Bahl and V. N. Padmanabhan, RADAR: An In-Building RF-Based User Location and Tracking System, In *Proceedings of the IEEE INFOCOM '00*, March 2000.

[2] J. Beutel, Geolocation in a PicoRadio Environment, M.S. Thesis, ETH Zurich, *Electronics Laboratory*, Dec. 1999.

[3] B. Blum, P. Nagaraddi, A. Wood, T. Abdelzaher, S. Son and Jack Stankovic, In Mobisys 2003, San Francisco, CA 2003

[4] N. Bulusu, J. Heidemann and D. Estrin, GPS-less Low Cost Outdoor Localization for Very Small Devices, *IEEE Personal Communications Magazine*, 7(5):28-34, October 2000.

[5] N. Bulusu, J. Heidemann and D. Estrin, Adaptive Beacon Placement, In *IEEE ICDCS '01*, Phoenix, AZ, April 2001.

[6] N. Bulusu, J. Heidemann, D. Estrin and T. Tran, Self-configuring Localization Systems: Design and Experimental Evaluation , In *TECS Special Issue on Networked Embedded Computing*, 2003.

[7] J. Caffery, Jr. A New Approach to the Geometry of TOA Location, In *IEEE Vehicular Technology Conference (VTC)*, Boston, Mass, September 2000.

[8] S. Capkun, M. Hamdi and J.P. Hubaux, GPS-Free Positioning in Mobile Ad-Hoc Networks, In *Proceedings of HICCSS '01*, Maui, Hawaii, January 2001.

[9] L. Doherty, L. E. Ghaoui and K. S. J. Pister, Convex Position Estimation in Wireless Sensor Networks, In *Proceedings of the IEEE INFOCOM '01*, Anchorage, AK, April 2001.

[10] D. Estrin, R. Govindan, J. Heidemann and S. Kumar, Next Century Challenges: Scalable Coordination in Sensor Networks,In *MOBICOM '99*, Seattle, Washington, 1999.

[11] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin and S. Wicker, Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks, *Technical Report UCLA/CSD-TR 02-0013*, 2002.

[12] L. Girod and D. Estrin, Robust Range Estimation using Acoustic and Multimodal Sensing, In *Proceedings of IROS '01*, Maui, Hawaii, October 2001.

[13] A. Harter, A. Hopper and P. Steggles, A. Ward and P. Webster, The anatomy of a context-aware application, In *Proceedings of MOBICOM '99*, Seattle, Washington, 1999.

[14] T. He, C. Huang, B. M. Blum, J. A. Stankovic and T. F. Abdelzaher, Range-Free Localization Schemes in Large Scale Sensor Networks, In Proceedings of MobiCom 2003.

[15] T. He, B. M. Blum, J. A. Stankovic and T. F. Abdelzaher, "A Lazy-Binding Communication Protocol for highly dynamic wireless sensor networks", In submission, 2003.

[16] J. Hightower and G. Boriello, Location Systems for Ubiquitous Computing, *IEEE Computer*, 34(8):57-66, August 2001.

[17] J. Hightower, G. Boriello and R. Want, SpotON: An indoor 3D Location Sensing Technology Based on RF Signal Strength, *University of Washington CSE Report #2000-02-02*, February 2000.

[18] X. Hong, K. Xu, and M. Gerla, Scalable routing protocols for mobile ad hoc networks, *IEEE Network magazine*, vol 16, No. 4, 2002.

[19] B. Karp and H. T. Kung, GPSR: Greedy Perimeter Stateless Routing for Wireless Networks, In *Proceedings of MOBICOM '00*, New York, August 2000.

[20] L. Kleinrock and J. Silvester, Optimum transmission radii for packet radio networks or why six is a magic number, In *proceedings of national Telecomm conf*erence, Pages 4.3.1-4.3.5, 1978

[21] Y. B. Ko and N. H. Vaidya, Location-Aided Routing (LAR) in Mobile Ad Hoc Networks, In *Proceedings of MOBICOM '98*, Dallas, TX, 1998.

[22] J. Li, J. Jannotti, D. S. J. De Couto, D. Karger and R. Morris, A Scalable Location Service for Geographic Ad-Hoc Routing, In *Proceedings of MOBICOM '00*, New York, August 2000.

[23] MICA Sensor Board Information, http://www.xbow.com

[24] R. Nagpal, Organizing a Global Coordinate System from Local Information on an Amorphous Computer, *A.I. Memo 1666*, MIT A.I. Laboratory, August 1999.

[25] R. Nagpal, H. Shrobe, J. Bachrach, Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network, In IPSN '03, Palo Alto, April, 2003.

[26] J. C. Navas and T. Imielinski, Geographic Addressing and Routing, In *Proceedings of MOBICOM '97*, Budapest, Hungary, September 26, 1997.

[27] D. Nicolescu and B. Nath, Ad-Hoc Positioning Systems, In *Proceedings of IEEE GLOBECOM '01*, November 2001.

[28] D. Niculescu and B. Nath, DV Based Positioning in Ad hoc Networks, In *Journal of Telecommunication* Systems, 2003.

[29] D. Niculescu and B. Nath, Ad Hoc Positioning System (APS) using AoA, *INFOCOM' 03*, San Francisco, CA,2003

[30] N. B. Priyantha, A. Chakraborty and H. Balakrishnan, The Cricket Location-Support System, In *Proceedings of MOBICOM '00*, New York, August 2000.

[31] C. Savarese, J. Rabay and K. Langendoen, Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks, *USENIX Technical Annual Conference*, Monterey, CA, June 2002.

[32] A. Savvides, C. C. Han and M. B. Srivastava, Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors, In *Proceedings of MOBICOM '01*, 2001, Rome, Italy, July 2001.

[33] A. Savvides, H. Park and M. Srivastava, The Bits and Flops of the N-Hop Multilateration Primitive for Node Localization Problems, In *WSNA'02*, Atlanta, GA, September 2002.

[34] R. Want, A. Hopper, V. Falcao and J. Gibbons, The Active Badge Location System, *ACM Transactions on Information Systems*, January 1992.

[35] B. H. Wellenhoff, H. Lichtenegger and J. Collins, *Global Positions System: Theory and Practice*, Fourth Edition. Springer Verlag, 1997.

[36] K. Whitehouse and D. Culler, Calibration as Parameter Estimation in Sensor Networks, In *First ACM International Workshop on Wireless Sensor Networks and Application*, Atlanta GA, September 2002.

[37] Y. Xu, J. Heidemann and D. Estrin, Geography-informed Energy Conservation for Ad Hoc Routing , In *Proceedings of MOBICOM '01*, Rome, Italy, July 2001.

[38] T. Yan, T. He, J. A. Stankovic, Differentiated Surveillance Service for Sensor Networks, In *SenSys 2003*, Los Angeles, CA, November 2003.

[39] G. Zhou, T. He, S. Krishnamurthy, J. A. Stankovic, Impact of Radio Irregularity on Wireless Sensor Networks, In MobiSys'04, Boston, MA, June 2004.

A Communication Architecture and Programming Abstractions for Real-Time Embedded Sensor Networks [*]

T. Abdelzaher, J. Stankovic, S. Son, B. Blum, T. He, A. Wood
Department of Computer Science
University of Virginia
Charlottesville, VA 22904

Chenyang Lu
Department of Computer Science
University of Washington, St. Louis
St. Louis, MO 63130

Abstract

Data distribution in embedded real-time sensor networks requires new protocols and programming environments that achieve time-sensitive message delivery and provide useful abstractions to the application programmer. Attainment of these goals requires changes to multiple layers of the communication protocol stack. In this paper, we review a protocol suite developed by the authors for data communication in embedded sensor networks. It takes into account time constraints and exports attribute-based connections that are tightly integrated with properties of the monitored environment. A programming language is described that allows external physical objects to be represented as first class abstractions in the computing system. The language facilitates writing monitoring applications. The system was implemented on a prototypical sensor network based on MICA motes.

Keywords: sensor networks, programming paradigms, tracking, QoS, distributed systems.

1 Introduction

Ad hoc wireless sensor networks, made possible by advances in communication technology and hardware miniaturization [11], raise the need for a new suite of communication protocols and new programming abstractions for distributed deeply embedded computing. Such sensor networks are especially useful when an inhospitable, poorly accessible, or delicate environment prevents the installation of needed computing infrastructure. An example could be the site of a natural disaster or a target behind enemy lines. Instead, myriads of tiny computationally equipped wireless sensor devices may be dropped to form an ad hoc network that operates autonomously to monitor its surroundings, react to distributed events, or alert appropriate authorities when specific activities are observed.

Sensor networks offer new challenges both from the perspective of building communication protocols and from the perspective of developing appropriate programming models. These challenges arise due to their large scale, autonomous operation, massively parallel interactions with a spatially distributed physical environment, and a more stringent set of resource constraints.

Communication protocols for sensor networks must provide real-time assurances. While ensuring proper timing behavior of systems has been a topic of real-time research for decades, sensor network applications offer physical *space* in addition to time, as a new dimension for interaction with the environment. Hence, while traditional real-time computing research has been concerned with meeting time constraints, a new branch of theory is needed to analyze systems that interact with the their surroundings both in real time and in the real dimensions of physical space. For example, in a network that tracks vehicles through the sensor field, the application must collect sensory measurements in real-time from the actual changing locale in which the vehicle is detected. Message communication must therefore be sensitive to both time and distance constraints, which may depend on external factors such as the physical speed of the monitored vehicle. In this paper, we describe a protocol suite in which both time and distance constraints are addressed.

A new programming paradigm is needed to facilitate the task of sensor network application development. Due to the large scale of sensor networks, programmers should not have to concern themselves with low-level abstractions and functions such as creating and destroying individual connections between pairs of nodes. Instead, the programming environment must offer a conceptual view in which global tasks can be defined in an abstract manner, leaving it for the underlying system to translate them into computational and communication activities on individual sensor nodes. This paper reports on the design of a programming system developed on top of our communication protocol suite, which provides the required high-level abstractions. The language allows external events in the environment to be represented as objects in the computing

[*]The work reported in this paper was supported in part by the National Science Foundation under grants CCR-02-05327 and CCR-00-92945, by DARPA under grant F33615-01-C-1905, and by MURI under grant N00014-01-1-0576.

system facilitating the monitoring of such events by the application. The reported architecture is a part of an ongoing research effort on developing a sensor network virtual machine for future distributed deeply embedded applications.

The rest of this paper is organized as follows. In Section 2, we describe a protocol suite that takes into account time and space constraints, and exports a useful transport-layer abstraction in which logical communication end-points can be associated with tracked objects in the external environment. Section 3 describes a new programming model for sensor networks which builds upon the aforementioned transport protocol to elevate environmental objects into first class programming abstractions. Related work is summarized in Section 4. The paper concludes with Section 5 which describes some of the remaining challenges and directions for future research.

2 A Protocol Suite for Sensor Networks

Communication protocols in sensor networks are the fundamental cornerstone that glues distributed applications together. The deeply embedded nature of sensor networks presents some of the most interesting challenges in the design of their communication protocols. New research topics span all protocol stack layers, primarily motivated by a tighter interaction between the network and its physical environment. At the MAC layer, new protocols are needed that enforce message priorities consistently with time and distance constraints that arise from environmental interactions [22]. Awareness of the physical environment must also be incorporated into the network layer. For example, location should be an essential attribute of addressable networked objects [15]. Location-assisted routing protocols such as LAR [19] and DREAM [4], as well as location services [21] have been described for ad hoc wireless networks. More generally, routing algorithms are needed in which destinations are described implicitly by their environmental attributes. For example, directed diffusion [18, 14] and the intentional naming system [3] provide addressing and routing based on data interests. A fundamental rethinking of basic protocols is required at the transport layer as well. Individual socket-style connections between nodes are too low-level to be a useful abstraction for the programmer. They must be replaced with higher-level alternatives that are more suitable for the main purpose of sensor networks, namely monitoring the external surroundings in which they are embedded.

This section describes our answer to the challenge of incorporating environmental awareness into the design of sensor network communication protocols. Our protocol stack features two important contributions. First, it implements new real-time message scheduling algorithms in which both time and physical distance requirements are observed. Second, it exports a transport-layer address space that associates unique network addresses with external environmental objects. The new addresses serve as connection end-points, thereby raising the level of connection abstraction to entities of direct interest to the application. The layers of our protocol stack are described in the following subsections.

2.1 Real-Time Distance-Aware Scheduling

Message communication in sensor networks must occur in bounded time, for example to prevent delivery of stale data on the status of detected events or intruders. In general, a sensor network may simultaneously carry multiple messages of different urgency, communicated among destinations that are different distances apart. The network has the responsibility of ordering these messages on the communication medium in a way that respects both time and distance constraints.

A protocol that achieves this goal in our architecture is called RAP [22]. It supports a notion of packet velocity and implements velocity monotonic scheduling (VMS) as the default packet scheduling policy on the wireless medium. Observe that for a desired end-to-end latency bound to be met, an in-transit packet must approach its destination at an average velocity given by the ratio of the total distance to be traversed to the requested end-to-end latency bound. RAP prioritizes messages by their required velocity such that higher velocities imply higher priorities. Two flavors of this algorithm are implemented. The first, called *static velocity-monotonic scheduling*, computes packet priority at the source and keeps it fixed thereafter regardless of the actual rate of progress of the packet towards the destination. The second, called *dynamic velocity-monotonic scheduling*, adjusts packet priority *en route* based on the remaining time and the remaining distance to destination. Hence, a packet's priority will increase if it suffers higher delays on its path and decrease if it is ahead of schedule.

To achieve consistent prioritization in the wireless network, not only do we need priority queues at nodes, but also a MAC layer that resolves contention on the wireless medium in a manner consistent with message priorities. We adopt a scheme similar to [1] to prioritize access to the wireless medium. The scheme is based on modifying two 802.11 parameters, namely the DIFS counter and the backoff window, such they are priority-aware. The DIFS counter determines the maximum time a node waits, after the communication channel becomes idle, prior to transmitting an RTS packet. The actual waiting time is randomly chosen between 0 and DIFS. An approximate prioritization effect is achieved by letting the DIFS value depend on the priority of the outgoing packet at the head of the transmission queue. A larger value is given to packets of lower priority. Hence, more urgent packets tend to contend on the medium more aggressively. The back-off window of 802.11 increases the maximum waiting time when collisions occur. To give preferential treatment to higher priority packets, we make this increase dependent on the priority of the head of the queue. A higher increase is incurred for packets of lower priority. Hence, collisions tend to be resolved in favor of higher-priority packets.

A detailed performance evaluation of this scheme can be found in [22]. It is shown that velocity-monotonic scheduling substantially increases the fraction of packets that meet their deadlines taking into consideration distance constraints. More accurate schemes for medium access prioritization remain an open research topic. An interesting related topic is that of schedulability analysis of velocity-monotonic scheduling. Ideally, such an analysis should allow a source node to determine whether a particular desired velocity is attainable between a source-destination pair given current network conditions. While an analytic expression for velocity feasibility is still an open problem, in the following, we describe a feedback-based technique that enforces velocity constraints dynamically by applying back-pressure to slow down the sources when such constraints are violated.

2.2 Enforcement of Velocity Constraints

Consider a network that supports multiple predefined velocities. An application can choose a velocity level for each message. The network guarantees that the chosen message velocity is observed with a very high probability as long as the message is accepted from the application. A network-layer protocol with the above property, called SPEED [13], has recently been developed by the authors. The protocol defines the velocity of an in-transit message as the rate of decrease of its straight-line distance to its final destination. Hence, for example, if the message is forwarded away from the destination, its velocity at that hop is negative.

The main idea of SPEED is as follows. Each node i in the sensor network maintains a neighborhood table that enumerates the set of its one-hop neighbors. For each neighbor, j, and each priority level, P, the node keeps a history of the average recently recorded local packet delay, $D_{ij}(P)$. Delay $D_{ij}(P)$ is defined as the average time that a packet of priority P spends on the local hop i before it is successfully forwarded to the next-hop neighbor j. Given a packet with some velocity constraint, V, node i determines the subset of all its neighbors that are closer to the packet's destination. If L_{ij} is the distance by which neighbor j is closer to the destination than i, the velocity constraint of the packet is satisfied at node i if there exists some priority level P and neighbor j such that $L_{ij}/D_{ij}(P) \geq V$. The packet is forwarded to one such neighbor non-deterministically. If the condition is satisfied at multiple priority levels, the lowest priority level is chosen. If no neighbor satisfies the velocity constraint, we say that a local deadline miss occurs.

A table at node i keeps track of the number of local deadline misses observed for each velocity level V. This table is exchanged between neighboring nodes. Nodes use this information in their forwarding decisions to favor more appropriate downstream hops among all options that satisfy the velocity constraint of a given packet. No messages are forwarded in the direction of nodes with a high miss-ratio. The mechanism exerts back-pressure on nodes upstream from congested areas. Congestion increases the local miss-ratio in its vicinity, preventing messages from being forwarded in that direction. Messages that cannot be forwarded are dropped thus increasing the local miss-ratio upstream. The effect percolates towards the source until a node is found with an alternative (non-congested) path towards the destination, or the source is reached and informed to slow down. The mentioned scheme is therefore effective in exerting congestion control and performing packet rerouting that guarantee the satisfaction of all velocity constraints in the network at steady state [13]. The protocol is of great value to real-time applications where different latency bounds must be associated with messages of different priority.

2.3 Entity-Aware Transport

Although RAP and SPEED allow velocity constraints to be met, the abstractions provided by them are too low-level for application programmers. We develop a transport layer whose main responsibility is to elevate the degree of abstraction to a level suitable for the application. In particular, we propose a transport layer in which connection end-points are directly associated with events in the physical environment. Events represent continuous external activities, such as the passage of a vehicle or the progress of a fire, which is precisely what an application might be interested in. By virtue of this layer, the programmer can describe events of interest and logically assign "virtual hosts" to them. Such hosts export communication ports and execute programs at the locations of the corresponding events. The programmer is isolated from the details of how these hosts and ports are implemented. When an external event (e.g., a vehicle) moves, the corresponding virtual host migrates with it transparently to the programmer.

We call the virtual host associated with an external event of interest an *entity*. Sensor nodes that can sense the event are called *entity members*. Members elect an *entity leader* that uniquely represents the entity and manages its state. Hence, an entity appears indivisible to the rest of the network. The fact that it is composed of multiple nodes with a changing membership is abstracted away.

When the external event moves outside the sensing horizon of the current entity leader, the leader hands-off leadership to another member. Connection state is handed off as well allowing communication with the entity to remain uninterrupted. To ensure unique representation of external events within the computational environment, a unique entity must be associated with each event. The transport protocol meets this constraint by announcing the existence of the entity to nearby nodes that cannot yet sense the event. These announcements are sent periodically by the entity leader and are called *heartbeats*. Nodes that hear a heartbeat but cannot sense the event are called *entity followers*. They are said to be within the *awareness horizon* of the named entity. Upon receiving a heartbeat, such nodes set an *entity timeout timer*. Upon timer expiration, their sta-

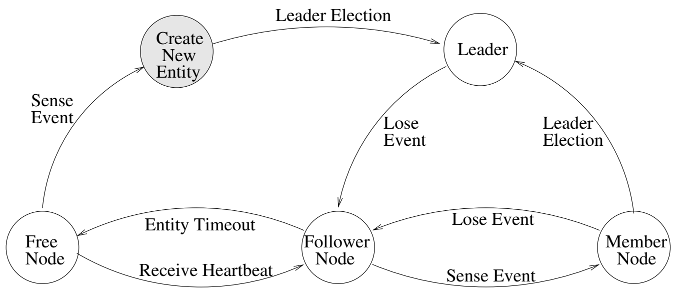


Figure 1. Node state transition diagram

tus as followers expires. The timer is reset to zero every time a new heartbeat is received. When the event enters the sensing horizon of a follower node, the node becomes a member of the entity it is following. If the node is not a follower, it recognizes that a new entity must be created. The node sets a random timer upon expiration of which it claims leadership of the new entity. If it receives a leadership claim message from another node prior to timer expiration, it clears the timer and becomes an entity member. The algorithm ensures that a newly sensed event is represented by a single entity and that current events do not spawn spurious entities as they move from one location to another. Figure 1 depicts the node state transition diagram between follower, member, and leader states, as well as the free state in which a node is not cognizant of any entities.

An evaluation of this architecture reveals that entity uniqueness is maintained as long as the target event moves in the environment at a speed slower than half the nodes' communication radius per second [7]. For example, if sensor nodes can communicate within a 200 meter radius, the transport layer can correctly maintain endpoints attached to targets that move as fast as 100 m/s (i.e., 360 km/hr). The combination of this transport layer and the guaranteed velocity protocols described earlier provides invaluable support to real-time applications. For example, communication regarding moving targets can be made to proceed in the network at a velocity that depends on target velocity itself. Hence, positions of faster targets, for example, can be reported quicker than those of slower ones. To the authors' knowledge no other protocols in sensor networks have explicitly addressed message timing constraints.

3 A Sensor-Network Programming Model

The transport layer described above gives rise to a programming model that elevates tracked activities in the physical environment into first class programming abstractions. In this model, the application developer specifies events to be monitored. The system automatically detects such events and instantiates a so called *context* every time an instance of an event is detected in the environment. From the programmer's perspective, the application is composed of a dynamic set of contexts, each representing a particular event. Objects can be attached to contexts. These objects will logically execute in the locale of the monitored event. Contexts have unique identifiers called *context labels*. Objects attached to a context can be addressed using the context label and object name. They can communicate remotely by remote method invocation. The programmer's view of the application is depicted in Figure 2.

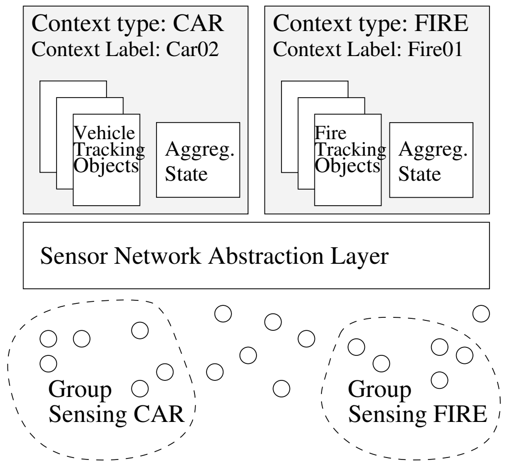


Figure 2. Programming model

A context label around some event, e, is completely defined by two elements, namely (i) the function $sense_e()$ which specifies an environmental condition that spawns the context label, and (ii) the function $state_e()$ which describes the environmental state to be encapsulated in the context label. The former function, for example, might dictate that a label is to be created if magnetic distortion (e.g., the presence of a vehicle) is sensed. The state function returns a set of aggregate variables, each computed using outputs of at least N_e nodes for which $sense_e()$ was true in the last L_e time units. We call N_e and L_e the critical mass and freshness constraints, respectively. For example, to obtain the approximate position of a vehicle we may define $state_e()$ to be the average coordinates of at least 5 nodes that have sensed the vehicle within the last 2 seconds. We define environmental tracking of event e as the process of maintaining the state of this event subject to given freshness and critical mass constraints.

Syntactically, an application consists of a list of context declarations, each specifying an activation condition $sense_e()$, a set of state variables $state_e()$, and a list of attached objects. An example declaration is shown in Figure 3. The example defines a context of type *tracker*, specifies its activation condition, $sense_e()$, as an appropriate magnetometer reading (presumably caused by a nearby vehicle), and defines $state_e()$ as the average $location$ of the tracked target. It specifies that $location$ must represent the average of at least 2 sensor readings measured no earlier than 1 second ago. The attached object is invoked periodically to report the current location of the vehicle to a virtual base station object. It passes the originating

```
begin context tracker
  activation: MAGNETOMETER == ON
  location : avg (position) mass=2, freshness=1s
  begin object reporter
    invocation: PERIOD(0.5s)
    report_function() {
      BaseStation.reportLocation (self.label, location);
    }
  end
end context
```

Figure 3. Sample code

context label as the identity of the reported vehicle. If there are several vehicles in the field, multiple reporter objects will be automatically instantiated. The programmer does not need to worry about instantiating these objects. Object execution and maintenance of aggregate state occurs automatically. Details of the underlying communication, group membership management, leader handoff, and mobility are handled transparently. Hence, the programmer's interaction with the sensor network is significantly simplified.

We have described real-time communication protocols and programming abstractions motivated by a tighter interaction between sensor networks and their physical environment. Our architecture might be a first step towards a comprehensive vision for next-generation programming systems supporting future real-time deeply embedded distributed sensor network applications.

4 Related Work

Classical distributed programming paradigms and middleware such as CORBA [27], group communication (e.g., ISIS [5]), remote procedure calls (RPC [6]), and distributed shared memory (e.g., MUNIN [9]) share in common the fact that their programming abstractions exist in a logical space that does not represent or interact with objects and activities in the physical world. Their main goal is to abstract distributed communication rather than facilitate distributed sensory interactions with an external physical environment. In contrast, sensor network applications call for a paradigm that revolves around "environmentally-inspired" abstractions aimed at simplifying the coding of interactions with the physical world that arise in distributed deeply embedded systems.

The work reported in this paper is closely related to several recent projects, such as Cricket [23], Sentient Computing [2] and Cooltown [10], which propose high-level paradigms in which an embedded distributed computing system is able to share humans' perceptions of the physical world. These systems allow the location of entities in the external environment to be tracked. One major difference is that they assume cooperative users who, for example, can wear beaconing devices that interact with location services in the infrastructure for the purposes of localization and tracking [23, 2]. Our interest, in contrast, is in situations where no cooperation is assumed from the tracked entity.

In the absence of cooperation, several research efforts proposed alternative addressing schemes that do not rely on having destinations with specific identities, but rather contact sensor nodes in the vicinity of a phenomenon of interest based on the attributes of data they sense. For example, DataSpace [17] exports abstractions of physical volumes addressable by their locations. Similarly, directed diffusion [18, 14] and the intentional naming system [3] provide addressing and routing based on data interests [18, 14]. Attributed-based naming is also related to the notion of content-addressable networks [24] proposed for an Internet environment, which allows queries to be routed depending on the requested content rather than on the identity of the target machine. We adopt context labels; a form of attribute-based naming. In our architecture, however, context labels are *active* elements. Not only do they provide a mechanism for *addressing* nodes that sense specific environmental conditions, but also they can *host context-specific computation* that tracks a target in the environment.

Recent research on system software for sensor networks has seen the introduction of distributed virtual machines designed to provide convenient high-level abstractions to application programmers, while implementing low-level distributed protocols transparently in an efficient manner [26]. This approach is taken in MagnetOS [12], which exports the illusion of a single Java virtual machine on top of a distributed sensor network. The application programmer writes a single Java program. The run-time system is responsible for code partitioning, placement, and automatic migration such that total energy consumption is minimized. Maté [20] is another example of a virtual machine developed for sensor networks. It implements its own bytecode interpreter, built on top of TinyOS [16].

A somewhat different approach of providing high-level programming abstractions is to view the sensor network as a distributed database, in which sensors produce series of data values and signal processing functions generate abstract data types. The database management engine replaces the virtual machine in that it accepts a query language that allows applications to perform arbitrarily complex monitoring functions. This approach is implemented in the COUGAR sensor network database [8]. A middleware implementation of the same general abstraction is also found in SINA [25], a sensor information networking architecture that abstracts the sensor network into a collection of distributed objects.

Our system is different in that it is geared for real-time environmental tracking. To the authors' knowledge, we describe the first programming language for sensor networks that explicitly facilitates the coding of tracking applications, and the first sensor network communication protocols that conider real-time constraints. These novel abstractions and underlying mechanisms are well-suited for monitoring targets that move in the physical world. They can therefore have a major impact on application development for sensor networks.

5 Conclusions

This paper reviewed a new protocol suite and programming system for sensor network applications, that may considerably improve real-time behavior and reduce the development cost of deeply embedded systems. This reduction comes from offloading from the application developer the details of managing low-level abstractions. Future work of the authors will involve refinement of the real-time protocols and the environmental tracking problem such that more precise semantics and failure models are achieved. With such refinements we hope to build a predictable sensor network "virtual machine" that exports timely, reliable behavior and well-defined semantics, implemented on the unreliable, unpredictable, and resource constrained hardware and communication infrastructure typical of sensor networks. Such a virtual machine would hide the complexity of sensor network programming from the application developer, making a new more robust and more dynamic realm of sensor network applications attaintable to impact future defense, surveillance, habitat monitoring, and disaster management systems.

References

[1] I. Aad and C. Castelluccia. Differentiation mechanisms for ieee 802.11. In *IEEE Infocom*, Anchorage, Alaska, April 2001.

[2] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper. Implementing a sentient computing system. *IEEE Computer*, 34(8):50–56, August 2001.

[3] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The design and implementation of an intentional naming system. In *ACM Symposium on Operating Systems Principles*, Kiawah Island, SC, December 1999.

[4] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. A distance routing effect algorithm for mobility (dream). In *ACM MOBICOM*, pages 76–84, October 1998.

[5] K. Birman, A. Schiper, and P. Stephenson. Lightweight causal and atmoic group multicast. *ACM Transactions on Computer Systems*, 9(3):272–314, August 1991.

[6] A. Birrel and B. Nelson. Implementing remote procedure calls. *ACM Transactions on Computer Systems*, 2(1), February 1984.

[7] B. Blum, P. Nagaraddi, A. Wood, T. Abdelzaher, J. Stankovic, and S. Son. An entity maintenance and connection service for sensor networks. In *The First International Conference on Mobile Systems, Applications, and Services (MobiSys)*, San Francisco, CA, May 2003.

[8] P. Bonnet, J. Gehrke, and P. Seshardi. Towards sensor database systems. In *2nd International Conference on Mobile Data Management*, pages 3–14, Hong Kong, January 2001.

[9] J. Carter, J. Bennet, and W. Zwaenepoel. Implementation and performance of munin. In *ACM Symposium on Operating Systems Principles*, pages 151–164, October 1991.

[10] P. Debaty and D. Caswell. Uniform web presence architecture for people, places, and things. *IEEE Personal Communications*, 8(4):46–51, August 2001.

[11] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Mobile networking for smart dust. In *ACM MOBICOM*, Seattle, WA, August 1999.

[12] R. B. et al. On the need for system-level support for ad hoc and sensor networks. *Operating System Review*, 36(2):1–5, April 2002.

[13] T. He, J. Stankovic, C. Lu, and T. Abdelzaher. Speed: A stateless protocol for real-time communication in sensor networks. In *International Conference on Distributed Computing Systems*, Providence, Rhode Island, May 2003.

[14] J. Heideman, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. *Operating Systems Review*, 35(5):146–159, December 2001.

[15] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8), August 2001.

[16] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. In *ASPLOS*, Cambridge, MA, November 2000.

[17] T. Imielinski and S. Goel. Dataspace - querying and monitoring deeply networked collections in physical space. *IEEE Personal Communications*, 7(5):4–9, October 2000.

[18] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *ACM MOBICOM*, Boston, Massachusetts, August 2000.

[19] Y.-B. Ko and V. Nitin. Location-aided routing (lar) in mobile ad hoc networks. In *ACM MOBICOM*, pages 66–75, October 1998.

[20] P. Levis and D. Culler. Mate: A tiny virtual machine for sensor networks. In *ASPLOS*, San Jose, CA, October 2002.

[21] J. Li, J. Jannotti, D. D. Couto, D. Karger, and R. Morris. A scalable location service for goegraphic ad hoc routing. In *ACM MOBICOM*, Boston, Massachusetts, August 2000.

[22] C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, and T. He. Rap: A real-time communication architecture for large-scale wireless sensor networks. In *Real-Time Technology and Applications Symposium*, San Jose, CA, September 2002.

[23] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *ACM MOBICOM*, Boston, MA, August 2000.

[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Sigcomm*, San Diego, CA, August 2001.

[25] C.-C. Shen, C. Srisathapornphat, and C. Jaikeo. Sensor information networking architecture and applications. *IEEE Personal Communications*, 8(4):52–59, August 2001.

[26] E. Sirer, R. Grimm, A. Gregory, and B. Bershad. 'design and implementation of a distributed virtual machine for networked computers. In *ACM Symposium on Operating System Principles*, pages 202–216, Kiawah Island, SC, December 1999.

[27] S. Vinoski. Corba: Integrating diverse applications within distributed heterogeneous environments. *IEEE Communications Magazine*, 14(2), February 1997.

Feedback Control-based Dynamic Resource Management in Distributed Real-Time Systems

Tian He[**], John A. Stankovic[*], Michael Marley [*], Chenyang Lu[***], Yin Lu[*****], Tarek Abdelzaher[****], Sang Son[*], Gang Tao[*]

Department of Computer Science, University of Virginia
151 Engineer's Way, Charlottesville, Virginia 22904-4740

Abstract

The resource management in distributed real-time systems becomes increasingly unpredictable with the proliferation of data-driven applications. Therefore, it is inefficient to allocate the resources statically to handle a set of highly dynamic tasks whose resource requirements (e.g., execution time) are unknown a prior. In this paper, we build a distributed real-time system based on the control theory, focusing on the computational resource management. Specifically, this work makes three important contributions. First, it allows the designer to specify the desired temporal behavior of system adaptation, such as the speed of convergence. This is in contrast to previous literature, specifying only steady-state metrics, e.g. the deadline miss ratio. Second, unlike QoS optimization approaches, our solution meets performance guarantees with no accurate knowledge of task execution parameters – a key advantage in a poorly modeled environment. Last, in contrast to ad hoc algorithms based on intuition and testing, we rigorously prove that our approach not only has excellent steady state behavior, but also meets stability, overshoot, and settling time requirements.

Key words: Real-time, Feedback Control, Quality of Service, Scheduling

1 Introduction

Distributed real-time systems are widely used in highly dynamic environments where the resource requirements are open, fluctuating and not amenable to the traditional worst-case real-time analysis. For example, a web farm can be used to distribute time-sensitive contents such as movies and video clips. They need to handle a changing number of requests with significantly different resource requirements that are unknown beforehand. In a stock market, a system needs to actively push real-time stock updates at various interval to a group of users. The number of users served by a server can change quickly over time. Although these systems differ significantly in term of applications, they all operate in open environments where both workloads and available resources are difficult to predict. Monitoring and feedback control are needed to meet performance constraints. Several difficulties are observed in dynamic resource management in these systems. One main difficulty lies in their data-dependent resource requirements, which cannot be predicted without interpreting input data. For example, the execution time of an information server (a web or database server) heavily depends on the content of requests, such as the particular web page requested. A second major challenge is that these systems have highly uncertain arrival workloads; it is not clear how many users will request some resource in the web. A third challenge involves the complex interactions among many distributed sites, often across an environment with poor or unpredictable timing behavior. Consequently, developing certain types of future real-time systems will involve techniques for modeling the unpredictability of the environment, handling imprecise or incomplete knowledge, reacting to overload and unexpected failures (i.e., those not expressed by design-time failure hypotheses), and achieving the required performance levels and temporal behavior. We envision a trend in real-time computing to provide performance guaran-

* Dept of Computer Science, University of Virginia

** Dept of Computer Science & Engineering, University of Minnesota

* * * Dept of Computer Science & Engineering, Washington Uni. in St. Louis

* * ** Dept of Computer Science, University of Illinois, Urban-Champaign

* * * * * Dept of Computer Science & Engineering, University of Nebraska, Lincoln

tees without the requirement of fine-grained task execution models, such as those depending on the precise estimation of individual task execution times. We shall see the emergence of coarse-grained models that describe the aggregate behavior of resource requirements. Coarse-grained models are easier to obtain and they need not be accurately computed. These models are more appropriate for dynamic resource management in the presence of uncertainties regarding load and resources.

In this paper, we explore one such model based on difference equations. Unlike the more familiar queuing theory models of aggregate behavior, difference equation models do not make assumptions regarding the statistics of the load arrival process. Independent of the load assumptions, difference equation models are more suitable for systems where load statistics are difficult to obtain or where the load does not follow a distribution that is easy to handle analytically. The latter is the case, for example, with web traffic, which cannot be modeled by a Poisson distribution. Our solution has a basis in the theory and practice of feedback control scheduling. This is in contrast to the more common ad hoc resource management based on intuition and testing where it is very difficult to characterize the aggregate performance of the system and where major overloads and/or anomalous behavior can occur since these designs are not developed to avoid these problems.

2 The Overview of DFCS Architecture

Traditional real-time computing provides guarantees in avoidance of undesirable effects such as overload and deadline misses. They assume worst-case resource requirements known a priori. In contrast, in highly uncertain environments, the main concern is to design adaptation capabilities that handle uncertain effects dynamically and in an analytically predictable manner. To address this issue, we propose a framework called Distributed Feedback Control Real-time Scheduling (DFCS). The framework is based on feedback control that incrementally corrects system performance to achieve its target in the absence of initial load and resource assumptions. One main performance metric of such a system is the quality of performance-convergence to the desired level. In our framework, the desired convergence attributes may be specified and enforced using mechanisms borrowed from control theory. These mechanisms have been applied successfully for decades in physical process control systems that are often non-linear and subject to random external disturbances. Before establishing our DFCS framework, we give an overview of the software system being controlled and describe the feed-back-control mechanism involved. Note that although we focus on computational resource management here, while the general methodology can be applied to other dynamic resource management as well.

We assume that the resource under investigation is a cluster of computing nodes connected via a network. Tasks arrive at nodes in unknown patterns. Each task is served by a periodically invoked schedulable entity (such as a thread) with each instance having a soft deadline equal to its period. The periodicity constraint is motivated by the requirements of real-time applications such as process control and streaming media. We abstract a typical dynamic system by two sets of performance metrics. The *primary set* represents metrics to be maintained at specified levels, for example, the deadline miss ratio of a server, or the desired altitude of an airplane. The *secondary set* represents negotiable metrics such as service quality. The objective of adaptation is to incur minimum degradation in secondary metrics while maintaining the primary metrics at their desired values. To represent multiple levels of degradation in secondary metrics, we assume that each task has several service levels of different qualities. For example, a task can execute for varying amounts of time with the quality of the results improving with greater execution time. The goal of our DFCS architecture is to maintain the primary performance metrics around their desired values. Unlike a centralized system, the dynamics of a distributed system manifest themselves on two different time-scales. Fast dynamics are observed on individual nodes, while slower dynamics are observed on the entire system. The fast dynamics arise from local load changes due to individual task arrivals and terminations, while the slower dynamics arise from changes in aggregate load distribution. Therefore, our feed-back architecture naturally includes two sets of control loops,

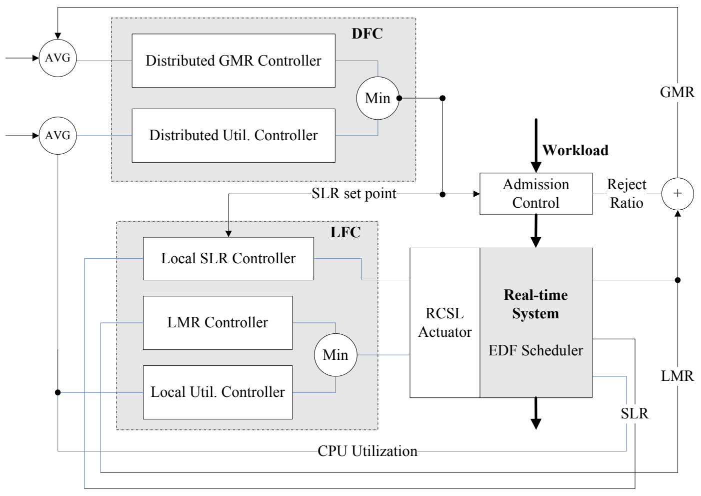


Fig. 1. DFCS Architecture Design

local and distributed ones, each tuned to the dynamics of the corresponding scale. Each node in the distributed system has a local feedback control system (LFC) and possibly a distributed feedback control system (DFC). The distributed feedback controller is responsible for maintaining the appropriate QoS balance between nodes. The local feedback controller is responsible for tracking the global QoS set point set by distributed controller and ensuring that tasks that are admitted to this node have a minimum miss ratio and the node remains fully utilized. It is important to note that these two types of controllers form the main parts of the distributed resource management in the system, but they are not the entire system.

Now consider a few more details about the DFCS architecture as shown in Figure 1. The distributed controller (DFC) commands a set of local controllers (LFC) via a QoS set point, termed as Service Level Ratio(SLR). The local controller (LFC) manipulates its actuators to achieve this SLR set point. In this architecture, we let the primary performance metric be the *deadline miss ratio* (MR). Since zero deadline miss ratio of admitted tasks can be trivially satisfied if the admitted task set is empty, it is especially important to quantify the loss of services due to task rejection to avoid trivial solutions. For this reason, we use two different miss ratio measurements, GMR and LMR. 1) GMR is the Global Miss Ratio of all submitted tasks, including both the admitted tasks and the rejected tasks. The distributed controller (DFC) is responsible for bounding the global miss ratio, GMR, of the system. 2) LMR is the Local Miss Ratio of admitted tasks in a single node. The local controller (LFC) is responsible for controlling the LMR miss ratio of locally admitted tasks as dictated by the distributed controller. Note that as shown in Figure 1 each of the local controller and the distributed controller has two similar parts, a miss ratio controller and a utilization controller. The miss ratio controller activates during overload, while the utilization controller activates at under-utilization when no deadline misses are observed, keeping the system sufficiently utilized. In addition, the LFC has a service level ratio controller (SLR) to address our secondary metric.

In addition, admission control (Figure 1) is based on estimated CPU utilization and the global SLR set point and decides to admit or reject tasks from the outside. If one task is rejected, it is offloaded to another node based on a certain routing policy. Finally, the real-time system is the *plant* under control, which processes the requests from the users. We can plug various scheduling algorithms into this real-time system based on different resource requirements. Here, we use EDF in our design.

3 Design and Model DFCS System

The DFC control design involves two components: a task model and difference equations describing the dynamics of the DFC in under-utilization and overload situations, respectively. The design process proceeds as follows: First, we specify the task model. Second, we specify the desired system performance using both transient and steady-state metrics. This step requires a mapping from the performance metrics of adaptive real-time systems to the dynamic response metrics of control systems used in control theory. Third we establish a mathematical model of the system for the purposes of feedback control. We take a high-level approach where our model aggregates the overall performance of the system in a single model. Our performance study shows this model works well, in spite of its simplicity. Finally, based on the performance specifications and the system model from first and second steps, we apply the mathematical techniques of control theory to design the controller that gives analytic guarantees on the desired transient and steady-state behavior at run-time. We map the controller design to various nodes in the system, depending on the network structure being studied.

3.1 Task Model

We assume a liquid task model in which a node can serve thousands of tasks, each with a small execution time. It is often the case in client-server architectures such as web servers. For each task T_i, there are N QoS service levels ($N \geq 2$). Task T_i running at Service Level q ($0 \leq q < N$) has a deadline $D_i[q]$ and an execution-time $C_i[q]$ that is *unknown* to the scheduler. The requested CPU utilization, $J_i(q) = C_i[q]/D_i[q]$, of the task is a monotonically increasing function of the service level q, which means that a higher QoS requires more CPU utilization. Let the average CPU utilization needed for a task set at level 0 be U_b. Without loss of generality, the average CPU utilization for a task set at level q is $f(q)U_b$, where $f(q)$ is a polynomial representing the Taylor's series expansion of the relation between CPU utilization and QoS level. Here we use the first order approximation of this relation to define the average requested CPU utilization $J(q)$ of a task set:$J(q) = (Aq+1)U_b$ where $q \in [0, N)$. Note that level $N-1$ is the best QoS a task can be served. It should be emphasized that the service level q of a single task T_i must be an integer value, however the service level q for a task set is the average service level, which can be a non-integer, if tasks are served at different levels in a single node. We make use of this approximation in the rest of the paper to derive the system model. Note that if this approximation is not appropriate in some situations, higher order ones can be used. However, the design process remains the same.

3.2 System Specification and Metrics

To evaluate the performance of a system, it is necessary to establish its specifications and performance metrics. Following the practice of the control community in specifying and evaluating the performance of control loops, we propose a series of canonic benchmarks that test system adaptation capabilities. These benchmarks generate a set of simple load profiles adapted from control theory; namely, the *step* load and the *ramp* load. The step load represents a worst-case load variation: a workload change that occurs in zero time. The ramp load represents a more moderate variation that features a slower rate of change. By experimenting with different rates of change, we can assess how well an adaptive system converges to the desired performance upon perturbations caused by changes in the workload. We can also analyze the effects of workload changes with different rates. If the change rate of the work is bounded, this analysis can yield guarantees on the convergence time and worst-case performance deviation. We measure the system load in the percentage of the full system capacity. The load corresponding to the full system capacity is said to be 100%. An overload is a system load that is higher than 100%. A load profile $L(t)$ is the system load as a function of time. In practice, this load is translated into system-specific parameters for evaluation purposes. For example, a 500% system load can be translated to the request rate of 8,000 Mbps in a specific web server.

Consider a time window $[(k-1)W, kW]$, where W is called the sampling period and k is called the sampling instant. During this window, let $M(k)$ be the number of task instances that miss their deadline, let $T(k)$ be the total number of task instances, and let $MR(k) = M(k)/T(k)$ be the miss ratio and M_S be the desired miss ratio performance, termed as the *set point* in control theory. To quantify the performance of adaptation, we have following metrics.

- **Overshoot** M_o**:** the maximum amount by which $M(k)$ exceeds its set point M_S, expressed as a percentage of the set point M_S.
- **Settling time** T_s: The time it takes the miss ratio to enter a steady state after a load change occurs.
- **Steady-state error** E_s: It is the difference between $M(k)$ and its set point M_S when no disturbance happens and after system transients have decayed. It indicates the DFCS's ability to regulate the controlled variable near the set point M_S in the long term. Ideally E_s should be zero.

These metrics provide a basis for us to compare the effectiveness of feedback control to other adaptive real-time scheduling policies. In addition, these metrics can be also used to specify the desired behavior of the adaptation process to guide the control loop design. To enforce these metrics, we need to establish a good aggregate model of the system, the central topic in the next section.

3.3 Modeling the Dynamics in DFCS

Before applying control theory to design a controller from specifications of adaptive behavior, it is necessary to model the system dynamics mathematically. Here the dynamics in DFCS is modeling as an integrated entity with aggregated behavior.

Let the utilization $U(k)$ be the fraction of time the CPU is busy in some sampling window k. Let $S(k)$ be the service level ratio (SLR) at the sampling window k, which can be formulated by Equation 1. In Equation 1, we assume each task has at least two service levels ($N \geq 2$).

$$\begin{gathered} S(k) = \frac{AvgServiceLevel}{N-1} = \frac{\sum_{q=0}^{N-1} (Num.of\ Tasks\ completed\ at\ level\ q) \cdot q}{(Num.of\ Tasks\ completed) \cdot (N-1)} \quad S(k) \in [0,1] \\ S(k) = 0 \quad Num.of\ Tasks\ completed = 0 \end{gathered} \tag{1}$$

Now we derive the relation between utilization $U(k)$, service level ratio $S(k)$ and the resulting number of miss $M(k)$. Note we use miss number $M(k)$ zero as desired the performance metrics (set point), which is equivalent to miss ratio $MR(k)$ zero. In each time window , CPU utilization is proportional to the number of tasks that finish successfully. This relationship can be modeled as:

$$U(k) = c(k) \cdot (T(k) - M(k)) \cdot J(q) \tag{2}$$

where $c(k)$ is the percentage of the arrived tasks that finish in the same sampling window. For example if $c(k) = 1$, all tasks arrive and finish in the same period. If $c(k) = 0.99$, 1% (relatively long) tasks neither miss their deadline, nor finish within the same period. From the perspective of control theory, worst-case conditions for convergence stability are those when system gain is maximum. The maximum gain condition corresponds to $c(k) = 1$. In other words, worst-case conditions occur when we assume the unfinished tasks consume their execution time in the sampling window of arrival. It is a reasonable assumption in a liquid task model where the task execution time is much smaller than the sampling window. Therefore, Equation 2 can be simplified as:

$$U(k) = (T(k) - M(k)) \cdot J(q) \tag{3}$$

From definitions of $J(q)$ in Section 3.1 and Equation 1, we have:

$$J(q) = (1 + A \cdot S(k) \cdot (N - 1)) \cdot U_b \tag{4}$$

Combining Equations 3 and 4, we get the relation desired:

$$U(k) = (T(k) - M(k)) \cdot (1 + A \cdot S(k) \cdot (N - 1)) \cdot U_b \tag{5}$$

Two important subcases arise in modeling the system: namely, overload and under-utilization. They are modeled separately in the two subsequent subsections, respectively.

3.4 Modeling Dynamics when Overload

In the overload situation, tasks begin to miss their deadlines, there are two approaches to tackle the situation: admission control and service level ratio (SLR) adjustment. Admission control reduces a node's local miss ratio by rejecting incoming requests. SLR adjustment tries to accommodate more tasks by degrading the service levels of individual tasks. Since DFCS treats task equally, it degrades tasks with the highest service level first, until the average task service level ratio reaches the desired SLR. In the DFCS design, we deem task rejection the same as missing the task's deadline. Hence, we adopt SLR adjustment whenever possible. Here we get a difference equation that describes how SLR adjustment affects the number of misses when the system is overloaded ($M(k) > 0$). Since we assume the EDF scheduling, when deadline misses occur it must be that the CPU utilization $U(k)$ is 100%. We differentiate

Equation 5, setting $U(k) = 1$, we get the linearized small signal model of the system in overload situations:

$$\begin{aligned} \Delta M(k) &= G_M \cdot \Delta S(k) + \Delta T(k) \\ where\ G_M &= A(N-1)/(U_b(1 + A(N-1)S(k-1))(1 + A(N-1)S(k))) \end{aligned} \tag{6}$$

3.5 Modeling Dynamics when Under-Utilization

The derivation in under-utilization situation is similar to Section 3.4. When DFC is underutilized under the EDF scheduling , the number of tasks missed deadlines $M(k)$ is obviously zero. Since the primary metric is satisfied, we focus on the SLR, the secondary metric presenting the QoS of admitted tasks. In this situation, we switch to utilization measurements. We can increase the SLR of the task set when the utilization is low to improve our service to the user. Here we obtain a difference equation that describes how SLR adjustment affects the CPU utilization when the system is underutilized ($U(k) < 100\%$). After we set $M(k) = 0$ and differentiate the Equation 5, we get

$$\begin{aligned} \Delta U(k) &= G_U \cdot \Delta S(k) + G_t \cdot \Delta T(k) \\ where\ G_U &= T(k-1)A(N-1)U_b\ \ and\ \ G_t = (1 + AS(k-1)(N-1)U_b \end{aligned} \tag{7}$$

3.6 Design the Distributed Controller

With the DFCS dynamics models defined by Equation 6 and 7, we can now design the distributed feedback control loop. In this section, we first define the performance specifications to achieve, then we apply a control design method called Root Locus to tune the distributed controller. Due to space limitations, we do not review local controller design here, which has been intensely studied in our previous work [1] [2].

3.6.1 Design of the Control Loop

In the distributed case, each node in the DFCS provides the same SLR to the user. This property is often preferred in many distributed applications. For example, in a web server farm, the SLR of each HTTP request should be independent of where this request is served in the farm. So the major goal of the distributed controller is to calculate the SLR set point for the system. Since the system dynamics can be modeled with two difference Equations 6 and 7. The former describes the relation between the changes of the service level ratio and the changes of miss number when the whole system is overloaded; the latter models the relation between the changes of the service level ratio and the changes of CPU utilization when the system is underutilized. Based on this knowledge, we design the distributed feedback control loop. Because the external workload is not under our control, we deem $\Delta T(k)$ as the external disturbance[1] . Let G_U be the gain from $\Delta S(z)$ to $\Delta U(z)$ when the system is underutilized and G_M be the gain from $\Delta S(z)$ to $\Delta U(z)$ when system is overloaded. We get:

$$\begin{gathered} M(k) = M(k-1) + \Delta M(k) = M(k-1) + G_M \Delta S(k) \; when \; M(k-1) > 0 \\ U(k) = U(k-1) + \Delta U(k) = U(k-1) + G_U \Delta S(k) \quad when \; U(k-1) < 1 \end{gathered} \tag{8}$$

where G_M and G_U are defined in Equations 6 and 7, respectively. We can now draw the block diagrams of the feedback control system as shown in Figures 2 and 3. When the system is overloaded, the distributed miss feedback control loop (Figure 2)is activated. The components inside the dotted rectangle describe the dynamics of the controlled process with input $\Delta S(z)$ and output $M(k)$, where $C_M(z)$ is the miss controller to be designed and M_S is the miss set point. We can easily obtain the miss ratio $MR(k)$ by dividing $M(k)$ by the total number of tasks. Note that while the controlled system gain does change by a multiplicative factor if the output metric used is miss ratio $MR(k)$ instead, the overall loop gain remains the same. This is because the designed controller gain in this case is multiplied by the inverse of that

[1] It is possible to include external workload dynamics in the model by modeling the admission control process, however we found this is necessary only when the workload changes significantly over a very short period of time, which is not the case for most distributed system. Extension on this aspect is left as future work.

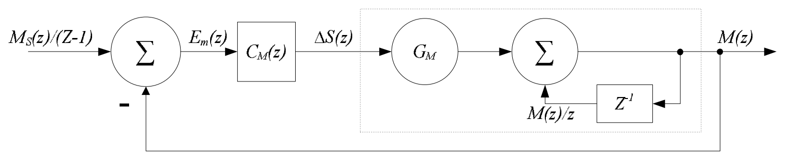


Fig. 2. Deadline Miss $M(k)$ Feedback Control Loop

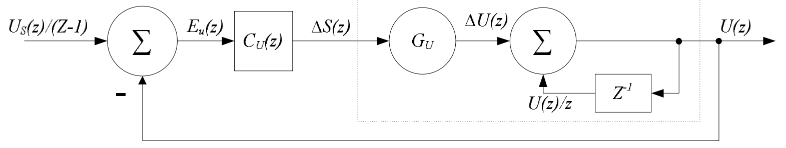


Fig. 3. CPU Utilization $U(k)$ Feedback Control Loop

factor. With the above observation in mind, the discussion below applies to both miss ratio $MR(k)$ and miss number $M(k)$ control. When the system is underutilized, we use the distributed utilization feedback control loop shown in Figure 3. $C_U(z)$ is the CPU utilization controller to be designed and U_S is the CPU utilization set point.

In z-transform notation we have:

$$\begin{aligned} M(z) &= H_M(z)\tfrac{M_S z}{z-1} \; where \; H_M(z) = \tfrac{C_M(z)G_M}{1-z^{-1}+C_M(z)G_M} \\ U(z) &= H_U(z)\tfrac{U_S z}{z-1} \quad where \; H_U(z) = \tfrac{C_U(z)G_U}{1-z^{-1}+C_U(z)G_U} \end{aligned} \tag{9}$$

Here the gains G_M and G_U are assumed to be set at some fixed values for nominal control design and analysis. Because our system intrinsically has an integral part, it is enough to use only a proportional controller to design $C_M(z)$ and $C_U(z)$ to guarantee the stability and zero steady state error. The general forms of the digital proportional controller in the time domain and z-domain are: $\Delta S(k) = K_p E(k)$(time domain)and $C(z) = K_P$ (z-domain). Here we denote K_M as the proportional term for the miss controller and K_U for the utilization controller. These values are substituted in Equations 9. Setting $C_M(z) = K_M$ and $C_U(z) = K_U$ we get:

$$\begin{aligned} M(z) &= H_M(z)\tfrac{M_S z}{z-1} \; where \; H_M(z) = \tfrac{K_M G_M z}{(1+K_M G_M)z-1} \\ U(z) &= H_U(z)\tfrac{U_S z}{z-1} \quad where \; H_U(z) = \tfrac{K_U G_U z}{(1+K_U G_U)z-1} \end{aligned} \tag{10}$$

3.6.2 Stability

According to control theory, system performance is determined by the poles of the closed loop transfer function. From Equations 10, we get $1/(1 + K_M G_M)$ as the pole for $H_M(z)$ and $1/(1 + K_U G_U)$ as the pole for $H_U(z)$. Since these poles are inside the unit cycle, according to the control theory, the stability is ensured in the DFCS system.

3.6.3 Steady State Error

Based on the Final-Value Theorem, the steady state values of and are:

$$\begin{aligned}\lim_{k\to\infty} M(k) &= \lim_{z\to 1}(z-1)M(z) = \lim_{z\to 1}\left\{(z-1)\tfrac{K_M G_M z}{(1+K_M G_M)z-1}\cdot\tfrac{M_s z}{z-1}\right\} = M_S \\ \lim_{k\to\infty} U(k) &= \lim_{z\to 1}(z-1)U(z) = \lim_{z\to 1}\left\{(z-1)\tfrac{K_U G_U z}{(1+K_U G_U)z-1}\cdot\tfrac{U_s z}{z-1}\right\} = U_S\end{aligned} \tag{11}$$

This result theoretically proves that the DFCS system can bring the miss number $M(k)$ and CPU utilization $U(k)$ to their set point(M_S and U_S)in steady state with zero error. It can also be verified that for a constant external disturbance $\Delta T(k) = \Delta T$, this asymptotic property still holds.

3.6.4 Settling Time

Settling time can be determined by the poles inside the unit cycle. The closer the pole is to the origin, the shorter the settling time. To deal with a worst case situation, we let $K_M G_M = 1$ and $K_U G_U = 1$, the poles of $H_M(z)$ and $H_M(z)$ are 0.5. According to control theory, the settling time is determined by the distance of the pole from the origin of the root locus plot. With a radius of 0.5, the theoretical settling time is about 8 sampling periods. In the experiment, based on the model, the calculated controller settings are 0.82 and 1.22 for the miss and utilization controllers, respectively.

3.7 Network Structures

For an effective distributed solution, we must consider the interaction among local controllers and the interaction between the global controller and local controllers. Two

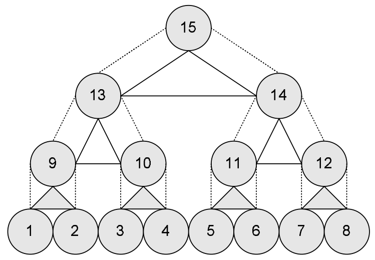


Fig. 4. Hierarchical Structure

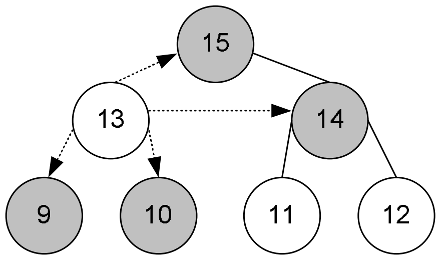


Fig. 5. Available routes in H-DFCS

aspects of the network to consider are the physical and logical network structure. The physical network structure could be a fully connected Ethernet, token ring, etc. For the purpose of this work, we are assuming that the physical structure of the network is either a hierarchy based point to point network, or a grid based point to point network running a Gigabit Ethernet.

The logical network structure defines information flow and route connectivity for the system. We propose two logical network structures: *hierarchical* and *neighborhood*, and design distributed real-time scheduling algorithm based on network structure.

3.7.1 Hierarchical Structure

Hierarchical Distributed Feedback Control System (H-DFCS) is based on the concept of information sharing in a hierarchical system. It allows a large distributed system to be broken down into a multi-level hierarchical system, as illustrated in Figure 4. By doing this, only the information that is required to coordinate subsystems needs to be exchanged at higher levels to coordinate the entire system. In the H-DFCS system, any node that has sub-nodes can be considered a coordinator. In the H-DFCS system, each node has a local feedback controller (LFC). The minimal requirement of this local controller is that it should be able to modify the service level set point of the node and report the local Miss Ratio (LMR) to other controllers (nodes).

The full scheduling algorithm for this system operates every sampling period in the following manner. Each node contains the LFC control system, with the exception of the top node in the hierarchy. This node contains the LFC control system as well as the DFC control system. The top node receives the MR and CPU utilization

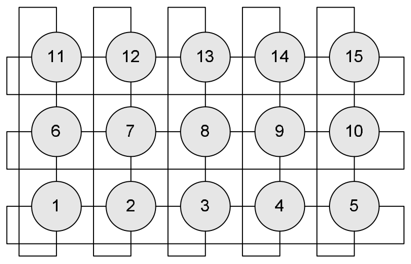


Fig. 6. Neighborhood Structure

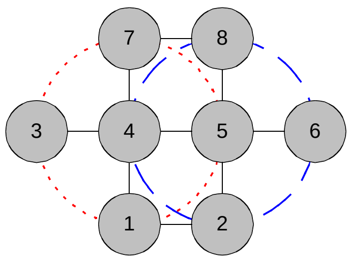


Fig. 7. Available routes in N-DFCS

averaged for the entire system and use this as inputs to its distributed controller to determine the new service level set point for the entire system. Such an operation allows the parent node to make decisions based on information from its children. The advantage is that the information from the children represents the subnet below that child, as the LMR values are weighted based on the number of nodes that the value represents.

Load balancing is achieved by migrating tasks between nodes through the network. Each node determines the route by comparing the MR values from the its children, parent and siblings. The MR values are weighted, to describe the number of nodes that they represent. This information is then used to assign a percentage to each entry in the route table, specifying the ratio of the off-loaded tasks to be sent along each route. In the hierarchical case that implements a binary tree, each node has up to four possible routes. For example, as shown in Figure 5, Node 13 has two routes to its two children, one to its parent, and one to its sibling. Routes are unidirectional, and are assigned only if the miss ratio of the other node in question is lower than that of the current node.

3.7.2 Neighborhood Structure

Neighborhood feedback control scheduling is based on the concept of information sharing in a neighborhood type system as shown in Figure 6. This means that a node shares its state information with its direct neighbors in the network and receives state information from these same neighbors.

Different from H-DFCS which has only one distributed controller at the root, in

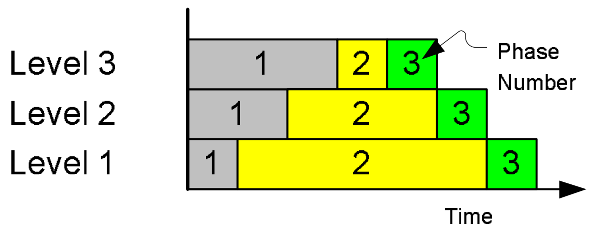


Fig. 8. H-DFCS in the 3-phase operation

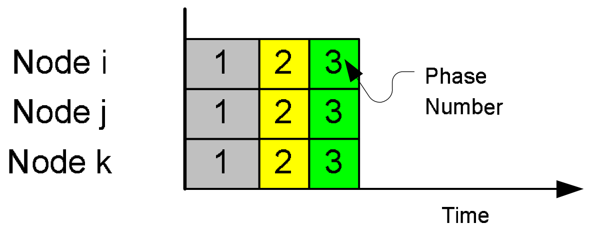


Fig. 9. N-DFCS in the 3-phase operation

neighborhood feedback control scheduling (N-DFCS) shown in Figure 6, every node contains both a local and distributed controller to control the state of the node as well as the state of its neighbor nodes. N-DFCS works within individual subnets instead of the whole distributed system. This design allows state information shared in a more decentralized manner. As a concrete example, Figure 7 shows the neighbors of two nodes (nodes 4 and 5) in a mesh structure (neighbors(4)=1,3,5,7 and neighbors(5)=2,4,6,8).

3.7.3 Comparison

Due to the difference in the network structure, H-DFCS and N-DFCS have different characteristics in terms of the delay and load balance.

- **H-DFCS:** Since H-DFCS needs to wait for the messages to be passed all the way up the hierarchy prior to making decisions and returning the set point, the duration of its operation depends on the number of levels in the hierarchy. For example, as shown in Figure 8, it takes seven time slots for a level-1 node finishes in a three-level hierarchy. This indicates that H-DFCS might not be quite suitable for some large scale systems. On the other hand, H-DFCS can make the SLR decision based on global information, leading to a quicker load balance.
- **N-DFCS:** N-DFCS allows a node to make a decision based on information from it nearest neighbors. The advantage here, is that state information is shared in a more decentralized manner. This helps in speeding up the computation time—as evident by comparing Figure 8 and Figure 9, but could result in a slower load balancing process. In comparison, the N-DFCS always works in a fixed time, regardless the number of nodes in the system.

4 Performance Evaluation

To evaluate the performance of DFCS, we compare N-DFCS and H-DFCS with two other well-known scheduling algorithms, QoS negotiation [3] and Dynamic QoS Management (DQM) [4]. Without a theoretical basis, QoS and DQM can be inefficient and difficult to tune. The results show that the feedback-control based approach is very effective. Due to space constraints, we are not able to show experimental results here. Please refer to [5] for complete evaluation results.

5 Related Work

Our DFCS architecture differs in two respects from early adaptive approaches. First, the performance of the adaptive system is modeled in a coarse-grained manner that represents the relation between aggregate QoS and aggregate resource consumption. This is different from fine-grained models, where the knowledge of individual task execution times is required as in [4]. Second, feedback control is used as a primary mechanism to adjust resource allocation in the absence of a priori knowledge of resource supply and demand. This is in contrast to early optimization-based QoS adaptation techniques that have assumed accurate models of application resource requirements. Examples of the new approach include [4,6]. In [6], a transaction scheduler called AED monitors the system deadline miss ratio and adjusts task priorities to improve the performance of EDF in overload. The DQM algorithm [4] features a feedback mechanism that changes task QoS levels according to the sampled CPU utilization or deadline misses. However, these algorithms are based on heuristics rather than a solid theoretical foundation.

Recently feedback control theory has been widely used as the underlying analytical foundation for building adaptive resource management systems. For example control theory has been applied to control throughput and delay [7] and to control congestion [8] at the network layer of the Internet. In addition, control-theoretic approaches have been adopted in a number of software systems such as realtime embedded systems [9,10], database servers [11], network storage systems [12]and web caching [2]. A survey on feedback performance control in software services is presented in [13]. Recent work on applying control-theoretic techniques in Real-Time systems is directly related to this work. In [14,1], feedback control real-time scheduling algorithms are developed to achieve deadline miss ratio guarantees in uniprocessor systems. Several recent paper also present feedback control algorithms (DEUCON [15]) and middleware (FC-ORB [16]) for enforcing desired utilizations of multiple processors in a distributed systems. There are several important differences between our work and those projects. First, those solutions control the resources by adapting the rates of end-to-end tasks. In contrast, our algorithms handle independent tasks via a combination of local QoS level adaptation and task (re)allocation.

6 Conclusions

To support data-driven applications with unpredictable and changing resource requirements, we develop here an effective computational resource management system, called DFCS, based on the feedback control. Different form other ad hoc approaches, DFCS has a basis in the theory. We have rigorously proven that our approach not only has excellent steady state behavior, but also meets stability, overshoot, and settling time requirements. We have demonstrated that DFCS is a better option for distributed resource management, than QoS [3] and DQM [4].

References

[1] C. Lu, J. A. Stankovic, T. F. Abdelzaher, G. Tao, Performance specifications and metrics for adaptive real-time systems, in: IEEE RTSS, 2000.

[2] Y. Lu, T. Abdelzaher, A. Sexana, Design, implementaion, and evaluation of differentiated caching services, accepted for IEEE Transactions on Parallel and Distributed Systems 15 (5).

[3] T. F. Adbelzaher, E. Atkins, K. Shin, QoS Negotiation in Real-Time Systems and Its Application to Automated Flight Control, IEEE Transactions on Computers 49 (11).

[4] S. Brandt, G. Nutt, T. Berk, J. Mankovich, A dynamic quality of service middleware agent for mediating application resource usage, in: RTSS '98, 1998.

[5] J. A. Stankovic, T. He, T. F. Abdelzaher, M. Marley, G. Tao, S. Son, C. Lu, Feedback Control Scheduling in Distributed Systems, in: IEEE RTSS, 2001.

[6] J. R. Haritsa, M. Livny, M. J. Carey, Earliest deadline scheduling for real-time database systems, in: IEEE Real-Time Systems Symposium, 1991, pp. 232–243.

[7] S. Keshav, A control-theoretic approach to flow control, in: Proceedings of the conference on Communications architecture & protocols, 1993, pp. 3–15.

[8] J. Wen, M. Arcak, A unifying passivity framework for network flow control (2002).

[9] L. Abeni, L. Palopoli, G. Lipari, J. Walpole, Analysis of a reservation-based feedback scheduler, in: RTSS '02, 2002.

[10] A. Cervin, J. Eker, B. Bernhardsson, , K.-E. Årzén, Feedback feedforward scheduling of control tasks, Real-Time Systems 23 (1-2) (2002) 25–53.

[11] S. Parekh, K. Rose, Y. Diao, V. Chang, J. Hellerstein, S. Lightstone, M. Huras, Throttling utilities in the ibm db2 universal database server, in: American Control Conference, 2004.

[12] M. Karlsson, C. T. Karamanolis, X. Zhu, Triage: performance isolation and differentiation for storage systems., in: IWQoS, 2004, pp. 67–74.

[13] T. F. Abdelzaher, J. A. Stankovic, C. Lu, R. Zhang, Y. Lu, Feedback performance control in software services, IEEE Control Systems Magazine 23 (3).

[14] C. Lu, J. A. Stankovic, G. Tao, S. H. Son, Design and evaluation of a feedback control edf scheduling algorithm., in: IEEE Real-Time Systems Symposium, 1999, pp. 56–67.

[15] X. Wang, D. Jia, C. Lu, X. Koutsoukos, Decentralized utilization control in distributed real-time systems, in: IEEE RTSS, 2005.

[16] X. Wang, C. Lu, X. Koutsoukos, Enhancing the robustness of distributed real-time middleware via end-to-end utilization control, in: IEEE RTSS, 2005.

Robust and Timely Communication over Highly Dynamic Sensor Networks

Tian He[§], Brian M. Blum[†], Qing Cao[‡],
John A. Stankovic[†], Sang H. Son[†] and Tarek F. Abdelzaher[‡]

[†]Department of Computer Science, University of Virginia
[§]Department of Computer Science and Engineering, University of Minnesota
[‡]Department of Computer Science, University of Illinois, Urbana-Champaign

Abstract

Highly dynamic sensor networks, such as mobile robotic sensor networks, have been applied in various kinds of application scenarios such as real-time planet exploration and deep-ocean discovery. In these types of networks, mobility and energy management protocols change the connectivity among the neighboring nodes quickly. Traditional state-based protocols, designed for static and/or low-mobility networks, suffer excessive delay in updating their routing or neighborhood tables, leading to severe packet loss and communication delay in the highly dynamic situations. To provide robust and timely communication, we exploit the concept of *Lazy-Binding* to deal with the elevated network dynamics. Based on this concept and the knowledge of the node positions, we introduce Implicit Geographic Forwarding (IGF), a new protocol for highly dynamic sensor networks that is altogether *state-free*. We compare our work against several typical routing protocols in static, mobile and energy-conserving networks under a wide range of system and workload configurations. In the presence of mobility and other dynamics, IGF achieves as much as 10 times improvement in the delivery ratio and significant reduction in both the end-to-end delay and control overhead. In addition to extensive simulations, we also implement and evaluate the IGF protocol on the Berkeley mote platform.

I. INTRODUCTION

Highly dynamic sensor networks, such as mobile robotic sensor networks, have been widely used to explore environments that are difficult and dangerous for humans. In the exploration missions of the red planet, scientists employ the robotic sensor devices (Rover) to discover the possibility of water activity. In the deep-ocean exploration, robotic sensors are used to access the risk of potential earthquakes, volcanoes and tsunamis. In addition, robotic sensors are used to investigate dangerous sites such as radioactive environments and mine fields. These robotic sensors are normally equipped with various kinds of sensors such as cameras, spectrometers and magnetometers. They can localize themselves in real-time through either GPS [1] or position tracking [2]. Distributed data processing, such as the collaborative exploration for map construction [3], requires a team of robotic sensors to communicate with each other constantly in real-time. This type of communication imposes several challenges. First, when robotic sensors move, the constant changes of the connectivity among the neighboring nodes make it difficult to maintain freshness of the routing states in traditional state-based routing protocols. Second, energy management protocols [4]–[6] transit the robotic sensors into and out of sleep states, and their participation in the network becomes probabilistic at any given point in time. These unique challenges demand a new routing solution. In this paper, we exploit the concept of **lazy-binding**, which is widely used in other research areas, such as the programming language design and operating systems. Specifically here, we define *lazy-binding as deferring mapping the system physics (e.g., the network topologies) into the volatile states (e.g., the route state), required by a certain operation, to the last possible moment allowed by the operations.* Since lazy binding defers binding the volatile states as late as possible, it enables the system to cope with real-time changes in the network topology. Our first installment based on this concept is a location-based routing protocol, called Implicit Geographic Forwarding (IGF). IGF allows a sender to determine a packet's next-hop online in real-time. By combining lazy-binding and location-address semantics, IGF becomes a pure state-free protocol, which does not depend on the knowledge of the network topology or the presence/absence of other nodes. This characteristic of being state-free is valuable to the highly dynamic sensor networks, as it supports fault tolerance and makes protocols robust to real-time topology shifts or node state transitions. Further, a state-free solution eliminates the bandwidth-consuming packets required in the state-based solutions for routing and neighbor table upkeep.

The remainder of the paper is organized as follows: Section II motivates the need for lazy-binding in highly dynamic networks. Section III describes the IGF protocol in detail. Section IV presents our simulation experiments and analysis in mobile and other environments. Section V describes our implementation on the MICA2 platform and its evaluation. We discuss the state-of-the-art and future work in Sections VI and VII, respectively. We conclude the paper in Section VIII.

II. THE MOTIVATION FOR LAZY-BINDING

Advances in the protocol design [7]–[11] continuously expand our ability to deal with the high dynamics inside the network. These protocols have been designed with robustness in mind, however, the level of fault tolerance is usually designed to adapt to occasional node failures and infrequent topology migration. In order to cope with the elevated transition of network topologies, the state-based solutions are required to refresh the routing states in real time to reflect changes, which consequently introduces significant overhead and network congestion. Eventually, the performance of these algorithms might degrade dramatically, as the real-time maintenance of the routing states might not keep up with the transition rate of the network topologies. We observe that **the delay** between **the time when a physical network topology maps to the routing states** and **the time when these states are actually used for packet forwarding** is the root cause of state invalidation and routing failures. We term this delay as the binding delay. A long binding delay leads to a high probability that recorded states are invalid by the time they are used. This problem increases as the network dynamics increase. In addition, since routing states are volatile and become outdated at a much faster rate in highly dynamic networks, it is inefficient to maintain state proactively and eagerly. According to the binding time, we categorize the routing protocols into four categories as shown in Figure 1.

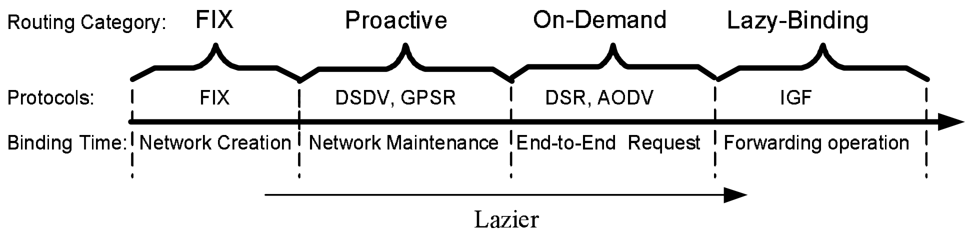


Fig. 1. Difference in Binding Time

- Fixed routing schemes are rarely used due to their rigid early-binding at the deployment time. The binding delay of this type of network could be infinite.
- Proactive schemes such as DSDV [10] and GPSR [11] maintain the network states aggressively. The routing states are refreshed regardless whether there is need of data delivery. This eager and proactive binding property is suitable for the networks with a small rate of topology change. The binding delay is the interval between consecutive routing updates.
- On-demand algorithms such as DSR [8], AODV [9] and Directed Diffusion [7] bind the routing state to physical topologies with a lazier approach - On-demand Route Discovery. The on-demand property allows them to defer the binding of the routing states to the physical network topologies until there is a need for end-to-end delivery. Those schemes have been proven effective [12] in dealing with moderate mobility and failures. The binding delay of on-demand algorithm is the time since the route discovery.
- Different from the on-demand schemes, the IGF protocol proposed in this work goes one-step further. It defers the binding of the routing states to the physical network topology *until the packet forwarding operation actually happens at a sending node.* This design allows: 1) The elimination of the communication overhead to maintain the state proactively, reducing the unnecessary update of the volatile routing states, 2) The real-time detection of the node failure, migration, and transition into a sleep state and 3) The real-time utilization of recently awoken or newly arriving nodes.

III. IGF PROTOCOL DESIGN

In this section, we introduce the IGF protocol as an exemplar instance of the lazy-binding concept applied to routing.

A. System Model and Assumptions

IGF is targeting to the high-end sensor networks (e.g., mobile sensor networks), where each sensor node can obtain its location (x, y) through GPS [1] or a position tracking technique [2]. The IGF communication supports the location-address semantic, in which locations are specified as the routing destinations, instead of using a particular node ID. This location-address semantic are valid in many sensor networks, because sensor data, such as temperature readings, are normally tagged with the location-context, and therefore can be addressed directly by the location, eliminating the overhead to translate the target destinations into a set of node IDs. Since the packet size in high-end sensor networks is relatively large, our main design uses RTS-CTS handshaking to avoid the hidden and exposed terminal problems in wireless communication [13], and an alternative solution for small-packet delivery (e.g., Tinyos Message) is discussed separately in Section III-H.4. For the sake of simplicity, we describe IGF in Section III-B assuming a sufficient node density. The issues related to the density, radio irregularity and localization error are resolved in the later sections.

B. IGF Details

We begin our introduction of IGF with an straightforward example. Figure 2 depicts a scenario where the node S is transmitting a packet towards the final destination D. We define the dark nodes within the 60-degree sector[1] as forwarding candidates (We address the case when there are no candidates inside the specified forwarding area in section III-G). Among these candidates, we highlight two nodes, R and A, to represent the chosen next-hop and an alternate ”competing” node, respectively. In addition, the gray node N represents a node within the communication range of S that is not a candidate node. When the node S initiates a packet transmission, the communication handshake goes through following steps: (the timeline of the IGF Handshake is shown in Figure 3)

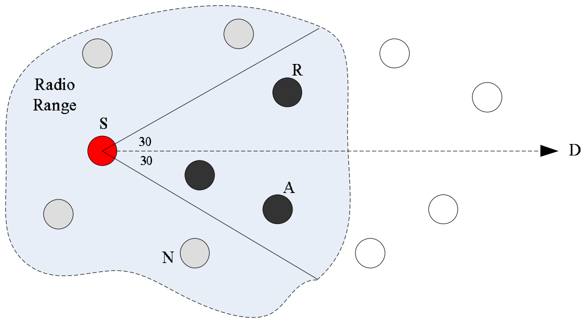


Fig. 2. Forwarding Area for Source S

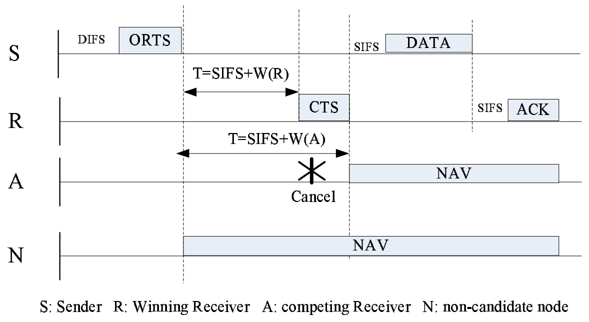

S
DIFS
ORTS
SIFS
DATA
T=SIFS+W(R)
R
CTS
SIFS
ACK
T=SIFS+W(A)
A
Cancel
NAV
N
NAV
S: Sender R: Winning Receiver A: competing Receiver N: non-candidate node

Fig. 3. IGF Handshake

1) ORTS PHASE: With modifications to the 802.11 DCF MAC protocol, the IGF handshake begins when the sender S's Network Allocation Vector (NAV) timer is zero and it senses an idle channel for DIFS time; at this point the node S sends, via broadcast, an Open RTS (ORTS) packet. This ORTS packet contains the locations of the sending node S and the final destination D.
2) CTS-WAIT: While all nodes within the communication radius of the node S receive and process this ORTS packet, only the forwarding candidates (dark nodes) set a CTS_Response timer (T_{cts_wait}). This timer controls an appropriate amount of time that a forwarding candidate must wait before responding to the received ORTS packet. The value of T_{cts_wait} can depend on the link quality, the progress in distance towards the destination and/or the energy remaining at the potential receiver. The nodes that are not forwarding candidates (gray nodes) set their NAV timer in accordance with 802.11 semantics to avoid interference with this ongoing transmission.
3) CTS: While all forwarding candidates set their CTS_Response timer, only a single node, with the shortest T_{cts_wait} value (the node R in the Figure 2 scenario), responds to the ORTS with a CTS packet. To prevent multiple responses, other forwarding candidates overhearing this CTS packet cancel their timers and set their appropriate NAV timers. In addition, the sender S, having already received a valid CTS packet, ignores further CTS packets heard in response to the now antiquated ORTS. We consider the IGF lazy-binding done, when the sender S decides that the node R is the receiver for this packet.
4) DATA: After the sender S is bound with a specific receiver (R), the sender S sends DATA to the node R.
5) ACK: The node R acknowledges the sender S, if DATA is received successfully.

[1]We choose this 60-degree sector because it is the maximal angle that ensures the distance between any two points within the sector is smaller than nominal communication range (for the purpose of overhearing). In case of high node density, a smaller angle can be chosen as well.

C. More on Forwarding Candidates

This section gives a detailed discussion on how a node determines whether it is a valid forwarding candidate. We prefer that every node within the forwarding area is capable of hearing one another, to prevent the interference among the forwarding candidates. Accordingly, as depicted in Figure 2, we choose the candidate nodes that reside within a ± 30-degree angle of the line connecting the sender and the final destination.

Using the sender and the receiver's own location, as well as the final destination location, each node (e.g., the node R) apply simple trigonometry to test whether itself is within the forwarding area. The formula to calculate the angle $\angle RSD$ in Figure 2 is:

$$Degree_{\angle RSD} = \text{acos}(\frac{|SR|^2 + |SD|^2 - |RD|^2}{2|SR||SD|}) \tag{1}$$

In the ideal case, the shape of the forwarding area ensures all forwarding candidates, responding to an ORTS packet, are located within the communication range of one another; this eliminates the chance multiple CTS are sent in response to a single ORTS packet. However, in reality, due to an irregular communication radius [14], the node A might still fail to know that a response to the node S's ORTS has already been transmitted by the node R. In this rare case, the sender S needs to resolve duplicate CTS packets by choosing only one of those responses.

As stated before, the neighboring nodes that receive an ORTS, but are not within the forwarding area, simply set their NAV timer to reflect the duration of communication. This prevents collisions due to the hidden terminal problem [13].

D. More on Setting Response Wait Times

This section provides more discussion on how to set the T_{cts_wait} value. Having determined that it is within the forwarding area of communication, a node can adopt different policies in setting its T_{cts_wait} according to any combination of available metrics including the reception quality of the link, the progress in distance toward the destination, the energy remaining at the receiver, the statistics of packet loss, the processor load or the single hop delay. While many metrics can be used to decide the T_{cts_wait} delay according to the application specifics, without loss of generality, in the current IGF implementation, we adopt following formula:

$$\begin{aligned} F &= \frac{W_P * (1 - progress/radius) + W_R * rand()}{W_P + W_R} \\ T_{cts\_wait} &= SIFS + (DIFS - SIFS) * rand() \end{aligned} \tag{2}$$

In Equation 2, $progress$ is the advance in distance toward the destination; $Radius$ is the nominal radio range. $Rand()$ generates a random number between 0 and 1; W_P and W_R are the weights of progress and randomness, respectively; A higher W_P tends to favor a few nodes that make most progress, leading to short routes, however, possibly less traffic distribution and lower link quality. A higher W_R distributes the traffic to more nodes, leading to a better load balance, however, possibly longer routes. The tradeoff between W_P and W_R depends on the link quality and traffic distribution. We leave the investigation on this tradeoff as future work. $SIFS$ delay is the Short Inter Frame Spacing and $DIFS$ delay is the Distributed Inter Frame Spacing as defined in the 802.11 standard. Equation 2 probabilistically allows the nodes that relay packets further to wait for a smaller period of time before responding. In addition, the randomization included in Equation 2 can disperse the system workload among multiple equally eligible nodes. It should be noted that Equation 2 is designed to be compatible with the timing rule of 802.11 DCF by guaranteeing: 1) The minimum value of T_{cts_wait} is larger than or equal to the SIFS delay. 2) The maximum value of T_{cts_wait} is smaller than the DIFS delay to prevent other nodes from initiating a new transmission.

E. More on identifying a unique candidate

If more than two forwarding candidates choose similar T_{cts_wait} values (within propagation delay τ), the transmission of CTS would overlap each other, leading to collision. This section provides analysis on the chance of collision under different node densities. Here we use the time slotted approach (e.g. in ALHOA and CSMA) to analyze the performance of the contention-based protocols and establish a system model. The analytic result from the slotted approach serves as the worst-case bound of the un-slotted case. Let N_{node} be the average number of competing nodes within the forwarding area and K_{slot} be the number of back-off slots. A CTS packet encounters a collision when it overlaps with the transmission of at least one other CTS packet from other competing nodes (two or more CTS packets choose the same slot). A unique candidate can be identified as long as the sender receives at least one CTS response from any node within the forwarding area. According to the generalized birthday problem [15], the expected number of slots containing exactly one CTS packets with N_{node} competing CTS packets is :

$$E(N_{node}) \ = N_{node}(1 - \frac{1}{K_{slot}})^{N_{node}-1} \tag{3}$$

According to Equation 3, we plot $E(N_{node})$ values under different N_{node} and K_{slot} settings. Figure 4 suggests that the collision-free slots increase almost linearly with the total slots available.

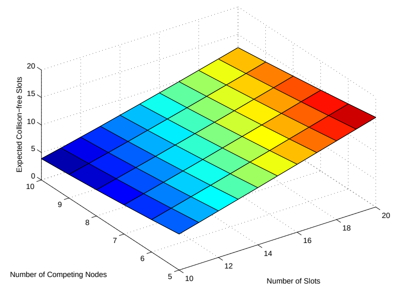


Fig. 4. Expected number of slots that are collision-free

We also simulate the process to identify a unique candidate. Figure 5 shows the probability of success under different N_{nodes} and K_{slots} values. Figure 5 indicates that with a sufficient and reasonable number of back-off slots (e.g., 20) the success ratio approaches 100%.

Although the chance of collision is very small under our design, it is still possible we can not identify a candidate uniquely in some cases. In this case, the standard 802.11 time-out mechanism will be activated to retransmit a RTS message.

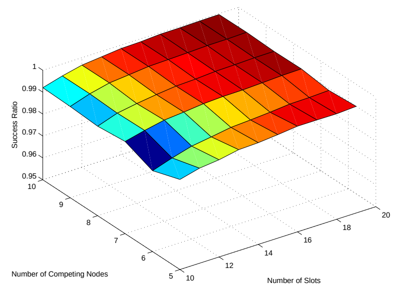


Fig. 5. Success ratio in identifying unique candidates

F. About lazy binding in IGF

IGF is an extension of the location-based protocols with the addition of lazy binding. In the location-based protocols such as GPSR, routing depends on up-to-date local neighborhood tables. Normally the neighbor table is updated through periodic beaconing. The binding of a specific forwarding node to a certain geographic location is eagerly established when the neighboring nodes exchange beacons. This eager-binding would be invalid quickly due to node mobility or sleep state transitions, which lead to stale routing information and unnecessary beacon exchanges. Moreover, this eager binding is not synchronized with the packet forwarding operations. In GPSR, AODV, DSR, if the chosen forwarding node of a sender fails or moves out of range, the MAC layer of the sender drops the packet and notifies the network layer about the routing failure. The network layer has to resolve this failure by attempting another backup route if available, which might be invalid too, or alternately waiting for an update to the neighborhood table, suffering a latency proportional to the beacon period in a scale of seconds. In contrast, IGF adopts lazy binding to discover the next hop the instant it is needed. The worst-case back-off delay introduced by lazy binding is tens of microseconds according to the 802.11a standard. This is four orders of magnitude shorter than the period of the neighbor table update through beaconing found in other protocols. The worst-case back-off delay in our implementation on the MICA platform is higher due to a low data rate; however, it is still two orders of magnitude shorter. In addition, the route maintenance found in Directed Diffusion [7], DSR [8], AODV [9] and LAR [16] normally takes at least tens of milliseconds or seconds to fix a broken link (depending on the size of network and the cause of failure). In contrast, IGF binds the node that is able to forward the packets, moments before (about 50us) the actual forwarding operation takes place. This lazy binding property dramatically reduces the chance that packets are forwarded to a node that fails or moves out of range. As a result, IGF shows as much as 10 times performance improvement in the delivery ratio when compared with several classical and state of the art solutions in the presence of high rate changes of the network topology.

G. Optimizations for Sparse Networks

IGF targets sensing-covered dense sensor networks in which greedy forwarding has been proven to guarantee delivery [17]. However, we note that without the capability of circumventing voids (e.g., the absence of forwarding candidate nodes), IGF results in communication failure in sparse sensor networks. The stateless property of IGF precludes utilizing the perimeter-forwarding rule for planarized graphs, such as the one used in GPSR [11].

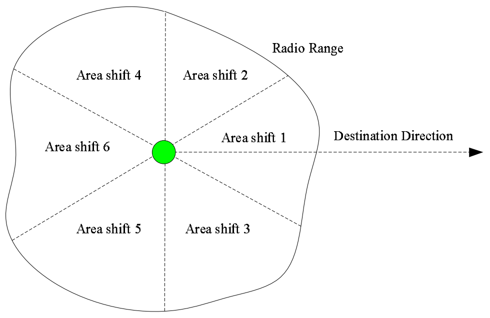


Fig. 6. IGF Handshake

To improve the delivery performance under sparse sensor networks, we have designed and implemented a history-based forwarding area shift technique in IGF. This mechanism activates when a void is detected through a MAC layer notification of failure to IGF. The sender then retransmits the packet, requesting a shift of the forwarding area to *search* for an available receiver. The sequence of shifts is shown Figure 6. Those shifts allow IGF to utilize communication areas outside of the initial forwarding area. Since the area shifts allow backtracking, we must make sure that IGF is loop-free while maintaining the state-free property. Unfortunately, it has been proven in [18], a memoryless location-based routing algorithm is not loop-free if backtracking is allowed. To address this issue, IGF places a trace-history into the packet header to remember the nodes this packet visited recently, and no state is maintained in the nodes. To avoid the infinite loop, during the backtracking, a node choose the next hop forwarding node with an ID that is not in the trace-history. We note that this trace-history starts to accumulate only when the backtracking is activated, it does not incur overhead whenever greedy forwarding is possible. With this void avoidance capability, in the empirical study later shown in the evaluation, IGF is able to achieve a 100% delivery ratio with a small length of history added to a packet header.

H. Design Issues

This section completes our approach with several practical design issues.

1) Radio Irregularity: For the sake of clarity, IGF is described with a nominal symmetric radio range. However, IGF does work with asymmetric irregular range [14]. First, we enforce a symmetric channel by an ORTS-CTS-DATA-ACK handshaking procedure; second, though it is possible that an asymmetric channel among forwarding candidates still exists, which might introduce multiple CTS responses to a single ORTS, the sender can resolve the duplicate packets by simply choosing one and ignoring the others.

2) Localization Error Impact: IGF can be regarded as an extension to location-based protocols, whose performance can be affected by localization errors. Results from [19] show that the performance of the Geographic Forwarding (GF) protocol degrades as the localization error increases. In our evaluation section IV-E, we demonstrate our IGF scheme as well as GPSR achieves 100% delivery ratios in the presence of up to 50% radio range errors.

3) Energy Implications: The 802.11b standard allows a node to turn off the radio [20] after the node overhears a RTS packet that is not targeted to itself. However, IGF requires forwarding candidates to remain in the listening mode to overhear any CTS for about 5×10^{-5} seconds. This causes a slightly increase in the energy consumption. We note, however, that this increase is negligible when considering the higher delivery ratio, the reduction in control packets, and the smaller end-to-end delay we are able to achieve.

4) Alternative MAC Implementation: Without loss of generality, IGF is currently built and evaluated on the top of 802.11 DCF. Considering the bandwidth available in the mobile robotic sensor networks, it is a good solution in dealing the hidden and exposure terminal problems. However, we note that IGF is not bound to 802.11 DCF and it can be implemented with several existing MAC protocols. For example, IGF can be built on the MAC protocol suggested in [21] that uses the implicit ACK. In this scenario, forwarding candidates wait for a random delay before starting to relay a packet, and the one with the smallest delay forwards the data packet first. This data packet serves as both an acknowledgement to the sender and a cancellation signal to the rest of the forwarding candidates. (These handshaking sequences are shown in Figure 7). Based on [32], we have built the IGF protocol on the Berkeley mote platform and evaluated it with results shown in Section V.

5) Other Implications: Depending on the localization method used, IGF requires either additional energy [1] or more control messages [19] to localize the nodes, especially in the mobile environments. In addition, IGF doesn't fix the routes during the forwarding, which might lead to packet reordering due to the MAC contention. Consequently, the final receiver should ensure the data can be re-assembled correctly.

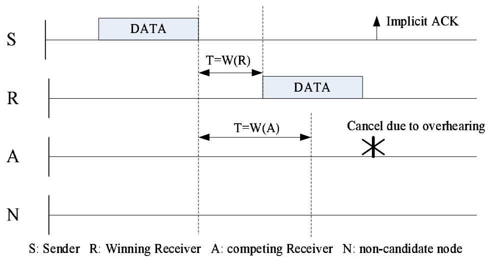

S
R
A
N
DATA
DATA
Implicit ACK
T=W(R)
T=W(A)
Cancel due to overhearing
S: Sender R: Winning Receiver A: competing Receiver N: non-candidate node

Fig. 7. IGF Handshake

TABLE I
SYSTEM PARAMETERS PARAMETERS

Parameters	Settings
Radio Range	$40.0m$
Terrain	$150X150m^2$
Collision Range	$71.2m$
Nodes	100 nodes,Uniform
Radio Range	$40.0m$
Bandwidth	$200kbps$
Radio	Lossy channel
Packet Size	32 byte Payload
W_P	W_P = 2
W_R	W_R = 1

IV. EXPERIMENTS AND ANALYSIS

To assess the performance, we implement the IGF protocol in GloMoSim [22], a simulator for wireless sensor, ad hoc, and mobile networks. GloMoSim provides a high fidelity simulation for wireless communication with detailed modeling of communication propagation, radio and MAC layers. In addition, we also implement the IGF protocol on Berkeley mote platform (section V).

To make our evaluation close to the latest Telos mote capability proposed for use in the WSN environments [23], we set our system parameters as shown in Table I. We expect the typical communication patterns inside a sensor network to be established based on request and retrieval semantics for data delivery between sensor nodes and a querying entity. One-to-one, many-to-one and many-to-many communication patterns are representative workloads in sensor networks. One-to-one communication happens when one node detects some activity that needs to be reported to a remote entity. Alternatively, a querying entity will require periodic reports from the whole sensor area, which take the form of many-to-one communication. It is more common that multiple applications run simultaneously and the traffic flows interleave with each other, shown by the many-to-many cross-traffic pattern.

We evaluate 120 system configurations under different traffic loads, node mobility and energy conserving schedules. For each configuration we average 60 runs with different random seeds (hence 60 different network topologies and node placements) to ensure adequate confidence of our results. The 90% confidence interval is within 3% to 10% of the mean for GPSR and IGF, and 8% to 15% for LAR. Due to the space limitation, we only present more complex and interesting many-to-many scenario ($40 \times 60 = 2400$ runs). The complete data set is available upon request. In the many-to-many tests, 6 nodes, randomly chosen from the left side of the terrain, send 6 CBR flows to 2 nodes (3 each) on the right side of the terrain. The average hop count is about $4 \sim 6$ hops. We note that most well-known sensor network protocols such as Directed Diffusion [7], TTDD [24] and TBF [25] are mostly designed for static sensor networks and have never been evaluated in mobile environments. For the sake of fairness, we choose the only protocols that evaluate the mobility extensively in their publications ([8], [11], [12], [16]). Moreover, since IGF is a location-based routing protocol, it is unfair to compare IGF with other ID-based procotocol. As a result, we decide to compare IGF with two protocols: 1) LAR [16] is a protocol optimized for mobility using the location information and it is suitable for sensor network; and 2) GPSR [11] is the standard location-based sensor network protocol with greedy and planar perimeter forwarding rules. We consider these protocols in three scenarios:

- A Static Network, where nodes are not mobile and energy conservation is not considered;
- A Mobile Network, with mobility ranging from walking to vehicular speeds;
- An Energy Conservation Network where nodes can transition into and out of dormant states.

For each experiment we choose three typical metrics on 1) the delivery ratio (the number of packets received / number of packets sent), 2) average end-to-end delay of received packets[2], and 3) overall communication overhead (total packets sent out by a node). In addition to the above experiments, we also evaluate the performance sensitivity of IGF in the presence of a low node density (voids), localization errors and location update delay in Section III-G, Section IV-E and Section IV-F, respectively.

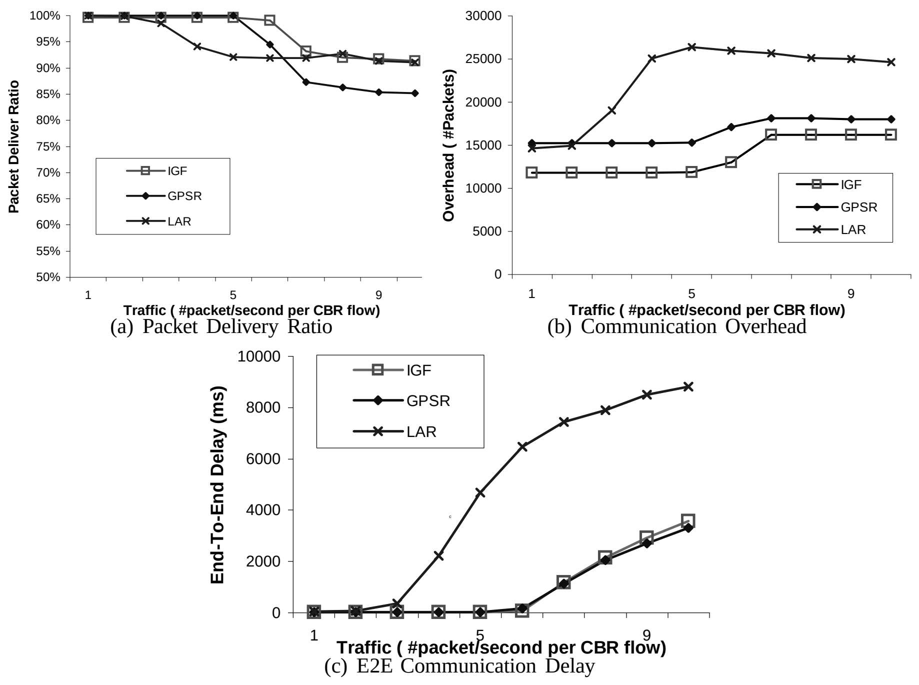


Fig. 8. Performance in Static Networks

A. Performance in Static Networks

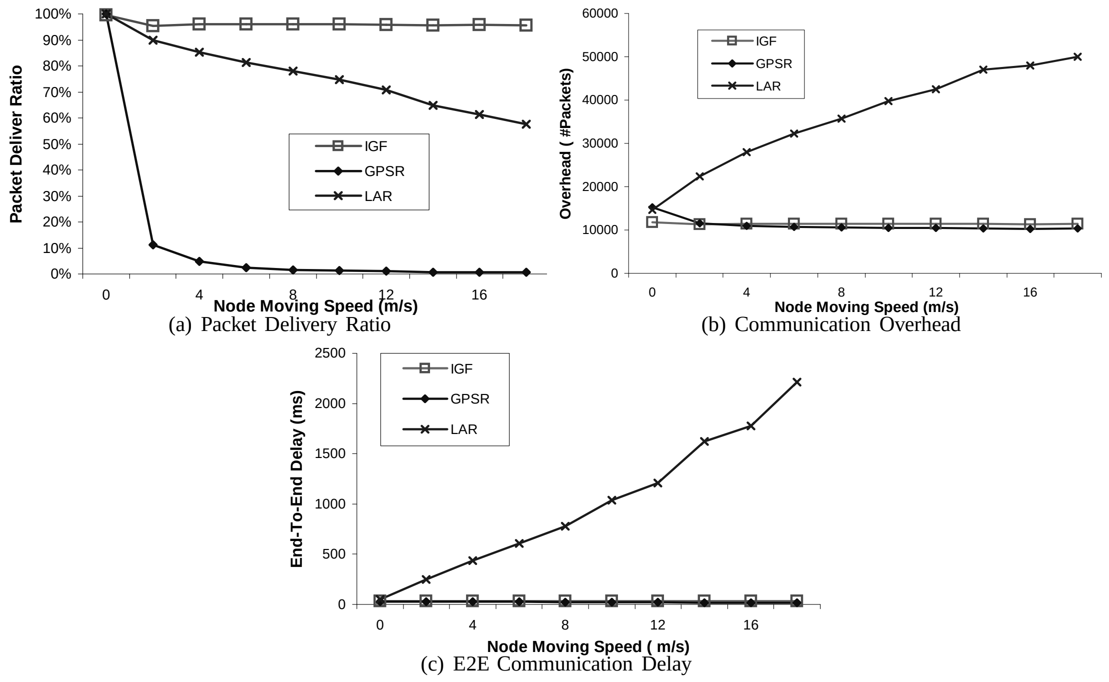
The evaluation in static networks shows that IGF performs as well as or slightly better than GPSR and LAR, when dynamics such as mobility and energy conserving sleep cycles are not considered. In these experiments, we increase many-to-many CBR flow rate until sufficient congestion is seen. Figure 8a shows that GPSR and IGF have comparable delivery ratios under light loads, while LAR loses packets early as these protocols quickly congest the network by sending route discovery packets. When the traffic flow rates increase enough to adequately congest the network in GPSR and IGF (6+ packets/second per CBR flow), performance in GPSR degrades due to limited intersecting routes, suffering additional collision caused by neighbor table update beacons (0.1 beacon per second). LAR uses location information to keep the effects of route discovery to a minimum allowing it to maintain delivery ratios comparable to IGF. However, LAR's frequent transmission of route discovery packets toward the destination, coupled with the latency incurred awaiting the route discovery response, lead to significantly more overhead (Figure 8b) and longer delay (Figure 8c) when compared with IGF. Figure 8b demonstrates IGF's savings at low traffic loads, as IGF does not require beaconing (GPSR) or route discovery packets (LAR) required in these protocols. As traffic loads increase, congestion increases the number of MAC layer collisions in both IGF and GPSR, resulting in retransmission attempts that add to overhead as shown in Figure 8b. In GPSR and IGF we see significantly lower end-to-end delay beyond 4 packets/second per CBR flow because LAR suffers latency awaiting the return of route discovery packets. Finally, under heavy traffic, we see a slightly longer delay in IGF over GPSR due to the fact that IGF manages to deliver 10% more packets (Figure 8a). We also note that Figure 8c demonstrates that the CTS back-off delay due to lazy binding in IGF has virtually no impact on the end-to-end delay.

B. Performance under Mobility

One scenario for the evaluation under high mobility would be a group of exploring robotic sensor nodes, trying to find the survivors underneath rubble after an earthquake. They periodically update their locations to each other while searching. Another scenario would be a group of mobile robots equipped with magnetic sensors, searching for mines in a battlefield.

[2]We note that our evaluation does not choose deadline miss ratios as the major metrics, because such an approach reveals less information about the tradeoff between actual delays and other system performance parameters

These robots report the detections to a base station by relaying packets among themselves. As we mentioned before, nodes' locations can be obtained through GPS in such a highly dynamic systems. We choose a standard waypoint mobility model during the simulations. It should be noted that in contrast to ad hoc networks where the mobility pattern is interleaved with burst movements and long pauses, sensor robots are normally continuously moving. To reflect this scenario, we set only a 1-second pause intervals between moves ($100 \sim 1000s$ pause intervals are normal settings in ad hoc network evaluations [1]). The settings stress-test the protocols' ability to deal with continuously high mobility and reflect the mobility patterns seen in mobile sensor networks. We model speeds up to 18 meters per second ($\sim 40mph$) to evaluate a wide range of mobile scenarios in which sensors can be attached to slow robots or to high-speed vehicles. We adopt a 40m range to confirm to current sensor ability, which is much smaller than 250m setting in used in WLAN ([1] and [15]). We note that the mobility is characterized by the number of neighborhood changes per second, which is affected by both the node speed and the radio range. With the same speed, a smaller range leads to a high mobility. To validate this point, in addition to this section, we investigate the impact of the radio ranges on the routing performance under mobility in Section IV-B.1.



(a) Packet Delivery Ratio (b) Communication Overhead

(c) E2E Communication Delay

Fig. 9. Performance in Mobile Networks

In the static network scenario (section IV-A), we use a low beacon rate (0.1 beacon per second) in GPSR to reduce the effect of congestion. To optimize GPSR to deal with mobility, we test GPSR with multiple beacon rates. Because beacons consume bandwidth, their cost offsets their savings, arriving at similar results in all beacon rates we tested. Consequently, we adopt 1 beacon per second to keep the state as fresh as possible without causing congestion. Rerouting is supported when a protocol experience a link break. From Figure 9a, we see that when nodes do not move (0 m/s), no packets are lost and the lowest delay and overhead are incurred due to minimal congestion. As we introduce mobility, increasingly affecting the validity of neighborhood and routing state with increased node speeds, we see the delivery ratios (Figure 9a) in GPSR and LAR drop off quickly while IGF continues to perform close to optimal. For example, when the node moving speed is at 4 meters/second, IGF demonstrates as much as 10 times performance gain in the delivery ratio over GPSR. For LAR, performance degrades as node migration invalidates eager-binding routes. Since LAR is specially designed to deal with mobility, its milder degradation, as seen in Figure 9a, results from location controlled flooding of route discovery packets to reestablish routes despite mobility. As an addendum to explain why GPSR performs so poorly, we note from Figure 9a that GPSR's delivery ratio quickly drops to zero at relatively low node speeds. One might assume this is because the beacon overhead leads to congestion in GPSR, hence a very low delivery ratio. However, from Figure 9b, we note that control overhead in GPSR is actually smaller than LAR. In fact, this low delivery ratio in GPSR happens because according to greedy forwarding rules in GPSR, the chosen next-hop node is normally located at the edge of the sender's communication radius. Because nodes are equally likely to move in any direction, there is a high chance that designated receiver will have moved out of communication range from the sender since the last beacon which was received seconds ago. Over multi-hop routes, the chances of failure grow exponentially. In contrast, IGF binds the next hop tens of microseconds before packet forwarding occurs. This significantly reduces the chance that a chosen node will move out of communication range during this tiny interval. Aside from the delivery ratio (Figure 9a), our evaluation shows that IGF significantly outperforms other protocols in metrics of overhead (Figure 9b) and end-to-end delay (Figure 9c) under all moving speeds. All these results are due to IGF's ability to defer the mapping between routing states and network topologies until this binding is absolutely required.

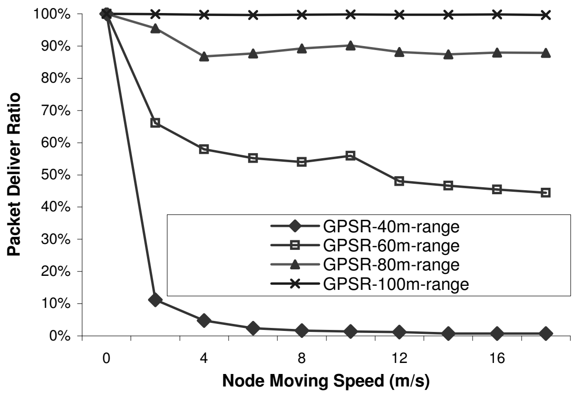


Fig. 10. Delivery Ratio under Different Radio Ranges and Speeds

1) Radio range impact on the routing performance under mobility: In this experiment, we investigate the impact of different radio ranges on the routing performance in mobile sensor networks. When nodes move around, mobility breaks old links and establishes new links. With the same node speed, a smaller radio range causes a higher rate of change in the network topology. Figure 10 proves that GPSR is able to achieve good deliver ratios with a large radio range, which leads to a smaller mobility. On the other hand, Figure 10 indicates GPSR's delivery performance reduces dramatically under high mobility situations.

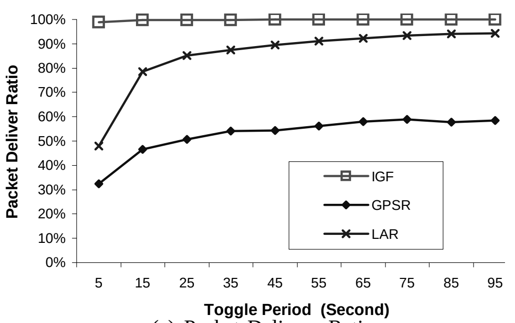
C. Performance under Energy Conservation

It is crucial for sensor network systems to support energy conservation. The most practical way to reduce total energy consumption is to turn on/off the nodes on demand of events [4]–[6]. However, these operations disrupt the network topologies. In this experiment, we test IGF, GPSR and LAR in the presence of orthogonal energy conserving protocols by periodically transiting nodes into and out of sleep states. To prevent congestion, and therefore isolate the effects of the awake-sleep transition in our analysis, we set the flow rate to 1 packet per second. We note that two key parameters in energy-conserving protocols can affect the routing performance:

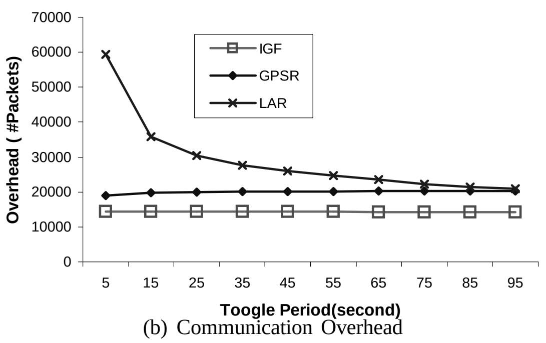
- **Toggle Period:** Toggle Period is the time interval between consecutive transitions into a sleep state. This parameter reflects how fast a routing state is invalidated due to sleep-awake transitions. We change this value from 5 seconds to 95 seconds in increments of 10.
- **Sleep Percentages:** The percentage of time a node is in the sleep mode. We note that sleeping can significantly affect the active node density, as this reduces the number of nodes participating in routing at any point in time.

1) Performance under Varied Toggle Periods: Figure 11a shows the results for many-to-many flows where the Sleep Percentage is set at 30% for varying Toggle Periods. It shows that IGF outperforms all other protocols at all toggle periods investigated. GPSR utilizes a beaconing mechanism to proactively bind network topologies into neighbor states. This binding can be quickly invalidated due to nodes' awake-sleep transitions. As a result, packets may be forwarded to nodes that were turned off since the last beacon and then dropped by the MAC layer. This leads to a poor delivery ratio in GPSR (Figure 11a). In LAR, a node requires the network layer to handle transmission failures by initiating route discovery. Due to the on-demand nature of those algorithms, LAR outperform GPSR, as the recently returned route discovery packet traverses nodes that are currently awake and therefore able to act as routers. Finally, we see IGF performing significantly better than other protocols, at times showing more than 3 times improvement in packets delivered when compared to GPSR. We attribute this performance to the IGF's ability to utilize whatever neighbors are currently awake en route to the destination. We note the Toggle Periods here only range from 5 to 95 seconds. When the Toggle Periods increase further, less dynamics are introduced into the network topologies and routing states can remain fresh for a longer period of time. In this scenario, higher delivery ratios are expected for other algorithms. Theoretically, when the Toggle Period approaches infinity, energy conserving networks become traditional static networks, for which we have shown performance comparisons in section IV-A.

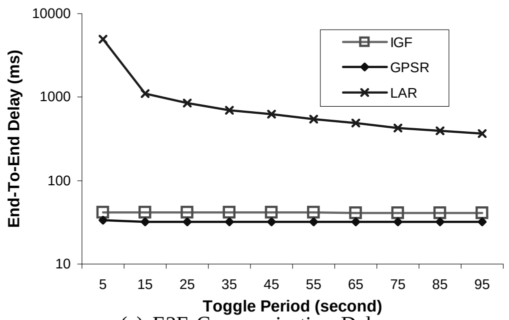
2) Performance under Varied Sleep Percentage: We next assess routing performance varying Sleep Percentage for the highly volatile case where the Toggle Period is set to 5 seconds. This not only allows us to compare our work under varied Sleep Percentage times, but allows us to stress test our protocol under highly dynamic system settings. In this experiment, we increase the sleep percentage of each node from 0% (always awake) to 100% (always asleep) in increments of 10%.



(a) Packet Delivery Ratio



(b) Communication Overhead



(c) E2E Communication Delay

Fig. 11. Performance under Varied Toggle Periods

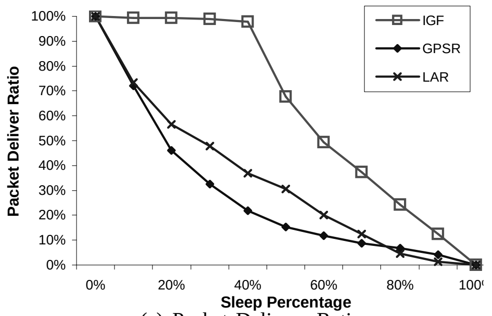
Figures 12a, b and c all demonstrate IGF's better performance over varied Sleep Percentages. Figure 12a shows that IGF delivers the highest percentage of packets under all Sleep Percentage settings, while incurring the small end-to-end delay (Figure 12c) and the lowest transmission overhead (Figure 12b). For example, Figure 12a shows that at a 50% sleep percentage, IGF delivers 340% more packets than the GPSR protocol. The drastic drop in overhead (Figure 12b) as seen in LAR also can also be attributed to this drop in the Packet Delivery Ratio. Since LAR is designed to adapt to occasional node failures, as we expect, in such highly dynamic networks, it takes a huge end-to-end delay to repeatedly fix these routes (Figure 12c). GPSR shows the lowest end-to-end delay (Figure 12c) because it delivers a tiny percentage of packets when compared to the IGF. Those packets go through the networks quickly by chance. Only IGF has a highest delivery ratio and a small delay. This is due to the fact that IGF can immediately detect node transitions into sleep states and immediately utilize recently awoken nodes.

D. Performance in Sparse Sensor Networks

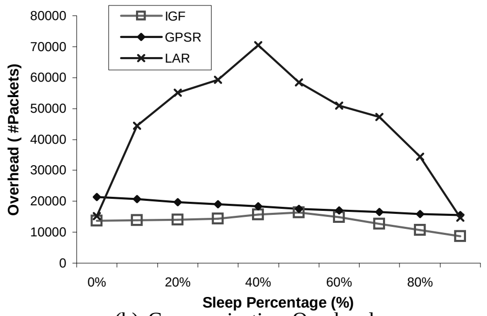
The typical density of sensing-covered sensor network systems [4] is about $20 \sim 25$ nodes/radio range in order to provide high fidelity in localization, detection and tracking. In previous evaluations, we use 22 node/radio ranges as a typical setting. However, it is important to understand how IGF performs under various node density settings. To prevent congestion, and therefore isolate the effects of density in our analysis, we set the per node flow rate to 1 packet per second. To change the density of the network, instead of increasing the number of nodes in the terrain, we keep the number of nodes constant at 100, and increase the side length of the square terrain from 100 meters to 250 meters in increments of 10. Figure 13 shows that with the history-based forwarding- area shifts, IGF achieves a 100% delivery ratio when the node density is larger than 12 nodes per nominal radio range. Figure 13 reveals that when density if relatively high (≥ 9 node/radio range), longer trace-history does not help much, however when the network become sparse, longer history can improve the delivery ratio.

E. Performance under Localization Errors

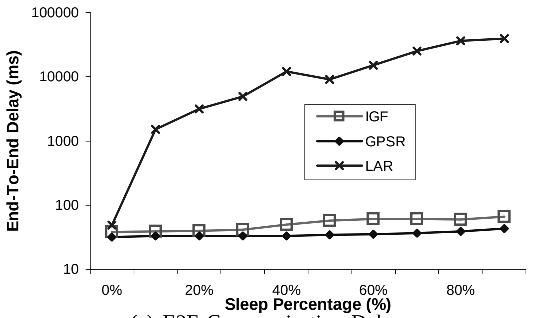
While most work in location-based routing assumes perfect location information, the fact is that erroneous location estimates are virtually impossible to avoid. In this experiment, we investigate location error impact on the IGF protocol. To prevent congestion, and therefore isolate the effects of the localization error, the traffic loads are set to the rate of 1 packet/second. We compare IGF, GPSR with the basic geographic forwarding(GF) [26], which forwards a packet to the node that makes the most progress toward the destination. We increase the localization error from 0% to 50% of the radio range in increments of 5% to measure the end-to-end delivery ratios. Figure 14 demonstrates that both the IGF and GPSR protocol perform much better in the presence of localization errors while the GF protocol suffers as location errors increase.



(a) Packet Delivery Ratio



(b) Communication Overhead



(c) E2E Communication Delay

Fig. 12. Performance under Varied Sleep Percentage

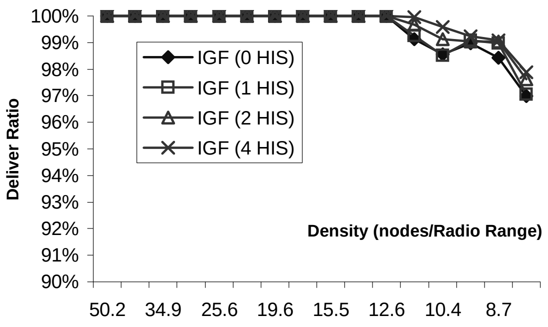


Fig. 13. Delivery Ratio Vs. Node Density

F. *Performance under Different Localization Update Intervals*

IGF obtains location updates from GPS or other localization schemes. Since the update rate affects the amount energy consumed to obtain the locations, the location is normally updated intermittently. Consequently, nodes have to make the routing decisions based on the last localization result, which might cause the routing failures if the the update delay is too long. In this section, we investigate the impact of the location update delay to the end-to-end delivery ratio. Figure 15 shows that the location update delay doesn't affect the static and energy conservation networks since nodes don't move in such networks. As for the mobile networks, a moderate location update delay (e.g., $\leq 1second$) doesn't noticeably affect the delivery ratio, however, a large delay cause more routing failures. Figure 15 also indicates that the impact of the update interval is affected by nodes' speed. With the same update intervals, a faster node speed leads to a lower delivery ratio.

V. IMPLEMENTATION ON MOTES

We have implemented the IGF protocol on the Berkeley motes platform [27] with a code size of 11,606 bytes (code is available through CVS at [28]). Currently, this implementation is built on the top of a MAC protocol with the implicit ACKs mentioned in section III-H.4. Three applications including data placement, target tracking and CBR data streaming are also built to run on top of IGF. Due to physical constraints and the un-availability of state-of-the-art protocols on such a platform, it is difficult to perform as extensive evaluation as we did in the wireless simulator. We, therefore, only present initial results here as a study for developing a more complete solution and evaluation in the future mote platform. As we mentioned in section III-B, IGF does not task a specific node to route packets a priori. This feature is beneficial for load balancing among the nodes inside the forwarding area. In this experiment, we use 25 motes to form a 5 by 5 grid. To evaluate the load balancing

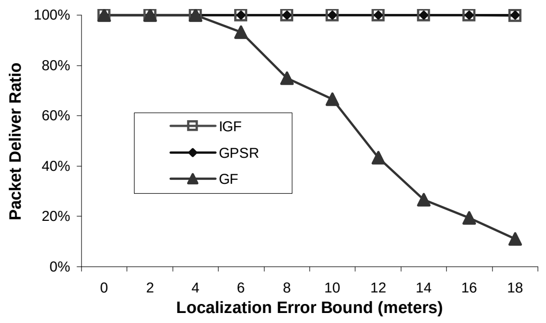


Fig. 14. Localization Error Impact

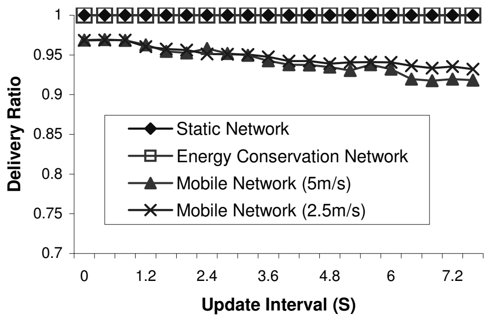


Fig. 15. Impact of Location Update Delay

capability of IGF we send a CBR data stream from node 24 to node 0, which is the base station. We collect the number of packets relayed by intermediate motes ($1 \sim 23$) and compare this with the result obtained from the GF protocol which we also implemented on the motes. While both GF and IGF achieve nearly 100% delivery ratio, GF tends to relay packets via a fixed route which might lead to unbalanced traffic. This is shown in Figure 16 as node 19 relays 250 packets while node 18 doesn't forward any packets. Instead, by distributing traffic loads, IGF effectively balances energy consumption. We argue that in sensor networks, balanced energy consumption can prevent some nodes from dying faster than others, therefore increasing the network lifetime.

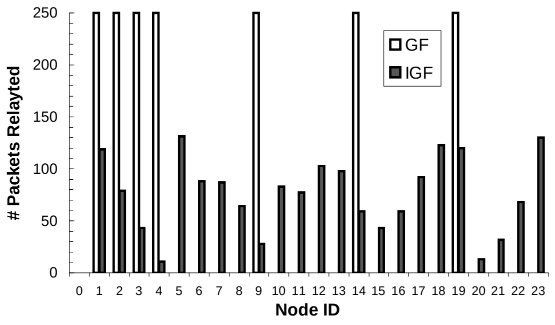


Fig. 16. Traffic Balance

VI. RELATED WORK

In this section we discuss prior research in distributed computing that is pertinent to the design of IGF. Various protocols [29]–[34] have been introduced to reduce packet loss through reliable communication in sensor networks. Alec Woo [33] chooses reliable routes based on link connectivity statistics obtained dynamically from a EWMA estimator. RMST [30] tracks packet fragments so that receiver initiated requests can be satisfied when individual pieces of an application payload get lost. PSFQ [31] caches packets along the path to the sender, initiating fragment recovery as required, starting with its local neighborhood. Robust data delivery [29] simultaneously sends packets along multiple paths at the expense of increases in communication overhead. While these ARQ/FEC-based solutions have proven effective when dealing with interference and collisions, their robust and reliable features might not be able to handle failures due to high dynamics in network topologies. We consider them orthogonal and complementary to our work.

Many routing algorithms have been proposed for ad hoc and sensor networks. With regard to the mechanisms used to bind network topology to the routing state, we divide these routing protocols into three categories. The first category we term as proactive eager-binding routing algorithms. DSDV [10] requires each node to proactively broadcast routing updates periodically. Global routing tables are refreshed regardless whether there is need for data delivery. Location-based routing algorithms such as GPSR [11] remove the requirement that a protocol maintains a global view of the network (i.e. end-to-end routing tables), therefore reduces communication overhead by eliminating its dependence on the network wide state information. However, they still depend on up-to-date local neighborhood tables, requiring control overhead to maintain such tables and suffering latency and packet loss when a node's neighborhood state changes between updates. To minimize unnecessary overhead incurred by proactive updates, a set of on-demand algorithms are proposed to defer route acquisition until data delivery is required. We term the second category as reactive eager-binding algorithms. It has been proved in [12] that AODV [9] and DSR [8] can successfully deal with moderate mobility with long pause intervals ($100 \sim 1000$ seconds). However, the eager binding of the routing states at the route acquisition phase make them less effective to deal with high dynamics in which network topologies change at a much faster rate than the duration of connections. Routing maintenance and rediscovery are proposed in [8] [9] to remedy this situation partially at the cost of higher delay and expensive control overhead. LAR [16] extends the on-demand idea proposed by AODV [9] and DSR [8], utilizing location information to limit the scope of route requests. While LAR significantly reduces routing overhead by only propagating queries to relevant portions of the network, it still needs to maintain or establish an explicit path before transmitting a packet. Current reactive eager-binding algorithms can successfully deal with occasional node failures and moderate mobility. However, the elevated dynamics due to the continuous mobility and power conservation inside sensor networks challenge researchers to develop a new category of routing protocols based on the lazy binding concept.

The first state-free protocol IGF belongs to this third category. ExOR [35] also decides the forwarding candidate on the fly. However, before transmitting a packet, the sender needs to specify the forwarding candidates in the packet header, which requires maintaining the state information about neighboring nodes. GeRaF [36] proposes a similar packet forwarding technique and it focuses on the multi-hop performance in terms of the average number of hops to reach a destination. Both ExOR and GeRaF do not model the effect of channel contention; while this work provides a detailed implementation and evaluation through both simulation and a running system.

VII. Future Work

In this work, IGF assumes a localization service or the GPS capability. This is justified as sensor network applications require location information to make sensor data meaningful. We note that lazy-binding is a general concept to deal with high network dynamics and its applicability does not intrinsically depend on the location service. It is promising to apply lazy-binding to ID-Based protocols such as Directed Diffusion [7]. To extend [7], we can keep the hop-count-to-a-sink as a non-volatile state with respect to the node failures, and we perform forwarding operations with the parents of each node. We note that in this ID-based case, the state-free property is not maintained, however, lazy-binding, which is independent of the state-free property, is still beneficial in dealing with the failure of the parent nodes. Due to the space constraints, we leave this as future work.

VIII. Conclusions

In highly dynamic sensor networks, the maintenance of freshness of routing states is costly. The state update, resulting from eager-binding, directly contributes to network congestion, wasting precious energy and increasing the end-to-end transmission latency. To prevent the adverse affects that dynamic factors such as high mobility have on the state-based eager-binding routing protocols, we advocate using the concept of lazy-binding to cope with high dynamics in sensor networks. Based on this concept, we introduce IGF, a unicast protocol that is altogether state-free. In simulation, we compare our work against protocols designed for mobile environments and sensor networks. IGF demonstrates more than 10 times improvement in the packet delivery ratio when the sensor network is highly mobile. IGF also achieves significant reduction in delay and overhead when considering mobility and energy-conservation. In addition, the IGF protocol has been implemented on the Berkeley motes platform to serve as an initial study in developing a more complete solution in the future.

Acknowledgments

This work was supported in part by NSF grant CCR-0098269 and NSF grant CSR-0615063.

References

[1] B. H. Wellenhoff, H. Lichtenegger, and J. Collins, *Global Positions System: Theory and Practice,Fourth Edition*. Springer Verlag, 1997.
[2] J. Borenstein, H. Everett, and L. Feng, *Navigating Mobile Robots: Systems and Techniques*. A. K. Peters, Ltd., Wellesley, 1996.
[3] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative Multi-Robot Exploration," in *In Proceeding of IEEE International Conferenceon Robotics and Automation (ICRA)*, 2000.
[4] T. He, S. Krishnamurthy, J. A. Stankovic, and T. Abdelzaher, "An Energy-Efficient Surveillance System Using Wireless Sensor Networks," in *The Second International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2004.

[5] T. Yan, T. He, and J. Stankovic, “Differentiated Surveillance Service for Sensor Networks,” in *First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, November 2003.
[6] F. Ye, G. Zhong, S. Lu, and L. Zhang, “PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks,” in *Proc. of International Conference on Distributed Computing Systems (ICDCS)*, May 2003.
[7] C. Intanagonwiwat, R. Govindan, and D. Estrin, “Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks,” in *the Sixth Annual International Conference on Mobile Computing and Networks*, 2000.
[8] J. Broch, D. B. Johnson, and D. A. Maltz, *DSR : The dynamic source routing protocol for multihop wireless ad hoc networks*. Ad Hoc Networks, Addison Wesley, 2001.
[9] C. E. Perkins and E. M. Royer, “Ad-hoc on demand distance vector routing,” in *WMCSA*, 1999.
[10] C. E. Perkins and P. Bhagwat, “Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers,” in *SIGCOMM*, 1994.
[11] B. Karp and H. T. Kung, “Greedy perimeter stateless routing for wireless networks,” in *International Conference on Mobile Computing and Networking*, 2000.
[12] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, “Performance comparison of multi-hop wireless ad hoc network routing protocols,” in *the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1998.
[13] I. . L. S. Committee, “Ansi/ieee, ieee 802.11 wireless local area networks,” http://grouper.ieee.org/groups/802/11/.
[14] G. Zhou, T. He, and J. A. Stankovic, “Impact of Radio Irregularity on Wireless Sensor Networks,” in *The Second International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2004.
[15] “Generalized birthday problem,” http://www.mathpages.com/home/kmath057.htm.
[16] Y. Ko and N. H. Vaidya, “Location-aided routing (lar) in mobile ad hoc networks,” in *International Conference on Mobile Computing and Networking*, 1998.
[17] G. Xing, C. Lu, R. Pless, and Q.Huang, “On greedy geographic routing algorithms in sensing-covered networks,” in *MobiHoc’04*, 2004.
[18] I. Stojmenovic and X. Lin, “Gedir: Loop-free location based routing in wireless networks,” in *Int. Conf. on Parallel and Distributed Computing and Systems*, 1999.
[19] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, “Range-Free Localization Schemes in Large-Scale Sensor Networks,” in *Proc. of the Intl. Conference on Mobile Computing and Networking (MOBICOM)*, September 2003.
[20] W. Ye, J. Heidemann, and D. Estrin, “An energy-efficient mac protocol for wireless sensor networks,” in *INFOCOM*, 2002.
[21] A. Woo and D. Culler, “A Transmission Control Scheme for Media Access in Sensor Networks,” in *Proc. of Mobile Computing and Networking (Mobicom)*, 2001.
[22] X. Zeng, R. Bagrodia, and M. Gerla, “Glomosim: a library for parallel simulation of large-scale wireless networks,” in *the 12th Workshop on Parallel and Distributed Simulations*, 1998.
[23] *Telos Mote IEEE 802.15.4 Compliant Mote Data Sheet*, CrossBow, http://www.moteiv.com/products/telos-brochure.pdf.
[24] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, “A Two-tier Data Dissemination Model for Large-scale Wireless Sensor Networks,” in *7th ACM MOBICOM Conference*, 2002.
[25] D. Niculescu and B. Nath, “Trajectory Based Forwarding and its Applications,” in *MobiCom*, 2003.
[26] G. G. Finn, “Routing and addressing problems in large metropolitan-scale internetworks,” USC/ISI, Tech. Rep. ISI/RR-87-180, 1987.
[27] *Mica2 data sheet*, CrossBow, 2003, available at http://www.xbow.com.
[28] T. He, L. Gu, B.Blum, and J. Xie, “Nest project source code,” http://sourceforge.net/projects/vert/.
[29] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, “Highly resilient, energy efficient multipath routing in wireless sensor networks,” *Mobile Computing and Communications Review*, vol. 1, no. 2, 2002.
[30] F. Stann and J. Heidemann, “RMST: Reliable Data Transport in Sensor Networks,” in *SNPA ’03*, 2003.
[31] C. Y. Wan, A. T. Campbell, and L. Krishnamurthy, “PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks,” in *WSNA ’02*, 2002.
[32] C. Wan, S. Eisenman, and A. T. Campbell, “CODA: COngestion Detection and Avoidance in Sensor Networks,” in *Sensys 03*, 2003.
[33] A. Woo, T. Tong, and D. Culler, “Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks,” in *First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, November 2003.
[34] J. Zhao and R. Govindan, “Understanding Packet Delivery Performance In Dense Wireless Sensor Networks,” in *Sensys 03*, 2003.
[35] S. Biswas and R. Morris, “Exor: opportunistic multi-hop routing for wireless networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 133–144, 2005.
[36] M. Zorzi and R. R. Rao, “Geographic Random Fowarding (GeRaF) for ad hoc and sensor networks: Multihop performance,” *IEEE Transaction on Mobile Computing*, vol. 2, no. 4, 2003.

Range-Free Localization

Radu Stoleru[1], Tian He[2] and John A. Stankovic[3]

[1] Department of Computer Science, University of Virginia,
stoleru@cs.virginia.edu
[2] Department of Computer Science and Engineering, University of Minnesota
tianhe@cs.umn.edu
[3] Department of Computer Science, University of Virginia,
stankovic@cs.virginia.edu

1 Introduction

Advances in micro-electro-mechanical systems have triggered an enormous interest in wireless sensor networks (WSN). WSN are formed by large numbers of densely deployed nodes enabled with sensing and actuating capabilities. These nodes have very limited processing and memory capabilities, limited energy resources and it is envisioned that they will be mass produced, to reduce costs.

Several challenging problems exist in wireless sensor networks. Among these is how to obtain location information for sensor nodes and events present in the network. From this perspective, we categorize the localization problem as: node localization, target localization and location service. Node localization is the process of determining the coordinates of the sensor nodes in the WSN. Target localization is the process of obtaining the coordinates of an event or a target present in the sensor network. The location of a target can be obtained either passively (the nodes sense the target) or actively, when the target cooperates and communicates with the sensor network. A location service acts as a repository that can be used to answer questions like “where is entity X?”. In the remaining part of this chapter we focus on the node localization problem in WSN.

Node localization is a complicated and important problem for wireless sensor networks (WSN). The aspects of this problem that have challenged the research community can be summarized as follows:

- **Assumptions** - The node localization problem remains a difficult challenge to be solved practically. To make the problem practically tractable, its complexity had to be reduced, by making simplifying assumptions. As a result, many localization schemes proposed solutions that are based on assumptions that do not always hold or are not practical. Examples of such assumptions are: circular radio range, symmetric radio connectivity, additional hardware (e.g., ultrasonic), lack of obstructions, lack of line-of-sight, no multipath and flat terrain.

- **Localization Protocol Design** - The problem of localization in WSN is further complicated by the large number of parameters that need to be considered when designing a localization system for a particular WSN deployment. Among these parameters are: the deployment method for the sensor network; the existence of a line-of-sight between sensor nodes and a remote, central point; the time required by the localization scheme; the presence of reference points (anchors) in the network, and the density; the cost for localization, represented by additional hardware (form factor) and energy expenditure (messages exchanged or time necessary for localization).
- **Cost/Accuracy trade-off** - Due to the mostly static nature of many WSN, obtaining the location information by each sensor node is often a one time or rare event. Adding hardware to each sensor node, to assist in the localization, is a costly solution, and, so far, has been ruled out from real system deployments. For example, GPS is a typical high-end solution, which requires sophisticated hardware to achieve high resolution time synchronization with satellites. The constraints on power and cost for tiny sensor nodes and the need for a line of sight from a sensor node to four or more satellites preclude this as a viable solution. Other solutions require per node devices that can perform ranging among neighboring nodes. The difficulties of these approaches are two-fold. First, under constraints of form factor and power supply, the effective ranges of such devices are very limited. For example the effective range of an ultrasonic transducer is on the order of a few meters, when the sender and receiver are not facing each other. Second, since most sensor nodes are static, i.e., the location is not expected to change, it is not cost-effective to equip these sensors with special circuitry just for a one-time localization.
- **Performance Evaluation** - The problem of localization in wireless sensor networks has been studied and evaluated predominantly in simulators. Due to the severe hardware constraints imposed on wireless sensor nodes, real system implementations of the proposed simulated solutions have not produced encouraging results. Solutions that use the most tempting means of evaluating relative distances between sensor nodes - RF signal strength, have largely failed in practice, due to the unreliable nature and irregular pattern of the radio communication. Localization schemes that are based on the receive signal strength indicator (RSSI) have been, however, intensively studied in simulators.
- **Security** - Since localization is a critical factor in WSN, attacks on it can render the sensor network ineffective. To date, very little work has been done on creating robust and secure localization schemes. A few notable exceptions are [15] [14] [17] [16] [5].

For wireless sensor networks ranging is a difficult option. The hardware cost (hardware used only for localization), the energy expenditure, the form factor, the small range, all are difficult compromises, and it is hard to envision cheap, unreliable and resource-constraint devices make use of range-based localization solutions. Their high accuracy in localization is very desirable, however.

To overcome the limitations of the range-based localization schemes, many range-free solutions have been proposed. These solutions estimate the location of sensor nodes by, either, exploiting the radio connectivity information among neighboring nodes, or exploiting the sensing capabilities that each sensor node possesses. Due to the distinct characteristics of these two approaches, we categorize the range-free localization schemes into: anchor-based schemes (which assume the presence of sensor nodes in the network that have knowledge about their location) and anchor-free schemes, which require no special sensor nodes for localization. The range-free localization schemes eliminate the need of high-cost specialized hardware on each sensor node. The fact that the radio propagation characteristics vary over time and are environment dependent, imposes higher calibration costs for the anchor-based localization schemes.

In this chapter we review a *representative* set of range-free localization schemes, from the perspective of the above proposed taxonomy: anchor-based and anchor-free solutions. We point out that hybrid solutions exist and, sometimes, one solution does not neatly fit in either one of the categories. Also, in addition to the localization schemes described below, many more have been proposed. To name a few: the ELA [32], Thunder [35], Hop-TERRAIN [26], KPS [7], RIPS [18], Resilient LSS [13], Robust Quadrilaterals [19] and MAL [23] . In the remaining part of this chapter, we use R to denote the radio range of a sensor node.

2 Anchor-Based Solutions

The location of a sensor node has to be expressed in a coordinate system. In a 2D space, three anchor nodes (three fixed points in the space) uniquely determine a coordinate system. In a 3D space, four anchor nodes are required. In this section, to demonstrate a wide range of possible solutions, we present several range-free localization schemes that use radio connectivity to infer proximity to a set of anchor nodes.

2.1 Centroid

The Centroid scheme was proposed by Bulusu et al. in [2]. This localization scheme assumes that a set of anchor nodes (A_i, $1 \leq i \leq n$), with overlapping regions of coverage, exist in the deployment area of the WSN. The main idea is to treat the anchor nodes, located at (X_i, Y_i), as point masses m_i and to find the center of gravity (centroid) of all these masses. In the most general form, the coordinates of the centroid of n point masses m_i are given by:

$$(X_G, Y_G) = \left(\frac{\sum_{i=1}^{n} m_i X_i}{\sum_{i=1}^{n} m_i}, \frac{\sum_{i=1}^{n} m_i Y_i}{\sum_{i=1}^{n} m_i} \right)$$

which, for equal masses m_i simplifies to:

$$(X_G, Y_G) = \left(\frac{\sum_{i=1}^{n} X_i}{n}, \frac{\sum_{i=1}^{n} Y_i}{n} \right)$$

An example of how the Centroid scheme works is shown in Figure 1, where a sensor node N_k is within communication range to four anchor nodes, $A_1 ... A_4$. The node N_k localizes itself to the centroid of the quadrilateral $A_1A_2A_3A_4$ (for the case of a quadrilateral, the centroid is at the point of intersection of the bimedians - the lines connecting the middle points of opposite sides).

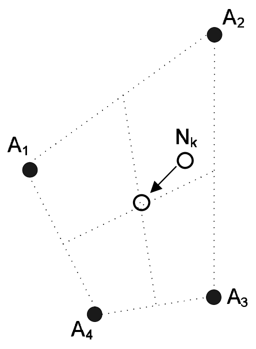


Fig. 1. Centroid localization - node N_k localizes to the centroid of the $A_1A_2A_3A_4$ quadrilateral.

The steps of the localization scheme are the following:

- Each anchor node A_i broadcasts its position.
- Each sensor node N_k listens for beacons from anchors and computes a connectivity metric, for each anchor node A_i it has received beacons from. This metric is defined as follows:

$$CM_{k,A_i} = \frac{N_{recv}(A_i, t)}{N_{sent}(A_i, t)}$$

  where $N_{recv}(A_i, t)$ and $N_{sent}(A_i, t)$ are the numbers of beacons received from anchor A_i and sent by anchor A_i, respectively, in the time interval t.
- A node N_k computes its location as the average of all the anchor nodes A_i it has heard from with a connectivity higher than a threshold, e.g., $CM_{k,A_i} > 90\%$, as follows:

$$(X_k, Y_k) = \left(\frac{X_{A_{i1}} + ... + X_{A_{ij}}}{j}, \frac{Y_{A_{i1}} + ... + Y_{A_{ij}}}{j} \right)$$

where j is the number of anchors with a higher connectivity than the threshold.

In subsequent work [3], the authors explored adaptive mechanisms for placing additional anchor nodes in a WSN, in order to reduce the average localization error.

2.2 APIT

APIT [8] is an area-based range-free localization scheme. It assumes that a small number of nodes, called anchors, are equipped with high-powered transmitters and know their location, obtained via GPS or some other mechanism. Using beacons from these anchors, APIT employs a novel area-based approach to perform location estimation by isolating the environment into triangular regions between anchor nodes as shown in Figure 2. A node's presence inside or outside of these triangular regions allows a node to narrow down the area in which it can potentially reside. By utilizing different combinations of anchors, the size of the estimated area in which a node resides can be reduced, to provide a good location estimate.

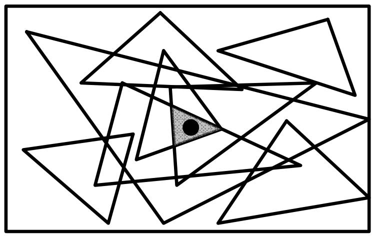


Fig. 2. Area-based APIT Algorithm Overview

The theoretical method used to narrow down the possible area in which a target node resides is called the *Point-In-Triangulation Test* (PIT). For three given anchors: $A(a_x, a_y), B(b_x, b_y), C(c_x, c_y)$, the Point-In-Triangulation test determines whether a point M with an unknown position is inside triangle $\triangle$ABC or not. APIT repeats this PIT test with different anchor combinations until all combinations are exhausted or the required accuracy is achieved. At this point, APIT calculates the center of gravity (COG) of the intersection of all of the triangles in which a node resides to determine its estimated position. These steps are shown in Algorithm 1.

In [8], the authors provide a perfect, albeit theoretical, solution for perfect Point-In-Triangulation test as follows:

Algorithm 1 APIT

```
1: Receive location beacons $(X_i, Y_i)$ from $N$ anchors;
2: $InsideSet = \emptyset$;
3: for each triangle $T_i \in \binom{N}{3}$ triangles do
4:     if Point-In-Triangle-Test $(T_i)$ == TRUE then
5:         $InsideSet = InsideSet \bigcup T_i$;
6:     end if
7: end for
8: Estimated Position = CenterOfGravity( $\bigcap T_i \in InsideSet$);
```

Perfect P.I.T. Test Theory: If there exists a direction such that a point adjacent to M is further/closer to points A, B, and C simultaneously, then M is outside of △ABC. Otherwise, M is inside △ABC (Figure 3).

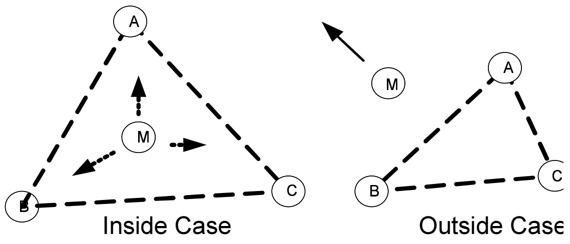


Fig. 3. Cases for Point-In-Triangulation Test

The perfect P.I.T. test can correctly decide whether a point M is inside triangle △ABC (the formal proofs can be found in [8]). However, there are two major issues to apply this theory practically in wireless sensor networks: First, how does a node recognize directions of departure from an anchor without moving? Second, how to exhaustively test all possible directions in which node M might depart/approach vertexes A, B, C simultaneously? The answer to the first question is to use RSSI comparisons. Through experiments, the authors confirm that in a narrow direction, the further away a node is from the anchor, the weaker the received signal strength (RSSI) will be. Through signal strength comparisons, a node can determine whether the direction towards a neighboring node is closer to a given anchor or not. To address the second issue, the authors propose an approximation (APIT) for the perfect PIT test, which uses neighbor information, through RSSI comparisons, to emulate the node movement in the Perfect PIT test. With a finite number of neighbors, APIT can only evaluate a limited number of directions. Consequently, APIT could make an incorrect decision. Fortunately, experiments indicate that the percentage of APIT tests exhibiting such an error is relatively small (14% in the worst case). Figure 4 demonstrates this error percentage as a function of node density. When node density increases, APIT can evaluate more directions, considerably reducing false positive, i.e. APIT returns true, while a node is outside of triangle (OutToInError). On the other hand, false negative (InToOutError) will slightly increase due to the increased chance of edge effects.

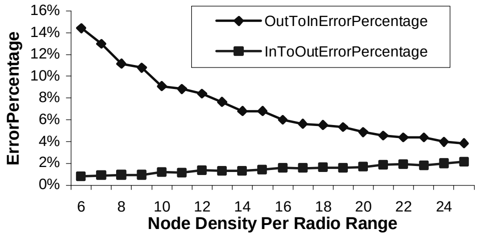


Fig. 4. APIT Error under Varying Node Densities

Once the individual APIT tests finish, APIT aggregates the results (inside/outside decisions among which some may be incorrect) through a grid SCAN algorithm (Figure 5). In this algorithm, a grid array is used to represent the maximum area in which a node likely resides. In the experiments, the length of a grid side is set to 0.1R, to guarantee that estimation accuracy is not noticeably compromised.

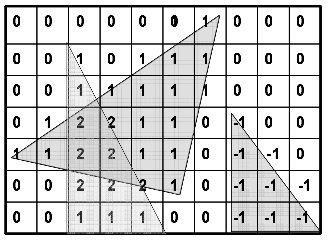


Fig. 5. APIT Error under Varying Node Densities

For each APIT inside decision (a decision where the APIT test determines the node is inside a particular region (Figure 5) the values of the grid regions over which the corresponding triangle resides are incremented. For an outside decision, the grid area is similarly decremented. Once all triangular regions are computed, the resulting information is used to find the maximum overlapping area (e.g., the grid area with value 2 in Figure 5). Since the majority (more than 85% in the worst case shown in Figure 4) of APIT tests are correct, the correct decisions build up on the grid and the small number of errors only serves as a slight disturbance to the final estimation. Evaluation in [8] indicates APIT works better than other range-free solutions under irregular radio patterns and random node placement. However, it should be pointed out that APIT has a more demanding requirement on the number of anchors used in localization.

2.3 SeRLoc

SeRLoc [15] is another area-based range-free localization. The authors assume two types of nodes: normal nodes and locators (i.e., anchors). Normal nodes are equipped with omnidirectional antennas, while locators are equipped with directional sectored antennas and their locations of locators are known a priori. In SeRLoc, a sensor estimates it location based on the information transmitted by the locators. Figure 6 shows the main idea, with node N_k within radio range to locators A_1, A_2 and A_3:

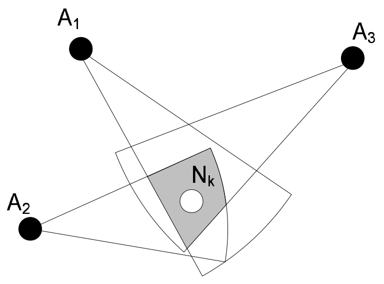


Fig. 6. SerLoc Localization

SeRLoc localizes the sensor nodes in four steps. First, a locator transmits directional beacons within a sector. Each beacon contains the locator's position and the angles of the sector boundary lines. A normal node collects the beacons from all locators it hears. Second, it determines an approximate search area within which it is located based on the coordinates of the locators heard. Third, it computes the overlapping sector region using a majority vote scheme. Finally, SeRLoc determines a node location as the center of gravity of the overlapping region. These steps are shown in Algorithm 2.

We note that SeRLoc is unique in its secure design. It can deal with various kinds of attacks including wormhole and Sybil attacks. We do not describe its security features here except to note that the authors prove in [15] that their approach is more secure, robust and accurate in the presence of attacks, compared with other state-of-the-art solutions that largely ignore this issue.

2.4 Multidimensional Scaling

The MDS-MAP algorithm proposed by Shang et al. in [28] is based on a data analysis technique, called MultiDimensional Scaling (MDS), extensively used

Algorithm 2 SeRLoc

```
1: Receive beacons from locators; each beacon contains the position of locator and
   the angles of sector boundary.
2: Find four values: $(X_{min}, Y_{min}, X_{max}, Y_{max})$ among all the locator positions
   heard.
3: Set the search area as the rectangle $(X_{min} - R, Y_{min} - R, X_{max} + R, Y_{max} + R)$,
   where R is the radio range.
4: Partition the search area into grids.
5: for each beacon received do
6:     Increase the value of a grid point by one if this grid point is within the sector
       defined in this beacon.
7: end for
8: Estimated Position = CenterOfGravity(the grid points with the largest values)
```

in psychometrics. MDS attempts to provide a visualization (2D or 3D) of the original data, and preserving the essential information present in the data set (i.e., similarities in a multidimensional space). The MDS-MAP algorithm uses the classical metric scaling, the simplest case of the MDS technique, which has a closed form solution, enabling a relatively efficient computation (requires no iterations).

An important concept for MDS is how to compute the distance between two points. If we denote by $\mathbf{X}$ the matrix of coordinates of points ($n \times m$ matrix of n points in m dimensions), and by $\mathbf{D} = [d_{ij}]$ the matrix of distances between points, it can be shown that the matrix of squared distances between points, i.e., $\mathbf{D}^{(2)}$, can be written as follows:

$$\mathbf{D^{(2)}} = \mathbf{c1'} + \mathbf{1c'} - 2\mathbf{XX'} = \mathbf{c1'} + \mathbf{1c'} - 2\mathbf{B}$$

where $\mathbf{c}$ is a vector with elements the diagonal elements of $\mathbf{XX'}$. The matrix $\mathbf{B} = \mathbf{XX'}$ is the scalar product matrix. So the questions becomes, given the matrix of squared distances $\mathbf{D}^{(2)}$ how can one obtain $\mathbf{B}$, and implicitly $\mathbf{X}$? It can be shown [1] that by double-centering $\mathbf{D^{(2)}}$, one can obtain $\mathbf{B}$:

$$-\mathbf{\frac{1}{2}}\mathbf{JD^{(2)}J} = \mathbf{B}$$

where $\mathbf{J} = \mathbf{I} - \mathbf{11'}/n$ (called the centering matrix), $\mathbf{I}$ the identity matrix and $\mathbf{1}$ the n-dimensional column vector with all elements one. Once $\mathbf{B}$ is obtained, the coordinates $\mathbf{X}$ can are computed by eigendecomposition.

The steps of classical scaling are summarized as follows:

1. Compute the squared distances matrix $\mathbf{D^{(2)}} = [d_{ij}^2]$
2. Double-center the $\mathbf{D^2}$ matrix:

$$\mathbf{B} = -\frac{1}{2}\mathbf{JD^{(2)}J}$$

3. Compute the singular value decomposition of $\mathbf{B} = \mathbf{VAV^T}$
4. Compute the coordinate matrix:

$$\mathbf{X} = \mathbf{V}_+\mathbf{A}_+^{\mathbf{1/2}}$$

where $\mathbf{A}_+$ is the matrix of the first m eigenvalues greater than zero and $\mathbf{V}_+$ the first m columns of V.

The MDS-Map algorithm that uses the classical metric scaling technique is shown in Algorithm 3.

Algorithm 3 MDS-MAP

```
1: Compute the shortest paths $d_{ij}$, $1 \leq i, j \leq n$. This gives the distances matrix $\mathbf{D}$.
2: Compute the relative positions (map), by applying classical MDS to the distance matrix $\mathbf{D}$, and retain the largest 2 (for a 2D space) or 3 (for a 3D space) eigenvalues and eigenvectors.
3: Transform the relative map, into an absolute map, based on the absolute positions of anchor nodes.
```

The main drawbacks of the MDS-MAP algorithm, the need for global information and centralized computation are addressed in subsequent work by the authors [27].

2.5 Gradient

In the Gradient algorithm, proposed by Nagpal et al. in [20], the anchor nodes initiate a gradient that self-propagates and allows a sensor node to infer its distance from the anchor. After estimating distances to three anchors a sensor node infers its own location through multilateration.

The steps of the algorithm are as follows:

- Each anchor node A_i initiates a flood of the network by broadcasting a packet containing its position and a counter with the initial value set to one.
- Each sensor node N_j keeps track of the shortest path (in terms of radio hop counts, h_{j,A_i}) to an anchor A_i from which it has received a beacon. A distance estimate, between the sensor node and anchor is obtained by:

$$d_{ji} = h_{j,A_i} d_{hop}$$

where d_{hop} is the estimated Euclidian distance covered by one radio hop, and it is given by the Kleinrock-Silvester formula [11]:

$$d_{hop} = r\left(1 + e^{-n_{local}} - \int_{-1}^{1} e^{-\frac{n_{local}}{\pi}\left(\arccos t - t\sqrt{1-t^2}\right)}\right)$$

- Each node N_j computes its coordinates such that the total error is minimized:

$$E_j = \sum_{i=1}^{n} (d_{ji} - \hat{d_{ji}})$$

where $d_{ji} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ and $\hat{d_{ji}}$ is the estimated distance computed through gradient propagation, as shown above.
The coordinate are incrementally updated in the following way:

$$\Delta x = -\alpha \frac{\partial E_j}{\partial x_j} \text{ and } \Delta y = -\alpha \frac{\partial E_j}{\partial y_j}$$

where:

$$\frac{\partial E_j}{\partial x_j} = \sum_{i=1}^{n} (x_j - x_i) \left(1 - \frac{d_{ji}}{\hat{d_{ji}}}\right) \text{ and } \frac{\partial E_j}{\partial x_j} = \sum_{i=1}^{n} (x_j - x_i) \left(1 - \frac{d_{ji}}{\hat{d_{ji}}}\right)$$

Sources for errors in location estimation arise in the Gradient scheme from: incorrect estimation of the one-hop distance (d_{hop}), and the multilateration procedure.

2.6 Ad-Hoc Positioning System

In a similar manner with the Gradient method, the Ad-Hoc Positioning System (APS) proposed by Niculescu and Nath [22], uses the hop-by-hop propagation of distances to known anchors (a set of anchors is assumed to be present in the WSN). After obtaining distance estimates to three or more anchors, a sensor node employs a multilateration (similar with that of GPS) for iteratively improving its location estimation. The main difference resides in how a sensor node N_j estimates its distance to an anchor A_i (d_{ji} from the Gradient method, presented before).

The steps of the APS localization scheme algorithm are the following:

- Each anchor node A_i initiates a flood of the network by broadcasting a packet containing its position and a counter with the initial value set to one.
- Each sensor node N_j keeps track of the shortest path (in terms of radio hop counts, h_{j,A_i}) to an anchor A_i from which it has received a beacon. In [22] the authors propose four methods for propagating the distances from anchors to sensor nodes: DV-Hop, DV-Distance, Euclidian and Coordinate. The method that does not assume ranging, DV-Hop is described below. An example of the DV-Hop scheme is shown in Figure 7. At the end of this phase, node N_j knows that it is 3 hops, 2 hops and 1 hop from A_1, A_2 and A_3, respectively.

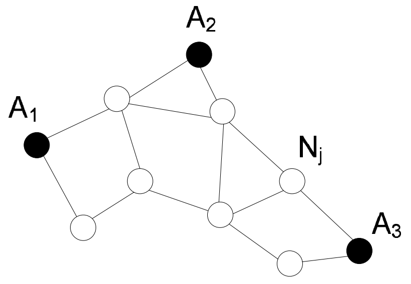


Fig. 7. The DV-Hop localization scheme

- Once an anchor node A_i obtains distances to other anchors, it computes a correction factor c_i (the estimated 1 radio hop Euclidian distance), which it propagates in the network. Corrections are propagated through controlled flooding, i.e., after a node receives and forwards the first correction, it will stop forwarding subsequent corrections. The correction factor is computed as follows:

$$c_i = \frac{\sum \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum h_i}$$

  for all anchors $A_j \neq A_i$ from which it has received a beacon (anchor A_j is positioned at (x_j, y_j) and h_i is the number of hops between the sensor node and anchor A_i).
  For the example shown in Figure 7, if the distances $\overline{A_1A_2}$, $\overline{A_2A_3}$ and $\overline{A_3A_1}$ are 30m, 40m and 50m, respectively, the correction factor for anchor A_3 is: c_3=(50+40)/(4+3)=12.9m/hop.
- A least square method (the authors used the Householder method) is employed for solving the non-linear system of equations:

$$\begin{bmatrix} \Delta\rho_1 \\ \Delta\rho_2 \\ \Delta\rho_3 \\ ... \\ \Delta\rho_n \end{bmatrix} = \begin{bmatrix} \hat{1}_{1x} & \hat{1}_{1y} \\ \hat{1}_{2x} & \hat{1}_{2y} \\ \hat{1}_{3x} & \hat{1}_{3y} \\ ... & ... \\ \hat{1}_{nx} & \hat{1}_{ny} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

  where $\Delta\rho_i = \hat{\rho}_i - \rho_i$, $\hat{\rho}_i$ and ρ_i are the estimated and the real distances between a sensor node and an anchor A_i, $\hat{1}_{ix}$ is the unit vector of $\hat{\rho}_i$ in the x direction and Δx and Δy are the corrections in the position estimate for the node N_j.
  For the example shown in Figure 7, the estimated distances between node N_j and anchors A_1, A_2 and A_3 are $\hat{\rho}_1$=4*12.9=51.6m, $\hat{\rho}_2$=38.7m and $\hat{\rho}_3$=12.9m.

2.7 Probability Grid

In a similar manner with the DV-Hop, a localization scheme that can be used in scenarios where the topology of deployment is known a priori to be a grid, is proposed in [31].

The steps of the localization scheme are the following:

- Each anchor node A_m initiates a flood of the network by broadcasting a packet containing its position and a counter with the initial value set to one.
- Each sensor node N_k keeps track of the shortest path (in terms of radio hop counts) to each of the anchors A_l from which it has received beacons.
- Once an anchor node A_m obtains distances to other anchors, it computes a correction factor c_m (the estimated radio range), and it propagates it in the network.
- After receiving hop-count estimates to three or more anchor nodes, and a correction factor c_m a sensor node N_k evaluates the probability of being located at any position in the grid (labeled (i, j)). For this, it computes an expected hop count:

  $$\lambda = d_{(i,j),l}/c_m$$

  where $d_{(i,j),l}$ is the Euclidian distance between anchor A_l and the point (i, j) being evaluated. It then computes the probability of it (node N_k) to be positioned at (i, j):

  $$p_{k,(i,j)} = \prod_{l=1}^{|A|} P_{(i,j)}^{h_{k,l}}$$

  where $P_{(i,j)}^{h_{k,l}}$ is the probability of node N_k, positioned at (i, j), to be $h_{k,l}$ hops from anchor A_l.
- A node N_k chooses as its location, the position in the grid (i, j) with the maximum probability $p_{k,(i,j)}$.

The authors make the observation that $h_{k,l}$ is a discrete random variable that represents the number of radio hops between one anchor and the point of interest, i.e. (i, j). The main features that the distribution function for $h_{k,l}$ needs to exhibit are: to have one parameter λ (defined above) , to be narrow and skewed positively for small values of λ and become broader and relatively symmetric for larger values of λ. This is illustrated in Figure 8:

These requirements follow the intuition that for smaller values of the parameter λ (i.e., grid points closer to the anchor) the number of hops (call it τ) has a limited range of possible values with higher and higher values being less and less probable (positively skewed). As the distance between the anchor and the node increases (λ increases), the number of possibilities for the hop count (τ) increases and the distribution becomes bell-shaped, i.e., smaller and larger

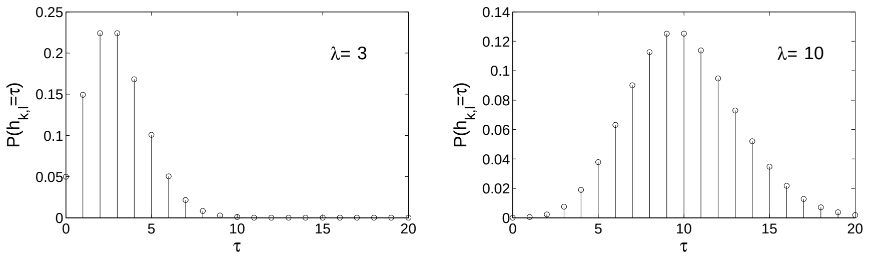


Fig. 8. The intuition behind the PMF of $h_{k,l}$

hop counts are equally probable. The authors found through simulations that a Poisson distribution is a good approximation for the $h_{k,l}$ discrete random variable. The distribution is given by:

$$P_{(h_{(k,l)}=\tau)} = \frac{\lambda^{\tau-1}e^{-\lambda}}{(\tau-1)!}$$

where $\tau = 1, 2, ...$

3 Anchor-Free Solutions

Anchor-free localization schemes exploit the proximity to an event with a known location: a light event in [30] [24] or a nearby radio packet in [29]. One common characteristic for these schemes is the moving of the complexity (hardware and computational, associated with an accurate localization) from the sensor node to a central, more sophisticated device. By controlling well the spatio-temporal properties of the events (light and radio packets), a much higher accuracy in localization (when compared with the anchor-based schemes) can be obtained. While anchor nodes are not required for any of the following schemes, anchor nodes can be beneficial for extensions of the proposed schemes.

3.1 Spotlight

The main idea of the Spotlight localization system [30] is to generate controlled events in the field where the sensor nodes are deployed. An event could be, for example, the presence of light in an area. Using the time when an event is perceived by a sensor node and the spatio-temporal properties of the generated events, spatial information (i.e. location) regarding the sensor node can be inferred. The system architecture for the Spotlight localization system is shown in Figure 9.

With the support of these three functions, the localization process goes as follows:

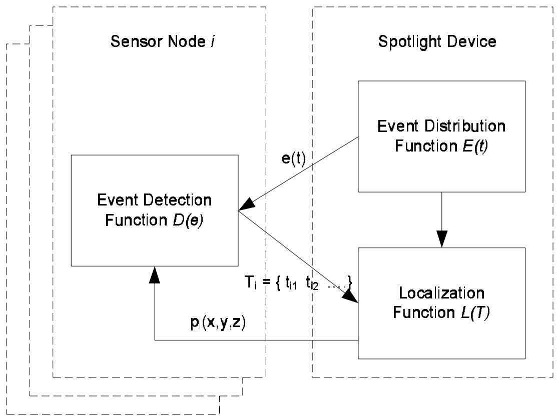

Sensor Node *i*
Spotlight Device
Event Distribution Function *E(t)*
e(t)
Event Detection Function *D(e)*
Localization Function *L(T)*
$T_i = \{ t_{i1}, t_{i2}, ... \}$
$p_i(x,y,z)$

Fig. 9. Spotlight system architecture

- A Spotlight device distributes events $e(t)$ in the space A over a period of time.
- During the event distribution, sensor nodes record the time sequence $T_i = \{t_{i1}, t_{i2}, ..., t_{in}\}$ at which they detect the events.
- After the event distribution, each sensor node sends the detection time sequence back to the Spotlight device.
- The Spotlight device estimates the location of a sensor node i, using the time sequence T_i and the known $E(t)$ function.

The Event Distribution Function $E(t)$ is the core technique used in the Spotlight system and the authors propose three designs for it, with different tradeoffs/costs. These designs are presented below.

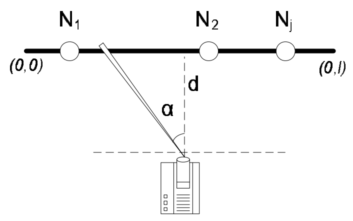


Fig. 10. The implementation of the Point Scan EDF

Point Scan

The Point Scan EDF is applicable to a simple sensor system where a set of nodes are placed along a straight line ($A = [0, l] \subset R$). The Spotlight device generates point events (e.g., light spots) along this line with constant speed s, as shown in Figure 10. The set of timestamps of events detected by a node i is $T_i = \{t_{i1}\}$. The Event Distribution Function $E(t)$ is:

$$E(t) = \{p \mid p \in A, p = t * s\}$$

where $t \in [0, l/s]$. The resulting localization function is:

$$L(T_i) = E(t_{i1}) = \{t_{i1} * s\}$$

Line Scan

Some devices, e.g. lasers, can generate an entire line of events simultaneously. With these devices, the Line Scan Event Distributed Function can be supported. Assuming that the sensor nodes are placed in a two dimensional plane ($A = [l \times l] \subset R^2$) and that the scanning speed is s. The set of timestamps of events detected by a node i is $T_i = \{t_{i1}, t_{i2}\}$.

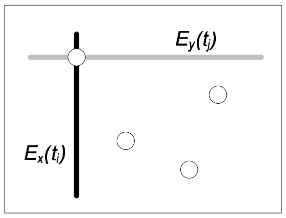


Fig. 11. The implementation of the Line Scan EDF

The Line Scan EDF, depicted in Figure 11, is defined as follows:

$$E_x(t) = \{p_k \mid k \in [0, l], p_k = (t * s, k)\} \texttt{ for } t \in 0, l/s$$

$$E_y(t) = \{p_k \mid k \in [0, l], p_k = (k, t * s - l)\} \texttt{ for } t \in l/s, 2l/s$$

and $E(t) = E_x(t) \cup E_y(t)$.

The location of a node can be calculated from the intersection of the two event lines, as shown in Figure 11. More formally:

$$L(T_i) = E(t_{i1}) \cup E(t_{i2})$$

Area Cover

Other devices, such as light projectors, can generate events that cover an area. This allows the implementation of the Area Cover EDF. The idea of Area Cover EDF is to partition the space A, where the sensor nodes are deployed, into multiple sections and assign a unique binary identifier, called code, to each section. Let's suppose that the localization is done within a plane ($A \in R^2$). Each section S_k within A has a unique code k.

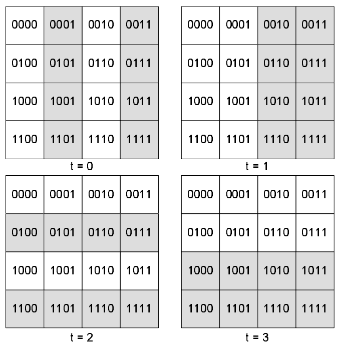


Fig. 12. The implementation of the Area Cover EDF

The Area Cover EDF, with its steps shown in Figure 12 is then defined as follows:

$$BIT(k, j) = \begin{cases} \text{true if } j^{th} \text{ bit of } k \text{ is } 1 \\ \text{false if } j^{th} \text{ bit of } k \text{ is } 0 \end{cases}$$

$$E(t) = \{p \mid p \in S_k, BIT(k, t) = true\}$$

and the corresponding localization algorithm is:

$$\begin{aligned} L(T_i) = \{p \mid p = COG(S_k), BIT(k, t) = true \ \texttt{if} \ t \in T_i, \\ BIT(k, t) = false \ \texttt{if} \ t \in T - T_i\} \end{aligned}$$

where $COG(S_k)$ denotes the center of gravity of S_k.

Cost Comparison

Although all three proposed techniques are able to localize the sensor nodes, they differ in the localization time, communication overhead and energy consumed by the Event Distribution Function (call it Event Overhead). Assume that all sensor nodes are located in a square with edge size D, and that the Spotlight device can generate N events (e.g. Point, Line and Area Cover events) every second and that the maximum tolerable localization error is r. Table 1 presents the execution cost comparison of the three different Spotlight techniques.

Criterion	Point Scan	Line Scan	Area Cover
Localization Time	$D^2/r^2)/N$	$D^2/r^2)/N$	$\log_r(D/N)$
# Detections	1	2	$\log_r D$
# Time Stamps	1	2	$\log_r D$
Event Overhead	D^2	$2D^2$	$D^2 \log_r(D/2)$

Table 1. Execution Cost Comparison

Table 1 indicates that the Event Overhead for the Point Scan method is the smallest - it requires a one-time coverage of the area, hence the D^2. However the Point Scan takes a much longer time than the Area Cover technique, which finishes in $\log_r D$ seconds. The Line Scan method trades the Event Overhead well with the localization time. By doubling the Event Overhead, the Line Scan method takes only $r/2D$ percentage of time to complete, when compared with the Point Scan method. From Table 1, it can be observed that the execution costs do not depend on the number of sensor nodes to be localized. It is important to remark the ratio "Event Overhead"/"Localization Time", which is indicative of the power requirement for the Spotlight device. This ratio is constant for the Point Scan (r^2N) while it grows linearly with area, for the Area Cover ($D^2N/2$). If the deployment area is very large, the use of the Area Cover EDF is prohibitively expensive, if not impossible. For practical purposes, the Area Cover is a viable solution for small to medium size networks, while the Line Scan works well for large networks.

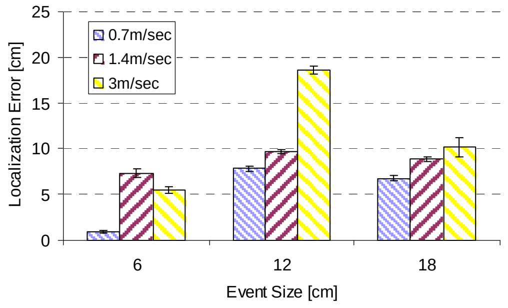


Fig. 13. Localization Error vs. Event Size for Spotlight system.

For the Spotlight system evaluation, the authors deployed 10 XSM [6] motes in a football field. The Spotlight device consisted of diode lasers, a computerized telescope mount, connected to a laptop. The Event Distribution Function investigated was the Point Scan. The range between the Spotlight device and the sensor nodes was approximately 170m.

Figure 13 shows the average localization errors versus the size of the event (diameter of the laser beam, on the ground), for different scanning speeds s. Localization errors of 10-20cm are reported.

3.2 Lighthouse

In a similar way to the Spotlight localization system, the Lighthouse scheme, proposed by Römer [24], makes use of the free-space optical channel between a device (called Lighthouse in this case) and sensor nodes.

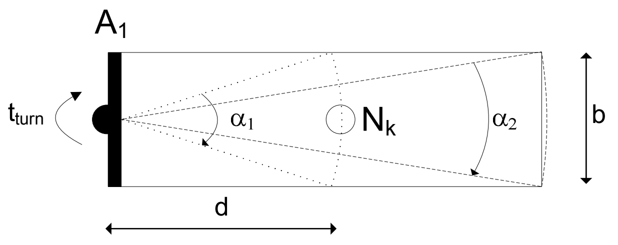


Fig. 14. Lighthouse Localization

The main idea of the Lighthouse system is exemplified in Figure 14. A parallel light beam of width b, emitted by anchor A_1 rotates with a certain period t_{turn}. A sensor node N_k detects this light beam for a period of time t_{beam}, which is dependent on the distance d between the Lighthouse device and the sensor node, in the following way:

$$d = \frac{b}{2\sin(\alpha_1/2)} = \frac{b}{2\sin(\pi t_{beam}/t_{turn})}$$

From measuring t_{beam} and knowing b and t_{turn}, one can compute the distance between the sensor node and the lighthouse device d. By constructing a device with three mutually perpendicular light emitting Lighthouses, a 3D location can be obtained.

The main difficulty encountered by the authors in the implementation of the Lighthouse prototype is ensuring that the light beam is perfectly parallel (zero divergence), having a width b. Instead, two laser beams of widths b_i and angle orientations β_i, γ_i and δ_i $i = 1, 2$, are used. To account for the misalignments, the authors develop a better approximation for the resulting beam width b:

$$b \approx C^b + \sqrt{d^2 + h^2}C^\beta + hC^\gamma + dC^\delta$$

where $C^b = b_1 + b_2$, $C^\beta = \sin\beta_1 + \sin\beta_2$, $C^\gamma = \tan\gamma_1 + \tan\gamma_2$ and $C^\delta = \sin\delta_1 + \sin\delta_2$. These parameters are constant for a particular Lighthouse system, and they are obtained through calibration, by localizing four points with known locations.

The experiments use 22 nodes placed in a 5x5m^2 area, with the Lighthouse device positioned at the coordinate (0,0). The accuracy of the localization algorithm is presented relative to the distance between the Lighthouse device and the sensor node (i.e., $|\hat{x} - x|/x$). The mean relative error (difference between the computed location and ground truth) in localization is 1.1% in one direction and 2.8% in the second direction (the difference is attributed to the calibration). This translates in localization errors of a few centimeters.

3.3 Walking-GPS

In many applications it is envisioned that WSN will be deployed from Unmanned Aerial Vehicles. In the meantime, manual deployments have been prevalent and the employed localization solutions have used some variant of associating the sensor node ID with prior knowledge of that ID's position in the field.

In [29] the authors propose a solution, called Walking GPS, in which the deployer (either person or vehicle) carries a GPS device that periodically broadcasts its location. The sensor nodes being deployed, infer their position from the location broadcast by the GPS device. The proposed solution is simple, cost effective and has very little overhead.

In the Walking GPS architecture the system is decoupled into two software components: the GPS Mote and the Sensor Mote. The GPS Mote runs on a Mica2 mote. The mote is connected to a GPS device, and outputs its location information at periodic intervals. The Sensor Mote component runs on all sensor nodes in the network. This component receives the location information broadcast by the GPS Mote and infers its position from the packets received. The proposed architecture pushes all complexity derived from the interaction with the GPS device to a single node, the GPS Mote, and to significantly reduce the size of the code and data memory used on the sensor node. Through this decoupling, a single GPS Mote is sufficient for the localization of an entire sensor network, and the costs are thus reduced.

A relatively simple design for the GPS Mote would have been to periodically broadcast the actual GPS location received from the GPS device. In order to reduce the overhead incurred when exchanging data containing global GPS coordinates, the Walking GPS system uses a local, Cartesian, coordinate system. The conversion between coordinate systems is performed by the GPS mote. A local coordinate system of reference is better suited for WSN, than a global coordinate system.

The localization scheme that makes use of the Walking GPS solution has two distinct phases:

1. The first phase is during the deployment of the sensor nodes. This is when the Walking GPS solution takes place. The deployer has a GPS-enabled mote attached to it; the GPS-enabled mote periodically beacons its location; the sensor nodes that receive this beacon infer their location based on the information present in this beacon.
2. The second phase is during the system initialization. If at that time, a sensor node does not have a location, it asks its neighbors for their location information. The location information received from neighbors is used in a triangulation procedure by the requester, to infer its position. This second phase enhances the robustness of the scheme.

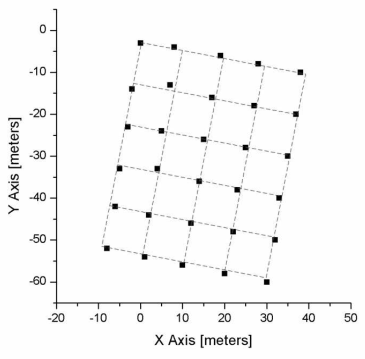


Fig. 15. Walking GPS system evaluation. Nodes deployed in a grid.

The experimental evaluation of the entire system, consisted of 30 MICA2 motes that were deployed in a 5x6 grid (for ease of measuring the localization error). The experimental results are shown in Figure 15. The average localization error obtained from fitting a grid to the experimental data is 0.8m with a standard deviation of 0.5m.

4 Open Problems

4.1 Security

Recently, several research groups have started to address robust and secure localization. For example, SeRLoc [15] demonstrates robustness against wormhole, Sybil and compromise of network nodes attacks. However, this work assumes a particular two-tier architecture and special hardware they call locators. In addition, the jamming of the wireless medium is not considered. It is an excellent start, but a lot more needs to be done especially for military domains and to meet various reality assumptions.

For securing localization, robust statistical methods (e.g. least median square) have been proposed [16]. The assumption is that an attacker selectively alters distance estimates to known anchor locations. The highest contamination ratio (i.e., affected readings) that the mathematical model supports is 50%, with significant degradation even at 35%. If an attacker possesses the capability of affecting distance estimates easily, then it is very likely that all distance estimates will be affected, making this set of solutions, less effective. The idea, however, of using robust statistical models, is a very good one.

In a similar approach, in [17], two solutions are proposed for secure localization: an attack resistant minimum mean square error (MMSE) which suffers from an unbounded localization error (the attack can result in an arbitrarily large localization error), and a voting-based scheme, which corrects the unbounded localization error, at a higher computational and storage cost.

Distance bounding has also recently been proposed as a technique for secure verification of localization. In [25] a combination of ultra-sound and radio communication is used for bounding the location claimed by a node, to a region. In this region, called region of interest, a set of trusted verifiers has to exist. This scheme is robust against attackers that can not be physically present in the region of interest. Similarly, [4] proposes a Verifiable Multilateration, that also relies on the distance bounding technique. The basic idea is to use the Time of Flight (ToF) of radio communication. Since the speed of light can not be exceeded, the location to be verified can not be closer than it actually is. It can only be further. However, claiming a longer distance would require a shorter distance to an even further positioned verifier. The main drawback is the hardware requirements (with nanosecond accuracy) imposed on the sensor nodes. In addition, [4] requires a relatively large number of anchor nodes.

In order to address some of the deficiencies of SeRLoc (e.g. jamming is not considered) and the Verifiable Multilateration (e.g., relatively high number of anchor nodes), a new scheme is proposed in [14]. This scheme can be used for both, location determination and location verification. The main idea is to fully utilize the strengths of both solutions: SeRLoc's use of sectored antennas and the distance bounding properties of Verifiable Multilateration. The deficiencies of both schemes, are still present.

The most recent effort on secure localization [5], attempts to depart from the aforementioned, "traditional", approaches, which require high speed hardware, sectored antennas or statistics, with a limited robustness. The idea is to use covert, hidden base stations (their position is known only to an authority), in addition to the "public", known base stations. The role of covert base stations is to perform TDoA (between radio and ultra-sound) ranging and verify the location computed and claimed by a node. For the effectiveness of this solution, the covert base stations communicate with a central location verification authority either in a wired manner or infra-red, to reduce the risk of being detected by the attacker. The authors also propose mobile base station assisting with the verification of location. While this direction for secure localization is novel, in its current form, has demanding requirements for the infrastructure of covert base stations.

4.2 Impact of Localization on Protocols

In localization for WSN, achieving better results (usually with regard to location accuracy) requires increasing the relative cost of the localization scheme via additional hardware, communication overhead, or the imposition of constraints and system requirements. Although more accurate location information is preferable, the desired level of granularity should depend on a cost/benefit analysis of the protocols that utilize this information. In this section, we investigate the impact of localization error on other communication protocols and proposed sensor network applications. Designers of sensor network systems with certain performance requirements can use this analysis to aid in their architectural design and in setting system parameters. Although requirements are expected to vary between deployments, we found that in the general case for the protocols studied, performance degradation is moderate and tolerable when the average localization error is less than 0.4R.

Routing Performance

A localization service is critical for location-based routing protocols such as geographic forwarding (GF) [21], [10], [12] and [34]. In these protocols, individual nodes make routing decisions based on knowledge of their geographic location. While most work in location-based routing assumes perfect location information, the fact is that erroneous location estimates are virtually impossible to avoid. Problems arise as error in the location service can influence location-based routing to choose the best next hop (the neighbor closest to the destination), or can make a node inadvertently think that the packet could not be routed because no neighbors are closer to the final destination.

To investigate the impact of localization error on routing, the authors of [8] studied the GF [21] routing protocol under the low traffic network conditions so that network congestion does not influence the results. The baseline was the perfect localization, the protocol where every sensor node knows its correct physical location.

Figure 16 shows the delivery ratio (the percentages of packets that reach destination over all packets sent) with regard to node density for various levels of location error. From this graph, we see that for average localization errors of 0.2R and 0.4R, the delivery ratios of GF are very close to the baseline (no error). Beyond these numbers, the results diminish with increased error; a trend that could be problematic and costly depending on the implemented architecture, reliability semantics, tolerance of message loss, and application requirements. For example, when localization error is the same as the node radio range, even with high node density (20 nodes per radio range), the delivery ratio still falls below 60%.

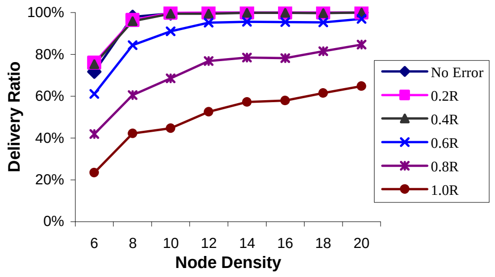


Fig. 16. Delivery ratio with different localization errors, changing node density

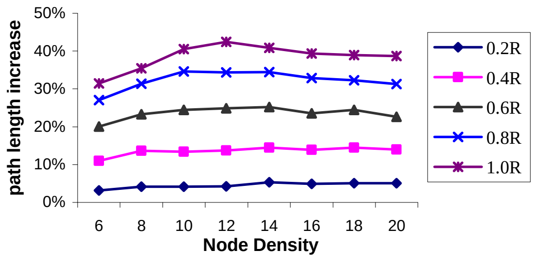


Fig. 17. Path length overhead with different localization errors under varying node density

Another metric affected by localization error is the route path length. Figure 17 measures the hop count increase (in percentage) due to location error to assess the cost in communication overhead of this error. We see from this graph that for low localization error (less than 0.4R), this routing overhead remains moderate (less than 15%). However, as was the case for the delivery ratio metric, when localization error grows beyond 0.4R, the routing overhead increases to as high as 45%. We also note that this trend occurs regardless of the network node density, a fact that was not true for our previous metric. We acknowledge here that GF was chosen as a representative protocol, and an in depth study about localization's impact on various routing protocols and its implications on the design of location-dependent systems is future work.

Target Estimation Performance

Many of the most frequently proposed applications for WSN utilize target position estimations for tracking, search and rescue, or other means. In these proposed applications, when a target is identified, some combination of the nodes that sensed that target report their location to a centralized node (leader or base station). This node then performs aggregation on the received data to estimate the actual location of the target. Because target information could be used for locating survivors during a disaster, or identifying an enemy's position for strategic planning, the accuracy of this estimation is crucial to the application that uses it.

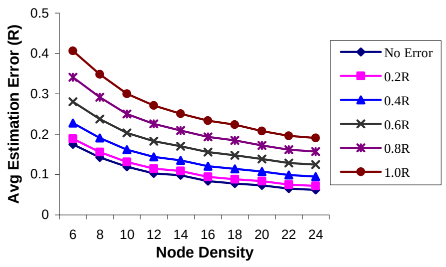


Fig. 18. Target estimation error with different localization errors under varying node density

Intuitively an increase in localization error directly leads to target estimation error. To better understand the degree to which this error propagates to other protocols, the authors of [8] investigate the average estimation error under different node densities for varying degrees of location error. For these experiments, a simple and widely used target estimation algorithm is used: the average X and Y coordinates of all reporting nodes are taken as the target location estimation. The sensing range is set to be equal to the node radio range so that the node density is equivalent to the average number of sensors involved in target estimation. The results of various experiments are depicted in Figure 18. This graph shows that target estimation error due to location error is dampened during the aggregation process. As before, the baseline occurs when no localization error exists. Aside from showing varying degrees of estimation error with respect to node location error, Figure 18 also shows that the absolute target estimation error decreases with increased node density. For example, when localization error is equal to 1.0R, and node density reaches 12 nodes per radio range, the estimation error is only about 67% as large as when the node density is 6 nodes per radio range. From this chart we see that more nodes participating in estimation results in more random estimation error being ameliorated through aggregation.

Object Tracking Performance

In [8], the authors further evaluate the performance of target estimation by simulating a tracking application that uses estimation in context. In this experiment, a mobile evader randomly walks around the specified terrain while a pursuer attempts to catch it. In this simple experiment, the pursuer is informed of the current location of the evader periodically via sensing nodes in the terrain that detect the evader, coordinate to estimate the targets position with regard to their own positions, and periodically report this result to the mobile pursuer. When receiving a report, the pursuer readjusts its direction in an attempt to intercept the evader. When the pursuer comes within the node communication radius of the evader, the evader is considered caught and the simulation ends. For this experiment, the average tracking time (the time from pursuer take-off to when the evader is caught) under different localization errors is compared to the tracking time in the case of no localization error. Figure 19 shows normalized tracking time in relation to the pursuer speed for various degrees of localization error.

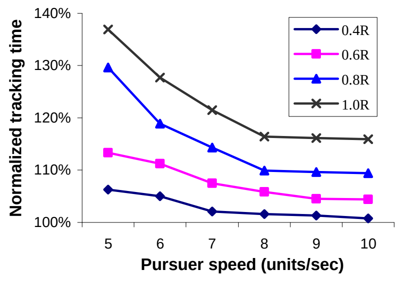


Fig. 19. Normalized tracking time with different localization errors varying pursuer speed. Terrain size 1000x1000m, Radio range = 40m, density = 8 nodes/radio range. Evader speed = 5m/s

From Figure 19 we see that the tracking time overhead decreases with increased pursuer speeds. More importantly, Figure 19 shows that the tracking time increases as localization error increases. This result implies that it is important for tracking applications with real-time requirements to take localization error into consideration. For example, when the average localization error is known to be 0.8R, and the pursuer speed is 5 units per second, the pursuer requires 30% more time in comparison to the ideal situation in which no localization error exists. To reduce this overhead to 10%, either the pursuers speed must be increased to 10 units per second, or the estimation error must be reduce to 0.4R. Again, Figure 19 shows that 0.4R is a tolerable bound for estimation error since tracking time only increases by 7% in the worst case.

4.3 Impact of Environment on Localization

The problem of range-free localization is further complicated by the diverse types of environments, where a WSN system can be deployed. Outdoor, real deployment environments very little resemble typical lab environments. Hence, issues like calibration, mobility (if nodes are “moved” by the environment, or the WSN is designed to be mobile), the lack of line-of-sight, the existence of obstructions and multipath effects often arise in realistic, outdoor environments.

Some preliminary work on the aforementioned issues are the following:

Calibration

Whitehouse formulates the calibration problem in WSN as a parameter estimation problem [33]. Each device in the WSN is parameterized and the values of the parameters are chosen such that the system performance is maximized (higher accuracy in location estimation). The author propose a macro-calibration procedure, called joint-calibration, that calibrates each device, by optimizing the overall system performance, instead of individual nodes. The steps of the joint calibration are the following:

1. Model the overall system, by using individual, device specific parameters.
2. Collection data.
3. Tune the parameters of individual devices, such that the overall system performance is improved.

The key insight into how to choose parameters to be tuned, such that the overall system performance improves is to look at trends in the transmitter/receiver pairs, and identify individual nodes for which the chosen parameters are problematic.

The proposed joint calibration is a good solution where manual calibration is possible. Obviously, in rugged, remote outdoor environments, auto-calibration (i.e., no manual intervention) is highly desirable.

Mobility

Hu and Evans [9] propose a sequential Monte-Carlo (SMC) localization algorithm for WSN in which sensor nodes and anchors are all mobile. The authors show that mobility can be used to enhance localization accuracy, a rather counterintuitive result - one would expect to be a significant impediment for an accurate positioning.

The proposed algorithm is an adaptation of the Sequential Monte Carlo localization scheme, frequently used in robot localization, target tracking and computer vision, to the domain of WSN. The main idea of the SMC localization algorithm is to represent the posterior distribution of possible locations using a set of weighted samples and to update them recursively in time.

From simulations of a 10Rx10R WSN, with an average number of nodes per transmission range of 10, the authors report localization errors of approximately 0.5R, when both sensor nodes and anchors move at a speed of R meters/sec. The localization error starts from high values (1.9R) and decreases rapidly, with the accumulation of new observations (nodes entering the ranges of new anchor nodes).

Line-of-Sight and Multipath

Real, outdoor environments pose significant challenges for range-free localization. Localization schemes designed and evaluated in "friendly" environments frequently fail to produce encouraging results in real deployments. When line-of-sight is a main assumption of the scheme [30], and does not always hold, or when obstructions and multipath for acoustic and radio waves are not considered [35], the performance of the localization scheme is degraded.

In order to address this, a potential direction to pursue is multimodal localization. In a multimodal localization system, more than one localization scheme is executed, in an attempt to reduce the impact the assumptions of a single localization scheme could have on the overall localization accuracy. By using Bayesian inference, and the knowledge (even if partial) obtained during the execution of one localization scheme, a finer, more accurate positioning can be obtained from the execution of subsequent localization schemes. For example, if a WSN is localized using the Line Scan scheme of the Spotlight system, described before, and due to some environmental conditions one of the two events created in the network is not detected (the Spotlight localization scheme fails to produce a location in this case), the knowledge gained from the detection of the other event can be used to initialize a subsequently executed localization scheme.

5 Conclusions

In this chapter we presented a suite of range-free localization schemes for WSN. We define ranging, in the context of sensor networks, as the ability of a sensor node to infer distances to its neighbor sensor nodes, either through localization specific hardware (e.g., ultrasound transceivers) or the strength of the received radio signal. Hence, the localization schemes presented here (i.e., range-free schemes) do not posses sophisticated hardware and do not rely on the received signal strength for inter-node ranging. The sensor nodes we consider have simple radio communication and sensing capabilities.

The taxonomy that we adopt for categorizing the range-free localization schemes is based on the (non)existence of an infrastructure of anchor nodes (i.e., at least three nodes, for a 2D localization, with known locations) in the WSN. An anchor-free localization scheme exploits the proximity to an event with a known location: a light event in Spotlight [30] and Lighthouse [24] and a nearby radio packet in Walking GPS [29].

One main observation is the high accuracy in localization of the anchor-free, event based, localization schemes, at a reduced, per node, cost. It is remarkable to obtain location accuracies of tens of centimeters, at zero dollar cost (if the sensor node is equipped with a photo sensor for the mission it was deployed for) and relatively low communication overhead (reduced energy cost). Characteristic to the anchor-free localization schemes, is the moving of the complexities associated with the localization from the sensor node to a capable, sophisticated device. While the cost of such device is not negligible, the possibility of its reuse make the event-based, anchor-free solutions very attractive. The anchor-free, event based, class of localization schemes seems a very promising direction for high accuracy, low cost localization in WSN.

Despite the extensive attention the range-free localization has received, several open problems remain. Among these are how to secure the radio communication and sensing channels that sensor nodes posses, how to make range-free localization more robust against attacks, node or protocol failures (possibly due to its strict assumptions), understand the impact of localization schemes on other protocols and how to design more robust, cost efficient, calibration techniques. The breadth and depth of all these issues present interesting opportunities for future research in the domain of range-free node localization in WSN.

References

1. Borg, I., and Groenen, P. *Modern Multidimensional Scaling.* Series in Statistics. Springer, 1997.
2. Bulusu, N., Heidemann, J., and Estrin, D. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine 7*, 5 (October 2000), 28–34.
3. Bulusu, N., Heidemann, J., and Estrin, D. Adaptive beacon placement. In *International Conference on Distributed Computing Systems (ICDCS)* (2001).
4. Capkun, S., and Hubaux, J. Secure positioning of wireless devices with application to sensor networks. In *IEEE Conference on Computer Communications (Infocom)* (2005).
5. Capkun, S., Srivastava, M., and Cagalj, M. Securing localization with hidden and mobile base stations. Tech. rep., NESL-UCLA, 2005.
6. Dutta, P., Grimmer, M., Arora, A., Bibyk, S., and Culler, D. Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In *International Symposium on Information Processing in Sensor Networks (IPSN)* (2005).
7. Fang, L., Du, W., and Ning, P. A beacon-less location discovery scheme for wireless sensor networks. In *IEEE Conference on Computer Communications (Infocom)* (2005).
8. He, T., Huang, C., Blum, B., Stankovic, J. A., and Abdelzaher, T. Range-Free localization schemes in large scale sensor networks. In *ACM International Conference on Mobile Computing and Networking (Mobicom)* (2003).
9. Hu, L., and Evans, D. Localization for mobile sensor networks. In *ACM International Conference on Mobile Computing and Networking (Mobicom)* (2004).
10. Karp, B., and Kung, H. T. Gpsr: Greedy perimeter stateless routing for wireless networks. In *ACM International Conference on Mobile Computing and Networking (Mobicom)* (2000).
11. Kleinrock, L., and Silvester, J. Optimum tranmission radii for packet radio networks or why six is a magic number. In *IEEE National Telecommunication Conference* (1978).

12. Ko, Y. B., and Vaidya, N. H. Location-aided routing (lar) in mobile ad hoc networks. In *ACM International Conference on Mobile Computing and Networking (Mobicom)* (1998).
13. Kwon, Y., Mechitov, K., Sundresh, S., Kim, W., and Agha, G. Resilient localization for sensor networks in outdoor environments. In *IEEE International Conference on Distributed Computing Systems (ICDCS)* (2005).
14. Lazos, L., Capkun, S., and Poovendran, R. Rope: Robust position estimation in wireless sensor networks. In *International Workshop on Information Processing in Sensor Networks (IPSN)* (2005).
15. Lazos, L., and Poovendran, R. SeRLoc: Secure range-independent localization for wireless sensor networks. In *ACM Workshop on Wireless Security (WiSe)* (2004).
16. Li, Z., Trappe, W., Zhang, Y., and Nath, B. Robust statistical methods for securing wireless localization in sensor networks. In *International Workshop on Information Processing in Sensor Networks (IPSN)* (2005).
17. Liu, D., Ning, P., and Du, W. Attack-resistant location estimation in sensor networks. In *International Workshop on Information Processing in Sensor Networks (IPSN)* (2005).
18. Maróti, M., Kusý, B., Balogh, G., Völgyesi, P., Nádas, A., Molnár, K., Dóra, S., and Lédeczi, Á. Radio interferometric geolocation. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)* (2005).
19. Moore, D., Leonard, J., Rus, D., and Teller, S. Robust distributed network localization with noisy range measurements. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)* (2004).
20. Nagpal, R., Shrobe, H., and Bachrach, J. Organizing a global coordinate system from local information on an ad hoc sensor network. In *International Workshop on Information Processing in Sensor Networks (IPSN)* (2003).
21. Navas, J. C., and Imielinski, T. Geographic addressing and routing. In *ACM International Conference on Mobile Computing and Networking (Mobicom)* (1997).
22. Niculescu, D., and Nath, B. Ad-hoc positioning system. In *IEEE Global Communications Conference (GLOBECOM)* (2001).
23. Priyantha, N. B., Balakrishnan, H., Demaine, E., and Teller, S. Mobile-assisted localization in wireless sensor networks. In *IEEE Conference on Computer Communications (Infocom)* (2005).
24. Römer, K. The lighthouse location system for smart dust. In *ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys)* (2003).
25. Sastry, N., Shankar, U., and Wagner, D. Secure verification of location claims. In *ACM Workshop on Wireless Security (WiSe)* (2003).
26. Savarese, C., Rabaey, J., and Langendoen, K. Positioning algorithms for distributed ad-hoc wireless sensor networks. In *USENIX Annual Technical Conference* (2002).
27. Shang, Y., and Ruml, W. Improved mds-based localization. In *IEEE Conference on Computer Communications (Infocom)* (2004).
28. Shang, Y., Ruml, W., Zhang, Y., and Fromherz, M. P. J. Localization from mere connectivity. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)* (2003).

29. Stoleru, R., He, T., and Stankovic, J. A. WalkingGPS: A practical localization system for manually deployed wireless sensor networks. In *IEEE Workshop on Embedded Networked Sensors (EmNetS)* (2004).
30. Stoleru, R., He, T., Stankovic, J. A., and Luebke, D. A high-accuracy low-cost localization system for wireless sensor networks. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)* (2005).
31. Stoleru, R., and Stankovic, J. A. Probability grid: A location estimation scheme for wireless sensor networks. In *IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON)* (2004).
32. Vicaire, P., and Stankovic, J. A. Elastic localization. Tech. Rep. CS-2004-35, University of Virginia, 2004.
33. Whitehouse, K., and Culler, D. Calibration as parameter estimation in sensor networks. In *ACM Intenational Workshop on Sensor Networks and Applications(WSNA)* (2002).
34. Xu, Y., Heidemann, J., and Estrin, D. Geography-informed energy conservation for ad hoc routing. In *ACM International Conference on Mobile Computing and Networking (Mobicom)* (2001).
35. Zhang, J., Yan, T., Stankovic, J. A., and Son, S. H. Thunder: A practical acoustic localization scheme for outdoor wireless sensor networks. Tech. Rep. CS-2005-13, University of Virginia, 2005.

Design and Comparison of Lightweight Group Management Strategies in EnviroSuite*

Liqian Luo, Tarek Abdelzaher, Tian He, and John A. Stankovic

Department of Computer Science, University of Virginia,
Charlottesville, VA 22904
{ll4p, zaher, th7c, stankovic}@cs.virginia.edu

Abstract. Tracking is one of the major applications of wireless sensor networks. *EnviroSuite*, as a programming paradigm, provides a comprehensive solution for programming tracking applications, wherein moving environmental targets are uniquely and identically mapped to logical objects to raise the level of programming abstraction. Such mapping is done through distributed group management algorithms, which organize nodes in the vicinity of targets into groups, and maintain the uniqueness and identity of target representation such that each target is given a consistent name. Challenged by tracking fast-moving targets, this paper explores, in a systematic way, various group management optimizations including semi-dynamic leader election, piggy-backed heartbeats, and implicit leader election. The resulting tracking protocol, *Lightweight EnviroSuite*, is integrated into a surveillance system. Empirical performance evaluation on a network of 200 XSM motes shows that, due to these optimizations, Lightweight EnviroSuite is able to track targets more than 3 times faster than the fastest targets trackable by the original EnviroSuite even when 20% of nodes fail.

1 Introduction

The increasing popularity of sensor networks in large-scale applications such as environmental monitoring and military surveillance motivates new high-level abstractions for programming-in-the-large. As a result, several programming models that encode the overall network behavior (rather than per-node behaviors) have been proposed in recent years. Examples include virtual machines [1][2], and database-centric [3][4][5], space-centric [6][7], group-based [8][9], and environment-based [10] programming models, which offer virtual machine instruction sets, queries, sensor node groups and environmental events, respectively, as the underlying abstractions with which the programmer operates. These abstractions capture the unique properties of distributed wireless sensor networks and expedite software development.

* The work reported in this paper is funded in part by NSF under grants CCR-0208769 and ITR EIA-0205327.

An important category of sensor network applications involves tracking environmental targets. In these applications, some internal representation of the external tracked entity is maintained such as a state record, a logical agent, or a logical object representing the physical target. A given target in the environment should be represented uniquely and its identity should be preserved consistently over time. One way to ensure unique, consistent representation is to relay all sensor readings to a centralized base-station which runs spatial and temporal correlation algorithms to infer the presence of targets, assign them unique identities, and maintain such identities consistently. Such a centralized approach, however, is both inefficient and vulnerable. In addition to relying on a single point of failure, it results in excessive power consumption due to communication with a centralized bottleneck and may unduly increase latency, especially when targets move far away from the base-station.

To avoid these limitations, in EnviroSuite, we take the alternative approach of processing target data at or near the location where the target is sensed. Hence, appropriate distributed group management policies are needed to ensure the uniqueness and identity of target representation such that targets are given consistent names and sensors agree on which target they are sensing. This paper systematically investigates different system optimizations in the design of such group management algorithms in a sensor network. The resulting tracking system, *Lightweight EnviroSuite*, is used in a sensor network surveillance prototype that has since been transferred to the Defense Intelligence Agency (DIA). It is evaluated on a sensor network of 200 XSM motes [11]. Results from field tests of the overall system are provided, focusing on tracking performance. (Other performance aspects of the system such as efficacy of energy management algorithms will be reported elsewhere.) It is seen that realistic targets can indeed be tracked correctly despite environmental noise using low-range sensors. Our field test results show that even when 20% of nodes fail, Lightweight EnviroSuite is able to track targets more than 3 times faster than the fastest targets trackable by the original EnviroSuite, due to the optimizations described in this paper. The improved tracking coincides with reduced communication cost.

The remainder of the paper is organized as follows. Section 2 presents the background information and enumerates limitations of the original EnviroSuite. Section 3 explores group management strategies and their effects that constitute Lightweight EnviroSuite. Section 4 analyzes the performance results of a surveillance system that is constructed from Lightweight EnviroSuite. Finally, Section 5 supplies a summary and concludes the paper.

2 Background

Target tracking has received special attention in recent ad hoc and sensor networks literature. Many prior approaches (e.g., in the ubiquitous computing and communication domains) focused on tracking cooperative targets. Cooperative targets are those that allow themselves to be tracked typically by exporting a unique identifier to the infrastructure (such as a cell-phone number). Examples of cooperative targets include cell phones, RFID tags [12] and smart badges [13]. Since such devices are preconfigured with a unique identity, the tracking problem is generally reduced to that of locating the uniquely identified device and performing hand-offs if needed (e.g., in cellular phones). In contrast, our goal is to track non-cooperative targets such as enemy vehicles that do not broadcast self-identifying information. The presence and identity of such targets can only be inferred from sensory signatures, as opposed to direct communication with the target. Tracking non-cooperative targets is more challenging due to the difficulty in associating sensory signatures with the corresponding targets (e.g., all tanks look the same to our unsophisticated motes). The presence of target mobility further complicates the tracking problem.

One of the first research efforts on group management for non-cooperative target tracking has been conducted by researchers at PARC [14]. Their group management method dynamically organizes sensors into collaborative groups, each of which tracks a single target. Typical tracking problems such as multi-target tracking and tracking crossing targets are solved elegantly in a distributed way. Other approaches to target tracking include [15] which presents a particle filtering style algorithm for tracking using a network of binary sensors which only detect whether the object is moving towards or away from the sensor. A scalable distributed algorithm for computing and maintaining multi-target identity information in described in [16]. In [17] a tree-based approach is proposed to facilitate sensor node collaboration in tracking a mobile target.

The authors have investigated the tracking problem in several of their own prior publications. Similar to [8], in [18] we present a set of group management algorithms which form sensor groups at the locations of environmental events of interest and attach logical identities to the groups. Based on [18], EnviroTrack [19] proposes an environmental computing paradigm which facilitates tracking application development. EnviroSuite [20] further extends the paradigm to support a broader set of applications that are not limited to target tracking. Geared for tracking of fast-moving targets with low communication cost on available hardware platforms that have limited sensing and communication abilities, this paper proposes lightweight group management algorithms for EnviroSuite. Optimizations described in this paper may be applicable to other systems such as [8] as well.

The design of EnviroSuite assumes that each node can independently detect the potential presence of a target (subject to false alarms). For example, the presence of a magnetic signature, motion, and engine sound can be independently detected by each XSM mote to signify the potential presence of a nearby moving vehicle in a desert surveillance scenario. It is further assumed that sensor readings do not interfere with each other. Hence, tracking reduces to the problem of correct mapping of nodes that detect target signatures to the actual physical target identities responsible for these signatures. EnviroSuite therefore organizes nodes that detect targets of interest into groups, each representing one target.

Different from traditional centralized tracking schemes, the data association between targets and groups in EnviroSuite is done in a distributed way. Namely, all nodes that sense a target and can communicate directly assume that they sense the same target and consequently join the same group. The resulting aggregate behavior is that connected regions of sensors that sense the same signature are fused into the same group. Observe that when the target moves, group membership changes reflecting the changing set of sensors that can sense it at the time. A leader is elected for the group among the current members. A leadership hand-off algorithm ensures that group state is passed to each new leader.

In this paper, we focus on point targets (i.e., those approximated by a point in space, such as a vehicle), as opposed diffuse region targets (given by an area, such as a chemical spill). It is assumed that the communication range of a node is sufficiently larger than its sensing range. Hence, all group members sensing a point target are within each other's radio range. Consequently, the data dissemination scheme within each group can simply use local broadcast to share sensory information. The leader performs data fusion in an application-specific manner to collect higher-level target information. Geographic forwarding [21] is used for communication with destinations external to the group. For example, in the evaluation section, each group leader estimates target position by averaging locations of group members and sends the result to remote base-stations periodically. Extensions of this scheme to diffuse region targets are described elsewhere [20].

This section briefly reviews the EnviroSuite programming paradigm and its core component, *MGMP* (*multi-target group management protocol*). It also analyzes and evaluates the limitations of MGMP when facing the practical requirements of a typical surveillance system.

2.1 EnviroSuite Abstractions and Challenges

EnviroSuite [20] is an object-based framework that supports *environmentally immersive programming* for sensor networks. Environmentally immersive programming refers to an object-based paradigm in which logical objects and objects representing physical environmental entities are seamlessly combined. Hence, EnviroSuite differs from other object-based systems in that its objects may be representations of elements in the external environment. At the implementation level, such objects are maintained by the corresponding group leaders. Upon detection of external elements of interest, nodes detecting the element self-organize into a group. The group leader in EnviroSuite dynamically creates an object instance to represent the tracked target. The group management protocol maintains a unique and identical mapping between object instances (or group leaders) and the corresponding environmental elements they track, such that object instances float across the network geographically following the elements they represent. This co-location is ideal for the execution of location sensitive object code that carries out sensing and actuation tasks. Objects encapsulate the aggregate state of the elements they represent (collected and stored by group leaders), making such state available to their methods. They are therefore the units that encapsulate program data, computation, communication, sensing and actuation. Object instances are destroyed when their corresponding environmental elements leave the network. This occurs naturally when the membership of the corresponding sensor group is reduced to zero.

Objects can be point objects (created for mobile targets that dynamically change their geographical locations), region objects (mapped to static or slowly moving regions), or function objects (not mapped to an environmental element). EnviroSuite is able to support both point objects and region objects in the same framework due to their similarities. Namely, (i) the corresponding external elements are detected by a group of geographically continuous nodes, and (ii) the aggregate state of the elements are collected by group leaders. However, the focus of this paper is on target tracking applications. Hence, we discuss mainly maintenance of point objects.

The biggest problem faced in EnviroSuite is the challenge of maintaining a unique and identical mapping between each object and the corresponding environmental element despite of distribution and possible mobility in the environment. In the rest of this paper, a *target* refers to a geographically continuous activity in the physical environment that persists over some interval of time. *Object uniqueness* dictates that each target be represented and that it be represented by exactly one logical object instance. *Object identity* dictates that the mapping between targets and objects be immutable. In other words, a target is always mapped to the same object instance identified by its *object ID*.

The problem of object uniqueness and identity is complicated by several factors. One is the need for seamless object migration across nodes as the target moves. Another is that sensor nodes that become aware of an external target should be able to tell whether it is a target previously seen by other neighboring sensors or not. Otherwise, an incorrect target list will be collectively maintained or an incorrect mapping will result between targets and objects. In the following we describe a solution to these problems.

2.2 MGMP and Its Limitations

The core component of EnviroSuite previously proposed by us is a set of multi-target group management protocols, named MGMP, which resolves the object uniqueness and identity problem. When predefined target signatures are detected by a set of nearby nodes, MGMP reacts by creating a *group* attached with a unique object ID. The set of nodes become group *members*, whose task is to periodically sense, calculate and report predefined object *attributes* (such as temperature, location, etc.) to the group *leader*. These reports are called *member reports*. A single leader is elected among these members to uniquely represent the group as one object to the external world. To avoid electing a node that is imminently going out of sensing range which results in yet another election, it is preferred to elect nodes near the target. Though current leader election doesn't enforce such preference, it can be easily adapted to do so if distances to the target can be inferred from detection results.

The leader is responsible for the maintenance of object attributes. It records member reports keeping only the most recent one from each member. It periodically creates a digest of the reports, and either keeps it as the internal state or

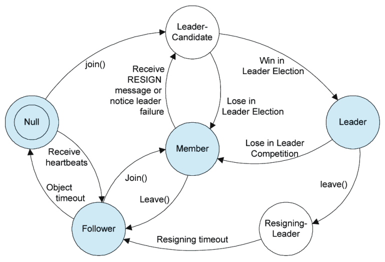


Fig. 1. Node state transitions in MGMP

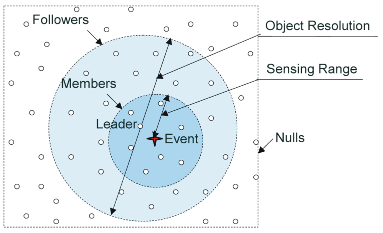


Fig. 2. States of nodes around a target

sends it to the external world (e.g., base stations) based on user specification. These reports are called *leader reports*. The leader is also responsible for object uniqueness and identity maintenance. It periodically sends leader heartbeats to nodes within half an *object resolution* (in meters, defined by users) to advertise the object ID as well as its internal states. By design, half the object resolution must be larger than twice the sensing range such that every member can receive leader heartbeats. Nodes within half the object resolution, which cannot sense, but are aware of the target through heartbeats, are called group *followers* as distinguished from members. Follower nodes are prevented from spawning new groups. Follower nodes are centered around the leader since leader locations provide a good approximation of target positions. Though centering around the target would be a better solution, given the fact that both communication range and sensing range are irregular, the extra complexity required to do so is not worth it.

Nodes dynamically join or leave the group whenever they detect or lose the target. If a leader loses the target, it sends out a `RESIGN` message to request a leadership handoff. Upon the reception of such messages, the most current members reelect a new leader to take over leadership. Figure 1 illustrates the complete node state transitions in MGMP and Figure 2 depicts a typical node state distribution around a target. MGMP employs two important strategies to enhance robustness in the face of failures:

1. **Dynamic Leader Election:** Leaders are always dynamically elected among all current members. There are no pre-designated leader candidates. When leader election starts, each member sets a timer at random from 0 up to *maximum back-off time* and, when the timer expires, claims its leadership by messages, the reception of which terminates other members' timers as well as their unsent messages. This strategy ensures robustness to node failures.
2. **Periodic Leader Heartbeats:** Leaders periodically send out heartbeats so that leader failure can be detected by neighboring nodes.

The main goal of our deployed system is to alert a military command and control unit of the occurrence of targets of interest in hostile regions. Targets of interest may include civilian persons (unarmed), armed persons or vehicles. The system is required to obtain and report current positions of such targets to a remote base station to create tracks in real time. Several application requirements must be satisfied to make this system useful in practice. First, the application must have the ability to track typical military vehicles with velocities varying from 5 mph to 35 mph. Object uniqueness and identity must be ensured. Second, in our application, real-time updates on target trajectory must be sent to a base-station to be used by other devices such as cameras, which requires high accuracy and low reporting latency. Given the severely constrained bandwidth of current mote platforms, the communication cost should be minimized to reduce communication latency and maximize information throughput.

Does MGMP satisfy these requirements? We first try to answer the question through experimental results on TOSSIM [22]; a simulator for TinyOS [23] that emulates the execution of application code on the motes. Our experiments consist of 120 nodes deployed in a 30×4 grid 10 meters apart. Sensing range is set to be 1 grid length and communication range is 3 grid lengths. These settings reflect our real system where sensor devices are deployed in a grid 10 meters apart, sensing range is around 10 meters, and communication range is approximately 30 meters. We simulate a target moving across the field in a straight line to test tracking performance. The target is tracked with a sensor polling period of 0.02 s. Consistent with the real system requirements, members report to leaders their own locations twice a second. Leaders triangulate received locations to estimate target position and report estimations to the base station (located in a corner node) twice a second. The same testbed is used in later sections to evaluate new schemes.

Figure 3 shows the number of objects formed for the single target during its presence in the field. The uniqueness of target representation requires that only one object be formed. As is seen in figure, this is not always the case. The number of objects formed is one at lower target velocities, but it increases as target velocity increases. This violation is due to the difficulty in reaching agreement on target identity quickly enough which leads some sensors to believe that they are seeing different targets. The effects of maximum back-off time are more subtle. As is seen from Figure 3, if maximum back-off time is too small such as 0.2 s, the number of objects generated can be large since multiple nodes may become leaders and create new groups at the same time to represent the same target. If it is too large, fast targets may move out of the sensing range of a node before its back-off timer expires, so that the current object is lost and spurious ones are created. In theory, it is possible to derive the appropriate back-off time analytically for a particular target velocity. The main idea is that leader migration (via election and hand-off) should be faster than target speed for the target to never escape its tracking group. Nevertheless, such a derivation would have to be experimentally validated since it is difficult to account for various imperfections such as the irregularity of the sensing and communication ranges and the non-uniform distributions of nodes in practice.

Observe that no back-off timer value in Figure 3 can maintain object uniqueness at target speeds more than 1 grid length per second (grid/s) or 22 mph (since grid length is 10 meters). These results are far from the desired perfor-

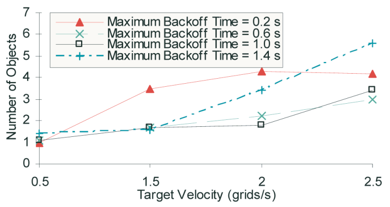


Fig. 3. Number of objects for varied maximum back-off time and target velocities

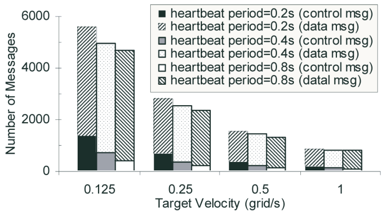


Fig. 4. Number of messages for varied heartbeat periods and target velocities

mance (35 mph). The overhead of dynamic leader election is the main reason why better results cannot be achieved, since long leader-election delays slow down the migration of groups, thus making fast-moving targets untrackable.

Figure 4 depicts the number of *control messages* (leader heartbeats and other group maintenance messages) and *data messages* (member reports and leader reports) sent during the presence of a target. The number of data messages decreases with increasing target velocity, because it is proportional to the duration of target presence due to periodicity of member reports and leader reports. The number of control messages exhibits similar trends since heartbeats that dominate control messages are also periodic. Obviously, control messages also decrease with longer heartbeat periods. However, minimizing communication cost by indefinitely increasing heartbeat periods is not feasible since longer heartbeat periods increase the vulnerability to message loss.

The above observations give insights into improvements to EnviroSuite that enhance tracking performance and reduce communication cost while maintaining robustness to failures. These improvements are described next.

3 Group Management Strategies in Lightweight EnviroSuite

This section explores in more detail the performance problems of current strategies, proposes a series of new strategies, and applies them one by one to EnviroSuite to verify their individual effects on the current system. These new group management strategies, as a whole, constitute a very practical and efficient version of EnviroSuite, called *Lightweight EnviroSuite*.

3.1 Semi-dynamic Leader Election

Dynamic leader election, as the main factor that limits tracking performance of MGMP, affects the maintenance of object uniqueness and identity in two ways. First, it causes long leader handoff delays. In dynamic leader election, all members are competitors for leadership. Hence, consensus has to be achieved among all on a single leader. Obviously, the more members participate, the slower

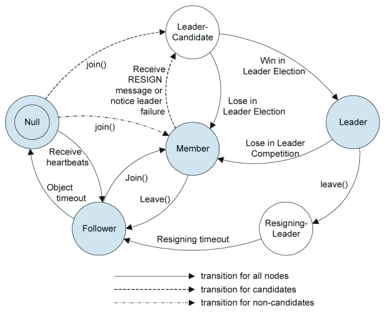


Fig. 5. Node state transitions for semi-dynamic leader election

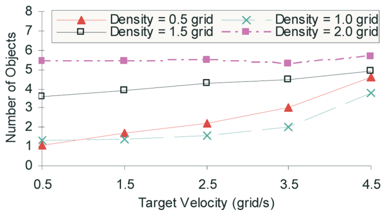


Fig. 6. Number of objects for varied target velocities and candidate densities

the consensus. Second, it increases the possibility of message collisions since all members are exchanging messages to compete for leadership.

A better solution is to allow only a portion of all members to compete for leadership, which we call *semi-dynamic leader election*. Semi-dynamic leader election includes an initialization phase which pre-elects a portion of the nodes to be *candidates* (the potential competitors for leadership); others become *non-candidates*. The pre-election of candidates is similar to dynamic leader election in EnviroSuite. Each node sets a random timer and, when the timer expires, claims itself as a candidate. Nodes within distance x that receive this claim message become non-candidates. Ideally, the algorithm elects at least one candidate within any circular area of radius x. We call this x the *candidate density*. The node state transition changes accordingly as shown in Figure 5. Transitions to `Leader-Candidate` occur only when the corresponding nodes are candidates. `Null` nodes become `Members` instead of `Leader-Candidates` when they are non-candidates. Since only `Leader-Candidate` nodes attend leader election, these changes make candidates the only ones competing for leaderships.

Figure 6 illustrates the number of objects created for targets with different velocities when different candidate densities are set. Semi-dynamic leader election allows for a smaller maximum back-off time (set to 0.2 s in the following experiment) in leader election due to a reduced number of competitors. The $Density = 0.5\,grid$ curve performs the same as dynamic leader election since all nodes are candidates (maximum back-off time is set to 0.6 s for better performance in this case). As seen from Figure 6, a proper candidate density, say 1.0 grid, makes the semi-dynamic scheme outperform the dynamic one.

Observe that, a very low candidate density results in worse performance than dynamic leader election since candidates are so scarce that, in most groups, no leader is elected to maintain objects. We call the phenomenon a *leader desert*. The dark grey circle in Fig. 7 shows a leader desert where no candidate exists. If the target moves further to the right and gets detected by the nearest candidate outside the follower set, a spurious object is created by the candidate since it is not aware of the existing object. Even when candidate density is 1.0 grid, a leader desert still appears occasionally, which hurts object uniqueness slightly.

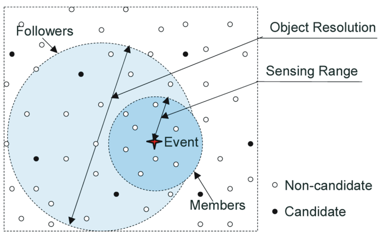


Fig. 7. Leader desert

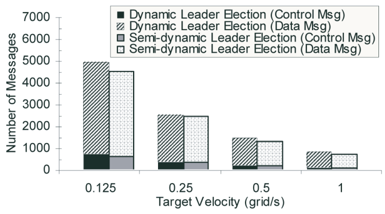


Fig. 8. Comparison of number of messages for different target velocities

This explains why semi-dynamic leader election performs a little worse when target velocity is 0.5 grid/s. The leader desert problem is solved in later sections. As a side-effect, semi-dynamic leader election also results in lower communication cost since fewer control messages are sent to compete for leadership, which is shown in Fig. 8.

On the disadvantage side, robustness to node failures is expected to degrade when using semi-dynamic leader election due to a higher vulnerability to failures of leader candidates. However, this can be partially compensated by executing candidate pre-election more frequently. We discuss the overall failure robustness of the new scheme in later sections.

3.2 Piggy-Backed Heartbeat

Periodic leader heartbeat entails big overheads that are not affordable in applications with severe bandwidth constraints. Yet, it plays the most critical role in MGMP. First, it recruits followers to prevent these boundary nodes from creating spurious groups. Second, its periodicity makes leader failures perceivable, and thus recoverable. Third, the periodicity also improves robustness to message loss. Therefore, the challenge is how to reduce overhead while retaining the advantages of frequent heartbeats.

Fortunately, another component in MGMP exhibits the behavior of sending periodic messages; namely, *object attribute collection*. Members periodically send sensed attribute data to leaders and leaders periodically aggregate received data, process it, and send results to the external world if required. If heartbeats can be piggy-backed into these member reports and leader reports, periodic heartbeat becomes almost free. We call this new scheme *piggy-backed heartbeat*, where heartbeats are transformed into *leader heartbeats* (heartbeats piggy-backed into leader reports) and *member heartbeats* (heartbeats piggy-backed into member reports). Leader desert is no longer an obstacle to object state dissemination, since members take over this task during leader absences. Since object uniqueness is ensured through leader uniqueness, members are only allowed to repeat heartbeats originated from leaders and leaders are still the only authority that may update object information.

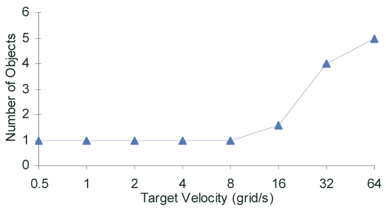


Fig. 9. Object maintenance in semi-dynamic leader election

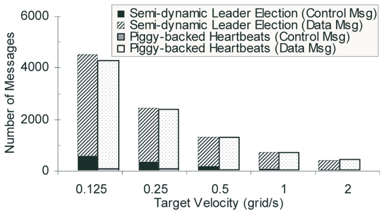


Fig. 10. Comparison of communication overhead

In the piggy-backed heartbeat scheme, member heartbeats and leader heartbeats are treated differently: only the reception of leader heartbeats transits pre-elected candidates from state `Null` to `Follower`, while member heartbeats transits them into an intermediate state, called `Null-Follower`. If a node detects a target while in this state, it transits to Leader-Candidate to compete for leadership. This is unlike a regular `Follower`, which becomes a `Member` upon target detection. Without these changes, member heartbeats may transit all potential leaders to `Follower` state and then to `Member` upon the detection of the target, making the group follow a target without any leaders.

The aforementioned efforts make the piggy-backed heartbeat scheme a big improvement in object maintenance. Compared with the maximum trackable velocity seen in MGMP (1 grid/s), this improved version maintains object uniqueness and identity for targets with velocities up to 8 grid/s as shown in Figure 9. Figure 10 suggests another big improvement in reducing control messages.

3.3 Implicit Leader Election

It is possible to further reduce the protocol costs by employing an *implicit leader election* scheme. An assumption is made in this scheme that monitoring tasks are periodic and that after each period monitoring results are communicated by each tracking group to the external world. This assumption is reasonable since periodicity is a typical property of sensor network applications. The scheme allows candidates to start the execution of leader tasks such as data aggregation, whenever they detect the target. Note that, their task periods are unsynchronized since nodes usually do not begin to sense the target at exactly the same time. As a result, multiple but limited potential leaders are executing tasks in a group. At any point in time, if the node that first reaches the end of a task period sends out a result report, other neighboring nodes including other potential leaders simply accept the results and become *inactive* in their current task periods, which prevents them from reporting the same redundant results when finishing their periods. Hence, the external world sees the illusion of a single group leader.

Figure 11 illustrates an example. Candidate A senses the target from time 0 to 2.5 and B from 0.25 to 3.5. The length of task period is 1. Both A and B are initially active. At time 1, A reaches the end of its period and sends a result report. Receiving this report, B admits A as the current leader, accepts

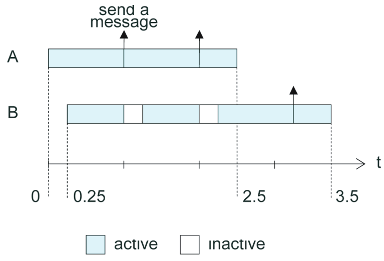


Fig. 11. An example of implicit leader election

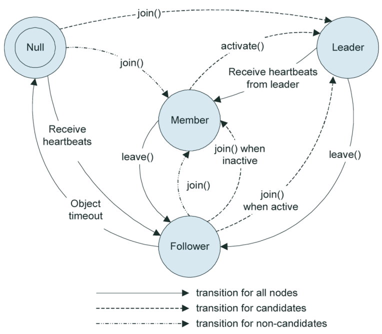


Fig. 12. Node state transitions for implicit leader election

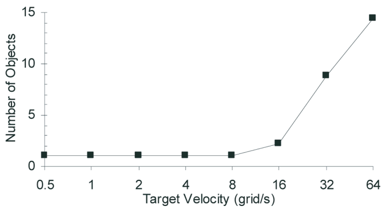


Fig. 13. Object maintenance in implicit leader election

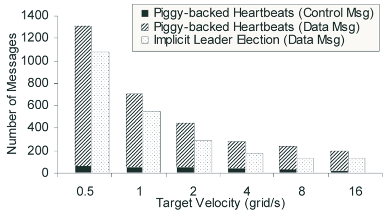


Fig. 14. Comparison of communication overhead

A's results and makes itself inactive, which makes B silent at time 1.25 when it finishes its task period. Similarly B is still silent in the next period since A sends a message first. Finally, A quits the leader competition at time 2.5 when it loses the target. B continues A's work and reports the results at the end of its third period (time 3.25). This way, tasks are executed continuously between different leaders, the results of which are exposed to the external world at a rate ($3/3.5 \approx 0.9$ report/s) that is very near to the defined rate (1 report/s).

Figure 12 depicts the new node state transition graph. `activate()` is called by each node when beginning a new task period. Different from all previous versions, intermediate states (`LeaderCandidate`, `ResigningLeader`) no longer exist since implicit leader election eliminates the need for nodes to stay at the `LeaderCandidate` state sending `CANDIDATE` messages to compete for leadership and to stay at `ResigningLeader` sending `RESIGN` messages to start new leader election. The elimination of control messages further improves communication performance as shown in Figure 14. Meanwhile, a comparable performance in object maintenance (maximum trackable velocity 8 grid/s) is achieved as Figure 13 shows.

Note that implicit leader election can not be applied to applications where duplicate execution of tasks is not allowed or where tasks are not periodic. However, since the EnviroSuite compiler [20] has the ability to dynamically select modules during compilation based on programmer's application definition, a version without implicit leader election can be composed in such cases.

Overall, due to the optimizations mentioned above, Lightweight EnviroSuite achieves roughly an order of magnitude improvement in maximum trackable velocity (8 grid/s compared with 1 grid/s in EnviroSuite). The number of messages per second for targets at 0.5, 1 and 2 grid/s is reduced 25%, 12% and 37%, respectively.

3.4 Failure Tolerance

This section discusses the overall robustness of Lightweight EnviroSuite to typical failures in the sensor network: message loss and node failures. Message loss harms object maintenance by making the existence of the current leader unnoticed by other candidates, which may result in multiple nodes sending duplicate leader reports in the same period to the external world. However, the external world can always recognize duplicate reports through version numbers attached in the reports. Message loss can also prevent nodes on a group's outer boundaries from getting heartbeats. Consequently, these nodes may not become aware of the object and create spurious objects. However, Lightweight EnviroSuite allows members to disseminate heartbeats, which maintains a comparatively high heartbeat frequency and makes the possibility that a node fails to get any heartbeats very low. At the same time, objects attach their ages, which increase with the increase of finished task periods, to heartbeats. Object information from younger objects is discarded in favor of older ones. Therefore, spurious objects are eventually terminated due to their young ages.

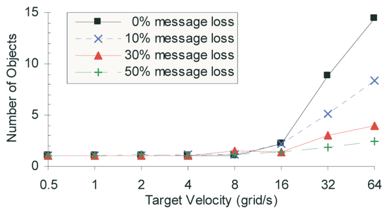


Fig. 15. Performance of object maintenance for varied message loss

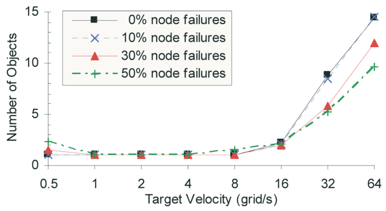


Fig. 16. Performance of object maintenance for varied node failures

Lightweight EnviroSuite is also robust to node failures. As stated earlier, it can go through leader deserts without losing object information or terminating task execution. In a similar way, it is able to overcome node deserts smaller than half an object resolution. Although we may temporarily lose track of the target inside a node desert, the target and its associated object can be picked up again in most cases after passing the node desert.

Experimental results confirm our conclusions as shown in Figure 15 and Figure 16. Lightweight EnviroSuite shows consistently good performance in object maintenance for targets with velocities up to 4 grid/s. However, after velocities exceed 8 grid/s, surprisingly bigger message loss results in fewer objects. This is because the number of objects is counted based on leader reports at the base station and a higher message loss results in fewer received leader reports, and thus fewer observed objects. For node failures up to 50%, Lightweight EnviroSuite exhibits comparable performance at velocities between 1 and 4 grid/s. However, when target velocities are as low as 0.5 grid/s, higher node failures do hurt performance. That is because higher node failures result in larger node or leader deserts. Slower targets may fail to cross the deserts before followers forget about the object.

4 System Evaluation

We integrated Lightweight EnviroSuite into an energy-efficient surveillance system, called *Vigilnet* [24], subsequently transitioned to the DIA. In December 2004, in the process of technology transition, we deployed 200 XSM motes running the Vigilnet system on sandy and grassy roads with a 3-way intersection and collected performance data in field tests. Figure 17 depicts the deployment of the system. Nodes are approximately deployed in a grid 10 meters apart, covering one 300-meter road and one 200-meter road. Each rectangular dot represents one XSM mote in the field. Several base-stations were deployed. Some nodes are missing in the GUI because they are turned off to emulate failures.

The XSM mote extends the MICA2 platform [25] by improved peripheral circuitry, new types of sensors and better enclosures. It communicates approximately 30 meters when deployed on grassy ground. The primary goal of the field test is to evaluate system ability to detect, classify and track one or multiple moving targets, which can be either SUVs, persons or persons carrying a ferrous object (suggestive of a weapon).

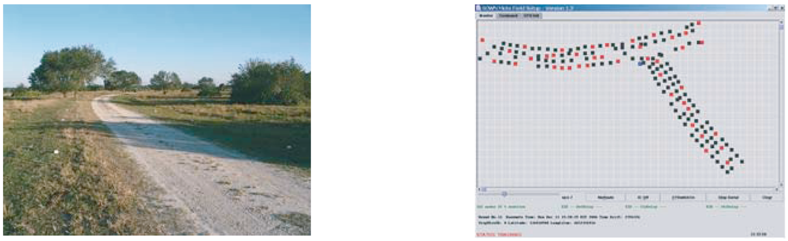


Fig. 17. System deployment

4.1 Overview of Vigilnet

Vigilnet is implemented on top of TinyOS. Figure 18 shows the layered architecture of Vigilnet. Components colored in dark grey are those implemented by Lightweight EnviroSuite.

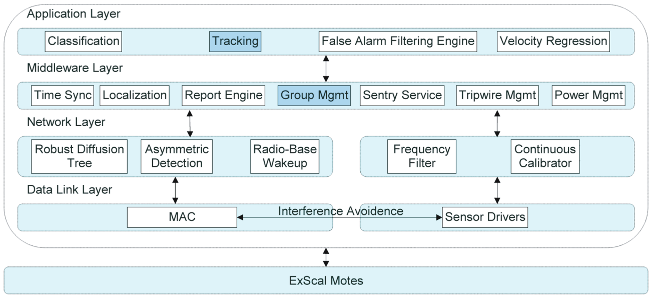


Fig. 18. System architecture of Vigilnet

Time synchronization (Time Sync), localization (Localization), and communication (MAC, Robust Diffusion Tree, Asymmetric Detection, and Report Engine) services constitute the lower-level components that are the basis for implementing higher-level services. Power management (Radio-Base Wakeup, Sentry Service, Power Mgmt, and Tripwire Mgmt), target classification (Sensor Drivers, Frequency-filter, Continuous Calibrator, Classification, and False Alarm Filtering Engine) and tracking (Group Mgmt, Tracking, and Velocity Regression) comprise main higher-level services. Target classification detects and classifies three types of targets with the help of collaborative group management provided by Lightweight EnviroSuite. Tracking components are responsible for estimating target positions and calculating target velocities.

Overall the system consists of 21,457 lines of source code, among which 2,884 are contributed by Lightweight EnviroSuite. The executable binary of Vigilnet occupies 85,926 bytes of code memory and 3,154 bytes of data memory, which can easily fit into XSMs equipped with 4KB data memory and 128KB code memory.

4.2 Tracking Performance Evaluation

Consistent with simulation, tracking modules in Vigilnet report to the base station estimations of target positions twice every second to provide sufficient data for false alarm processing and classification. A spanning-tree based routing [24] is used to disseminate such reports. The communication latency of these reports plays a critical role in achieving good tracking performance. Therefore, we suggest that when the system scales up to cover bigger fields, multiple base stations should be deployed. The false alarm filtering engine component executed in the base mote filters these reports and slows down the report rate to upper layers to once every 3 seconds due to bandwidth limitations. Target velocity calculation takes such reports as inputs.

Table 1 lists the comparison of tracking performance between Vigilnet equipped with the original EnviroSuite (measured at a previous field test conducted in August 2003) and Vigilnet equipped with Lightweight EnviroSuite. As is seen, without Lightweight EnviroSuite the maximum trackable velocity is about 5 to 10 mph, while the new Vigilnet system tracks targets up to 35 mph.

Table 1. Tracking performance comparison

Target vel.	Vigilnet with EnviroSuite	Vigilnet with Lightweight EnviroSuite
5 mph	successful	successful
10 mph	partially successful	successful
20 mph	failed	successful
30 mph	untested	successful
35 mph	untested	partially successful

Table 2. Tracking performance of Vigilnet with Lightweight EnviroSuite

Target type		Avg. tracking error	Std. dev. of tracking errors	Actual vel.	Calculated vel.
walking person		6.19 meter	3.28 meter	3±1 mph	2.9 mph
running person		6.67 meter	3.89 meter	7±1 mph	6.9 mph
vehicle		7.06 meter	3.98 meter	10±1 mph	10.5 mph
vehicle		5.91 meter	3.02 meter	20±1 mph	23.5 mph
two vehicles	1	5.58 meter	4.76 meter	10±1 mph	9.2 mph
	2	6.33 meter	3.52 meter	10±1 mph	9.9 mph

Note that, the success of tracking is an end-user metric measured by the accuracy of position and velocity calculations, which depend on several factors besides EnviroSuite group management protocols. A track is said to be successful only when the final calculated velocity within a 20% error. Due to the limited length of the field and the fixed report rate (once every 3 seconds), velocity calculation does not perform well when the velocity reaches 35 mph. However, the tracking performance of Lightweight EnviroSuite itself is actually better than the reported results for the integrated system.

Table 2 shows in more details the tracking performance of the new Vigilnet system. As seen, tracking errors are between 5.5 meters and 7.5 meters. These results were collected with 20% of the nodes randomly turned off to emulate failures. In all listed targets whose velocities vary from 3 mph to 20 mph, the maximum error of velocity calculation is less than 10%, which reflects the good tracking performance supplied by Lightweight EnviroSuite.

To give a more concrete view of the tracking performance of Lightweight EnviroSuite, Figure 19 shows the tracking trajectories for the following scenarios: (i) one vehicle drives across the field from left to right; (ii) two vehicles keep a distance of about 50 meters before they separate (the first one goes from left to right and turns right at the intersection and the second one goes from left to right). In the one-vehicle-tracking case, the rugged trajectory in the center of the horizontal road shows explicitly that existing node failures do affect tracking accuracy. The two-vehicle-tracking case proves the ability of Lightweight EnviroSuite to track multiple targets with the same sensory signatures as long as they keep a distance (50 meters) that is more than half an object resolution (set to 30 meters in the system). This was deemed sufficient by the client for operational use.

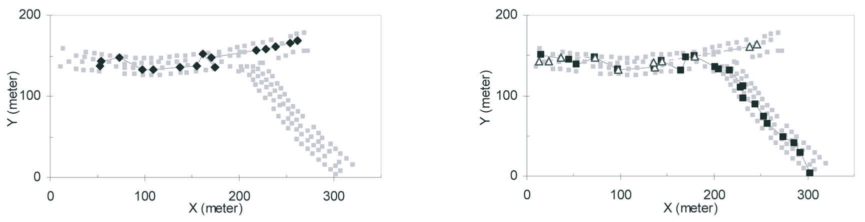


Fig. 19. Tracking trajectories for one vehicle and two vehicles

Field test results show that Lightweight EnviroSuite successfully improves the maximum trackable speed from near 10 mph to near 35 mph. Given our 10-meter-apart grid deployment and 20% node failures, tracking errors are still as small as about 6 meters and the maximum error in velocity calculation doesn't exceed 10%. These results from physically deployed systems validate that Lightweight EnviroSuite is practical, effective, and efficient on current hardware with limited communication and sensing capabilities. In this paper, we did not supply an experimental comparison between EnviroSuite and other high-level sensor network programming systems. This is, in part, due to the difficulty in porting application code across the different systems. If applications are re-implemented (as opposed to ported), it is hard to separate the effects of application-level implementation decisions from the inherent strengths and weaknesses of the underlying programming frameworks when interpretting performance comparison results.

5 Summary

Research on programming paradigms and frameworks in sensor networks has been very active in recent years. This paper presents our effort in improving an existing programming paradigm (EnviroSuite) into a more practical and efficient version (Lightweight EnviroSuite) that can be directly utilized to build practical large-scale systems with realistic requirements. The resulting version is shown to be efficient via experimental testing on a physically deployed large-scale surveillance system consisting of 200 XSM motes. This is one of the first attempts to use high-level sensor network programming languages in building and deploying real sensor network applications.

References

1. Boulis, A., Srivastava, M.B.: A framework for efficient and programmable sensor networks. In: OPENARCH '02. (2002)
2. Levis, P., Culler, D.: Mate: a virtual machine for tiny networked sensors. In: ASPLOS. (2002)
3. Li, S., Son, S.H., , Stankovic, J.: Event detection services using data service middleware in distributed sensor networks. In: IPSN '03. (2003)

4. Madden, S.R., Franklin, M.J., Hellerstein, J.M., , Hong, W.: The design of an acquisitional query processor for sensor networks. In: SIGMOD. (2003)
5. Yao, Y., Gehrke, J.E.: The cougar approach to in-network query processing in sensor networks. In: Sigmod Record, Volume 31, Number 3. (2002)
6. Welsh, M., Mainland, G.: Programming sensor networks using abstract regions. In: NSDI '04. (2004)
7. Newton, R., Welsh, M.: Region streams: functional macroprogramming for sensor networks. In: DMSN '04: Proceeedings of the 1st international workshop on Data management for sensor networks, New York, NY, USA, ACM Press (2004) 78–87
8. Liu, J., Liu, J., Reich, J., Cheung, P., , Zhao, F.: Distributed group management for track initiaition and maintenance in target localization applications (2004)
9. Whitehouse, K., Sharp, C., Brewer, E., Culler, D.: Hood: A neighborhood abstraction for sensor networks. In: MobiSYS '04. (2004)
10. Liu, J., Chu, M., Liu, J., Reich, J., Zhao, F.: State-centric programming for sensor-actuator network systems. In: IEEE Pervasive Computing. (2003)
11. XSM motes: (http://www.cast.cse.ohio-state.edu/exscal/index.php?page=main)
12. Stockman, H.: The active badge location system. In: Proceedings of the IRE. (1948) 1196–1204
13. Want, R., Hopper, A., Falcao, V., Gibbons, J.: The active badge location system (1992)
14. Liu, J., Chu, M., Liu, J., Reich, J., Zhao, F.: Distributed state representation for tracking problems in sensor networks. In: IPSN '04. (2004)
15. Aslam, J., Butler, Z., Constantin, F., Crespi, V., Cybenko, G., Rus, D.: Tracking a moving object with a binary sensor network. In: SenSys '03, New York, NY, USA, ACM Press (2003) 150–161
16. Shin, J., Guibas, L., Zhao, F.: A distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks. In: IPSN '03. (2003)
17. Zhang, W., Cao, G.: Optimizing tree reconfiguration for mobile target tracking in sensor networks. In: INFOCOM '04. (2004)
18. Blum, B., Nagaraddi, P., Wood, A., Abdelzaher, T., Son, S., Stankovic, J.: An entity maintenance and connection service for sensor networks. In: MobiSys '03. (2003)
19. Abdelzaher, T., Blum, B., Cao, Q., Evans, D., George, J., George, S., He, T., Luo, L., Son, S., Stoleru, R., Stankovic, J., Wood, A.: Envirotrack: Towards an environmental computing paradigm for distributed sensor networks. In: ICDCS '04. (2004)
20. Luo, L., Abdelzaher, T., He, T., Stankovic, J.A.: Envirosuite: An environmentally immersive programming framework for sensor networks. (In: Submitted to ACM Transactions on Embedded Computing Systems)
21. Karp, B.: Geographic Routing for Wireless Networks. PhD thesis, Harvard University (2000)
22. Levis, P., Lee, N., Welsh, M., , Culler, D.: Tossim: Accurate and scalable simulation of entire tinyos applications. In: SenSys '03. (2003)
23. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System architecture directions for networked sensors. In: ASPLOS-IX, New York, NY, USA, ACM Press (2000) 93–104
24. He, T., Krishnamurthy, S., Stankovic, J.A., Abdelzaher, T., Luo, L., Stoleru, R., Yan, T., Gu, L., Hui, J., Krogh, B.: Energy-efficient surveillance system using wireless sensor networks. In: MobiSYS '04, New York, NY, USA, ACM Press (2004) 270–283
25. UC Berkeley. MICA motes: (http://www.tinyos.net/scoop/special/hardware/)

ATPC: Adaptive Transmission Power Control for Wireless Sensor Networks

Shan Lin, Jingbin Zhang, Gang Zhou, Lin Gu, Tian He[†], and John A. Stankovic

Department of Computer Science, University of Virginia

[†]Department of Computer Science and Engineering, University of Minnesota

{shanlin,jz7q,gzhou,lingu}@cs.virginia.edu, tianhe@cs.umn.edu, stankovic@cs.virginia.edu

Abstract

Extensive empirical studies presented in this paper confirm that the quality of radio communication between low power sensor devices varies significantly with time and environment. This phenomenon indicates that the previous topology control solutions, which use static transmission power, transmission range, and link quality, might not be effective in the physical world. To address this issue, online transmission power control that adapts to external changes is necessary. This paper presents ATPC, a lightweight algorithm of Adaptive Transmission Power Control for wireless sensor networks. In ATPC, each node builds a model for each of its neighbors, describing the correlation between transmission power and link quality. With this model, we employ a feedback-based transmission power control algorithm to dynamically maintain individual link quality over time. The intellectual contribution of this work lies in a novel pairwise transmission power control, which is significantly different from existing node-level or network-level power control methods. Also different from most existing simulation work, the ATPC design is guided by extensive field experiments of link quality dynamics at various locations and over a long period of time. The results from the real-world experiments demonstrate that 1) with pairwise adjustment, ATPC achieves more energy savings with a finer tuning capability and 2) with online control, ATPC is robust even with environmental changes over time.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*wireless communication*

General Terms

Algorithms, Design, Experimentation, Measurement, Performance

Keywords

Adaptive, Feedback, Link Quality, Wireless Sensor Network, Transmission Power Control

1 Introduction

With the integration of sensing and communication abilities in tiny devices, wireless sensor networks are widely deployed in a variety of environments, supporting military surveillance [1] [24], emergency response [41], and scientific exploration [36]. The in-situ impact from these environments, together with energy constraints of the nodes, makes reliable and efficient wireless communication a challenging task. Under a constrained energy supply, reliability and efficiency are often at odds with each other. Reliability can be improved t by transmitting packets at the maximum transmission power [13] [38], but this situation introduces unnecessarily high energy consumption. To provide system designers with the ability to dynamically control the transmission power, popularly used radio hardware such as CC1000 [6] and CC2420 [7] offers a register to specify the transmission power level during runtime. It is desirable to specify the minimum transmission power level that achieves the required communication reliability for the sake of saving power and increasing the system lifetime.

Although theoretical study and simulation provide a valuable and solid foundation, solutions found by such efforts may not be effective in real running systems. Simplified assumptions can be found in these studies, for example, static transmission power, static transmission range, and static link quality. These studies do not consider the spatial-temporal impact on wireless communication. In this paper, we present systematic studies on these impacts. There are a number of empirical studies on communication reality conducted with real sensor devices [43] [40] [44] [4] [29] [20]. Their results suggest that for a specified transmission power and communication distance, the received signal power varies and the link quality is unstable. But they do not focus on a systematic study on the radio and link dynamics in the context of different transmission power settings. Our extensive experiments with MICAz [8] confirm the observations presented in previous work. We also go further and explore the radio and link dynamics when different transmission power levels are applied. Our experimental results identify that link quality changes differently according to spatial-temporal factors in a real wireless sensor network. To address this issue, we design a pairwise transmission power control. Our empirical study also reveals that it is feasible to choose a minimal and environment-adapting transmission power level to save power, while guaranteeing specified link quality at the same time.

To achieve the optimal transmission power consumption for specified link qualities, we propose ATPC, an adaptive transmission power control algorithm for wireless sensor networks. The result of applying ATPC is that every node knows the proper transmission power level to use for each of its neighbors, and every node maintains good link qualities with its neighbors by dynamically adjusting the transmission power through on-demand feedback packets. Uniquely, ATPC adopts a feedback-based and pairwise transmission power control. By collecting the link quality history, ATPC builds a model for each neighbor of the node. This model represents an in-situ correlation between transmission power levels and link qualities. With such a model, ATPC tunes the transmission power according to monitored link quality changes. The changes of transmission power level reflect changes in the surrounding environment. ATPC supports packet-level transmission power control at runtime for MAC and upper layer protocols. For example, routing protocols with transmission power as a metric [33] [35] [12] [9] [5] can make use of ATPC by choosing the route with optimal power consumption to forward packets.

The topic of transmission power control is not new, but our approach is quite unique. In state-of-art research, many transmission power control solutions use a single transmission power for the whole network, not making full use of the configurable transmission power provided by radio hardware to reduce energy consumption. We refer to this group as network-level solutions, and typical examples in this group are [27] [25] [2] [18] [31]. Also, some other work takes the configurable transmission powers into consideration. They either assume that each node chooses a single transmission power for all the neighbors [2] [18] [19] [28] [37] [17] [26] [30] [22], which we refer to as node-level solutions, or nodes use different transmission powers for different neighbors [23] [42] [3], which we call neighbor-level solutions. While these solutions provide a solid foundation for our research, ATPC goes further to support packet-level transmission power control in a pairwise manner.

Also, most existing real wireless sensor network systems use a network-level transmission power for each node, such as in [13] [38]. These coarse-level power controls lead to high energy consumption. The authors of [34] present a valuable study about the impact of variable transmission power on link quality. Through our empirical experiments with the MICAz platform, it is observed that different transmission powers are needed to achieve the same link quality over time. This leads to our feedback-based transmission power control design, which is not addressed in [34]. Also, the authors of [34] use a fixed number of transmission powers (13 levels), which fixes the maximum accuracy for power tuning. The ATPC we propose chooses different transmission power levels based on the dynamics of link quality, and it also allows for better tuning accuracy and more energy savings. Our approach essentially represents a good tradeoff between accuracy and cost, a finer control at each node in exchange for less energy consumption when transmitting the packets.

In this work, we invest a fair amount of effort to obtain empirical results from three different sites and over a reasonably long time period. These results give practical guidance to the overarching design of ATPC. We demonstrate that ATPC greatly extends the system lifetime by choosing a proper transmission power for each packet transmission, without jeopardizing the quality of data delivery. In our 3-day experiment with 43 MICAz motes, ATPC achieves above a 98% end-to-end Packet Reception Ratio in natural environment through fair and rainy days. The solutions without online tuning can barely deliver half of packets. Compared to other solutions, ATPC also significantly saves transmission power. With equivalent communication performance, ATPC only consumes 53.6% of the transmission energy of the maximum transmission power solution and 78.8% of the transmission energy of the network-level transmission power solution. More specifically, the contributions of our work lie in two aspects.

- Our systematic study and experiments reveal the spatiotemporal impacts on wireless communication and identify the relationship between dynamics of link quality and transmission power control.
- With run-time pairwise transmission power control, we achieve high packet delivery ratio successfully with small energy consumption under realistic scenarios.

The rest of this paper is organized as follows: the motivation of this work is presented in Section 2. In Section 3, the design of ATPC is stated. In Section 4, ATPC is evaluated in real world experiments. The state of the art is analyzed in Section 5. In Section 6, conclusions are given and future work is pointed out.

2 Motivation

Radio communication quality between low power sensor devices is affected by spatial and temporal factors. The spatial factors include the surrounding environment, such as terrain and the distance between the transmitter and the receiver. Temporal factors include surrounding environmental changes in general, such as weather conditions. In this section, we present experimental results for investigation of these impacts. We note that previous empirical studies on communication reality [43] [4] [44] [10] [29] [20] suggest that for a specified transmission power, fixed communication distance, and antenna direction, the received signal power and the link quality vary. But they do not focus on a systematic study of the radio and link dynamics when different transmission powers are considered. We conducted these measurements, and we are the first to study systematically the spatial and temporal impacts on the correlation between transmission power and Received Signal Strength Indicator (RSSI)/ Link Quality Indicator (LQI) [15]. Both RSSI and LQI are useful link metrics provided by CC2420 [7]. RSSI is a measurement of signal power which is averaged over 8 symbol periods of each incoming packet. LQI is a measurement of the "chip error rate" [7] which is also implemented based on samples of the error rate for the first eight symbols

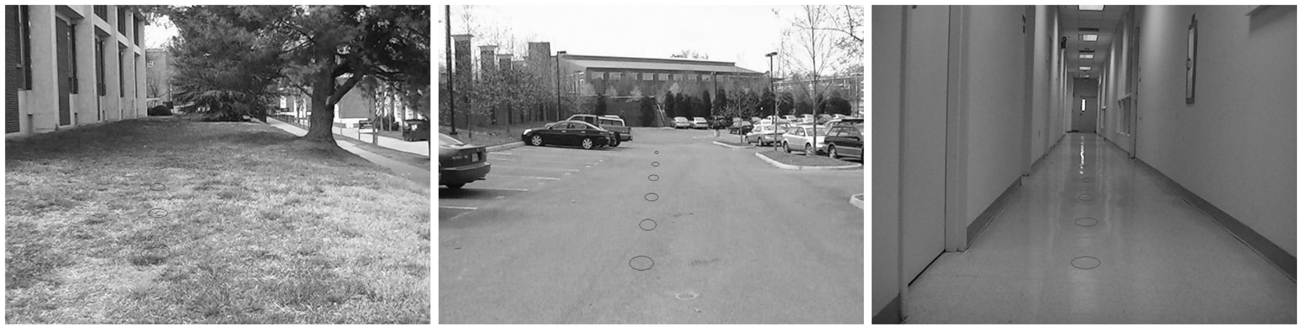

(a) Experiments on a Grass Field (b) Experiments in a Parking Lot (c) Experiments in a Corridor

Fig. 1. Experimental Sites

of each incoming packet. Transmission power level index refers to the value specified for the RF output power provided by CC2420 [7]. It can be mapped to output power in units of dBm.

Our empirical results show that link quality is significantly influenced by spatiotemporal factors, and that every link is influenced to a different degree in a real system. This observation proves that the assumptions made from previous work about the static impact of the environment on link quality do not hold. Solutions based on these simplifying assumptions may not accurately capture the dynamics of communication quality, and may result in highly unstable communication performance in real wireless sensor networks. Therefore, the in-situ transmission power control is essential for maintaining good link quality in reality.

2.1 Investigation of Spatial Impact

To investigate the spatial impact, we study the correlation between transmission power and link qualities in three different environments: a parking lot, a grass field, and a corridor, as shown in Figure 1. We use one MICAz as the transmitter and a second MICAz as the receiver. They are put on the ground at different locations, maintaining the same antenna direction. The transmitter sends out 100 packets (20 packets per second) at each transmission power level. The receiver records the average RSSI, the average LQI, and the number of packets received at each transmission power level. The experiments are repeated with 5 different pairs of motes in the same environmental conditions to obtain statistical confidence.

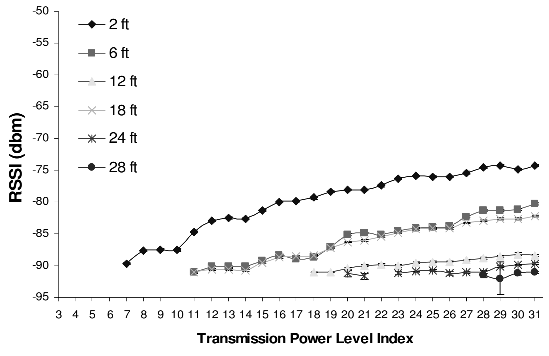
Figure 2 shows our experimental data obtained from one pair of nodes in different environments. Each curve demonstrates the correlation between the transmission power and RSSI/LQI at a certain distance of that pair. The confidence intervals (97%) of RSSI/LQI are also plotted on Figure 2. Clearly, there is a strong correlation between transmission power level and RSSI/LQI. We note that there is an approximately linear correlation between transmission power and RSSI in Figures 2 (a) (c) (e). The LQI curves in Figures 2 (b) (d) (f) also present approximately linear correlations when the LQI readings are small. However, the LQI readings suffer saturation when they get close to 110, which is the maximum quality frame detectable by the CC2420 [7]. We also notice that each LQI curve and its corresponding RSSI curve demonstrate similar trends and variations. This is because the LQI reading is also a representation of the SNR value, which is the ratio of the received signal power level to the background noise level.

The slopes of RSSI curves generally decrease as the distance increases, but this is not always true. According to [32], RSSI is inversely proportional to the square of the distance. To obtain the same amount of RSSI increase, a larger transmission power increase is needed at a longer distance. However, in reality, this rule doesn't always hold. For example, in Figures 2 (a) and (c), the slopes of RSSI curves at a distance of 18 feet are bigger than those at a distance of 12 feet, which is caused by multi-path reflection and scattering [43]. Therefore, this measured correlation is a better reflection of the communication reality.

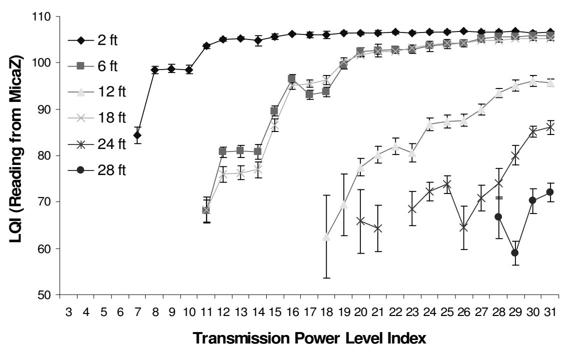
The shapes of RSSI/LQI curves based on the results from a grass field (Figures 2 (a) and (b)), a parking lot (Figures 2 (c) and (d)) and a corridor (Figures 2 (e) and (f)) are significantly different from one another, even with the same distance and antenna direction between a pair of nodes. For example, with a transmission power level of 20 and a distance of 12 feet, the RSSI is -90 dBm on a grass field (Figure 2 (a)), while above -70 dBm in a corridor (Figure 2 (e)). Even though the curves for 12 feet on a grass field and on a parking lot are similar (Figures 2 (a) and (c)), the 6 feet curves in these two environments are not quite the same (Figures 2 (a) and (c)). These experimental results confirm that radio propagation among low power sensor devices can be influenced largely by environment [43] [44] [10]. Moreover, RSSI/LQI with specified transmission power and distance varies in a very small range and the degree of variations is related to the environment. According to the confidence intervals (97%) shown on Figure 2, RSSI readings are more stable than LQI. The confidence intervals of RSSI are not observable at most of the sampling points in Figures 2 (a) (c) and (e).

2.2 Investigation of Temporal Impact

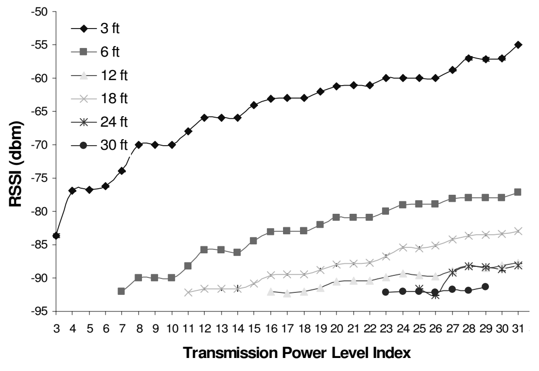
We also investigate the impact of time on the correlation between transmission power and link quality. Empirical results in this section suggest that this correlation changes



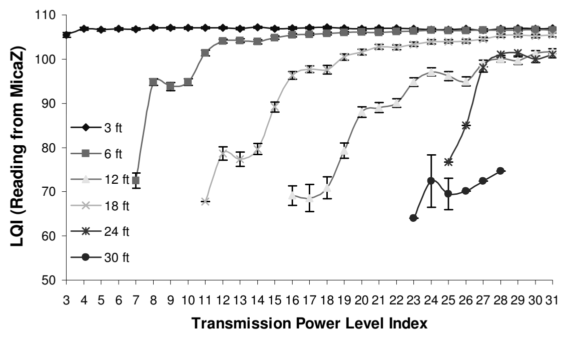
(a) RSSI Measured on a Grass Field



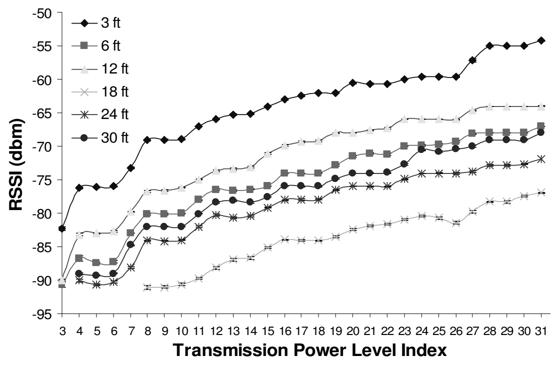
(b) LQI Measured on a Grass Field



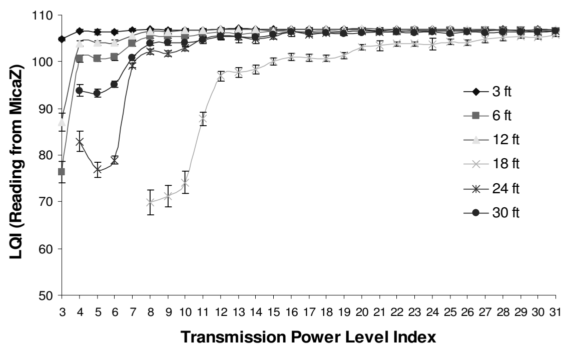
(c) RSSI Measured in a Parking Lot



(d) LQI Measured in a Parking Lot



(e) RSSI Measured in a Corridor



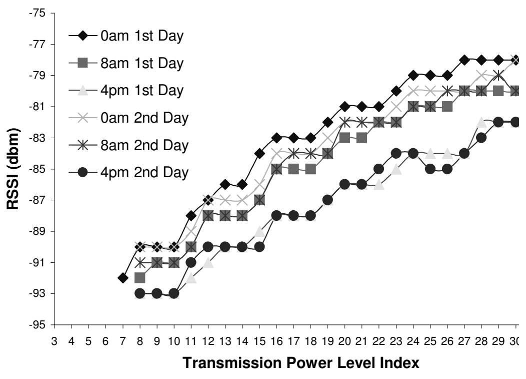
(f) LQI Measured in a Corridor

Fig. 2. Transmission Power *vs.* RSSI/LQI at Different Distances in Different Environments

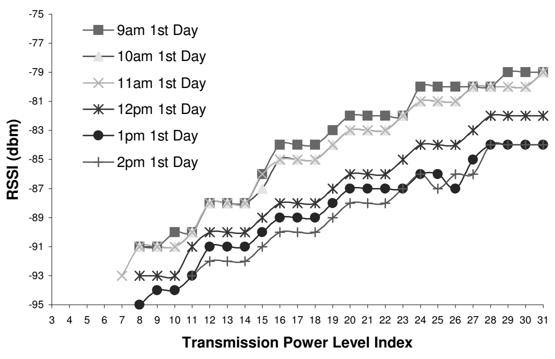
slowly but noticeably over a long period of time. Therefore, online transmission power control is requisite to maintain the quality of communication over time.

A 72-hour outdoor experiment is conducted to demonstrate the variations of the radio communication quality over time. We place 9 MICAz motes in a line with a 3-feet spacing. These motes are wrapped in tupperware containers to protect against the weather. The tupperware containers are placed in brushwood. They are about 0.5 feet high above the ground because the brushwood is very dense. During the experiment, each mote sends out a group of 20 packets at each transmission power level every hour. The transmission rate is 10 packets per second. All the other motes receive and record the average RSSI and the number of packets they received at each transmission power level. The transmissions of different motes are scheduled at different times to avoid collision.

In this experiment, data obtained from different pairs ex-



(a) Transmission Power *vs*. RSSI every 8-hour



(b) Transmission Power *vs*. RSSI every Hour

Fig. 3. Transmission Power *vs*. RSSI at Different Times

hibit similar trends. Figure 3 presents our empirical data obtained from a pair of motes at a distance of 9 feet apart. Each curve represents the correlation between transmission power and RSSI at a specific time. The correlation between transmission power and RSSI every 8-hour is plotted in Figure 3 (a). The shapes of these curves are different due to environmental dynamics. As a result, different transmission power levels are needed to reach the same link quality at different times. For example, to maintain RSSI value at -89 dBm, the transmission power level needs to be 11 at 0 AM on the first day, while at 4 PM on the second day the transmission power level needs to be 20. Figure 3 (b) shows the hourly changes of the correlation. From Figure 3 (b), we can see that the relation between transmission power and RSSI changes more gradually and continuously than that in Figure 3 (a). For example, the maximum change in RSSI is 8 dBm over an 8-hour period in Figure 3 (a), while it is 3 dBm over a one-hour period in Figure 3 (b).

These curves are approximately parallel, and the relationship between transmission power and RSSI varies differently at different times of day. For example, in Figure 3 (a) the curve at 4 PM on the first day is much lower than the curve at 8 AM on the first day. The same variation happens on curves at 8 AM and 4 PM on the second day, but the degree of variation is different. All these results indicate that it is critical for transmission power control algorithms proposed for sensor networks to address the temporal dynamics of communication quality.

2.3 Dynamics of Transmission Power Control

To establish an effective transmission power control mechanism, we need to understand the dynamics between link qualities and RSSI/LQI values. In this section, we present empirical results that demonstrate the relation between the link quality and RSSI/LQI. The key observations, which serve as the basis of our work, are as follows:

- Both RSSI and LQI can be effectively used as binary link quality metrics for transmission power control.
- The link quality between a pair of motes is a detectable function of transmission power.

2.3.1 Link Quality Threshold

Wireless link quality refers to the radio channel communication performance between a pair of nodes. PRR (packet reception ratio) is the most direct metric for link quality. However, the PRR value can only be obtained statistically over a long period of time. Our experiments indicate that both RSSI and LQI can be used effectively as binary link quality metrics for transmission power control[1]. We record the PRR and the average RSSI/LQI for every group of 100 packets from a grass field (Figures 4 (a) and (d)), a parking lot (Figures 4 (b) and (e)) and a corridor (Figures 4 (c) and (f)). All experimental results show that both RSSI and LQI have a strong relationship with PRR. There is a clear threshold to achieve a nearly perfect PRR. However, these thresholds are slightly different in different environments. Take RSSI as an example: the 95% PRR threshold of RSSI is around -90 dBm on the grass field (Figure 4 (a)), -91 dBm on the parking lot (Figure 4 (b)), and -89 dBm in the corridor (Figure 4 (c)).

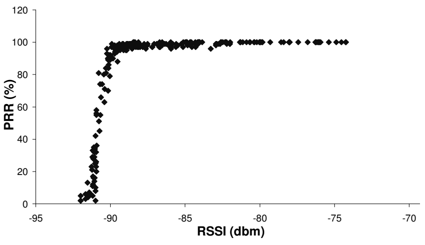
2.3.2 Relations between Transmission Power and RSSI/LQI

Radio irregularity results in radio signal strength variation in different directions, but the signal strength at any point within the radio transmission range has a detectable correlation with transmission power in a short time period.

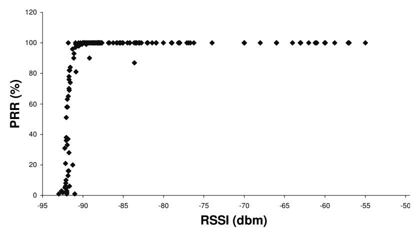
In short term experiments, the correlation between transmission power and RSSI/LQI for a pair of motes at a certain distance is generally monotonic and continuous. From Figure 2, the overall trend of RSSI increases linearly when the transmission power increases.

However, RSSI/LQI fluctuates in a small range at any fixed transmission power level. So, the correlation between transmission power and RSSI/LQI is not deterministic. For example, Figure 5 shows the RSSI upper bound and lower bound of 100 received packets at each transmission power level when we place two motes 6-feet apart on a grass field. This result confirms the observation from previous studies [43] [44] [10].

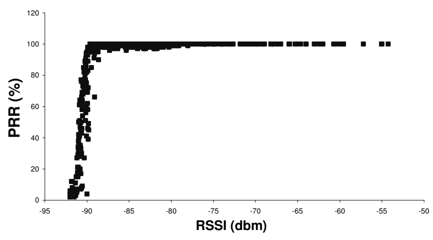
[1]It is still controversial whether RSSI or LQI is a better indicator on link quality [43] [29] [20].



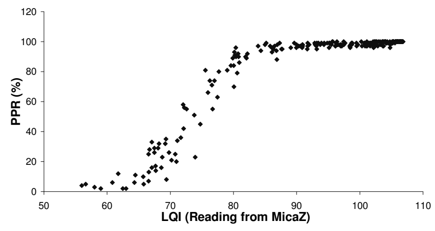
(a) RSSI *vs*. PRR on Grass Field



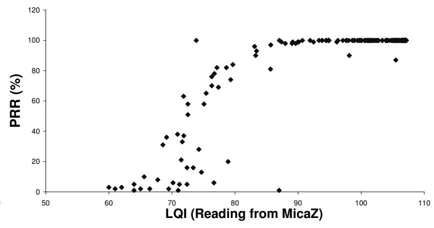
(b) RSSI *vs*. PRR on Parking Lot



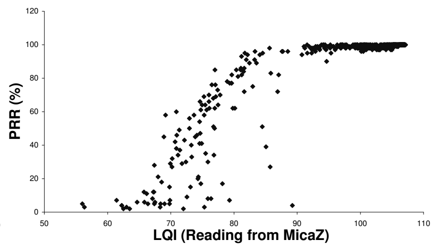
(c) RSSI *vs*. PRR in Corridor



(d) LQI *vs*. PRR on Grass Field



(e) LQI *vs*. PRR on Parking Lot



(f) LQI *vs*. PRR in Corridor

Fig. 4. RSSI *vs*. PRR in Different Environments

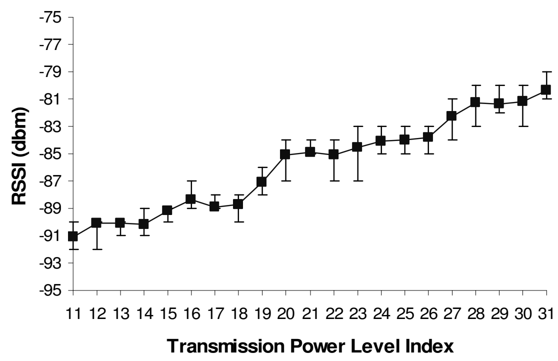


Fig. 5. Transmission Power *vs*. RSSI

There are three main reasons for the fluctuation in the RSSI and LQI curves. First, fading [32] causes signal strength variation at any specific distance. Second, the background noise impairs the channel quality seriously when the radio signal is not significantly stronger than the noise signal. Third, the radio hardware doesn't provide strictly stable functionality [7].

Since the variation is small, this relation can be approximated by a linear curve. The correlation between RSSI and transmission power is approximately linear, and the correlation between LQI and transmission power is also approximately linear in a range. From the confidence intervals in Figure 2, we can see that RSSI and LQI are both relatively stable when these values are not small. All the points with confidence intervals bigger than 1 correspond to low link quality points in Figure 4, and the RSSI/LQI values which have the most fluctuations are below the good link quality thresholds. Since we are only interested in RSSI/LQI samplings that are above or equal to the good link quality threshold, it is feasible to use a linear curve to approximate this correlation. This linear curve is built based on samples of RSSI/LQI. This curve roughly represents the in-situ correlation between RSSI/LQI and transmission power.

This in-situ correlation between transmission power and RSSI/LQI is largely influenced by environments, and this correlation changes over time. Both the shape and the degree of variation depend on the environment. This correlation also dynamically fluctuates when the surrounding environmental conditions change. The fluctuation is continuous, and the changing speed depends on many factors, among which the degree of environmental variation is one of the main factors.

3 Design of ATPC

Guided by the observations obtained from empirical experiments, in this section, we propose our Adaptive Transmission Power Control (ATPC) design. The objectives of ATPC are: 1) to make every node in a sensor network find the minimum transmission power levels that can provide good link qualities for its neighboring nodes, to address the spatial impact, and 2) to dynamically change the pairwise transmission power level over time, to address the temporal impact. Through ATPC, we can maintain good link qualities between pairs of nodes with the in-situ transmission power control.

Figure 6 shows the main idea of ATPC: a neighbor table is maintained at each node and a feedback closed loop for transmission power control runs between each pair of nodes. The neighbor table contains the proper transmission power levels that this node should use for its neighboring nodes and the parameters for the linear predictive models of transmission power control. The proper transmission power level is

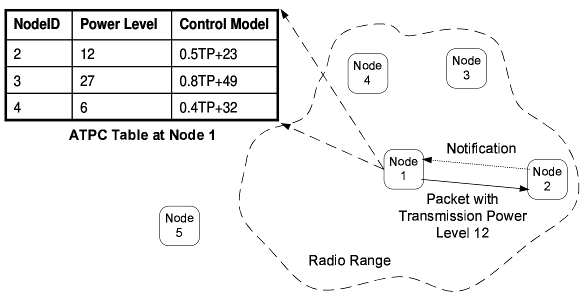


Fig. 6. Overview of the Pairwise ATPC Design

defined here as the minimum transmission power level that supports a good link quality between a pair of nodes. The linear transmission power predictive model is used to describe the in-situ relation between the transmission powers and link qualities. Our empirical data indicate that this in-situ relation is not strictly linear. Therefore, this predictive model is an approximation of the reality. To obtain the minimum transmission power level, we apply feedback control theory to form a closed loop to gradually adjust the transmission power. It is known that feedback control allows a linear model to converge within the region when a non-linear system can be approximated by a linear model, so we can safely design a small-signal linear control for our system, even if our linear model is just a rough approximation of reality.

3.1 Predictive Model for ATPC

The design objective is to establish models that reflect the correlation of the transmission power and the link quality between the senders and the receivers. Based on our empirical study and analysis in Section 2, we formulate a predictive model to characterize the relation between transmission power and link quality. Since no single model can capture precisely the per-network, or even per-node behavior, we shall establish pairwise models, reflecting the in-situ impact on individual links. Based on these models, we can predict the proper transmission power level that leads to the link quality threshold.

The idea of this predictive model is to use a function to approximate the distribution of RSSIs at different transmission power levels, and to adapt to environmental changes by modifying the function over time. This function is constructed from sample pairs of the transmission power levels and RSSIs via a curve-fitting approach. To obtain these samples, every node broadcasts a group of beacons at different transmission power levels, and its neighbors record the RSSI of each beacon that they can hear and return those values.

We formulate this predictive model in the following way. Technically, this model uses a vector TP and a matrix R. $TP = \{tp_1, tp_2, ..., tp_N\}$. TP is the vector containing different transmission power levels that this mote uses to send out beacons. $|TP| = N$. N, the number of different transmission power levels, is subject to the accuracy requirement for applications. Ideally the more sampling data we have, the more accurate this model could be. Matrix R consists of a set of RSSI vectors R_i, one for each neighbor ($R = \{R_1, R_2, ..., R_n\}^T$). $R_i = \left\{r_i^1, r_i^2, ..., r_i^N\right\}$ is the RSSI vector for the neighbor i, in which r_i^j is a RSSI value measured at node i corresponding to the beacon sent by transmission power level tp_j. We use a linear function (Equation 1) to characterize the relationship between transmission power and RSSI on a pairwise basis.

$$r_i(tp_j) = a_i \cdot tp_j + b_i \tag{1}$$

We adopt a least square approximation, which requires little computation overhead and can be easily applied in sensor devices. Based on the vectors of samples, the coefficients a_i and b_i of Equation 1 are determined through this least square approximation method by minimizing S^2.

$$\sum \left(r_i(tp_j) - r_i^j\right)^2 = S^2 \tag{2}$$

Accordingly, the value of a_i and b_i can be obtained in Equation 3:

$$\begin{bmatrix} a_i \\ b_i \end{bmatrix} = \frac{1}{N\sum_{j=1}^{N}(tp_j)^2 - (\sum_{j=1}^{N} tp_j)^2} \times \begin{bmatrix} \sum_{j=1}^{N} r_i^j \sum_{j=1}^{N}(tp_j)^2 - \sum_{j=1}^{N} tp_j \sum_{j=1}^{N} tp_j \cdot r_i^j \\ N\sum_{j=1}^{N} tp_j \cdot r_i^j - \sum_{j=1}^{N} tp_j \sum_{j=1}^{N} r_i^j \end{bmatrix}, \tag{3}$$

where i is the neighboring node's ID and j is the number of transmissions attempted. Using a_i and b_i together with a link quality threshold $RSSI_{LQ}$ identified based on experiments in Section 2.3, we can calculate the desired transmission power $tp_j = \frac{RSSI_{LQ} - b_i}{a_i}$.

Note that Equation 3 only establishes an initial model. We need to update this model continuously while the environment changes over time in a running system. Basically, the values of a_i and b_i are functions of time. These functions allow us to use the latest samples to adjust our curve model dynamically. Based on our experimental results in Section 2, a_i, the slope of a curve, changes slightly in our 3-day experiment, while b_i changes noticeably over time. Therefore, once the predictive model of ATPC is built, a_i does not change any longer. $b_i(t)$ is calculated by the latest transmission power and RSSI pairs from the following feedback-based equation.

$$b_i(t) = \frac{\sum_{t=1}^{K}\left[RSSI_{LQ} - r_i(t-1)\right]}{K} \tag{4}$$

Here $r_i(t-1)$ is the RSSI value of the neighboring node i during time period $t-1$. K is the number of feedback responses received from this neighboring node at time period $t-1$. Although the link quality varies significantly over a long period of time, it changes gradually and continuously at a slow rate. Our experiments indicate that one packet per hour between a pair is enough to maintain the freshness of the model in a natural environment. If the network has a reasonable amount of traffic, such as several packets per hour, nodes can use these packets to measure link quality change and piggyback RSSI readings. In this way, these models are refreshed with little overhead.

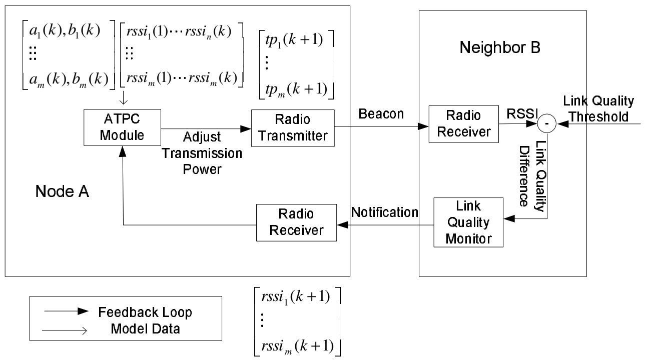


Fig. 7. Feedback Closed Loop Overview for ATPC

3.2 Implementation of ATPC

The implementation of ATPC on sensor devices is presented in this subsection. We discuss mainly four aspects: 1) the two phase design and the feedback closed loop for pairwise transmission power control, 2) the parameters that affect system performance, 3) the techniques that optimize system performance and reduce the cost, and 4) the other issues.

ATPC has two phases, the initialization phase and the runtime tuning phase.

In the initialization phase, a mote computes a predictive model and chooses a proper transmission power level based on that model for each neighbor. Since wireless communication is broadcast in nature, all the neighbors can receive beacons and measure link qualities in parallel. Based on this property, every node broadcasts beacons with different transmission power levels in the initialization phase, and its neighbors measure RSSI/LQI values corresponding to these beacons and send these values back by a notification packet.

In the runtime tuning phase, a lightweight feedback mechanism is adopted to monitor the link quality change and tune the transmission power online. Figure 7 is an overview picture of the feedback mechanism in ATPC. To simplify the description, we show a pair of nodes. Each node has an ATPC module for transmission power control. This module adopts a predictive model described in the previous subsection for each neighbor. It also maintains a list of proper transmission power levels for neighbors of this mote. When node A has a packet to send to its neighbor B, it first adjusts the transmission power to the level indicated by its neighbor table in the ATPC module, and then transmits the packet. When receiving this packet, the link quality monitor module at its neighbor B takes a measurement of the link quality. Based on the difference between the desired link quality and actual measurements, the link quality monitor module decides whether a notification packet is necessary. A notification packet is necessary when 1) the link quality falls below the desired level or 2) the link quality is good but the current signal energy is so high that it wastes the transmission energy. The notification packet contains the measured link quality difference. When node A receives a notification from its neighbor B, the ATPC module in node A uses the link quality difference as the input to the predictive model and calculates a new transmission power level for its neighbor B.

If achieving good link quality requires using the maximum transmission power level, ATPC adjusts the transmission power to the maximum level. If using the maximum transmission power level could not achieve good link quality, this link is marked so that routing protocols, like [33] [35] [12] [9] [5], can choose another route based on the neighbor table provided by ATPC. If all the routes cannot provide good link quality, the mote can do best-effort transmission to a neighbor with relative good link quality by using the maximum transmission power level.

There is a tradeoff between accuracy and cost when applying ATPC. The practical values of these parameters are obtained from analysis and empirical results. These important parameters include the link quality thresholds, the sampling rate of transmission power control, the number of sample packets in the initialization phase, and the small-signal adjustment of transmission power control, which is proportional to the link quality error. Choices of parameters are essential for obtaining good performance.

The link quality monitor can have any of the following three criteria to estimate link quality changes. The first one is the link quality reflected by the RSSI value, the second one is the LQI value if available, and the last one is the packet reception ratio as detected by sequence number monitoring. Our design is compatible with all these methods. Without loss of generality, we use both RSSI and PRR in our experiments. We note that the theory described in section 3.1 is good guidance in ideal conditions.

To monitor the link quality by referring to RSSI values, we set two link quality thresholds. LQ_{upper} is an upper threshold and LQ_{lower} is a lower threshold. As long as the RSSI value of the received packet lies within this range, the system is in steady state. When a link is in steady state, the receiver does not need to send a notification packet to the sender, and the sender does not adjust the transmission power. The range of [LQ_{lower}, LQ_{upper}] is critical to energy savings and tuning accuracy. If the range of [LQ_{lower}, LQ_{upper}] is too small, radio signal fading may result in the oscillation of transmission power. If the range of [LQ_{lower}, LQ_{upper}] is too big, the transmission power control result may not be accurate enough, and the optimal power control will not be achieved. In our implementation, the value of LQ_{lower} is chosen to guarantee that the link quality does not drop below the tolerance level. With respect to LQ_{upper} in our design, its value is chosen to trade off the energy cost paid to transmit notifications and the energy saved to transmit data packets. This is a simple calculation for choosing LQ_{upper} which compares the energy consumed by sending a control packet with the energy saved for n data packets after tuning the transmission power. In our experiment, we use n = 2 for simplicity. Thus, energy savings are achieved when at least two data packets are transmitted using the tuned transmission power level, compared to the energy consumed by transmitting a notification packet.

A good feedback sampling rate is essential to maintain the link quality at a desired level while minimizing the control overhead. Two main factors influence the feedback sampling rate: link quality dynamics and network traffic. On one hand, the higher the link quality dynamics, the higher the sampling rate needed. Based on our empirical results in Figure 3, the maximum link quality variation per 8-hour is 8 dBm and the maximum link quality variation per hour is 3 dBm. In order to keep link quality error under 3 dBm, a sampling rate of 1 packet per hour is necessary. On the other hand, the regular network traffic can be used for ATPC sampling purposes and considered as ATPC's input. When the network traffic is higher than this sampling rate, notification packets can be sent on demand. There is only a low number of notification packets needed and the control overhead is minimized. Our running system evaluation demonstrates that this design is very efficient. On average, 8 on-demand notification packets are sent per link per day to deal with the runtime link quality dynamics.

In applications with periodic multi-hop traffic, an overhearing approach can save the overhead of notification packets. Along the data transfer route, when a node is forwarding packets to its next hop, it can incorporate an extra byte to record the RSSI value of the previous hop transmission in the packet, and then the sender of the previous hop can overhear the corresponding RSSI, thus eliminating explicit notifications.

Another optimization technique is to use ATPC only on critical paths with heavy traffic, so ATPC can extend the system lifetime while supporting a high quality end-to-end communication with little control overhead. For those links with a low traffic load, directly using a conservative transmission power level is a good tradeoff between communication quality and energy savings. This is because nodes do not need to periodically generate control packets to monitor link quality.

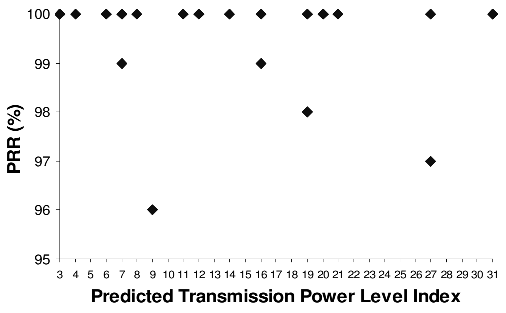
Based on our empirical results, the RSSI readings can be affected by stochastic environmental noise. For example, the RSSI with a certain beacon packet can be unexpectedly high or low, which is inconsistent with the monotonic relationship between transmission power and RSSI. Filtering such noise input can enhance the accuracy of ATPC's modeling. On the other hand, if some RSSI with a certain transmission power level falls in our desired link quality range, using the corresponding transmission power level directly also enhances ATPC's performance.

The code for ATPC mainly includes functions for linear approximation. The code size is 14122 bytes in ROM. The data structures in ATPC mainly include a neighbor table, a vector TP and a matrix R as described in Section 3.1. For a node with 20 neighbors, the data size is 2167 bytes in RAM.

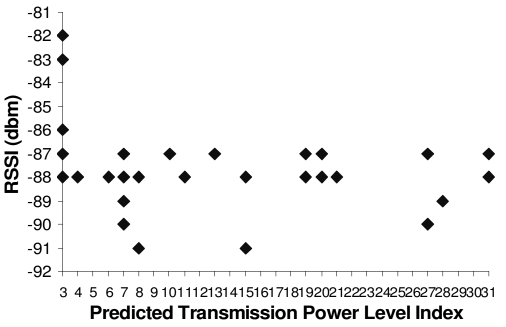
4 Experimental Evaluations

ATPC is evaluated in outdoor environments. We first evaluate ATPC's predictive model described in Section 3.1 with a short term experiment. We then describe a 72-hour experiment to compare ATPC against network-level uniform transmission power solutions and a node-level non-uniform transmission power solution. According to our empirical results, ATPC's advantages lie in three core aspects:

1. ATPC maintains high communication quality over time in changing weather conditions. It has significantly better link qualities than using static transmission power in a long term experiment, which confirms our observations in Section 2.2. Moreover, it maintains equivalent link qualities as using the maximum transmission power solution.
2. ATPC achieves significant energy savings compared to other network-level transmission power solutions. ATPC only consumes 53.6% of the transmission energy of the maximum transmission power solution, and 78.8% of the transmission energy of the network-level transmission power solution.
3. ATPC accurately predicts the proper transmission power level and adjusts the transmission power level in time to meet environmental changes, adapting to spatial and temporal factors.



(a) Predicated Transmission Power *vs.* PRR



(b) Predicated Transmission Power *vs.* RSSI

Fig. 8. Prediction Accuracy

4.1 Initialization Phase

In the initialization phase of ATPC, each mote broadcasts a group of beacons. Its neighbors record the RSSI and the corresponding transmission power level of each beacon that they can hear, and then send them back to the beaconing node. Using these pairs of values as input for the ATPC module, the beaconing node builds the predictive models and computes the transmission power level for each of its neighbors.

To evaluate the accuracy of the initialization phase, an experiment is conducted in a parking lot with 8 MICAz motes; it is repeated 5 times. These motes are put in a line 3 feet apart from adjacent nodes. Each mote runs ATPC's initialization phase in a different time slot, sending out 8 beacons at a rate of 5 packets per second using different trans-

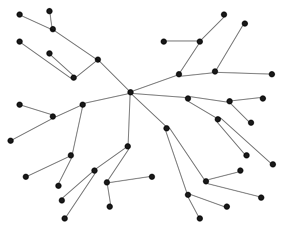


Fig. 9. Topology

Fig. 10. Experimental Site

Date	March 19	March 20	March 21	March 22
High	56º F	54º F	41º F	49º F
Low	27º F	31º F	31º F	30º F
Precip.	0 inch	0 inch	0.05 inch	0 inch
Condition	Fair	Mostly Fair	Cloudy, Light Rain during 10am ~ 12am	Mostly Fair

Fig. 11. Weather Conditions over 72 Hours

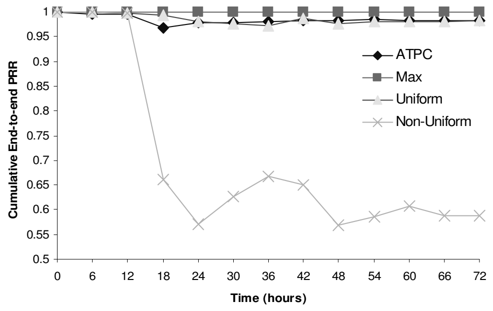


Fig. 12. E2E PRR over Time

mission power levels. These transmission power levels are distributed uniformly in the transmission power range supported by the CC2420 radio chip. After the initialization phase, each mote sends a group of 100 packets to its neighbors using predicted transmission power levels. Its neighbors record the average RSSI and PRR.

The experimental results are shown in Figure 8 (a) and Figure 8 (b). Every point in Figure 8 (a) demonstrates a pair of the predicted transmission power level and the PRR when using that power level. In all these experiments, the average PRR is 99.0%. From Figure 8 (a), we can see that all the RSSI readings are above or equal to -91 dBm. The standard deviation of the RSSI is 2. According to Section 2.3.1, RSSIs that are above -91 dBm means good link quality in a parking lot. These results prove that the predictive model of ATPC works well. Moreover, in our long term experiments, the predicted transmission power levels obtained in ATPC's initialization phase of most nodes are in the desired range.

4.2 Runtime Performance

To evaluate the runtime performance, we compare ATPC against existing transmission power control algorithms: network-level uniform solutions and a node-level non-uniform solution (Non-uniform). Two kinds of network-level transmission power levels are used: the max transmission power level (Max) and the minimum transmission power level over nodes in the network that allows them to reach their neighbors (Uniform). A 72-hour continuous experiment is conducted to evaluate the energy savings and communication quality of ATPC over time. The empirical data shows that ATPC achieves the best overall performance in terms of communication quality and energy consumption. The 3-hop end-to-end PRR of ATPC is constantly above 98% over three days, and ATPC greatly saves transmission power consumption compared to network-level uniform transmission power solutions.

4.2.1 Experiment Setup

A 72-hour experiment is conducted on a grass field with 43 MICAz motes. These motes are deployed according to a randomly generated topology. They form a spanning tree as shown in Figure 9. The root of the spanning tree is at the center of Figure 9. The deployed area is a 15-by-15 meter square. Figure 10 is a picture of the node deployment for one of our experiments on a grass field. All the motes are placed in tupperware containers to protect against the weather. According to our experiments, these plastic boxes (non-conducting material) do not attenuate radio waves significantly.

There are 24 total leaf nodes in this spanning tree. These leaf nodes report data to the base node hourly. Each hour is evenly divided into 24 time slots and different leaf nodes are assigned to different time slots. Transmissions of different motes are scheduled at different times to avoid collision. Each leaf node reports 32 packets to the base node at a transmission rate of 15 packets per minute in its time slot. These packets are divided into 4 groups, corresponding to 4 transmission power control solutions: ATPC, Max, Uniform, and Non-Uniform. These four algorithms are evaluated in the same environment. The predicted transmission power level obtained in ATPC's initialization phase is used for Non-Uniform, which satisfies the assumption that it is the minimum transmission power for each node to reach its neighbors. We use the maximum predicted transmission power level of all nodes obtained in ATPC's initialization phase for Uniform. This transmission power level is the minimum transmission power level over all nodes to reach their neighbors. Max, Uniform, and Non-Uniform all use static transmission power. The statistical data about number of packets sent and received and the transmission power level used for each solution are recorded at each mote. In this experiment, for simplicity, each node considers its parent in the spanning tree as its neighbor. This experiment is deployed on 6 PM on March 19, and finished on 7 PM on March 22. There was a shower that lasted for 2 hours on the morning of March 21. Figure 11 shows the weather conditions of these days.

4.2.2 Data Delivery Ratio

Figure 12 shows the cumulative end-to-end PRR over time. From this figure, we can see that Max achieves 100% end-to-end PRR all the time. As using the maximum transmission power makes the RSSI values at the receiver the highest of all solutions, it is robust to random environmental changes and noise.

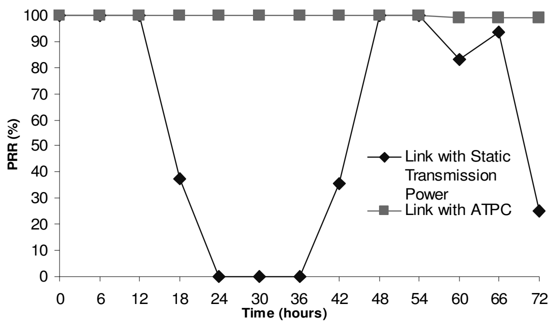


Fig. 13. Link Quality over Time

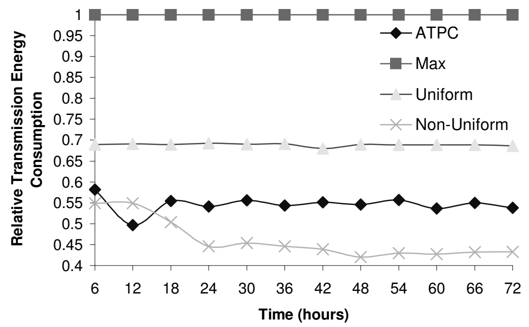


Fig. 14. Transmission Energy Consumption over Time

ATPC and Uniform both achieve around 98% cumulative end-to-end PRR. ATPC has a little better performance than Uniform for 83% of the experimental time. However, the reasons for packet loss of these two solutions are quite different. For ATPC, half of these end-to-end links have 100% PRR. The other 12 links from leaves to the base node suffer from random packet loss from time to time. For Uniform, the packet loss mainly happens at 2 specific links. These links have the same predicted transmission power level as the uniform transmission power level. We pick up one of these two links and plot its PRRs over time in Figure 13. From Figure 13, we compare the PRRs of this link when it works in Uniform and ATPC. This link quality maintained by this static transmission power level is much more vulnerable to environmental changes. After the first 12 hours, the PRR of the link with static transmission power in Uniform drops dramatically, and it is above 95% PRR only 25% of the time. On the other hand, the same link with ATPC constantly achieves above 99% PRR while exposed in the same environment and using the same radio hardware. These two weak links are between leaf nodes and first-level parent nodes, so the packet loss they caused does not have a big impact on the average end-to-end PRR. However, if such a static transmission power level is used at links with more traffic, such as a link between a 2-level parent and the base, the end-to-end communication quality would drop severely.

Non-Uniform solution has weak performance over time. All the links in this solution are vulnerable to link quality variation. However, in the short term and in relatively static weather conditions, Non-Uniform can achieve more than 99% end-to-end PRR, as shown in Figure 12. After the first 12 hours, the communication quality of Non-Uniform becomes poor and unstable. We also notice that the variation of its trend is much bigger than other solutions. It means the end-to-end PRR with these static transmission power levels at certain time periods can be significantly better or worse than at other time periods of the day. This observation confirms our judgment that the dynamics of link quality may make communication performance unstable and unpredictable when assuming static transmission power.

Considering the quality of wireless communication, ATPC and maximum transmission power solutions are proper to apply in real systems.

4.2.3 Power Consumption

The total energy consumption of the network is measured in the radio's transmission mode when different schemes are used. We calculate the total energy spent in the transmit state of the system by the following formula,

$$E = \sum_{i=1}^{n} \left(\sum_{j=min}^{max} ((NumD_{ij} \times TE_j) \times LD) + NumC_i \times maxTE \times LC \right), \quad (5)$$

where i is the node ID and j is the transmission power level. $NumD_{ij}$ is the number of data packets sent at node i with transmission power level j. TE_j is the transmission energy consumed per bit from [7]. LD is the length of a data packet, which is 45 bytes. All the control packets are sent with the maximum transmission power level. $NumC_i$ is the number of control packets (beacons and notifications) sent at node i. $maxTE$ is the transmission energy per bit when using the maximum transmission power level. We get $maxTE$ also from [7]. LC is the length of a control packet, which is 19 bytes. In our experiments, the ratio of the number of control packets and the number of data packets is 3.9%. The ratio of the energy consumed by control packets and the energy consumed by data packets is 1.9%. ATPC achieves energy-efficient transmission with small control overhead.

For better comparison, we take the energy consumption of the Max scheme as the base line, which is unit 1 in Figure 14. The power consumptions of the other three schemes are represented as percentage values compared with this base line. The empirical data demonstrate that ATPC and Non-Uniform consume the least transmission energy. Considering that ATPC has much better communication quality than Non-Uniform, ATPC is the most energy-efficient solution. In Figure 14, ATPC has much less transmission energy consumption than Max and Uniform. Although ATPC has extra beacon and feedback packets, the average transmission energy consumption of ATPC is about 53.6% of Max and 78.8% of Uniform.

The trend of ATPC's energy consumption varies a little bit. The main factor causing this variation is the transmission power level variation. There are only 3 feedback packets per link per day on average. Comparing ATPC with Non-Uniform in the first 6 hours, ATPC has similar energy consumption as Non-Uniform. The reason is that the transmission power level of each mote does not change much in the first 6 hours. In the next 6 hours, Non-Uniform has higher energy consumption than ATPC because a large number of nodes decrease their transmission power level to save energy

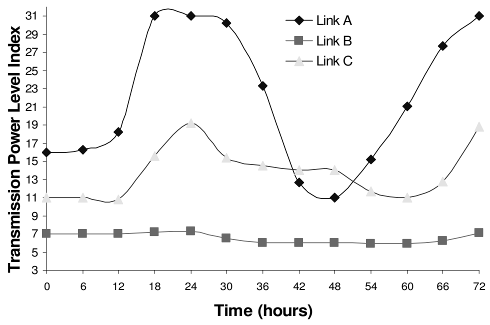


Fig. 15. Average Transmission Power Level over Time

in ATPC. Later, the transmission energy of Non-Uniform drops mainly because of its low PRR, which reduces the number of transmission relays.

Max and Uniform have relatively stable transmission energy consumptions because they use a static transmission power level and their network throughput is stable. The transmission power level used in Uniform largely depends on the topology. In a network with long distance neighbors, this uniform transmission power level tends to get close to the maximum transmission power level. Both solutions waste significant transmission energy compared to ATPC.

The total energy consumption of the Non-Uniform varies because its network throughput varies. Compared to the other solutions, it consumes the least transmission energy over time. It doesn't have the overhead of feedback in ATPC, but the energy is not used efficiently due to its low communication quality. However, it may provide good communication quality and save energy in the short term.

We choose three links and plot the average transmission power they used over time in Figure 15. All these links constantly have above 98% PRR. From Figure 15, we have two main observations as follows.

From a historical record of the tuning process in ATPC, it is confirmed that link qualities vary significantly in reality. Though all these links work in the same environment, the tuning rate and range of transmission power for different links can be significantly different. We can see Link A has a large varying range, which means high sensitivity to environmental changes. Transmission power of Link C is quite stable; it is a robust link to environmental changes. The variation of transmission power of Link B is in between. Link B is a more typical case in our experiments.

ATPC is robust in handling dynamics of link quality in reality, according to differences of link conditions. Although all these links are exposed to the same environment, the impacts of the environment on them are link-specific. ATPC successfully adjusts the transmission power differently. It also confirms our judgments in Section 2.3.2 both that environmental change is a major reason for the transmission power adjustment, and that the adjustment speed depends on the variation speed of the environment.

To summarize, ATPC maintains above 98% end-to-end communication quality while saving transmission power significantly. The static non-uniform transmission power solution may work well on the short term in static environments, but its communication qualities are very vulnerable to environmental changes. The maximum transmission power solution is robust with regard to environmental changes but wastes transmission energy.

5 State of the Art

There are three categories of research topics related to our ATPC: Transmission Power Control, Topology Control and empirical studies on wireless radio communication.

There are a small number of researches on realistic transmission power control for wireless sensor networks. The authors of [34] provide a valuable study about the impact of transmission power control on link qualities and propose a novel blacklisting approach. The ATPC we propose is different from their work. First, since link quality varies with time, different transmission powers are needed to maintain the same desired link quality. ATPC uses a feedback-based scheme to pick optimal power levels at different times; this is not addressed in [34]. Second, protocol [34] fixes the number of configurable power levels, reducing the design flexibility and also limiting the maximum power tuning accuracy that can be achieved. Also, [16] makes an experimental comparison of several existing transmission power control algorithms, and in [14], the authors give a short survey of transmission power control.

There is some other work on transmission power control evaluated in simulation. In [28], the authors formulate the transmission power adjustment problem for static and dynamic network topologies. The authors of [37] describe a power control algorithm to increase transmission power to reach neighbors. Protocol [25] introduces cluster-based transmission power control. The authors of [21] propose an algorithm which increases transmission power to reach neighbors in every cone of a certain degree. Most of these works are simulation-based and they ignore the in-situ impact on communication quality in reality. Our approach is based on systematic empirical studies and we adopt a unique feedback-based approach, tuning link quality pairwise.

Topology control research is a well-studied area in ad hoc and sensor network communities. The goal of a significant portion of these efforts is to achieve better network performance, considering throughput, connectivity, network size, traffic load, and so on. These works can be classified in three major categories according to the transmission range and power assumptions: network-level uniform transmission power [27] [25] [2] [18] [31], node-level non-uniform transmission power [11] [2] [18] [19] [28] [37] [17] [26] [30] [22], and neighbor-level transmission power solutions [23] [42] [3]. Most of these works are based on simulations, which carry the assumptions that the transmission range is static, circular, and within the transmission range the link quality is perfect and never changes. However, such assumptions do not hold in reality. Therefore, solutions making these assumptions may lead to unstable and unpredictable communication qualities. ATPC, based on empirical studies about communication reality, addresses the practical issues of radio and link dynamics.

There are a number of experimental research results on radio communication reality in wireless sensor networks.

In [10] [40], the authors extensively study communication reality in a large scale sensor network. The authors of [43] study the impact of spatial-temporal characteristics on packet loss, and its environmental dependence on packet delivery performance in a wireless sensor network. The authors of [44] give a lot of insight on causes of the radio irregularity phenomenon. In [29], the authors suggest using RSSI value as a reliable parameter to predict a reception rate. The authors of [20] study the relationship between SNR and PRR. With different foci, these experimental works are complementary to our work.

Although the literature is rich, simplifying assumptions may hinder most work from being applied directly to physically deployed sensor networks. We believe a practical transmission power control algorithm like ATPC is the key to apply previous theoretical work to real-world wireless sensor networks.

6 Conclusions and Future Work

We believe there is a serious gap between existing theory work and the in-situ practice. As a solid step towards the in-situ topology control in sensor networks, ATPC presents a lightweight transmission power control technique in a pairwise manner. This fine-granularity tuning trades off computation and local memory (e.g., need a table in each node) with communication, a much more costly operation in terms of energy. Our in-situ experiments reveal the correlation between RSSI/LQI and link quality. Such observations guide us to set up a model to predict the proper transmission power, which is enough to guarantee a good packet reception ratio. We acknowledge that this work is by no means conclusive. However, it indicates a worthwhile direction for future research, so that we can build sensor systems for practical deployment.

Our experiments are designed without congestion and collision. According to our experimental results, ATPC works very well in TDMA protocols. In a low utilization network, where collision and congestion do not happen very frequently, ATPC can still work well. This is because feedback control is renowned for its ability to handle stochastic disturbances.

Conflicting transmissions and interferences may impact the performance of ATPC. However, the capture effect makes the influence of collision and interference on ATPC less serious. Since a packet can be received even when there are overlapped radio signals raised by simultaneous transmission, using RSSI/LQI of such a packet may drive ATPC to unsteady state. In [39], the authors address a technique to detect packet collision. In [45], the authors create an approach to detect interferences. By adopting such techniques, RSSI/LQI for packets identified from packet collision is not considered as input for ATPC. Therefore, ATPC is expected to work equally well in a CSMA network by filtering disturbances caused by collision and interference. This is one of the major future works for ATPC.

7 Acknowledgements

This work is supported in part by National Science Foundation grants CNS-0615063 and CNS-0414870. We thank the anonymous reviewers and our shepherd, Shivakant Mishra, for their insightful comments. We are grateful to Stephen G. Wilson and Sang Son for valuable discussions.

8 References

[1] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A Line in the Sand: A Wireless Sensor Network for Target Detection, Classification, and Tracking. *Comput. Networks*, 46(5):605 – 634, 2004.

[2] C. Bettstetter. On the Connectivity of Wireless Multihop Networks with Homogeneous and Inhomogeneous Range Assignment. In *IEEE VTC*, volume 3, pages 1706 – 1710, September 2002.

[3] D. Blough, M. Leoncini, G. Resta, and P. Santi. The k-Neigh Protocol for Symmetric Topology Control in Ad Hoc Networks. In *ACM MobiHoc*, pages 141 – 152, June 2003.

[4] A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, and D. Estrin. Statistical Model of Lossy Links in Wireless Sensor Networks. In *ACM/IEEE IPSN*, April 2005.

[5] O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J. Stankovic, and T. Abdelzaher. Real-time Power Aware Routing in Wireless Sensor Networks. In *IWQOS*, June 2006.

[6] CC1000 A unique UHF RF Transceiver. http://www.chipcon.com.

[7] CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver. http://www.chipcon.com.

[8] XBOW MICAz Mote Specifications. http://www.xbow.com.

[9] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks. In *ACM Mobile Computing and Communications Review*, volume 5, October 2001.

[10] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks. In *Technical Report UCLA/CSD-TR 02-0013*, 2002.

[11] J. Gomez and A. Campbell. A Case for Variable-Range Transmission Power Control in Wireless Multihop Networks. In *IEEE INFOCOM*, volume 3, pages 1425– 1436, March 2004.

[12] J. Gomez, A. Campbell, M. Naghshineh, and C. Bisdikian. PARO: Supporting Dynamic Power Controlled Routing in Wireless Ad Hoc Networks. In *ACM/Kluwer WINET*, volume 9, pages 443 – 460, 2003.

[13] T. He, S. Krishnamurthy, J. A. Stankovic, T. F. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh. Energy-Efficient Surveillance System Using Wireless Sensor Networks. In *ACM MobiSys*, pages 270– 283, June 2004.

[14] J. Heidemann and W. Ye. Energy Conservation in Sensor Networks at the Link and Network Layers. In *Technical Report USC/ISI-TR-2004-599*, 2004.

[15] IEEE 802.15.4, Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs), 1999. IEEE Std. 802.15.4, 2003.

[16] J. Jeong, D. E. Culler, and J. H. Oh. Empirical Analysis of Transmission Power Control Algorithms for Wireless Sensor Networks. In *Technical Report No. UCB/EECS-2005-16*, 2005.

[17] V. Kawadia, S. Narayanaswamy, R. S. Sreenivas, R. Rozovsky, and P.R. Kumar. Protocols for Media Access Control and Power Control in Wireless Networks. In *40th IEEE Conference on Decision and Control*, pages 1935 – 1940, December 2001.

[18] L. M. Kirousis, E. Kranakis, D. Krizanc, and A. Pelc. Power Consumption in Packet Radio Networks. In *Theoretical Computer Science*, volume 243, pages 289 – 305, July 2000.

[19] M. Kubisch, H. Karl, A. Wolisz, L. C. Zhong, and J. M. Rabaey. Distributed Algorithms for Transmission Power Control in Wireless Sensor Networks. In *IEEE WCNC*, March 2003.

[20] D. Lal, A. Manjeshwar, F. Herrmann, E. Uysal-Biyikoglu, and A. Keshavarzian. Measurement and Characterization of Link Quality Metrics in Energy Constrained Wireless Sensor Networks. In *IEEE GlobeCom*, volume 1, pages 446 – 452, December 2003.

[21] L. Li, J. Halpern, V. Bahl, Y. M. Wang, and R. Wattenhofer. A Cone-Based Distributed Topology-Control Algorithm for Wireless Multi-Hop Networks. In *IEEE/ACM Transactions on Networking*, volume 13, pages 147 – 159, Feburary 2005.

[22] X. Y. Li, P. J. Wan, Y. Wang, and O. Frieder. Sparse Power Efficient Topology for Wireless Networks. In *HICSS*, January 2002.

[23] J. Liu and B. Li. MobileGrid: Capacity-Aware Topology Control in Mobile Ad Hoc Networks. In *IEEE ICCCN*, pages 570 – 574, October 2002.

[24] J. Liu, J. Reich, and F. Zhao. Collaborative In-Network Processing for Target Tracking. *EURASIP JASP*, 2003(4):378 – 391, April 2003.

[25] S. Narayanaswamy, V. Kawadia, R. S. Sreenivas, and P. R. Kumar. Power Control in Ad Hoc Networks: Theory, Architecture, Algorithm and Implementation of the COMPOW Protocol. In *European Wireless Conference*, pages 156 – 162, Feburary 2002.

[26] S. J. Park and R. Sivakumar. Load-sensitive transmission power control in wireless ad-hoc networks. In *IEEE GlobeCom*, volume 1, pages 42 – 46, November 2002.

[27] S. J. Park and R. Sivakumar. Quantitative Analysis of Transmission Power Control in Wireless Ad-hoc Networks. In *IWAHN*, pages 56 – 63, August 2002.

[28] R. Ramanathan and R. R-Hain. Topology Control of Multihop Wireless Networks using Transmit Power Adjustment. In *IEEE INFOCOM*, volume 2, pages 404 – 413, March 2000.

[29] N. Reijers, G. Halkes, and K. Langendoen. Link Layer Measurements in Sensor Networks. In *IEEE MASS*, pages 224 – 234, October 2004.

[30] V. Rodoplu and T. H. Meng. Minimum Energy Mobile Wireless Networks. In *IEEE JSAC*, volume 17, pages 1333 – 1344, August 1999.

[31] P. Santi and D. M. Blough. The Critical Transmitting Range for Connectivity in Sparse Wireless Ad Hoc Networks. In *IEEE Transactions on Mobile Computing*, volume 2, pages 25 – 39, 2003.

[32] P. M. Shankar. *Introduction to Wireless Systems*. John Wiley and Sons, Inc., 2001.

[33] S. Singh, M. Woo, and C. S. Raghavendra. Power-Aware Routing in Mobile Ad Hoc Networks. In *ACM MobiCom*, pages 181 – 190, October 1998.

[34] D. Son, B. Krishnamachari, and J. Heidemann. Experimental Study of the Effects of Transmission Power Control and Blacklisting in Wireless Sensor Networks. In *IEEE SECON*, pages 289 – 298, October 2004.

[35] M. W. Subbarao. Dynamic Power-Conscious Routing for MANETs: An Initial Approach. In *IEEE VTC*, pages 1232 – 1237, September 1999.

[36] G. Tolle, J. Polastre, R. Szewczyk, N. Turner, K. Tu, S. Burgess, D. Gay, P. Buonadonna, W. Hong, T. Dawson, and D. Culler. A Macroscope in the Redwoods. In *ACM SenSys*, November 2005.

[37] R. Wattenhofer, L. Li, P. Bahl, , and Y. M. Wang. Distributed Topology Control for Power Efficient Operation in Multihop Wireless Ad Hoc Networks. In *IEEE INFOCOM*, pages 1388–1397, April 2001.

[38] G. Werner-Allen, K. Lorincz, M. C. Ruiz, O. Marcillo, J. B. Johnson, J. M. Lees, and M. Welsh. Deploying a Wireless Sensor Network on an Active Volcano. In *IEEE Internet Computing, Special Issue on Data-Driven Applications in Sensor Networks*, volume 10, pages 18 – 25, March 2006.

[39] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, and D. Culler. Exploiting the Capture Effect for Collision Detection and Recovery. In *IEEE EmNetS-II*, May 2005.

[40] A. Woo, T. Tong, and D. Culler. Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks. In *ACM SenSys*, November 2003.

[41] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A Wireless Sensor Network for Structural Monitoring. In *ACM SenSys*, November 2004.

[42] F. Xue and P. R. Kumar. The Number of Neighbors Needed for Connectivity of Wireless Networks. In *Wireless Networks*, volume 10, pages 169 – 181, March 2004.

[43] J. Zhao and R. Govindan. Understanding Packet Delivery Performance in Dense Wireless Sensor Networks. In *ACM SenSys*, November 2003.

[44] G. Zhou, T. He, S. Krishnamurthy, and J.A. Stankovic. Impact of Radio Irregularity on Wireless Sensor Networks. In *ACM MobiSys*, pages 125 – 138, June 2004.

[45] G. Zhou, T. He, J.A. Stankovic, and T.F. Abdelzaher. RID: Radio Interference Detection in Wireless Sensor Networks. In *IEEE INFOCOM*, volume 2, pages 891 – 901, March 2005.

StarDust: A Flexible Architecture for Passive Localization in Wireless Sensor Networks[*]

Radu Stoleru, Pascal Vicaire, Tian He[†], John A. Stankovic

Department of Computer Science, University of Virginia
[†]Department of Computer Science and Engineering, University of Minnesota

{stoleru, pv9f}@cs.virginia.edu, tianhe@cs.umn.edu, stankovic@cs.virginia.edu

ABSTRACT

The problem of localization in wireless sensor networks where nodes do not use ranging hardware, remains a challenging problem, when considering the required location accuracy, energy expenditure and the duration of the localization phase. In this paper we propose a framework, called StarDust, for wireless sensor network localization based on passive optical components. In the StarDust framework, sensor nodes are equipped with optical retro-reflectors. An aerial device projects light towards the deployed sensor network, and records an image of the reflected light. An image processing algorithm is developed for obtaining the locations of sensor nodes. For matching a node ID to a location we propose a constraint-based label relaxation algorithm. We propose and develop localization techniques based on four types of constraints: node color, neighbor information, deployment time for a node and deployment location for a node. We evaluate the performance of a localization system based on our framework by localizing a network of 26 sensor nodes deployed in a $120 \times 60 ft^2$ area. The localization accuracy ranges from $2ft$ to $5ft$ while the localization time ranges from 10 milliseconds to 2 minutes.

Categories and Subject Descriptors: C.2.4 [Computer Communications Networks]: Distributed Systems; C.3 [Special Purpose and Application Based Systems]: Real-time and embedded systems

General Terms: Algorithms, Measurement, Performance, Design, Experimentation

Keywords: Wireless Sensor Networks, Localization

1. INTRODUCTION

Wireless Sensor Networks (WSN) have been envisioned to revolutionize the way humans perceive and interact with the surrounding environment. One vision is to embed tiny sensor devices in outdoor environments, by aerial deployments from unmanned air vehicles. The sensor nodes form a network and collaborate (to compensate for the extremely scarce resources available to each of them: computational power, memory size, communication capabilities) to accomplish the mission. Through collaboration, redundancy and fault tolerance, the WSN is then able to achieve unprecedented sensing capabilities.

A major step forward has been accomplished by developing systems for several domains: military surveillance [1] [2] [3], habitat monitoring [4] and structural monitoring [5]. Even after these successes, several research problems remain open. Among these open problems is sensor node localization, i.e., how to find the physical position of each sensor node. Despite the attention the localization problem in WSN has received, no universally acceptable solution has been developed. There are several reasons for this. On one hand, localization schemes that use ranging are typically high end solutions. GPS ranging hardware consumes energy, it is relatively expensive (if high accuracy is required) and poses form factor challenges that move us away from the vision of dust size sensor nodes. Ultrasound has a short range and is highly directional. Solutions that use the radio transceiver for ranging either have not produced encouraging results (if the received signal strength indicator is used) or are sensitive to environment (e.g., multipath). On the other hand, localization schemes that only use the connectivity information for inferring location information are characterized by low accuracies: $\approx 10ft$ in controlled environments, $40 - 50ft$ in realistic ones.

To address these challenges, we propose a framework for WSN localization, called StarDust, in which the complexity associated with the node localization is completely removed from the sensor node. The basic principle of the framework is localization through passivity: each sensor node is equipped with a corner-cube retro-reflector and possibly an optical filter (a coloring device). An aerial vehicle projects light onto the deployment area and records images containing retro-reflected light beams (they appear as luminous spots). Through image processing techniques, the locations of the retro-reflectors (i.e., sensor nodes) is determined. For inferring the identity of the sensor node present at a particular location, the StarDust framework develops a constraint-based node ID relaxation algorithm.

The main contributions of our work are the following. We

[*]This work was supported by the DARPA IXO office (grant F336616-01-C-1905), by the ARO grant W911NF-06-1-0204 and by the NSF grants CCR-0098269, CNS-0626616, CNS-0626614. We would like to thank the anonymous reviewers and our shepherd, Nirupama Bulusu, for their valuable feedback.

propose a novel framework for node localization in WSNs that is very promising and allows for many future extensions and more accurate results. We propose a constraint-based label relaxation algorithm for mapping node IDs to the locations, and four constraints (node, connectivity, time and space), which are building blocks for very accurate and very fast localization systems. We develop a sensor node hardware prototype, called a SensorBall. We evaluate the performance of a localization system for which we obtain location accuracies of $2 - 5ft$ with a localization duration ranging from 10 milliseconds to 2 minutes. We investigate the range of a system built on our framework by considering realities of physical phenomena that occurs during light propagation through the atmosphere.

The rest of the paper is structured as follows. Section 2 is an overview of the state of art. The design of the StarDust framework is presented in Section 3. One implementation and its performance evaluation are in Sections 4 and 5, followed by a suite of system optimization techniques, in Section 6. In Section 7 we present our conclusions.

2. RELATED WORK

We present the prior work in localization in two major categories: the range-based, and the range-free schemes.

The range-based localization techniques have been designed to use either more expensive hardware (and hence higher accuracy) or just the radio transceiver. Ranging techniques dependent on hardware are the time-of-flight (ToF) and the time-difference-of-arrival (TDoA). Solutions that use the radio are based on the received signal strength indicator (RSSI) and more recently on radio interferometry.

The ToF localization technique that is most widely used is the GPS. GPS is a costly solution for a high accuracy localization of a large scale sensor network. AHLoS [6] employs a TDoA ranging technique that requires extensive hardware and solves relatively large nonlinear systems of equations. The Cricket location-support system (TDoA) [7] can achieve a location granularity of tens of inches with highly directional and short range ultrasound transceivers. In [2] the location of a sniper is determined in an urban terrain, by using the TDoA between an acoustic wave and a radio beacon. The PushPin project [8] uses the TDoA between ultrasound pulses and light flashes for node localization. The RADAR system [9] uses the RSSI to build a map of signal strengths as emitted by a set of beacon nodes. A mobile node is located by the best match, in the signal strength space, with a previously acquired signature. In MAL [10], a mobile node assists in measuring the distances (acting as constraints) between nodes until a rigid graph is generated. The localization problem is formulated as an on-line state estimation in a nonlinear dynamic system [11]. A cooperative ranging that attempts to achieve a global positioning from distributed local optimizations is proposed in [12]. A very recent, remarkable, localization technique is based on radio interferometry, RIPS [13], which utilizes two transmitters to create an interfering signal. The frequencies of the emitters are very close to each other, thus the interfering signal will have a low frequency envelope that can be easily measured. The ranging technique performs very well. The long time required for localization and multi-path environments pose significant challenges.

Real environments create additional challenges for the range based localization schemes. These have been emphasized by several studies [14] [15] [16]. To address these challenges, and others (hardware cost, the energy expenditure, the form factor, the small range, localization time), several range-free localization schemes have been proposed. Sensor nodes use primarily connectivity information for inferring proximity to a set of anchors. In the Centroid localization scheme [17], a sensor node localizes to the centroid of its proximate beacon nodes. In APIT [18] each node decides its position based on the possibility of being inside or outside of a triangle formed by any three beacons within node's communication range. The Gradient algorithm [19], leverages the knowledge about the network density to infer the average one hop length. This, in turn, can be transformed into distances to nodes with known locations. DV-Hop [20] uses the hop by hop propagation capability of the network to forward distances to landmarks. More recently, several localization schemes that exploit the sensing capabilities of sensor nodes, have been proposed. Spotlight [21] creates well controlled (in time and space) events in the network while the sensor nodes detect and timestamp this events. From the spatio-temporal knowledge for the created events and the temporal information provided by sensor nodes, nodes' spatial information can be obtained. In a similar manner, the Lighthouse system [22] uses a parallel light beam, that is emitted by an anchor which rotates with a certain period. A sensor node detects the light beam for a period of time, which is dependent on the distance between it and the light emitting device.

Figure 1: Stars, Planets or Smart Dust?

Many of the above localization solutions target specific sets of requirements and are useful for specific applications. StarDust differs in that it addresses a particular demanding set of requirements that are not yet solved well. StarDust is meant for localizing air dropped nodes where node passiveness, high accuracy, low cost, small form factor and rapid localization are all required. Many military applications have such requirements.

3. STARDUST SYSTEM DESIGN

The design of the StarDust system (and its name) was inspired by the similarity between a deployed sensor network, in which sensor nodes indicate their presence by emitting light, and the Universe consisting of luminous and illuminated objects: stars, galaxies, planets, etc. We depict this similarity in Figure 1, which shows the duality between stars, galaxies and sensor nodes equipped with light emitting capabilities.

The main difficulty when applying the above ideas to the real world is the complexity of the hardware that needs to be put on a sensor node so that the emitted light can be detected from thousands of feet. The energy expenditure for

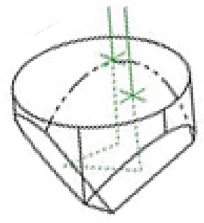


Figure 2: Corner-Cube Retroreflector (a) and an array of CCRs molded in plastic (b)

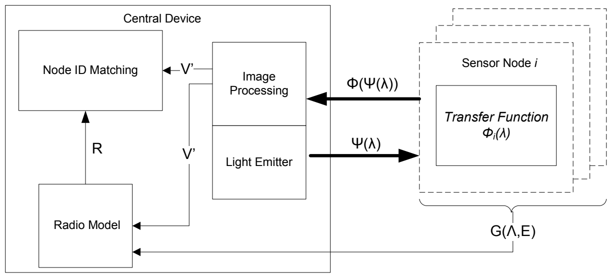


Figure 3: The StarDust system architecture

producing an intense enough light beam is also prohibitive.

Instead, what we propose to use for sensor node localization is a passive optical element called a retro-reflector. The most common retro-reflective optical component is a Corner-Cube Retroreflector (CCR), shown in Figure 2(a). It consists of three mutually perpendicular mirrors. The interesting property of this optical component is that an incoming beam of light is reflected back, towards the source of the light, irrespective of the angle of incidence. This is in contrast with a mirror, which needs to be precisely positioned to be perpendicular to the incident light. A very common and inexpensive implementation of an array of CCRs is the retro-reflective plastic material used on cars and bicycles for night time detection, shown in Figure 2(b).

In the StarDust system, each node is equipped with a small (e.g. $0.5in^2$) array of CCRs and the enclosure has self-righting capabilities that orient the array of CCRs predominantly upwards. It is critical to understand that the upward orientation does not need to be exact. Even when large angular variations from a perfectly upward orientation are present, a CCR will return the light in the exact same direction from which it came.

In the remaining part of the section, we present the architecture of the StarDust system and the design of its main components.

3.1 System Architecture

The envisioned sensor network localization scenario is as follows:

- The sensor nodes are released, possibly in a controlled manner, from an aerial vehicle during the night.
- The aerial vehicle hovers over the deployment area and uses a strobe light to illuminate it. The sensor nodes, equipped with CCRs and optical filters (acting as coloring devices) have self-righting capabilities and retro-reflect the incoming strobe light. The retro-reflected light is either "white", as the originating source light, or colored, due to optical filters.
- The aerial vehicle records a sequence of two images very close in time (msec level). One image is taken when the strobe light is on, the other when the strobe light is off. The acquired images are used for obtaining the locations of sensor nodes (which appear as luminous spots in the image).
- The aerial vehicle executes the mapping of node IDs to the identified locations in one of the following ways: a) by using the color of a retro-reflected light, if a sensor node has a unique color; b) by requiring sensor nodes to establish neighborhood information and report it to a base station; c) by controlling the time sequence of sensor nodes deployment and recording additional images; d) by controlling the location where a sensor node is deployed.
- The computed locations are disseminated to the sensor network.

The architecture of the StarDust system is shown in Figure 3. The architecture consists of two main components: the first is centralized and it is located on a more powerful device. The second is distributed and it resides on all sensor nodes. The Central Device consists of the following: the Light Emitter, the Image Processing module, the Node ID Mapping module and the Radio Model. The distributed component of the architecture is the Transfer Function, which acts as a filter for the incoming light. The aforementioned modules are briefly described below:

- Light Emitter - It is a strobe light, capable of producing very intense, collimated light pulses. The emitted light is non-monochromatic (unlike a laser) and it is characterized by a spectral density $\Psi(\lambda)$, a function of the wavelength. The emitted light is incident on the CCRs present on sensor nodes.
- Transfer Function $\Phi(\Psi(\lambda))$ - This is a bandpass filter for the incident light on the CCR. The filter allows a portion of the original spectrum, to be retro-reflected. From here on, we will refer to the transfer function as the color of a sensor node.
- Image Processing - The Image Processing module acquires high resolution images. From these images the locations and the colors of sensor nodes are obtained. If only one set of pictures can be taken (i.e., one location of the light emitter/image analysis device), then the map of the field is assumed to be known as well as the distance between the imaging device and the field. The aforementioned assumptions (field map and distance to it) are not necessary if the images can be simultaneously taken from different locations. It is important to remark here that the identity of a node can not be directly obtained through Image Processing alone, unless a specific characteristic of a sensor node can be identified in the image.
- Node ID Matching - This module uses the detected locations and through additional techniques (e.g., sensor

Algorithm 1 Image Processing

```
1: Background filtering
2: Retro-reflected light recognition through intensity filtering
3: Edge detection to obtain the location of sensor nodes
4: Color identification for each detected sensor node
```

node coloring and connectivity information ($G(\Lambda, E)$) from the deployed network) to uniquely identify the sensor nodes observed in the image. The connectivity information is represented by neighbor tables sent from each sensor node to the Central Device.

- Radio Model - This component provides an estimate of the radio range to the Node ID Matching module. It is only used by node ID matching techniques that are based on the radio connectivity in the network. The estimate of the radio range R is based on the sensor node density (obtained through the Image Processing module) and the connectivity information (i.e., $G(\Lambda, E)$).

The two main components of the StarDust architecture are the Image Processing and the Node ID Mapping. Their design and analysis is presented in the sections that follow.

3.2 Image Processing

The goal of the Image Processing Algorithm (IPA) is to identify the location of the nodes and their color. Note that IPA does not identify which node fell where, but only what is the set of locations where the nodes fell.

IPA is executed after an aerial vehicle records two pictures: one in which the field of deployment is illuminated and one when no illuminations is present. Let P_{dark} be the picture of the deployment area, taken when no light was emitted and P_{light} be the picture of the same deployment area when a strong light beam was directed towards the sensor nodes.

The proposed IPA has several steps, as shown in Algorithm 1. The first step is to obtain a third picture P_{filter} where only the differences between P_{dark} and P_{light} remain. Let us assume that P_{dark} has a resolution of $n \times m$, where n is the number of pixels in a row of the picture, while m is the number of pixels in a column of the picture. Then P_{dark} is composed of $n \times m$ pixels noted $P_{dark}(i,j)$, $i \in 1 \leq i \leq n, 1 \leq j \leq m$. Similarly P_{light} is composed of $n \times m$ pixels noted $P_{light}(i,j)$, $1 \leq i \leq n, 1 \leq j \leq m$.

Each pixel P is described by an RGB value where the R value is denoted by P^R, the G value is denoted by P^G, and the B value is denoted by P^B. IPA then generates the third picture, P_{filter}, through the following transformations:

$$\begin{array}{l} P^R_{filter}(i,j) = P^R_{light}(i,j) - P^R_{dark}(i,j) \\ P^G_{filter}(i,j) = P^G_{light}(i,j) - P^G_{dark}(i,j) \\ P^B_{filter}(i,j) = P^B_{light}(i,j) - P^B_{dark}(i,j) \end{array} \tag{1}$$

After this transformation, all the features that appeared in both P_{dark} and P_{light} are removed from P_{filter}. This simplifies the recognition of light retro-reflected by sensor nodes.

The second step consists of identifying the elements contained in P_{filter} that retro-reflect light. For this, an intensity filter is applied to P_{filter}. First IPA converts P_{filter} into a grayscale picture. Then the brightest pixels are identified and used to create $P_{reflect}$. This step is eased by the fact that the reflecting nodes should appear much brighter than any other illuminated object in the picture.

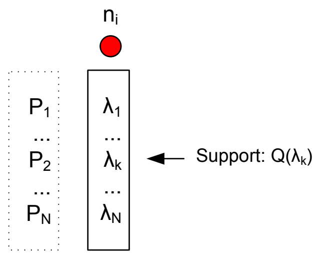


Figure 4: Probabilistic label relaxation

The third step runs an edge detection algorithm on $P_{reflect}$ to identify the boundary of the nodes present. A tool such as Matlab provides a number of edge detection techniques. We used the *bwboundaries* function. For the obtained edges, the location (x, y) (in the image) of each node is determined by computing the centroid of the points constituting its edges. Standard computer graphics techniques [23] are then used to transform the 2D locations of sensor nodes detected in multiple images into 3D sensor node locations. The color of the node is obtained as the color of the pixel located at (x, y) in P_{light}.

3.3 Node ID Matching

The goal of the Node ID Matching module is to obtain the identity (node ID) of a luminous spot in the image, detected to be a sensor node. For this, we define $V = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$ to be the set of locations of the sensor nodes, as detected by the Image Processing module and $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_m\}$ to be the set of unique node IDs assigned to the m sensor nodes, before deployment. From here on, we refer to node IDs as labels.

We model the problem of finding the label λ_j of a node n_i as a probabilistic label relaxation problem, frequently used in image processing/understanding. In the image processing domain, scene labeling (i.e., identifying objects in an image) plays a major role. The goal of scene labeling is to assign a label to each object detected in an image, such that an appropriate image interpretation is achieved. It is prohibitively expensive to consider the interactions among all the objects in an image. Instead, constraints placed among nearby objects generate local consistencies and through iteration, global consistencies can be obtained.

The main idea of the sensor node localization through probabilistic label relaxation is to iteratively compute the probability of each label being the correct label for a sensor node, by taking into account, at each iteration, the "support" for a label. The support for a label can be understood as a hint or proof, that a particular label is more likely to be the correct one, when compared with the other potential labels for a sensor node. We pictorially depict this main idea in Figure 4. As shown, node n_i has a set of candidate labels $\{\lambda_1, ..., \lambda_k\}$. Each of the labels has a different value for the Support function $Q(\lambda_k)$. We defer the explanation of how the Support function is implemented until the subsections that follow, where we provide four concrete techniques. Formally, the algorithm is outlined in Algorithm 2, where the equations necessary for computing the new probability $P_{n_i}(\lambda_k)$ for a label λ_k of a node n_i, are expressed by the

Algorithm 2 Label Relaxation

```
 1: for each sensor node $n_i$ do
 2:     assign equal prob. to all possible labels
 3: end for
 4: repeat
 5:     $converged \leftarrow true$
 6:     for each sensor node $n_i$ do
 7:         for each each label $\lambda_j$ of $n_i$ do
 8:             compute the Support label $\lambda_j$: Equation 4
 9:         end for
10:         compute $K$ for the node $n_i$: Equation 3
11:         for each each label $\lambda_j$ do
12:             update probability of label $\lambda_j$: Equation 2
13:             if $|new\ prob. - old\ prob.| \geq \epsilon$ then
14:                 $converged \leftarrow false$
15:             end if
16:         end for
17:     end for
18: until $converged = true$
```

following equations:

$$P_{n_i}^{s+1}(\lambda_k) = \frac{1}{K_{n_i}} P_{n_i}^{s}(\lambda_k) Q_{n_i}^{s}(\lambda_k) \tag{2}$$

where K_{n_i} is a normalizing constant, given by:

$$K_{n_i} = \sum_{k=1}^{N} P_{n_i}^{s}(\lambda_k) Q_{n_i}^{s}(\lambda_k) \tag{3}$$

and $Q_{n_i}^{s}(\lambda_k)$ is:

$$Q_{n_i}^{s}(\lambda_k) = \text{“}\textit{support for label } \lambda_k \textit{ of node } n_i\text{”} \tag{4}$$

The label relaxation algorithm is iterative and it is polynomial in the size of the network(number of nodes). The pseudo-code is shown in Algorithm 2. It initializes the probabilities associated with each possible label, for a node n_i, through a uniform distribution. At each iteration s, the algorithm updates the probability associated with each label, by considering the Support $Q_{n_i}^{s}(\lambda_k)$ for each candidate label of a sensor node.

In the sections that follow, we describe four different techniques for implementing the Support function: based on node coloring, radio connectivity, the time of deployment (time) and the location of deployment (space). While some of these techniques are simplistic, they are primitives which, when combined, can create powerful localization systems. These design techniques have different trade-offs, which we will present in Section 3.3.6.

3.3.1 Relaxation with Color Constraints

The unique mapping between a sensor node's position (identified by the image processing) and a label can be obtained by assigning a unique color to each sensor node. For this we define $C = \{c_1, c_2, ..., c_n\}$ to be the set of unique colors available and $M : \Lambda \rightarrow C$ to be a one-to-one mapping of labels to colors. This mapping is known prior to the sensor node deployment (from node manufacturing).

In the case of color constrained label relaxation, the support for label λ_k is expressed as follows:

$$Q_{n_i}^{s}(\lambda_k) = 1 \tag{5}$$

As a result, the label relaxation algorithm (Algorithm 2) consists of the following steps: one label is assigned to each sensor node (lines 1-3 of the algorithm), implicitly having a probability $P_{n_i}(\lambda_k) = 1$; the algorithm executes a single iteration, when the support function, simply, reiterates the confidence in the unique labeling.

However, it is often the case that unique colors for each node will not be available. It is interesting to discuss here the influence that the size of the coloring space (i.e., $|C|$) has on the accuracy of the localization algorithm. Several cases are discussed below:

- If $|C| = 0$, no colors are used and the sensor nodes are equipped with simple CCRs that reflect back all the incoming light (i.e., no filtering, and no coloring of the incoming light). From the image processing system, the position of sensor nodes can still be obtained. Since all nodes appear white, no single sensor node can be uniquely identified.

- If $|C| = m - 1$ then there are enough unique colors for all nodes (one node remains white, i.e. no coloring), the problem is trivially solved. Each node can be identified, based on its unique color. This is the scenario for the relaxation with color constraints.

- If $|C| \geq 1$, there are several options for how to partition the coloring space. If $C = \{c_1\}$ one possibility is to assign the color c_1 to a single node, and leave the remaining $m - 1$ sensor nodes white, or to assign the color c_1 to more than one sensor node. One can observe that once a color is assigned uniquely to a sensor node, in effect, that sensor node is given the status of "anchor", or node with known location.

It is interesting to observe that there is an entire spectrum of possibilities for how to partition the set of sensor nodes in equivalence classes (where an equivalence class is represented by one color), in order to maximize the success of the localization algorithm. One of the goals of this paper is to understand how the size of the coloring space and its partitioning affect localization accuracy.

Despite the simplicity of this method of constraining the set of labels that can be assigned to a node, we will show that this technique is very powerful, when combined with other relaxation techniques.

3.3.2 Relaxation with Connectivity Constraints

Connectivity information, obtained from the sensor network through beaconing, can provide additional information for locating sensor nodes. In order to gather connectivity information, the following need to occur: 1) after deployment, through beaconing of HELLO messages, sensor nodes build their neighborhood tables; 2) each node sends its neighbor table information to the Central device via a base station.

First, let us define $G = (\Lambda, E)$ to be the weighted connectivity graph built by the Central device from the received neighbor table information. In G the edge (λ_i, λ_j) has a weight g_{ij} represented by the number of beacons sent by λ_j and received by λ_i. In addition, let R be the radio range of the sensor nodes.

The main idea of the connectivity constrained label relaxation is depicted in Figure 5 in which two nodes n_i and n_j have been assigned all possible labels. The confidence in

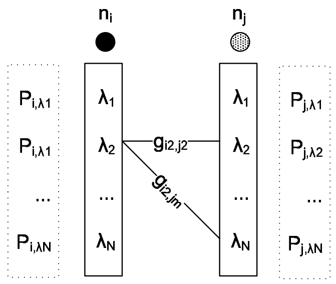


Figure 5: Label relaxation with connectivity constraints

each of the candidate labels for a sensor node, is represented by a probability, shown in a dotted rectangle.

It is important to remark that through beaconing and the reporting of neighbor tables to the Central Device, a global view of all constraints in the network can be obtained. It is critical to observe that these constraints are among labels. As shown in Figure 5 two constraints exist between nodes n_i and n_j. The constraints are depicted by $g_{i2,j2}$ and $g_{i2,jM}$, the number of beacons sent the labels λ_{j2} and λ_{jM} and received by the label λ_{i2}.

The support for the label λ_k of sensor node n_i, resulting from the "interaction" (i.e., within radio range) with sensor node n_j is given by:

$$Q^s_{n_i}(\lambda_k) = \sum_{m=1}^{M} g_{\lambda_k \lambda_m} P^s_{n_j}(\lambda_m) \tag{6}$$

As a result, the localization algorithm (Algorithm 3 consists of the following steps: all labels are assigned to each sensor node (lines 1-3 of the algorithm), and implicitly each label has a probability initialized to $P_{n_i}(\lambda_k) = 1/|\Lambda|$; in each iteration, the probabilities for the labels of a sensor node are updated, when considering the interaction with the labels of sensor nodes within R. It is important to remark that the identity of the nodes within R is not known, only the candidate labels and their probabilities. The relaxation algorithm converges when, during an iteration, the probability of no label is updated by more than ϵ.

The label relaxation algorithm based on connectivity constraints, enforces such constraints between pairs of sensor nodes. For a large scale sensor network deployment, it is not feasible to consider all pairs of sensor nodes in the network. Hence, the algorithm should only consider pairs of sensor nodes that are within a reasonable communication range (R). We assume a circular radio range and a symmetric connectivity. In the remaining part of the section we propose a simple analytical model that estimates the radio range R for medium-connected networks (less than 20 neighbors per R). We consider the following to be known: the size of the deployment field (L), the number of sensor nodes deployed (N) and the total number of unidirectional (i.e., not symmetric) one-hop radio connections in the network (k). For our analysis, we uniformly distribute the sensor nodes in a square area of length L, by using a grid of unit length $L/\sqrt{N}$. We use the substitution $u = L/\sqrt{N}$ to simplify the notation, in order to distinguish the following cases: if $u \le R \le \sqrt{2u}$ each node has four neighbors (the expected $k = 4N$); if $\sqrt{2}u \le R \le 2u$ each node has eight neighbors (the expected $k = 8N$); if $2u \le R \le \sqrt{5}u$ each node has twelve neighbors (the expected $k = 12N$); if $\sqrt{5}u \le R \le 3u$ each node has twenty neighbors (the expected $k = 20N$)

For a given $t = k/4N$ we take R to be the middle of the interval. As an example, if $t = 5$ then $R = (3 + \sqrt{5})u/2$. A quadratic fitting for R over the possible values of t, produces the following closed-form solution for the communication range R, as a function of network connectivity k, assuming L and N constant:

$$R(k) = \frac{L}{\sqrt{N}}\left[-0.051\left(\frac{k}{4N}\right)^2 + 0.66\left(\frac{k}{4N}\right) + 0.6\right] \tag{7}$$

We investigate the accuracy of our model in Section 5.2.1.

Algorithm 3 Localization

```
1: Estimate the radio range $R$
2: Execute the Label Relaxation Algorithm with Support
   Function given by Equation 6 for neighbors less than $R$
   apart
3: for each sensor node $n_i$ do
4:     node identity is $\lambda_k$ with max. prob.
5: end for
```

3.3.3 Relaxation with Time Constraints

Time constraints can be treated similarly with color constraints. The unique identification of a sensor node can be obtained by deploying sensor nodes individually, one by one, and recording a sequence of images. The sensor node that is identified as new in the last picture (it was not identified in the picture before last) must be the last sensor node dropped.

In a similar manner with color constrained label relaxation, the time constrained approach is very simple, but may take too long, especially for large scale systems. While it can be used in practice, it is unlikely that only a time constrained label relaxation is used. As we will see, by combining constrained-based primitives, realistic localization systems can be implemented.

The support function for the label relaxation with time constraints is defined identically with the color constrained relaxation:

$$Q^s_{n_i}(\lambda_k) = 1 \tag{8}$$

The localization algorithm (Algorithm 2 consists of the following steps: one label is assigned to each sensor node (lines 1-3 of the algorithm), and implicitly having a probability $P_{n_i}(\lambda_k) = 1$; the algorithm executes a single iteration, when the support function, simply, reiterates the confidence in the unique labeling.

3.3.4 Relaxation with Space Constraints

Spatial information related to sensor deployment can also be employed as another input to the label relaxation algorithm. To do that, we use two types of locations: the node location p_n and the label location p_l. The former p_n is defined as the position of nodes (x_n, y_n, z_n) after deployment, which can be obtained through Image Processing as mentioned in Section 3.3. The latter p_l is defined as the location (x_l, y_l, z_l) where a node is dropped. We use $D^{n_i}_{\lambda_m}$ to denote the horizontal distance between the location

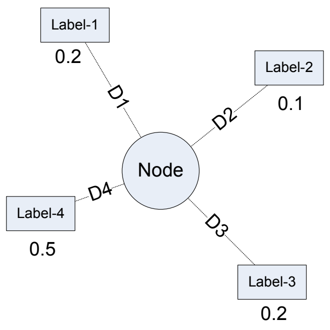


Figure 6: Relaxation with space constraints

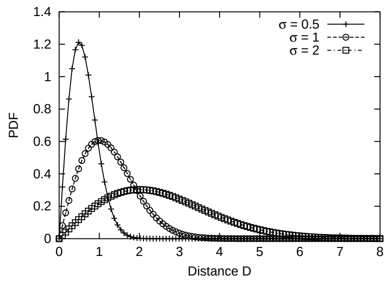


Figure 7: Probability distribution of distances

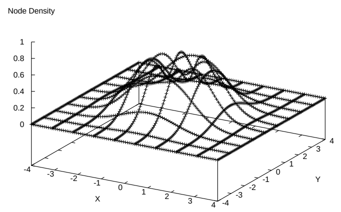


Figure 8: Distribution of nodes

of the label λ_m and the location of the node n_i. Clearly, $D^{n_i}_{\lambda_m} = \sqrt{(x_n - x_l)^2 + (y_n - y_l)^2}$.

At the time of a sensor node release, the one-to-one mapping between the node and its label is known. In other words, the label location is the same as the node location at the release time. After release, the label location information is partially lost due to the random factors such as wind and surface impact. However, statistically, the node locations are correlated with label locations. Such correlation depends on the airdrop methods employed and environments. For the sake of simplicity, let's assume nodes are dropped from the air through a helicopter hovering in the air. Wind can be decomposed into three components $\vec{X}, \vec{Y}$ and $\vec{Z}$. Only $\vec{X}$ and $\vec{Y}$ affect the horizontal distance a node can travel. According to [24], we can assume that $\vec{X}$ and $\vec{Y}$ follow an independent normal distribution. Therefore, the absolute value of the wind speed follows a Rayleigh distribution. Obviously the higher the wind speed is, the further a node would land away horizontally from the label location. If we assume that the distance D is a function of the wind speed V [25] [26], we can obtain the probability distribution of D under a given wind speed distribution. Without loss of generality, we assume that D is proportional to the wind speed. Therefore, D follows the Rayleigh distribution as well. As shown in Figure 6, the spatial-based relaxation is a recursive process to assign the probability that a nodes has a certain label by using the distances between the location of a node with multiple label locations.

We note that the distribution of distance D affects the probability with which a label is assigned. It is not necessarily true that the nearest label is always chosen. For example, if D follows the Rayleigh(σ^2) distribution, we can obtain the Probability Density Function (PDF) of distances as shown in Figure 7. This figure indicates that the possibility of a node to fall vertically is very small under windy conditions ($\sigma > 0$), and that the distance D is affected by the σ. The spatial distribution of nodes for $\sigma = 1$ is shown in Figure 8. Strong wind with a high σ value leads to a larger node dispersion. More formally, given a probability density function $PDF(D)$, the support for label λ_k of sensor node n_i can be formulated as:

$$Q^s_{n_i}(\lambda_k) = PDF(D^{n_i}_{\lambda_k}) \tag{9}$$

It is interesting to point out two special cases. First, if all nodes are released at once (i.e., only one label location for all released nodes), the distance D from a node to all labels is the same. In this case, $P^{s+1}_{n_i}(\lambda_k) = P^s_{n_i}(\lambda_k)$, which indicates that we can not use the spatial-based relaxation to recursively narrow down the potential labels for a node. Second, if nodes are released at different locations that are far away from each other, we have: (i) If node n_i has label λ_k, $P^s_{n_i}(\lambda_k) \to 1$ when $s \to \infty$, (ii) If node n_i does not have label λ_k, $P^s_{n_i}(\lambda_k) \to 0$ when $s \to \infty$. In this second scenario, there are multiple labels (one label per release), hence it is possible to correlate release times (labels) with positions on the ground. These results indicate that spatial-based relaxation can label the node with a very high probability if the physical separation among nodes is large.

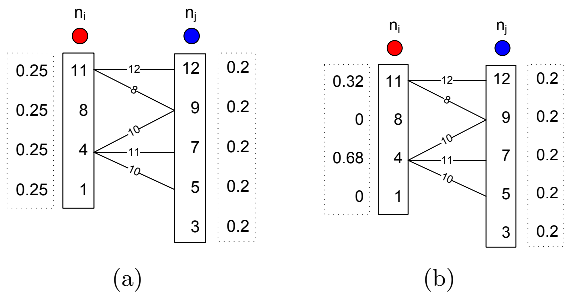


Figure 9: A step through the algorithm. After initialization (a) and after the 1st iteration for node n_i (b)

3.3.5 Relaxation with Color and Connectivity Constraints

One of the most interesting features of the StarDust architecture is that it allows for hybrid localization solutions to be built, depending on the system requirements. One example is a localization system that uses the color and connectivity constraints. In this scheme, the color constraints are used for reducing the number of candidate labels for sensor nodes, to a more manageable value. As a reminder, in the connectivity constrained relaxation, all labels are candidate labels for each sensor node. The color constraints are used in the initialization phase of Algorithm 3 (lines 1-3). After the initialization, the standard connectivity constrained relaxation algorithm is used.

For a better understanding of how the label relaxation algorithm works, we give a concrete example, exemplified in Figure 9. In part (a) of the figure we depict the data structures associated with nodes n_i and n_j after the initialization steps of the algorithm (lines 1-6), as well as the number of beacons between different labels (as reported by the network, through $G(\Lambda, E)$). As seen, the potential labels (shown inside the vertical rectangles) are assigned to each node. Node n_i can be any of the following: $11, 8, 4, 1$. Also depicted in the figure are the probabilities associated with each of the labels. After initialization, all probabilities are equal.

Part (b) of Figure 9 shows the result of the first iteration of the localization algorithm for node n_i, assuming that node n_j is the first w_i chosen in line 7 of Algorithm 3. By using Equation 6, the algorithm computes the "support" $Q(\lambda_i)$ for each of the possible labels for node n_i. Once the $Q(\lambda_i)$'s are computed, the normalizing constant, given by Equation 3 can be obtained. The last step of the iteration is to update the probabilities associated with all potential labels of node n_i, as given by Equation 2.

One interesting problem, which we explore in the performance evaluation section, is to assess the impact the partitioning of the color set C has on the accuracy of localization. When the size of the coloring set is smaller than the number of sensor nodes (as it is the case for our hybrid connectivity/color constrained relaxation), the system designer has the option of allowing one node to uniquely have a color (acting as an anchor), or multiple nodes. Intuitively, by assigning one color to more than one node, more constraints (distributed) can be enforced.

3.3.6 Relaxation Techniques Analysis

The proposed label relaxation techniques have different trade-offs. For our analysis of the trade-offs, we consider the following metrics of interest: the localization time (duration), the energy consumed (overhead), the network size (scale) that can be handled by the technique and the localization accuracy. The parameters of interest are the following: the number of sensor nodes (N), the energy spent for one aerial drop (ϵ_d), the energy spent in the network for collecting and reporting neighbor information ϵ_b and the time T_d taken by a sensor node to reach the ground after being aerially deployed. The cost comparison of the different label relaxation techniques is shown in the table below.

Criteria	Color	Connectivity	Time	Space
Duration	0	NT_b	NT_d	0
Overhead	ϵ_d	$\epsilon_d + N\epsilon_b$	$N\epsilon_d$	ϵ_d
Scale	$\|C\|$	$\|N\|$	$\|N\|$	$\|N\|$
Accuracy	High	Low	High	Medium

Table 1: Comparison of label relaxation techniques

As shown, the relaxation techniques based on color and space constraints have the lowest localization duration, zero, for all practical purposes. The scalability of the color based relaxation technique is, however, limited to the number of unique color filters that can be built. The narrower the Transfer Function $\Psi(\lambda)$, the larger the number of unique colors that can be created. The manufacturing costs, however, are increasing as well. The scalability issue is addressed by all other label relaxation techniques. Most notably, the time constrained relaxation, which is very similar to the color-constrained relaxation, addresses the scale issue, at a higher deployment cost.

4. SYSTEM IMPLEMENTATION

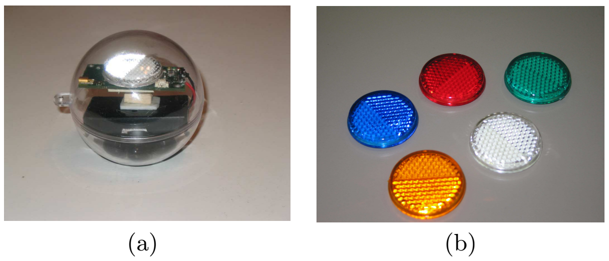

(a) (b)

Figure 10: SensorBall with self-righting capabilities (a) and colored CCRs (b)

The StarDust localization framework, depicted in Figure 3, is flexible in that it enables the development of new localization systems, based on the four proposed label relaxation schemes, or the inclusion of other, yet to be invented, schemes. For our performance evaluation we implemented a version of the StarDust framework, namely the one proposed in Section 3.3.5, where the constraints are based on color and connectivity.

The Central device of the StarDust system consists of the following: the Light Emitter - we used a common-off-the-shelf flash light (QBeam, 3 million candlepower); the image acquisition was done with a 3 megapixel digital camera (Sony DSC-S50) which provided the input to the Image Processing algorithm, implemented in Matlab.

For sensor nodes we built a custom sensor node, called SensorBall, with self-righting capabilities, shown in Figure 10(a). The self-righting capabilities are necessary in order to orient the CCR predominantly upwards. The CCRs that we used were inexpensive, plastic molded, night time warning signs commonly available on bicycles, as shown in Figure 10(b). We remark here the low quality of the CCRs we used. The reflectivity of each CCR (there are tens molded in the plastic container) is extremely low, and each CCR is not built with mirrors. A reflective effect is achieved by employing finely polished plastic surfaces. We had 5 colors available, in addition to the standard CCR, which reflects all the incoming light (white CCR). For a slightly higher price (ours were 20cents/piece), better quality CCRs can be employed. Higher quality (better mirrors) would translate in more accurate image processing (better sensor node detection) and smaller form factor for the optical component (an array of CCRs with a smaller area can be used).

The sensor node platform we used was the micaZ mote. The code that runs on each node is a simple application which broadcasts 100 beacons, and maintains a neighbor table containing the percentage of successfully received beacons, for each neighbor. On demand, the neighbor table is reported to a base station, where the node ID mapping is performed.

5. SYSTEM EVALUATION

In this section we present the performance evaluation of a system implementation of the StarDust localization framework. The three major research questions that our evaluation tries to answer are: the feasibility of the proposed framework (can sensor nodes be optically detected at large distances), the localization accuracy of one actual imple-

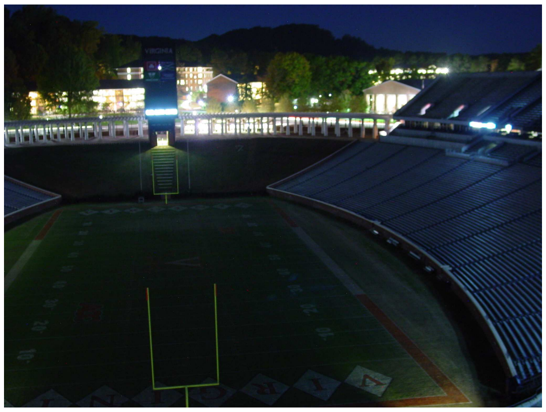


Figure 11: The field in the dark

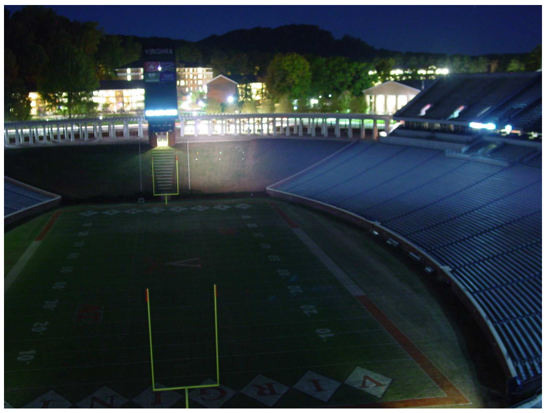


Figure 12: The illuminated field

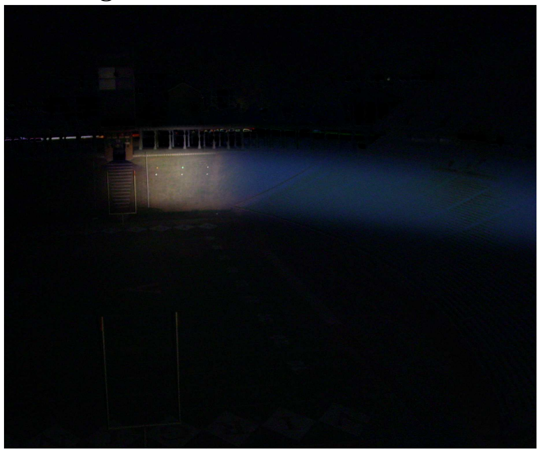


Figure 13: The difference: Figure 11 - Figure 12

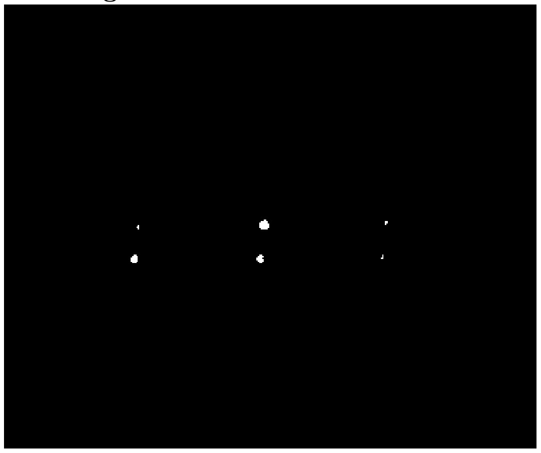


Figure 14: Retroreflectors detected in Figure 13

mentation of the StarDust framework, and whether or not atmospheric conditions can affect the recognition of sensor nodes in an image. The first two questions are investigated by evaluating the two main components of the StarDust framework: the Image Processing and the Node ID Matching. These components have been evaluated separately mainly because of lack of adequate facilities. We wanted to evaluate the performance of the Image Processing Algorithm in a long range, realistic, experimental set-up, while the Node ID Matching required a relatively large area, available for long periods of time (for connectivity data gathering). The third research question is investigated through a computer modeling of atmospheric phenomena.

For the evaluation of the Image Processing module, we performed experiments in a football stadium where we deploy 6 sensor nodes in a 3×2 grid. The distance between the Central device and the sensor nodes is approximately $500ft$. The metrics of interest are the number of false positives and false negatives in the Image Processing Algorithm.

For the evaluation of the Node ID Mapping component, we deploy 26 sensor nodes in an $120 \times 60ft^2$ flat area of a stadium. In order to investigate the influence the radio connectivity has on localization accuracy, we vary the height above ground of the deployed sensor nodes. Two set-ups are used: one in which the sensor nodes are on the ground, and the second one, in which the sensor nodes are raised 3 inches above ground. From here on, we will refer to these two experimental set-ups as the low connectivity and the high connectivity networks, respectively because when nodes are on the ground the communication range is low resulting in less neighbors than when the nodes are elevated and have a greater communication range. The metrics of interest are: the localization error (defined as the distance between the computed location and the true location - known from the manual placement), the percentage of nodes correctly localized, the convergence of the label relaxation algorithm, the time to localize and the robustness of the node ID mapping to errors in the Image Processing module.

The parameters that we vary experimentally are: the angle under which images are taken, the focus of the camera, and the degree of connectivity. The parameters that we vary in simulations (subsequent to image acquisition and connectivity collection) the number of colors, the number of anchors, the number of false positives or negatives as input to the Node ID Matching component, the distance between the imaging device and sensor network (i.e., range), atmospheric conditions (light attenuation coefficient) and CCR reflectance (indicative of its quality).

5.1 Image Processing

For the IPA evaluation, we deploy 6 sensor nodes in a 3×2 grid. We take 13 sets of pictures using different orientations

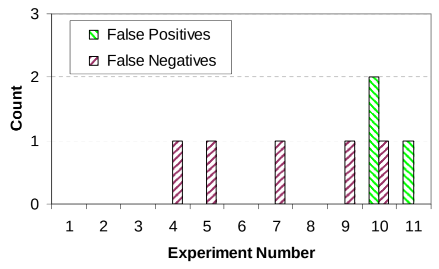


Figure 15: False Positives and Negatives for the 6 sensor nodes

of the camera and different zooming factors. All pictures were taken from the same location. Each set is composed of a picture taken in the dark and of a picture taken with a light beam pointed at the nodes. We process the pictures offline using a Matlab implementation of IPA. Since we are interested in the feasibility of identifying colored sensor nodes at large distance, the end result of our IPA is the 2D location of sensor nodes (position in the image). The transformation to 3D coordinates can be done through standard computer graphics techniques [23].

One set of pictures obtained as part of our experiment is shown in Figures 11 and 12. The execution of our IPA algorithm results in Figure 13 which filters out the background, and Figure 14 which shows the output of the edge detection step of IPA. The experimental results are depicted in Figure 15. For each set of pictures the graph shows the number of false positives (the IPA determines that there is a node while there is none), and the number of false negatives (the IPA determines that there is no node while there is one). In about 45% of the cases, we obtained perfect results, i.e., no false positives and no false negatives. In the remaining cases, we obtained a number of false positives of at most one, and a number of false negatives of at most two.

We exclude two pairs of pictures from Figure 15. In the first excluded pair, we obtain 42 false positives and in the second pair 10 false positives and 7 false negatives. By carefully examining the pictures, we realized that the first pair was taken out of focus and that a car temporarily appeared in one of the pictures of the second pair. The anomaly in the second set was due to the fact that we waited too long to take the second picture. If the pictures had been taken a few milliseconds apart, the car would have been represented on either both or none of the pictures and the IPA would have filtered it out.

5.2 Node ID Matching

We evaluate the Node ID Matching component of our system by collecting empirical data (connectivity information) from the outdoor deployment of 26 nodes in the $120 \times 60ft^2$ area. We collect 20 sets of data for the high connectivity and low connectivity network deployments. Off-line we investigate the influence of coloring on the metrics of interest, by randomly assigning colors to the sensor nodes. For one experimental data set we generate 50 random assignments of colors to sensor nodes. It is important to observe that, for the evaluation of the Node ID Matching algorithm (color and connectivity constrained), we simulate the color assignment to sensor nodes. As mentioned in Section 4 the size of

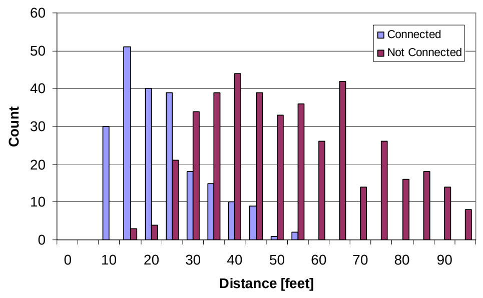


Figure 16: The number of existing and missing radio connections in the sparse connectivity experiment

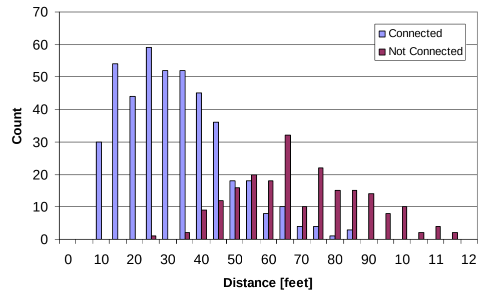


Figure 17: The number of existing and missing radio connections in the high connectivity experiment

the coloring space available to us was 5 (5 colors). Through simulations of color assignment (not connectivity) we are able to investigate the influence that the size of the coloring space has on the accuracy of localization. The value of the parameter ϵ used in Algorithm 2 was 0.001. The results presented here represent averages over the randomly generated colorings and over all experimental data sets.

We first investigate the accuracy of our proposed Radio Model, and subsequently use the derived values for the radio range in the evaluation of the Node ID matching component.

5.2.1 *Radio Model*

From experiments, we obtain the average number of observed beacons (k, defined in Section 3.3.2) for the low connectivity network of 180 beacons and for the high connectivity network of 420 beacons. From our Radio Model (Equation 7, we obtain a radio range $R = 25ft$ for the low connectivity network and $R = 40ft$ for the high connectivity network.

To estimate the accuracy of our simple model, we plot the number of radio links that exist in the networks, and the number of links that are missing, as functions of the distance between nodes. The results are shown in Figures 16 and 17. We define the average radio range R to be the distance over which less than 20% of potential radio links, are missing. As shown in Figure 16, the radio range is between $20ft$ and $25ft$. For the higher connectivity network, the radio range was between $30ft$ and $40ft$.

We choose two conservative estimates of the radio range: $20ft$ for the low connectivity case and $35ft$ for the high connectivity case, which are in good agreement with the values predicted by our Radio Model.

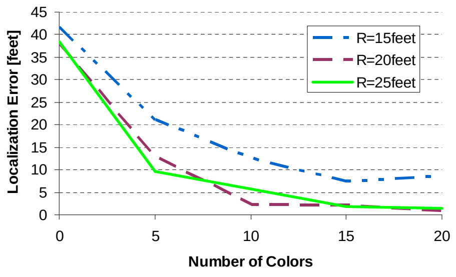


Figure 18: Localization error

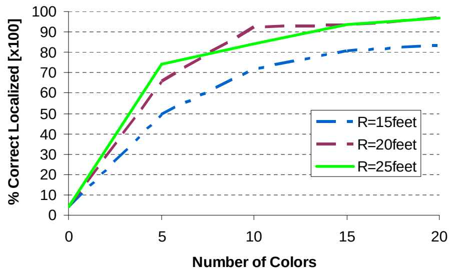


Figure 19: Percentage of nodes correctly localized

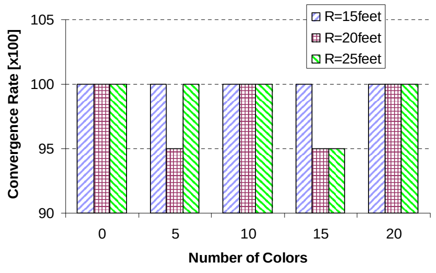


Figure 20: Convergence error

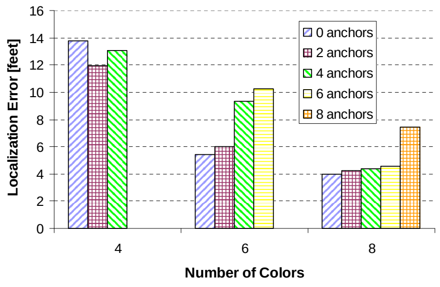


Figure 21: Localization error vs. number of colors

5.2.2 *Localization Error vs. Coloring Space Size*

In this experiment we investigate the effect of the number of colors on the localization accuracy. For this, we randomly assign colors from a pool of a given size, to the sensor nodes. We then execute the localization algorithm, which uses the empirical data. The algorithm is run for three different radio ranges: 15, 20 and $25ft$, to investigate its influence on the localization error.

The results are depicted in Figure 18 (localization error) and Figure 19 (percentage of nodes correctly localized). As shown, for an estimate of $20ft$ for the radio range (as predicted by our Radio Model) we obtain the smallest localization errors, as small as $2ft$, when enough colors are used. Both Figures 18 and 19 confirm our intuition that a larger number of colors available significantly decrease the error in localization.

The well known fact that relaxation algorithms do not always converge, was observed during our experiments. The percentage of successful runs (when the algorithm converged) is depicted in Figure 20. As shown, in several situations, the algorithm failed to converge (the algorithm execution was stopped after 100 iterations per node). If the algorithm does not converge in a predetermined number of steps, it will terminate and the label with the highest probability will provide the identity of the node. It is very probable that the chosen label is incorrect, since the probabilities of some of labels are constantly changing (with each iteration).The convergence of relaxation based algorithms is a well known issue.

5.2.3 *Localization Error vs. Color Uniqueness*

As mentioned in the Section 3.3.1, a unique color gives a sensor node the statute of an anchor. A sensor node that is an anchor can unequivocally be identified through the Image Processing module. In this section we investigate the effect unique colors have on the localization accuracy. Specifically, we want to experimentally verify our intuition that assigning more nodes to a color can benefit the localization accuracy, by enforcing more constraints, as opposed to uniquely assigning a color to a single node.

For this, we fix the number of available colors to either 4, 6 or 8 and vary the number of nodes that are given unique colors, from 0, up to the maximum number of colors (4, 6 or 8). Naturally, if we have a maximum number of colors of 4, we can assign at most 4 anchors. The experimental results are depicted in Figure 21 (localization error) and Figure 22 (percentage of sensor node correctly localized). As expected, the localization accuracy increases with the increase in the number of colors available (larger coloring space). Also, for a given size of the coloring space (e.g., 6 colors available), if more colors are uniquely assigned to sensor nodes then the localization accuracy decreases. It is interesting to observe that by assigning colors uniquely to nodes, the benefit of having additional colors is diminished. Specifically, if 8 colors are available and all are assigned uniquely, the system would be less accurately localized (error $\approx 7ft$), when compared to the case of 6 colors and no unique assignments of colors ($\approx 5ft$ localization error).

The same trend, of a less accurate localization can be observed in Figure 22, which shows the percentage of nodes correctly localized (i.e., $0ft$ localization error). As shown, if we increase the number of colors that are uniquely assigned, the percentage of nodes correctly localized decreases.

5.2.4 *Localization Error vs. Connectivity*

We collected empirical data for two network deployments with different degrees of connectivity (high and low) in order to assess the influence of connectivity on location accuracy. The results obtained from running our localization algorithm are depicted in Figure 23 and Figure 24. We varied the number of colors available and assigned no anchors (i.e., no unique assignments of colors).

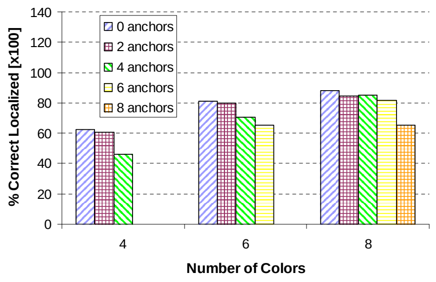


Figure 22: Percentage of nodes correctly localized vs. number of colors

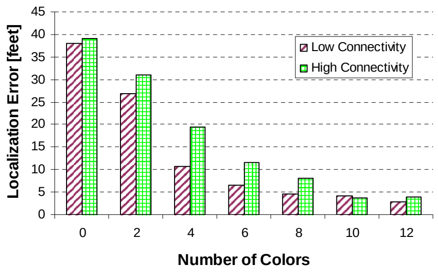


Figure 23: Localization error vs. number of colors

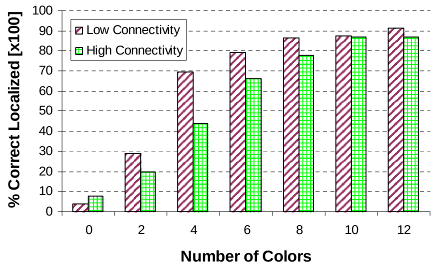


Figure 24: Percentage of nodes correctly localized

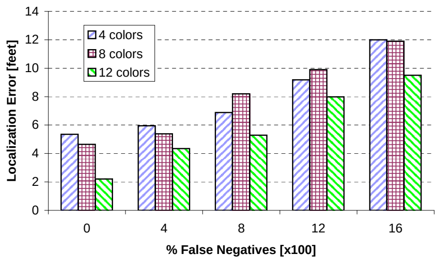


Figure 25: Impact of false negatives on the localization error

In both scenarios, as expected, localization error decrease with an increase in the number of colors. It is interesting to observe, however, that the low connectivity scenario improves the localization accuracy quicker, from the additional number of colors available. When the number of colors becomes relatively large (twelve for our 26 sensor node network), both scenarios (low and high connectivity) have comparable localization errors, of less that $2ft$. The same trend of more accurate location information is evidenced by Figure 24 which shows that the percentage of nodes that are localized correctly grows quicker for the low connectivity deployment.

5.3 Localization Error vs. Image Processing Errors

So far we investigated the sources for error in localization that are intrinsic to the Node ID Matching component. As previously presented, luminous objects can be mistakenly detected to be sensor nodes during the location detection phase of the Image Processing module. These false positives can be eliminated by the color recognition procedure of the Image Processing module. More problematic are false negatives (when a sensor node does not reflect back enough light to be detected). They need to be handled by the localization algorithm. In this case, the localization algorithm is presented with two sets of nodes of different sizes, that need to be matched: one coming from the Image Processing (which misses some nodes) and one coming from the network, with the connectivity information (here we assume a fully connected network, so that all sensor nodes report their connectivity information). In this experiment we investigate how Image Processing errors (false negatives) influence the localization accuracy.

For this evaluation, we ran our localization algorithm with empirical data, but dropped a percentage of nodes from the list of nodes detected by the Image Processing algorithm (we artificially introduced false negatives in the Image Processing). The effect of false negatives on localization accuracy is depicted in Figure 25. As seen in the figure if the number of false negatives is 15%, the error in position estimation doubles when 4 colors are available. It is interesting to observe that the scenario when more colors are available (e.g., 12 colors) is being affected more drastically than the scenario with less colors (e.g., 4 colors). The benefit of having more colors available is still being maintained, at least for the range of colors we investigated (4 through 12 colors).

5.4 Localization Time

In this section we look more closely at the duration for each of the four proposed relaxation techniques and two combinations of them: color-connectivity and color-time. We assume that 50 unique color filters can be manufactured, that the sensor network is deployed from $2,400ft$ (necessary for the time-constrained relaxation) and that the time required for reporting connectivity grows linearly, with an initial reporting period of 160sec, as used in a real world tracking application [1]. The localization duration results, as presented in Table 1, are depicted in Figure 26.

As shown, for all practical purposes the time required by the space constrained relaxation techniques is 0sec. The same applies to the color constrained relaxation, for which the localization time is 0sec (if the number of colors is sufficient). Considering our assumptions, only for a network of size 50 the color constrained relaxation works. The localization duration for all other network sizes (100, 150 and 200) is infinite (i.e., unique color assignments to sensor nodes can not be made, since only 50 colors are unique), when only color constrained relaxation is used. Both the connectivity constrained and time constrained techniques increase linearly with the network size (for the time constrained, the Central device deploys sensor nodes one by one, recording

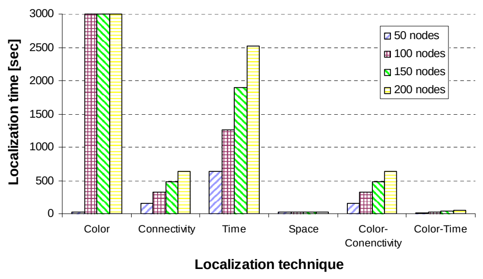


Figure 26: Localization time for different label relaxation schemes

an image after the time a sensor node is expected to reach the ground).

It is interesting to notice in Figure 26 the improvement in the localization time obtained by simply combining the color and the connectivity constrained techniques. The localization duration in this case is identical with the connectivity constrained technique.

The combination of color and time constrained relaxations is even more interesting. For a reasonable localization duration of 52seconds a perfect (i.e., $0ft$ localization error) localization system can be built. In this scenario, the set of sensor nodes is split in batches, with each batch having a set of unique colors. It would be very interesting to consider other scenarios, where the strength of the space constrained relaxation (0sec for any sensor network size) is used for improving the other proposed relaxation techniques. We leave the investigation and rigorous classification of such technique combination for future work.

5.5 System Range

In this section we evaluate the feasibility of the StarDust localization framework when considering the realities of light propagation through the atmosphere.

The main factor that determines the range of our system is light scattering, which redirects the luminance of the source into the medium (in essence equally affecting the luminosity of the target and of the background). Scattering limits the visibility range by reducing the apparent contrast between the target and its background (approaches zero, as the distance increases). The apparent contrast C_r is quantitatively expressed by the formula:

$$C_r = (N_r^t - N_r^b)/N_r^b \tag{10}$$

where N_r^t and N_r^b are the apparent target radiance and apparent background radiance at distance r from the light source, respectively. The apparent radiance N_r^t of a target at a distance r from the light source, is given by:

$$N_r^t = N_a + \frac{I\rho_t e^{-2\sigma r}}{\pi r^2} \tag{11}$$

where I is the intensity of the light source, ρ_t is the target reflectance, σ is the spectral attenuation coefficient ($\approx 0.12km^{-1}$ and $\approx 0.60km^{-1}$ for a clear and a hazy atmosphere, respectively) and N_a is the radiance of the atmospheric backscatter, and it can be expressed as follows:

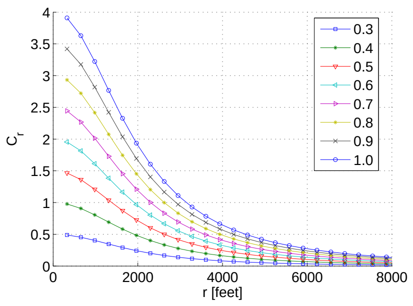


Figure 27: Apparent contrast in a clear atmosphere

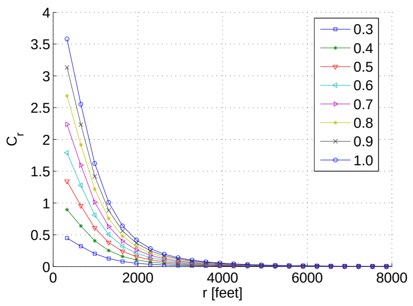


Figure 28: Apparent contrast with haze

$$N_a = \frac{G\sigma^2 I}{2\pi} \int_{0.02\sigma r}^{2\sigma r} \frac{e^{-x}}{x^2}\,dx \tag{12}$$

where $G = 0.24$ is a backscatter gain. The apparent background radiance N_r^b is given by formulas similar with Equations 11 and 12, where only the target reflectance ρ_t is substituted with the background reflectance ρ_b. It is important to remark that when C_r reaches its lower limit, no increase in the source luminance or receiver sensitivity will increase the range of the system. From Equations 11 and 12 it can be observed that the parameter which can be controlled and can influence the range of the system is ρ_t, the target reflectance.

Figures 27 and 28 depict the apparent contrast C_r as a function of the distance r for a clear and for a hazy atmosphere, respectively. The apparent contrast is investigated for reflectance coefficients ρ_t ranging from 0.3 to 1.0 (perfect reflector). For a contrast C of at least 0.5, as it can be seen in Figure 27 a range of approximately $4,500ft$ can be achieved if the atmosphere is clear. The performance dramatically deteriorates, when the atmospheric conditions are problematic. As shown in Figure 28 a range of up to $1,500ft$ is achievable, when using highly reflective CCR components.

While our light source (3 million candlepower) was sufficient for a range of a few hundred feet, we remark that there exist commercially available light sources (20 million candlepower) or military (150 million candlepower [27]), powerful enough for ranges of a few thousand feet.

6. STARDUST SYSTEM OPTIMIZATIONS

In this section we describe extensions of the proposed architecture that can constitute future research directions.

6.1 Chained Constraint Primitives

In this paper we proposed four primitives for constraint-based relaxation algorithms: color, connectivity, time and space. To demonstrate the power that can be obtained by combining them, we proposed and evaluated one combination of such primitives: color and connectivity. An interesting research direction to pursue could be to chain more than two of these primitives. An example of such chain is: color, temporal, spatial and connectivity. Other research directions could be to use voting scheme for deciding which primitive to use or assign different weights to different relaxation algorithms.

6.2 Location Learning

If after several iterations of the algorithm, none of the label probabilities for a node n_i converges to a higher value, the confidence in our labeling of that node is relatively low. It would be interesting to associate with a node, more than one label (implicitly more than one location) and defer the label assignment decision until events are detected in the network (if the network was deployed for target tracking).

6.3 Localization in Rugged Environments

The initial driving force for the StarDust localization framework was to address the sensor node localization in extremely rugged environments. Canopies, dense vegetation, extremely obstructing environments pose significant challenges for sensor nodes localization. The hope, and our original idea, was to consider the time period between the aerial deployment and the time when the sensor node disappears under the canopy. By recording the last visible position of a sensor node (as seen from the aircraft) a reasonable estimate of the sensor node location can be obtained. This would require that sensor nodes posses self-righting capabilities, while in mid-air. Nevertheless, we remark on the suitability of our localization framework for rugged, non-line-of-sight environments.

7. CONCLUSIONS

StarDust solves the localization problem for aerial deployments where passiveness, low cost, small form factor and rapid localization are required. Results show that accuracy can be within $2ft$ and localization time within milliseconds. StarDust also shows robustness with respect to errors. We predict the influence the atmospheric conditions can have on the range of a system based on the StarDust framework, and show that hazy environments or daylight can pose significant challenges.

Most importantly, the properties of StarDust support the potential for even more accurate localization solutions as well as solutions for rugged, non-line-of-sight environments.

8. REFERENCES

[1] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh, "An energy-efficient surveillance system using wireless sensor networks," in *MobiSys*, 2004.

[2] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton, "Sensor network-based countersniper system," in *SenSys*, 2004.

[3] A. Arora, P. Dutta, and B. Bapat, "A line in the sand: A wireless sensor network for trage detection, classification and tracking," in *Computer Networks*, 2004.

[4] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler, "An analysis of a large scale habitat monitoring application," in *ACM SenSys*, 2004.

[5] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A wireless sensor network for structural monitoring," in *ACM SenSys*, 2004.

[6] A. Savvides, C. Han, and M. Srivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Mobicom*, 2001.

[7] N. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Mobicom*, 2000.

[8] M. Broxton, J. Lifton, and J. Paradiso, "Localizing a sensor network via collaborative processing of global stimuli," in *EWSN*, 2005.

[9] P. Bahl and V. N. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," in *IEEE Infocom*, 2000.

[10] N. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Mobile-assisted topology generation for auto-localization in sensor networks," in *IEEE Infocom*, 2005.

[11] P. N. Pathirana, A. Savkin, S. Jha, and N. Bulusu, "Node localization using mobile robots in delay-tolerant sensor networks," *IEEE Transactions on Mobile Computing*, 2004.

[12] C. Savarese, J. M. Rabaey, and J. Beutel, "Locationing in distribued ad-hoc wireless sensor networks," in *ICAASSP*, 2001.

[13] M. Maroti, B. Kusy, G. Balogh, P. Volgyesi, A. Nadas, K. Molnar, S. Dora, and A. Ledeczi, "Radio interferometric geolocation," in *ACM SenSys*, 2005.

[14] K. Whitehouse, A. Woo, C. Karlof, F. Jiang, and D. Culler, "The effects of ranging noise on multi-hop localization: An empirical study," in *IPSN*, 2005.

[15] Y. Kwon, K. Mechitov, S. Sundresh, W. Kim, and G. Agha, "Resilient localization for sensor networks in outdoor environment," UIUC, Tech. Rep., 2004.

[16] R. Stoleru and J. A. Stankovic, "Probability grid: A location estimation scheme for wireless sensor networks," in *SECON*, 2004.

[17] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices," *IEEE Personal Communications Magazine*, 2000.

[18] T. He, C. Huang, B. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-Free localization schemes in large scale sensor networks," in *ACM Mobicom*, 2003.

[19] R. Nagpal, H. Shrobe, and J. Bachrach, "Organizing a global coordinate system from local information on an ad hoc sensor network," in *IPSN*, 2003.

[20] D. Niculescu and B. Nath, "Ad-hoc positioning system," in *IEEE GLOBECOM*, 2001.

[21] R. Stoleru, T. He, J. A. Stankovic, and D. Luebke, "A high-accuracy low-cost localization system for wireless sensor networks," in *ACM SenSys*, 2005.

[22] K. Römer, "The lighthouse location system for smart dust," in *ACM/USENIX MobiSys*, 2003.

[23] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE JRA*, 1987.

[24] C. L. Archer and M. Z. Jacobson, "Spatial and temporal distributions of U.S. winds and wind power at 80m derived from measurements," *Geophysical Research Jrnl.*, 2003.

[25] "Team for advanced flow simulation and modeling." [Online]. Available: http://www.mems.rice.edu/TAFSM/RES/

[26] K. Stein, R. Benney, T. Tezduyar, V. Kalro, and J. Leonard, "3-D computation of parachute fluid-structure interactions - performance and control," in *Aerodynamic Decelerator Systems Conference*, 1999.

[27] Headquarters Department of the Army, "Technical manual for searchlight infrared AN/GSS-14(V)1," 1982.

SIGF: A Family of Configurable, Secure Routing Protocols for Wireless Sensor Networks

Anthony D. Wood Lei Fang
John A. Stankovic
Department of Computer Science
University of Virginia
{wood—leifang—stankovic}@cs.virginia.edu

Tian He
Dept. of Computer Science and Engineering
University of Minnesota
tianhe@cs.umn.edu

Abstract

As sensor networks are deployed in adversarial environments and used for critical applications such as battlefield surveillance and medical monitoring, security weaknesses become a big concern. The severe resource constraints of WSNs give rise to the need for *resource bound security* solutions.

In this paper we present SIGF (Secure Implicit Geographic Forwarding), a configurable secure routing protocol family for wireless sensor networks that provides "good enough" security and high performance. By avoiding or limiting shared state, the protocols prevent many common attacks against routing, and contain others to the local neighborhood.

SIGF makes explicit the tradeoff between security provided and state which must be stored and maintained. It comprises three protocols, each forming a basis for the next: SIGF-0 keeps no state, but provides probabilistic defenses; SIGF-1 uses local history and reputation to avoid attackers; and SIGF-2 uses neighborhood-shared state to provide stronger security guarantees.

Our performance evaluation shows that SIGF achieves high packet delivery ratios with low overhead and end-to-end delay. We evaluate the security of SIGF protocols under various security attacks and show that it effectively contains the damage from compromised nodes and defends against black hole, selective forwarding, Sybil, and some denial of service attacks.

1. Introduction

Security is critical for many wireless sensor network applications such as battlefield surveillance, medical monitoring, and emergency response. However, many security mechanisms developed for the Internet or ad-hoc networks cannot be applied directly to wireless sensor networks (WSNs) due to their limited resources in computation, memory, communication bandwidth, and energy.

The severe resource constraints of WSNs give rise to the need for *resource bound security* solutions. There are at least two interesting aspects of this concept. First, individual security mechanisms must be efficient in memory, computation, energy and bandwidth. For example, certain cryptographic schemes are inappropriate because ciphertext message expansion results in costly memory, bandwidth and energy use. Second, the resource consumption of all security mechanisms installed together at a node must not exceed the amount of resources allocated for security and they cannot degrade performance to an unacceptable level during normal operation nor when an attack is underway.

It is not possible in today's state of the art to include strong security mechanisms for *each* of the services at a node such as medium access control, routing, localization, time synchronization, power management, sensing, and group management. Consequently, even if a secure (to a wide variety of attacks) routing protocol is implemented, it may suffer from low efficiency and would not protect against attacks on the other services.

Our approach for resource bound security is to have minimal *active* security protection. This results in very high performance and minimal resource consumption when no attacks are underway. Then upon detecting an attack or if the system designers expect increased threats, the appropriate security mechanism is activated. The result is not 100% security protection—but *good enough* security, activated at the right time. This general approach makes it possible to have high performance and to react to current security attacks, and is even more evolvable to new attacks than approaches that fix a set of solutions into a node.

In this paper we present Secure Implicit Geographic Forwarding (SIGF), a family of configurable secure routing protocols that follow this general approach. For a complete WSN solution similar families of protocols would be required for each of the other services.

SIGF is based on IGF [1], a nondeterministic Network/MAC hybrid routing protocol that is completely stateless. This allows it to handle network dynamics effortlessly, and intrinsically limits the effects of a compromised node to a local area. There are no routing tables to corrupt, since forwarding decisions are made as late as possible—when a packet is ready to transmit. Nevertheless, it is susceptible in the local neighborhood to a simple CTS rushing attack.

SIGF comprises three protocols which extend IGF and populate the gap between pure statelessness and traditional shared-state security. SIGF-0 keeps no state, but uses nondeterminism and candidate sampling to achieve high packet delivery ratios probabilistically. SIGF-1 keeps local state, building reputations for its neighbors to aid in next-hop selection. SIGF-2 uses state shared with neighbors to provide the strongest defense against attack, yet at the greatest cost. Each protocol encompasses the features of the previous, layering additional mechanisms to defend against more sophisticated attacks. The layered family of protocols allows a network to activate only the protections currently necessary, and to change to stronger ones only if they are warranted.

We evaluate the performance of each protocol by simulating with no attacks, and with black hole, selective forwarding, Sybil, and denial of service attacks. We show that each protocol represents a tradeoff between state and security, and that despite keeping no state, SIGF-0 performs well.

We make several contributions in this work. First, we show that even with no security countermeasures, the base protocol IGF has desirable attack containment properties, but nevertheless falls to several attacks that completely disrupt communication in a local neighborhood. Then we present the design and evaluation of SIGF, a secure routing protocol family built upon IGF. We show that the stateless SIGF-0 protocol maintains 45% packet delivery ratio (PDR) under black hole attack from a single node, and that the reputation-based SIGF-1 achieves 83% PDR under Sybil attack. To the best of our knowledge, SIGF is the first configurable routing protocol for WSNs that makes explicit the tradeoffs between resources required and security provided, and enables resource bound security that is both efficient and effective.

It is possible that some WSNs require much stronger security than what our dynamic approach offers. However, no perfect solution exists—nor is likely to exist on severely resource constrained devices. Our approach, as exemplified by the SIGF routing protocols presented in this paper, can significantly improve security (not make it absolute), allow operation in the presence of attacks, and support a high performance system.

The rest of this paper is organized as follows. In the next section we briefly review our foundational routing protocol, IGF. Then Section 3 describes system assumptions and routing attacks on IGF. Section 4 presents SIGF, our secure routing protocol family. In Section 5 we present our experiments and a detailed evaluation of the protocols under various attacks. Finally, we discuss related work in Section 6 and then conclude.

2. Implicit Geographic Forwarding

Our foundational routing protocol is Implicit Geographic Forwarding (IGF), which is completely stateless, without dependence on knowledge of the network topology or the presence/absence of any other nodes [1]. It makes nondeterministic routing decisions, implicitly allowing receiving nodes to determine a packet's next-hop at transmission time. IGF couples the routing and MAC components into a single integrated Network/MAC protocol. It identifies the best forwarding candidate during MAC-layer handshaking at the instant a packet is sent.

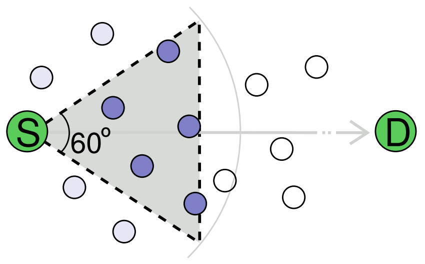


Figure 1. Forwarding area for message sender S.

Figure 1 presents an example topology, where source node S transmits a message toward D. Routing is integrated with the RTS/CTS hand-shake of MACA/802.11 [9] DCF MAC protocols. Overhead from the small control messages is acceptable for the stateless benefits they provide, and are negligible in WSNs that carry moderate or large packet loads of aggregated data.

The communication handshake for this example topology is shown in Figure 2. It begins when the sender S's NAV timer is zero and it carrier senses an idle channel for DIFS time. Having verified that the channel is free, S broadcasts an Open RTS (ORTS) containing its location S and destination D.

Neighbors are eligible to forward the message if they are within a 60° sextant centered on the direct line from the sender to the destination (the forwarding area). We call the nodes in the forwarding area *candidate nodes*. Such nodes set a CTS Response timer inversely proportional to a weighted sum of their distance from the sender, remaining energy, and perpendicular distance to a line from the sender to the destination. This favors the nodes that are more desirable for forwarding.

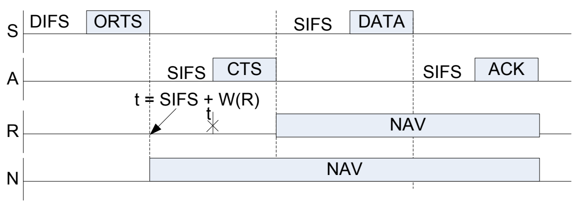


Figure 2. IGF handshake timeline.

On the expiry of a node's CTS response timer, it responds with a CTS packet, and the data is transferred from the Open RTS sender in a DATA message. The valid duration of the CTS timers is called the CTS-response window. Ideally, other candidate nodes can hear the CTS (by virtue of lying inside the sextant) and cancel their timers before the end of the window. Therefore, in IGF, a single node with the shortest CTS response timer responds to the ORTS. [1]

Since IGF keeps no routing state information, it provides fault tolerance and is robust under topology changes. It also eliminates expensive communication for routing and neighbor information maintenance, and the associated routing latency. Using the concept of *lazy binding*, the IGF protocol defers next-hop selection until the packet forwarding operation actually happens at a sending node. Lazy binding dramatically reduces the chance that packets are forwarded to a node that fails, moves out of range, or transits to a sleep state, and also enables the use of recently awakened or newly arriving nodes.

Compared with other established protocols for sensor and ad-hoc networks (such as GPSR [13], DSR [10], and LAR [15]), IGF achieves up to a ten-fold increase in delivery ratio and significantly reduces both end-to-end delay and control overhead. It is therefore a good protocol to serve as a foundation for secure routing.

[1] IGF deals with network voids by shifting the forwarding sextant to the side and retrying [1]. SIGF inherits this mechanism, which we do not discuss further in this paper.

IGF has no routing tables, so it naturally confines the attacker's impact to the neighborhood and prevents attacks such as spoofing, altering, or replacing routing information. This is a significant advantage over link-state and distance vector routing protocols, which must carefully manage updates and route requests to avoid contamination by attackers.

Unfortunately, a single attacker can completely disrupt routing for all of its neighbors. For example, the attacker arranges for itself to be chosen as the next-hop relay simply by sending an immediate CTS message upon receiving an ORTS. When the attacker gets the DATA, it replies with an ACK, but drops the DATA packet. The packet delivery ratio becomes zero—a simple attack, but devastating in the local neighborhood.

We designed SIGF to secure routing in the local neighborhood while preserving the performance and attack containment properties of IGF. Section 4 presents our secure routing protocol family in detail.

3. Assumptions and Attacks

Routing is an essential service for enabling communication in sensor networks, and is therefore potentially the target of many different attacks. First, we identify our assumptions about the system. We review the general classes of attacks on sensor network routing, then focus on attack mechanisms specific to our protocol in the next section.

3.1 System Assumptions

We assume that radio links are insecure, i.e., attackers may eavesdrop on radio transmissions, inject messages, and record and later replay messages. If an attacker is able to interact with the routing protocol, it can also drop messages for which it is responsible. Attackers possess hardware capabilities similar to that of legitimate nodes, and wireless transmissions use the same power levels.

All nodes know their own location, and may additionally know that of their neighbors (in SIGF-1 and SIGF-2). Nodes know the location of important resources, like base stations, and use it for geographic routing.

We do not require time synchronization among nodes. For SIGF-0 and SIGF-1, no shared keys are required between nodes in the network. SIGF-2 assumes the presence of pairwise-shared keys in the neighborhood, which may be fulfilled by many different key distribution schemes in the literature [4, 2, 25]. Nodes trust their own clocks, measurements, and storage.

3.2 Routing Attacks

Karlof and Wagner [12] and others [22, 18] have systematically studied attacks on routing protocols. We summarize these attacks below, noting whether they are applicable to IGF (and therefore to SIGF). Then we discuss those attacks which are not obviously thwarted in greater detail.

1. *Routing state corruption.* By spoofing, altering, or replaying routing information, attackers are able to create routing loops, attract or redirect network traffic, increase end-to-end delay, etc. IGF keeps no information, and SIGF keeps only locally generated information.
2. *Wormholes.* In this attack, an adversary tunnels messages received in one part of the network over a low latency link and replays them elsewhere. Since IGF chooses the next-hop dynamically, a wormhole does not cause disruption when it ceases to operate.
3. *HELLO floods.* An attacker convinces nodes in the network that the attacker is a neighbor by broadcasting HELLO messages with high energy. As with the wormhole attack, dynamic routing in IGF prevents disruption by a HELLO flood.
4. *Black holes.* In a black hole attack, an adversary or compromised node lures nearly all the traffic from a particular area through itself, where the messages are dropped. We further discuss this attack below.
5. *Selective forwarding.* Attackers selectively forward packets instead of faithfully forwarding all received packets or completely dropping all packets. At one end of the spectrum, messages are rarely dropped. At the other end is a black hole attack. We group this attack with the black hole attack since its mechanism is the same and consider its impact on IGF.
6. *Sybil attack.* In the Sybil attack, a malicious node behaves as if it were a larger number of nodes by impersonating other nodes or simply by claiming false identities. We further discuss this attack below.
7. *Denial of Service.* Most attacks result in a denial of service of some sort, but this moniker is usually reserved for attacks that waste resources or disrupt service in a way that far exceeds the effort required of an attacker. Message amplification and jamming are general examples. We consider specific mechanisms for mounting this attack on IGF below.

In an insider attack, a compromised node uses any means available to legitimate nodes to disrupt the protocol or perform one of the other specific attacks listed above. All state, including keys possessed by the node, may be used by the attacker.

Since IGF keeps no routing tables, it prevents attacks such as state corruption, wormholes, and HELLO floods. Further, the impact of attacks is limited to the local area, since routing is fully distributed and independent from hop to hop. IGF and SIGF do not trust neighboring nodes to behave correctly, so they are resistant to attacks from outsiders and insiders alike.

The main attacks available to an adversary are to create a black hole, pose as multiple identities (Sybil attack), or disrupt the routing protocol through denial of service attacks. We describe specific mechanisms for performing these attacks on IGF in the next sections. When we describe and evaluate SIGF in Sections 4–5, we focus particularly on its resilience to these attacks.

3.2.1 Black Hole / Selective Forwarding Attack

Within the local neighborhood, the easiest way for an attacker to create a black hole is to manage to always be selected by neighbors as the next hop, whether this is proper, or not.

In the *CTS rushing attack*, an attacker exploits the cooperative nature of IGF's next-hop selection. When an Open RTS (ORTS) message is received, neighbors set timers proportional to their desirability as forwarding candidates. The attacker disregards this mechanism and always replies immediately with a CTS, volunteering to forward the packet. Once selected as the next relay, the attacker may modify, totally drop (black hole attack) or selectively forward the DATA message. This attack is very effective against IGF, easy to perform, and requires moderate power consumption, as it is completely reactive.

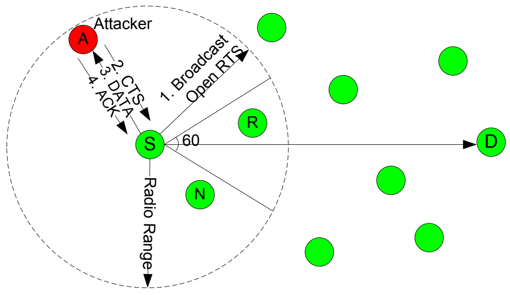


Figure 3. CTS Rushing Attack by A against S.

Figure 3 shows how this attack works. When attacker A overhears an ORTS message, it sends a CTS message, whether it is in the forwarding area or not. Other nodes overhear the CTS from the attacker and abort the protocol. Unsuspecting ORTS senders in the neighborhood of the attacker always choose to send their messages into the black hole created by A.

3.2.2 Sybil Attack

In a Sybil attack, an attacker illegitimately claims to be multiple nodes by sending messages with different identities and locations. Its additional identities are virtual Sybil nodes. Without cryptographic authentication, a receiver of a message cannot determine the true identity of its originator, and does not know how many of the claimed identities are truly unique. Our foundational routing protocol IGF is vulnerable to Sybil attack because it does not maintain any neighborhood state with which to validate the identities.

Identity and Location. A Sybil node can either fabricate a new identity, or steal an identity from a legitimate node [18]. In our experiments, an attacker creates several Sybil nodes surrounding its true location and assigns each either a random or fixed location.

Communication. We assume Sybil nodes can communicate directly with legitimate nodes in the following way. When a legitimate node sends a message to a Sybil node, the attacker overhears the message. Likewise, messages sent from Sybil nodes are actually from the attacker, but with the proper identity enclosed.

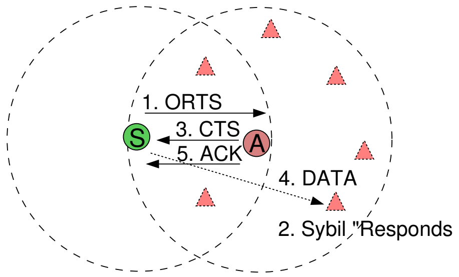


Figure 4. Node A performs a Sybil attack against S.

Communication with a Sybil node is illustrated in Figure 4. After receiving an ORTS message, the attacker sends a CTS addressed from one of the Sybil nodes. Once the Sybil node is selected as the next relay, the attacker overhears and acknowledges the DATA. It can then drop, tamper, or forward the DATA in a black hole or selective forwarding attack.

3.2.3 Denial of Service Attack

The goal of this type of attack is to deny service to the nearby nodes in a manner that is less intrusive and costly than jamming. The attacker partially executes the IGF protocol to cause nearby nodes to waste energy transmitting messages, waste time waiting on completion of the protocol, or prematurely abort the protocol. We describe two specific attacks which cause denial of service by recording and replaying legitimate messages.

In an *ORTS replay* attack, a node captures an overheard ORTS message and subsequently replays it repeatedly. Each time it is replayed, neighbors of the attacker respond with CTS messages and wait for data exchange. The wireless channel cannot be used in this local neighborhood for legitimate traffic during the CTS collection window.

In a *CTS replay* attack, the old CTS message falsely causes other eligible receivers in IGF to abort the protocol (cancel their CTS response timers). The ORTS sender selects an unsuspecting or absent node (the originator of the captured CTS) as the next hop. The sender transmits the DATA, wasting energy and channel capacity, and then must retry or drop the DATA message when no acknowledgement is forthcoming. A captured ACK could be replayed by the attacker as well, causing the sender to believe the transmission was successful.

This attack is less costly to the attacker than an ORTS replay because it is reactive: the protocol is only disrupted when a neighbor actually tries to send a message.

4. SIGF: Secure IGF

We propose a novel secure routing protocol family, called Secure IGF (SIGF) which keeps the advantages of dynamic binding in IGF, yet provides effective defenses against the attacks discussed above. The protocols provide tradeoffs between security and state maintenance, and configurability that can be adapted at runtime.

The configurability of the SIGF protocol family gives a significant advantage over other more static routing protocols. Some provide no security, while others provide strong guarantees—but at the cost of more assumptions, computation, and communication. These higher costs must be borne even when no attacks are occurring. SIGF protocols can be selected and configured for the security requirements of a particular deployment.

Network planners can select among three classes of security solutions, grouped by the amount of state they keep: no state (SIGF-0), locally generated state (SIGF-1), and pairwise-shared state within the neighborhood (SIGF-2). This choice is currently static, but in the future will be dynamically adjustable.

SIGF-0 is a stateless protocol that maintains no routing information, but provides only probabilistic defenses against attack. SIGF-1 keeps limited information learned from interactions with neighbors. SIGF-2 uses keys and sequence numbers shared among neighbors to provide cryptographic guarantees in routing. Each protocol is a subset of the next. That is, SIGF-1 uses mechanisms from SIGF-0, and likewise SIGF-2 uses some from SIGF-1.

The main weakness of a last-instant dynamic binding approach, as used by IGF, is in the selection of the next-hop relay. Each of these protocols uses different means to prevent or minimize the probability of select-

```
1  if (include destination)
2    ORTS $\leftarrow \langle S, S_{location}, D, D_{location}, FwdArea \rangle$
3  else
4    ORTS $\leftarrow \langle S \rangle$

6  broadcast ORTS message

8  /* Every neighbor N receives ORTS message,
       and if in FwdArea, sets CTS response
       timer proportional to next−hop
       desirability , sending CTS = $\langle N, N_{location} \rangle$
       upon expiry. */

10 $CTS_{candidates} \leftarrow \emptyset$
11 while (collection window open)
12   if (CTS received AND $N_{location} \in FwdArea$)
13     add N to $CTS_{candidates}$

15 choose $C \in CTS_{candidates}$ for next hop
16 send DATA to C
```

Algorithm 1. SIGF-0 next-hop selection for message from current node S to ultimate destination D.

ing an attacker as the relay, while achieving high packet delivery rates with low delay and overhead.

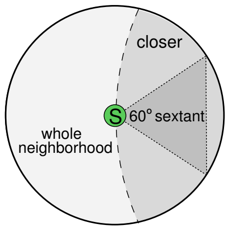
In the following sections we present each protocol in turn.

4.1 SIGF-0: Stateless Secure IGF

SIGF-0 is the basis of the other protocols in the Secure IGF family. Without keeping forwarding history or information about neighbors, it chooses the next-hop relay non-deterministically and dynamically. This lessens, but does not eliminate, the chance of selecting an attacker in the neighborhood.

The logic for sending a message from source S to destination D is shown in Algorithm 1. The ORTS message (as described in Section 2) is constructed in Lines 1–4 and broadcast to the one-hop neighbors in Line 6. Neighbors of S that receive the ORTS and which are in the forwarding area start CTS response timers. Upon timer expiry, a node sends a CTS response that includes its own location. In Lines 10–13, node S collects CTS responses until the collection window closes. Then a candidate C is chosen among the responders and the DATA is relayed to node C.

The algorithm is configurable in four dimensions, each of which is described here. Each is annotated with the list of options and the line number in Algorithm 1 where it appears.

1. **Forwarding Area** $\in$ {60° sextant, closer, whole neighborhood} *Line 2*
   In the foundational routing protocol IGF, a 60° sextant toward the destination is always used as the forwarding area. This gives some assurance that CTS responders can overhear each other and cancel their timers.
   

   In the presence of multiple neighboring adversaries, however, this sextant may not provide enough responses from which to select. Low-density deployments allow attackers to fill the CTS candidate set to the exclusion of legitimate forwarders. SIGF-0 allows the use of larger forwarding areas, since for a given number of attackers this increases the probability of selecting a legitimate node. In addition to the 60° sextant, any node that is closer to the destination than the sender may respond, or all neighbors may respond.

   Performance is affected both by allowing messages to take longer paths, and by lengthening the collection window to accomodate greater CTS candidates. However, this is offset by the ease with which multiple attackers may capture forwarding when the narrower sextant is used. Allowing more neighbors to be considered in the forwarding area does not automatically cause worse performance when there are no attacks, since correct nodes still respond according to their desirability for forwarding (as described in Section 2).
2. **Collection Window** $\in$ {one responder, fixed multiple, dynamically lengthened} *Line 11*
   SIGF-0 collects one *or more* CTS messages before choosing the next-hop relay among them. IGF closes the collection window immediately upon receiving the first CTS, but this is vulnerable to the CTS rushing attack presented earlier. The attacker disregards the correct response delay and responds first, creating a black hole in the neighborhood. Still, this option is available in SIGF-0 since it provides best performance (lowest delay and overhead) when no attacker is present.

   By allowing a longer collection window, SIGF-0 collects more CTS messages before selecting a relay. The ORTS sender waits a fixed amount of time, storing CTS responses. One is chosen according to the criteria given in the next part. A fixed-length window gives predictability and constant cost, and allows CTS response timers to be scaled *a priori* to avoid unnecessary contention during the window. A flag is included in the ORTS to prevent CTS responders from aborting the protocol when another CTS is overheard.

If not enough CTS responses are received, the window may optionally be extended dynamically. At a greater cost in delay, this allows the ORTS sender to collect enough responses to give better assurance that an attacker is not chosen.

3. **Forwarding Candidate Choice** $\in$ {first, by priority, random, multiple} *Line 15*

   Given a set of forwarding candidates collected during the window ($CTS_{candidates}$ in Algorithm 1), this parameter determines how one is chosen to be the next-hop relay. IGF always chooses the first responder, which is vulnerable to the CTS rushing attack. We allow this option since it is compatible with IGF and because it is most efficient when no attackers are present.

   Selecting by priority means choosing the node that makes the most progress toward the ultimate destination of the message. For other protocols, this is extended to include other criteria. This option has the advantage of minimizing path dilation when no attacker is present.

   Random selection is robust against a wide variety of attackers, since it does not give credence to the location information contained in the CTS. The larger the pool of forwarding candidates, the less likely that a neighboring attacker performing a CTS rushing attack or masquerading as a legimitate node is chosen. Performance suffers, however, since progress toward the destination is erratic. Compared with the impact of a black hole attack, this is most likely an acceptable tradeoff.

   More than one candidate may be chosen to relay messages along multiple paths from sender to receiver. This redundancy lessens the impact of attackers met along the way, though if a fixed number of attackers is present, the higher cost may be justified by its effectiveness.

4. **Omit Location** $\in$ {yes, no} *Line 1*

   Even when selecting among multiple responses in the collection window, an attacker can manipulate the choice if it is made by priority. Since the ORTS includes the ultimate destination, an attacker can fabricate an optimal location for inclusion in its CTS to maximize its chances of being selected.

   An option to omit the source and destination locations in the ORTS message mitigates this threat. In this case, the neighbors of S cannot determine whether they are in the forwarding area, nor how close they lie to the line $\overline{SD}$. Therefore, all neighbors respond by setting timers proportional to their remaining energy only. The ORTS sender then chooses the relay according to the previous configuration setting.

   When the DATA message is relayed to the selected node, it must contain the destination's location to enable subsequent routing.

   Omitting the destination does not eliminate the threat of a black hole attack, since an adversary may infer the ultimate destination from a stream of messages using traffic analysis. We do not consider that attack in this paper.

Note that during protocol operation, both a sender and its neighbors (forwarding candidates) retain some state. It is transient, however, since it need not be retained after the message is relayed. For this reason we classify SIGF-0 as stateless.

SIGF-0 is compatible with IGF when the forwarding area is a 60° sextant, the collection window is short enough to contain one response, the first response is selected, and the destination is included in the ORTS.

The configuration options presented give SIGF-0 robustness against a black hole caused by CTS rushing. They are similar enough to IGF to allow a smooth, run-time transition between option settings, according to the current attack situation. We are exploring the dynamic transition between settings, and between protocols in future work.

4.2 SIGF-1: Local-State Secure IGF

SIGF-1 builds on the capabilities and operation of SIGF-0, while aiming to further reduce the chances of selecting an attacker as the next-hop relay. By keeping some limited information about its current state and statistics of neighbor performance, a node can also defend against Sybil attacks. This state is summarized by a per-neighbor reputation value that influences the choice of forwarding candidates.

Since the state kept is not shared with neighbors, there is no overhead associated with initialization, synchronization, or repair. By limiting the information to that which can be verified locally, the protocol avoids state corruption attacks. Further, neighborhood dynamics due to mobility, failure, or transient communication are still supported.

We classify state kept in SIGF-1 in three categories: data about the local node, statistics about neighboring nodes, and values derived from both together. Each is presented below.

For the local node, we maintain T, the total number of messages sent by the node to all neighbors. It is used to calculate derived values for each neighbor. Nodes also have a small buffer B in which recently relayed messages are stored.

For each neighbor N among those discovered dynamically (i.e., **neighbor tables are not exchanged**), we keep the following:

1. $\boldsymbol{N_{sent}}$ = number of messages sent to neighbor N for forwarding. It is increased by one each time N is selected as the next-hop relay.
2. $\boldsymbol{N_{forward}}$ = number of messages forwarded by neighbor N on this node's behalf. This is counted by overhearing a message on its retransmission by node N to a downstream node, albeit imperfectly due to collisions and asymmetries.
3. $\boldsymbol{N_{location}}$ = last claimed location of node N in its CTS message.
4. $\boldsymbol{N_{delay}}$ = average delay between relaying a message to node N and overhearing the subsequent relay of the same.

After transmitting a message to a neighboring node, a copy of the message is stored in the message buffer B, along with a timestamp. If the message is overheard on its relay to a downstream node, the difference between the recorded and current times is calculated and the message is flushed from the buffer. $N_{forward}$ and N_{delay} are updated as described above. In case the buffer fills due to message loss or failure to overhear the relay, the oldest message is replaced and the associated N_{delay} is updated with a fixed maximum delay D.

From the data collected during routing, other values are derived which combine to determine a node's reputation. These are also maintained per-neighbor as they are discovered.

5. $\boldsymbol{N_{success}} = \frac{\boldsymbol{N_{forward}}}{\boldsymbol{N_{sent}}}$ = forwarding success ratio, a measure of reliability. Neighbors which always (verifiably) forward messages achieve high success ratios. When first discovered and until the neighbor forwards a message, it is given a neutral initial value of 0.5.
6. $\boldsymbol{N_{fairness}} = \frac{\boldsymbol{T-N_{sent}}}{\boldsymbol{T}}$ = forwarding fairness ratio, a measure of the distribution of relay choices among neighbors. This promotes dispersion of next-hop relay choices among similarly performing neighbors, and reduces the likelihood of selecting an attacker even before its misbehavior is detected. Initial value is 0.5.
7. $\boldsymbol{N_{consistency}}$ = a consistency score based on the variance of neighbor N's claimed locations. When $N_{location}$ changes, the score is decreased additively to penalize nodes which are either moving constantly or lying about their locations. When the claimed location remains the same, a small additive reward is granted, increasing the score. The consistency score saturates so as always to be in the interval $[0, 1]$. A neutral initial value of 0.5 is assigned.

Parameter	Description
α	Forwarding success weight
β	Forwarding fairness weight
γ	Location consistency weight
ζ	Forwarding performance weight
$R_{threshold}$	Reputation threshold

Table 1. System parameters for SIGF-1 to be determined statically by the network designer, or dynamically at runtime.

8. $\boldsymbol{N_{performance}} = \frac{\boldsymbol{D-N_{delay}}}{\boldsymbol{D}}$ = forwarding performance of the neighbor in terms of the maximum delay D, a static system parameter. This favors nodes which are able to quickly relay messages, due to light congestion and correct behavior, and penalizes nodes which are heavily congested or deliberately delay or drop messages. A neutral initial value of 0.5 is assigned.

Each neighbor is assigned a reputation R comprising a weighted linear combination of the above computed values:

$$\begin{aligned} R &= \alpha N_{success} + \beta N_{fairness} \\ &\quad + \gamma N_{consistency} + \zeta N_{performance} \end{aligned} \tag{1}$$

The terms are weighted according to the network designer's choice, with the limitation that all weights must sum to unity. All terms and the computed reputation are in the interval $[0, 1]$. The reputation is not shared externally; it is used only on the local node for ranking forwarding candidates.

SIGF-1 allows the weights to be assigned flexibly so the designer can favor some neighbor properties over others. The weights may also be adjusted dynamically based on current conditions. For example, a high weight for forwarding success ratio α may improve performance during a black hole attack by degrading attackers' reputations more quickly. Table 1 summarizes the configuration parameters of SIGF-1.

SIGF-1 builds upon the stateless algorithm and protocol. All the options described above for SIGF-0 are still available. The Forwarding Area, Omit Location, and Collection Window settings are orthogonal, although a window for only one CTS responder does not provide any real choice of forwarding candidate. The key interaction is the use of reputation for choosing among eligible candidates.

A reputation threshold $R_{threshold}$ is used to cull undesirable relays before applying the Forwarding Candidate Choice policy. All responders with reputations below the threshold are eliminated from consideration. Then the next-hop is chosen depending on the option in use: the *first* (i.e., earliest) responder, a *random* responder, or the responder with highest routing *priority* (based on distance to destination, etc.).

The threshold is a system parameter that allows nodes to avoid wasting energy sending a message to a neighbor with known poor performance, even if it claims to be the best route. The tradeoff is that a high threshold may cause premature routing failure by eliminating too many neighbors from consideration. For this reason, if no responders' reputations are above $R_{threshold}$, we select the node with the highest reputation, even if it is a sub-optimal route. This favors routing success over performance when under attack.

Our experiments show that SIGF-1 effectively defends against the black hole, selective forwarding, and Sybil attacks when the next hop is selected using reputation. When an attacker drops messages, its reputation degrades quickly, as desired.

4.3 SIGF-2: Shared-State Secure IGF

Protection against attacks in SIGF-0 and SIGF-1 is gained by adding nondeterminism to the already dynamic forwarding candidate selection. However, some attacks still result in poor performance, since they go beyond the protections afforded by probabilistic, fully decentralized means.

SIGF-2 addresses this limitation by using state that is shared among neighbors for cryptographic operations. This provides guarantees for authenticity, confidentiality, integrity, and freshness that some other secure routing protocols provide (discussed in Section 6), but in the framework of a protocol family that can also operate without them. It builds upon SIGF-0 and SIGF-1, and inherits their configuration options.

The state required for use of SIGF-2 is described below along with the protocol configuration options. Many key pre-distribution or online key establishment protocols have been proposed, many of which are suitable for supporting this protocol. One example is LEAP [25], which provides both pairwise-shared keys between neighbors and neighborhood-shared keys for broadcast.

Each option is described below, including its shared-state requirements.

1. **Message Authentication** $\in$ {all messages, only DATA, none}

   Authenticating messages cryptographically ensures that they originate from a neighbor with which a node has pre-shared information. This prevents an outside attacker from entering the network and being able to inject arbitrary messages. Using an appropriate key, a message authentication code (MAC) is computed over the header and payload and is appended to the message before transmission. Message integrity is provided by the same mechanism.

   All protocol messages (ORTS, CTS, DATA, ACK) may be authenticated, or only the DATA portion. The latter has lower computation and communication overhead, but does not prevent replay attacks, but may prevent an attacker from hijacking a protocol exchange to insert false data.

   Note that message authentication does not prevent compromised nodes from participating in this or any other protocol. Since they possess all the security information of the original nodes, they may send any authenticated messages that the original nodes could.

   CTS, DATA, and ACK authentication uses a shared key between the ORTS sender and the selected relay. When authenticating the ORTS message, a broadcast key must be used that is shared with all potential forwarding candidates in the neighborhood.

2. **Message Sequencing** $\in$ {yes, no}

   When message sequencing is enabled, protocol messages include a monotonically increasing sequence number s. A receiver accepts a message from neighbor N only if $s > N_{seq}$, the highest sequence number verifiably received from N. This ensures that each message is fresh and prevents an attacker from capturing and replaying old messages. It requires that N_{seq} be stored for each neighbor and updated upon each reception of an authentic message.

   Message sequencing only provides defense against replay attacks if authentication is also in use. Otherwise, an attacker can simply change the sequence number when replaying a message, and it will not be detected at the receiver.

3. **Payload Encryption** $\in$ {yes, no}

   Payload encryption uses a shared key between the ORTS sender and the selected relay to conceal the contents of a DATA message from eavesdropping by attackers. This may also help to thwart traffic analysis based on semantic contents of messages.

The use of authentication and sequencing in SIGF-2 prevents message injection by outsiders, since they do not possess the keys to create valid MACs. Attackers also may not replay ORTS and CTS messages to cause denial of service from spuriously invoking or aborting the protocol. Messages with old sequence numbers are dropped.

SIGF-2 does not by itself prevent compromised nodes from creating a black hole or other attack de-

Protocol	Approach	Corruption	Wormhole	HELLO flood	Black hole	Sybil	Replay DoS
IGF	Dynamic Binding	☑	☑	☑	–	–	–
SIGF-0	Nondeterminism	☑	☑	☑	☑	–	–
SIGF-1	Local Reputation	☑	☑	☑	☑	☑	–
SIGF-2	Cryptography	☑	☑	☑	☑	☑	☑

Table 2. Attacks resisted by IGF and SIGF protocols.

scribed in previous sections. It must be layered atop SIGF-0 and SIGF-1 to retain these defenses.

End-to-end cryptographic protections may be employed at a higher level in the protocol stack. Such mechanisms would affect only the payload of the DATA message transparently to SIGF.

4.4 Discussion and Comparison

SIGF-0 inherits resistance to state corruption, wormhole, and HELLO flood attacks. In addition, it provides robustness against a black hole by CTS rushing. However, its effectiveness is reduced when an attacker creates multiple identities and responds to an ORTS with several CTS messages. Even with random selection from a large window, this greatly increases the chances of selecting the attacker.

SIGF-1 adds resistance to a Sybil attack by exploiting locally generated information about neighbors. The reputation calculation helps to distinguish between the stable, well-behaved legitimate neighbors and the attackers that lie about locations or do not forward packets reliably.

An attacker that replays others' messages can still mount a denial of service attack. To partially address this, we allow SIGF-2 to use state that is shared with its neighbors. Though the overhead is greater, this allows for cryptographic guarantees of authenticity, integrity, freshness, and confidentiality. It does not prevent flooding-type denial of service attacks, but mitigates against attackers using the protocol itself to cause disruption.

Even for authentic messages in SIGF-2, nodes do not completely trust neighbors. Methods of SIGF-0 and SIGF-1 for sampling among multiple candidates and ranking according to reputation are used to limit the impact of a compromised node.

Table 2 summarizes which attacks from Section 3 are addressed by the SIGF protocols.

The SIGF family of protocols is designed to provide several incremental steps between IGF and symmetric-cryptography-based routing protocols. This gives the network designer flexibility to choose a protocol statically based on application security requirements and available resources. They can also be selected dynamically through control logic that remains for future work.

SIGF-0 reduces to emulating IGF operation with the following settings: forwarding in a 60° sextant, a collection window for one CTS, selecting the first one, and not omitting the destination location.

A node using SIGF-0 can dynamically change to SIGF-1 if notified by out-of-band means that attackers are present, or upon detecting degraded performance. Changing back again is as simple as releasing the state collected. SIGF-2 requires that keys be shared *a priori*, and so may not be available for dynamic selection at runtime if the network started completely statelessly. We investigate these issues in future work.

5. Evaluation

To evaluate the performance of our secure routing protocol, we implementd SIGF in GloMoSim, a wireless simulator for sensor, ad hoc, and mobile networks. GloMoSim models the communication architecture from physical-layer bit transmissions, including signal interference and attenuation patterns, up the stack to application-layer traffic loads. Our system parameters are listed in Table 3.

Terrain	150 x 150 meters
Number of Nodes	196
Node Placement	Grid + $\mathcal{N}(0, 16)$ noise
Application	CBR streams
Payload Size	32 bytes
Simulation Length	100 packets, 10 runs
Radio Range	40 meters
Radio Bandwidth	200 kb/s

Table 3. GloMoSim simulation parameters.

For our experiments, we configured a terrain of 150 square meters, with 196 sensor nodes having communication radii of 40 meters. The terrain was subdivided uniformly into 196 cells. A node was placed at the center of each and then perturbed using a Gaussian distribution with standard deviation of four meters. We limited the duration of the CBR streams to 100 packets to emulate the type of traffic expected in low-bandwidth networks, and to avoid swamping initial reputation transients with steady-state behavior. Data shown in the graphs are the mean of ten simulation runs. Figure 5 shows the final node locations and labels for the source S, destination D, and attackers $A1$–4 used in the experiments.

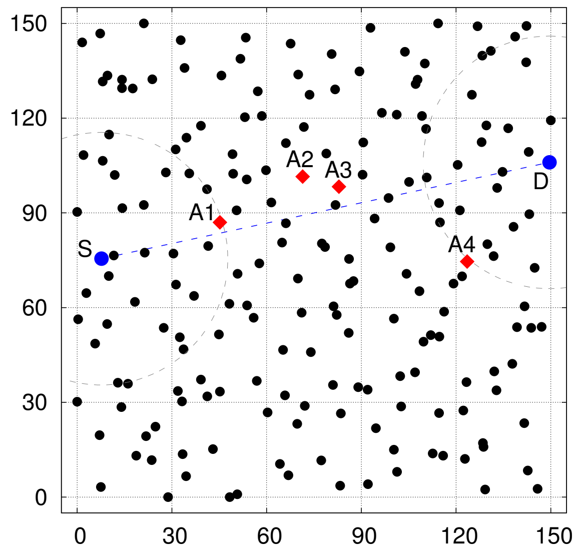


Figure 5. Locations for 196 nodes in GloMoSim 150x150m field. Nodes are first placed uniformly on a grid, then perturbed by a $\mathcal{N}(0, 16)$ distribution.

We evaluated the performance of SIGF-0, SIGF-1, and SIGF-2 under six scenarios. In the base system test, we compare GF [5], DSR [10], IGF, and SIGF without any attacks. Then we evaluate SIGF under black hole, selective forwarding, and Sybil attacks. Denial of service attacks are considered in the last two experiments.

Configurations used for the experiments that follow are shown in Table 4. The labels given there are used in the discussions and legends of figures that follow.

5.1 Base System (No Attacks)

We consider the first set of experiments as a baseline for comparison. It tests many-to-many constant bit rate (CBR) traffic flows, which mimic the periodic point-to-point communication expected in such systems, for example from an event of interest back to a base station.

Results show that under increasing traffic load, SIGF modestly increases the overhead from message exchange and the end-to-end delay, but maintains high packet delivery ratios.

From Figure 6(a) we see that GF, IGF, and SIGF have comparable delivery ratios (90–100%) under light traffic load. When traffic flow rates increase to more than 7 packets/second per CBR flow, the network begins to suffer congestion in all protocols except IGF.

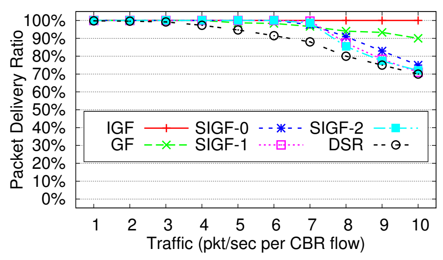
Label	**Configuration details**
SIGF-0 or SIGF-0-priority	$60°$ forwarding area, fixed $5\,ms$ collection window, choose by priority, include destination
SIGF-0-random	SIGF-0, but with random candidate selection
SIGF-1 or SIGF-1-reputation	SIGF-0 limited to high reputation neighbors, $\alpha = \frac{5}{8}$, $\beta = \frac{1}{8}$, $\gamma = \frac{1}{8}$, $\zeta = \frac{1}{8}$, $R_{threshold} = 0.45$
SIGF-1-random	SIGF-1, but with random candidate selection of nodes above $R_{threshold}$
SIGF-2	SIGF-0 and SIGF-1 with all messages authenticated, message sequencing, payload encryption

Table 4. Experimental protocol configurations.

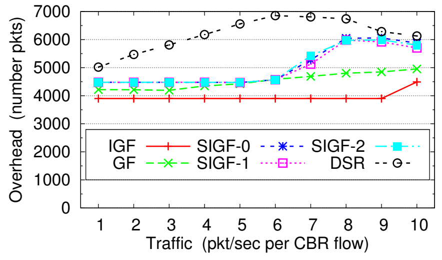
Performance in GF degrades along limited intersecting routes, suffering additional congestion caused by neighbor table update beacons. SIGF suffers congestion since multiple CTS responses are collected by each ORTS sender. DSR has significant message loss from its flooded route discovery packets.

IGF saves in communication overhead (shown in Figure 6(b)) because it does not require beaconing as in GF. Here the overhead packets are all MAC control packets including ORTS, CTS and ACK packets. Under light traffic loads, SIGF has similar communication overhead as GF, about 15% higher than IGF. As traffic loads increase, congestion increases the number of MAC layer collisions in IGF, GF and SIGF, resulting in retransmission attempts that add to the overhead. In particular, for SIGF the number of CTS packets increase quickly. DSR has more overhead because of route discovery packets. Its overhead ultimately diminishes because packet loss and the failure of route discovery packets to return to the source lead to fewer transmission attempts as messages are dropped early.

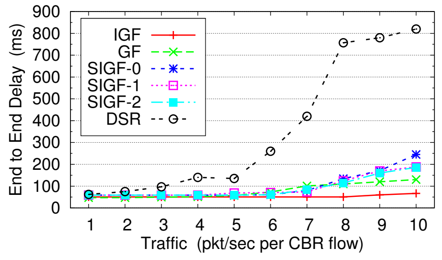
Local routing decisions introduce less end-to-end delay compared with routing protocols that require complete paths between a source and destination *a priori*. Figure 6(c) shows that IGF and SIGF have significantly lower end-to-end delay than DSR because DSR suffers latency awaiting the return of route discovery packets. This effect becomes less apparent in DSR under heavy traffic because DSR's low delivery ratio leads to fewer packets contributing to this metric. Besides that, we also can see that SIGF causes only a gradual increase in end-to-end delay (from 59 to 188 ms for



(a) Packet Delivery Ratio (PDR)



(b) Message Overhead



(c) End-to-End Delay

Figure 6. Baseline performance of routing protocols under increasing CBR traffic load, with no attacks.

SIGF-1) after hitting congestion, even though it collects multiple CTS packets for every ORTS packet.

In summary, IGF has very good performance without attacks—better than GF. The SIGF protocols add minimal overhead, and though there is little to distinguish them in the baseline results, differences become clear as we add attacks in the next sections.

5.2 Black Hole Attack

To create a black hole, attackers rush their CTS responses (as described in Section 3.2.1) so that they are received first by the ORTS sender. If an attacker is selected as the next hop, it simply drops the packet. We deal with incorrect locations in the Sybil attack experiments—here the locations reported are correct.

To eliminate the impact of network congestion, only a single CBR stream is considered. Node S sends a stream of packets to node D, shown in Figure 5. One-hop neighborhoods of S and D, and the direct line between them are also shown in the figure.

SIGF-0-random and SIGF-0-priority refer to the configurations shown in Table 4. SIGF-1-random and SIGF-1-reputation, also detailed in the table, discard responders whose reputations fall below $R_{threshold} = 0.45$. The former selects among the remaining nodes randomly, while the latter chooses the remaining node with the highest routing priority (based on distance, energy, etc). In both protocols, if no nodes have reputations that exceed the threshold, the node with the highest reputation is chosen.

We study the effect of the number of attackers and their locations on the packet delivery ratio in four scenarios. Figure 7 shows the performance of IGF and SIGF, grouped by the particular attack scenario. Results are nearly identical for CBR flows of 1–10 packets per second, hence the data shown are for six packets per second. Error bars in the graph show 95% confidence intervals on the mean.

In the first scenario, attacker $A2$ is near the routing path from source S to destination D (refer to Figure 5).

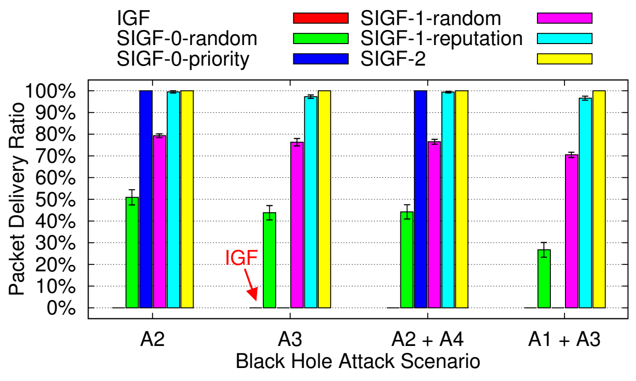


Figure 7. Performance under black hole attack scenarios, six pkts/sec CBR flow from S to D.

However, it is not an optimal relay compared with other nodes, for example $A3$, which lies closer to the shortest geographic routing path.

As shown in Figure 7, under a black hole attack the Packet Delivery Ratio (PDR) of IGF becomes zero—it is unable to deliver a single packet, since the attacker is always the first responder. SIGF-0-priority, SIGF-1-reputation, and SIGF-2 all better than 99% PDR: the former two protocols do not select attacker $A2$ since it is not an optimal choice, while the latter discards inauthentic messages. Using the random selection method in SIGF-0-random and SIGF-1-random degrades performance, since $A2$ will sometimes be chosen to relay the message. Still, the PDR is maintained at 50% and 79%, respectively.

In the second scenario, attacker $A3$ creates a black hole. From the node distribution, $A3$ is seen to lie near the optimal route from S to D. Hence, SIGF-0-priority performs very poorly, with zero PDR, since it always selects the attacker as the next-hop relay. SIGF-1-reputation achieves a very high PDR of 98%, even though it also selects $A3$, since the reputation degrades quickly causing other nodes to be selected instead.

The combination of attackers $A2$ and $A4$ in the third scenario shows the cumulative effect of two black holes along the path from S to D. Since neither attacker is an optimal relay, SIGF protocols using priority, reputation, or authentication maintain PDR of 100%. Random selection in SIGF-0-random suffers most, since packet loss is incurred at two hops along the path. Performance in SIGF-1-random is unchanged at 73%, indicating that the attackers' reputations eventually degrade below $R_{threshold}$ and so cease to be among the neighbors chosen at random to relay.

Finally, attackers $A1$ and $A3$ in scenario four are both optimal relays. Performance degrades for the randomized and reputation-based protocols only slightly, since the attackers' reputations degrade to allow other nodes to be selected more frequently.

We note that unlike the other protocols, results for SIGF-2 assume an outsider is performing the black hole attack. In an attack by a compromised node, messages are not authentic, and the protocol therefore performs as SIGF-1-reputation does—which is nearly as good.

In summary, SIGF protocols continue to deliver packets successfully when neighbors perform black hole attacks. Success rates vary depending on the amount of state and mechanisms used: SIGF-0 provides some defense with low PDRs (0–43%), SIGF-1 achieves moderate PDRs (70–99%), and SIGF-2 provides the best performance (100%).

5.3 Selective Forwarding Attack

If an attacker drops all messages completely, as in the black hole attack, it runs the risk that neighboring nodes can quickly conclude that an attack is under way and use other routes to avoid the attacker. It is more difficult to detect the attack if messages are selectively suppressed [12].

In this experiment, node $A3$ lies on the path of messages from S to D and mounts a selective forwarding attack. In Figure 8, successful packet delivery is plotted against an increasing packet drop ratio by the attacker. Error bars show 95% confidence intervals on the mean.

From zero to 100% dropped packets, IGF and SIGF-0-priority decline linearly from 100% to zero PDR. The randomized protocols, SIGF-0-random and SIGF-1-random, show greater robustness, but still decline to 43% and 76%, respectively. The latter levels off due to its limited use of reputation. Delivery success for SIGF-1-reputation dips to 82% when the attacker drops 30% of packets, but improves thereafter since the packet loss is sufficient to degrade $A3$'s reputation with its neighbors. Despite 50% dropped packets, SIGF-1-reputation has recovered to 96% PDR. SIGF-2 discards

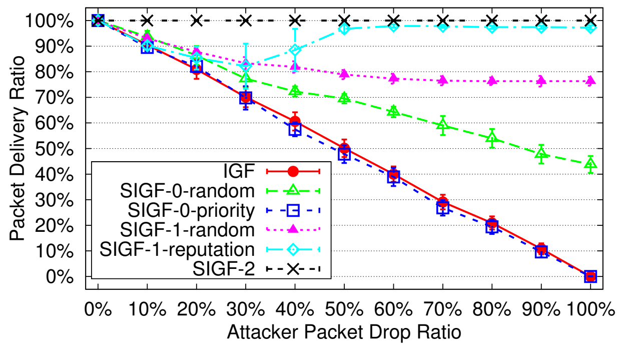


Figure 8. Performance under selective forwarding attack by $A3$ for increasing packet drop ratios.

all inauthentic messages, reliably achieving a 100% PDR.

Here we clearly see the ability of SIGF-1-reputation to adapt to worsening attacks, using history to learn to avoid unproductive neighbors. All the SIGF protocols react smoothly, without discontinuities or phase changes that may lead to unpredictable runtime behavior.

5.4 Sybil Attack

Now we evaluate our secure routing protocol under a Sybil attack by node $A3$. Figures 9–11 show the experimental results from the different scenarios we describe.

In the first scenario, attacker $A3$ creates six Sybil nodes randomly located about itself in a circle with a radius of the radio transmission range, the *Sybil distribution radius*. Each virtual node performs a black hole attack when the attacker receives an ORTS message. The locations of its virtual nodes are fixed to improve their reputations, since location inconsistency is penalized according to weight γ in Equation 1.

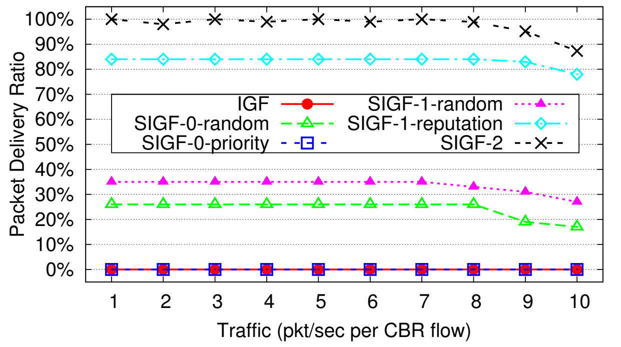


Figure 9. Performance under Sybil attack by $A3$, with six fixed-location virtual nodes.

Despite the Sybil black hole attack, SIGF-2 and SIGF-1-reputation achieve high packet delivery ratios, as shown in Figure 9. In SIGF-2, the attacker and its virtual Sybil nodes fail authentication, hence PDR is near 100%. In SIGF-1-reputation, Sybil nodes' reputations degrade quickly because they drop or modify the packets, resulting in a PDR of about 84%. Randomized protocols fare worse, but still achieve 26% and 35% PDRs. Overall, delivery ratios are less than in the single-node black hole attacks (Section 5.2) due both to more attackers and to network congestion caused by the Sybil neighbors.

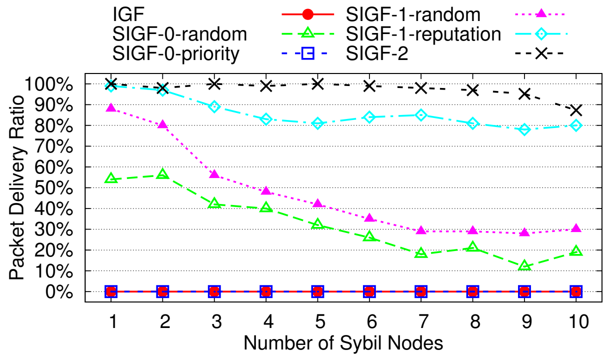


Figure 10. Performance under Sybil attack by $A3$, with increasing number of virtual nodes.

When the number of virtual Sybil nodes increases, the delivery ratio is reduced because a Sybil node is more likely to be chosen as the next-hop relay. In the second scenario we simulated an increasing number of fake Sybil nodes to determine their impact on performance. Figure 10 shows the results. Although delivery ratios decline overall as the attacker uses more Sybil nodes, SIGF-1-reputation stabilizes for more than four Sybil nodes at about 80% PDR.

An attacker can maximize the impact of its virtual Sybil nodes by "locating" them entirely within the forwarding area of a nearby message stream, if possible. In the last scenario, we examine the delivery ratio for increasing Sybil distribution radii.

When the Sybil distribution radius is small, the attack is more effective if the attacker is close to the optimal forwarding path for a message stream. Such is the case here, in which $A3$ is the Sybil attacker. We observe a clustered PDR of about 30–34% for all but IGF and SIGF-2 (see Figure 11). Larger radii decrease the attack's effectiveness (PDR improves to 58% or better) since fewer virtual Sybil nodes are in contention for relaying. These effects would be opposite for an attacker farther away from a message stream.

Together, these scenarios show that SIGF-0 and SIGF-1 can defend against Sybil attacks without requiring the initialization, synchronization, and state maintenance overhead of SIGF-2's use of authentication. Although performance is best for SIGF-2, this may pose an acceptable tradeoff if the threat of Sybil attacks is low.

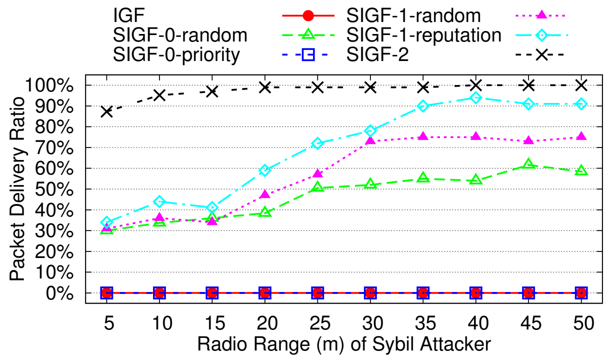


Figure 11. Performance under Sybil attack by $A3$, when virtual nodes are distributed in circles of increasing radii.

5.5 ORTS Replay DoS Attack

An attacker may capture and replay an ORTS message to cause a denial of service attack. For each ORTS message, this monopolizes the channel which may not be used except for collection of CTS messages from neighboring nodes.

In this experiment, node $A3$ replays an old ORTS message every $100\,ms$ while messages are in transit between S and D. Figure 12 shows that IGF, SIGF-0, and SIGF-1 are unable to defend against the attack, with less than 8% PDR in all cases. The congestion caused by the attacker's denial of service causes almost all packets to be dropped in the neighborhood of the attacker.

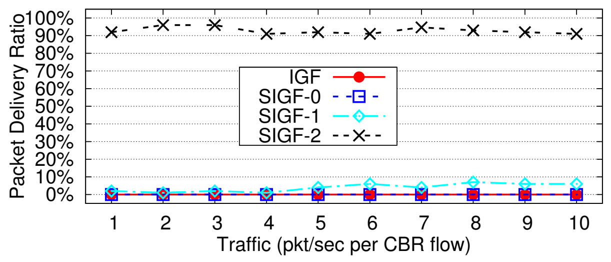


Figure 12. Performance under ORTS Replay attack by $A3$, which replays an old ORTS every $100\,ms$.

Only SIGF-2 can determine that the message is inauthentic by examining the sequence number contained in it. The congestion causes a mere 10% loss of PDR. As with many denial of service attacks [22], defense against the ORTS Replay is difficult without the stronger guarantees of SIGF-2.

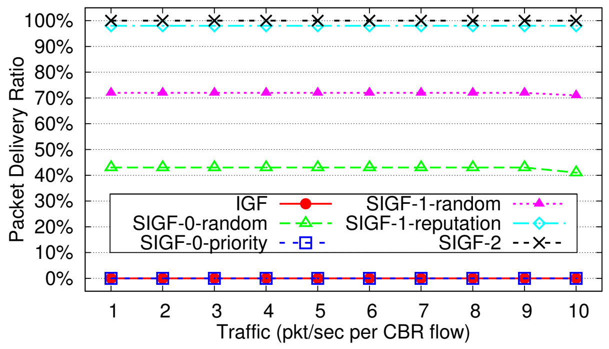


Figure 13. Performance for CTS Replay attack by $A3$.

5.6 CTS Replay DoS Attack

In a CTS Replay attack, a node attempts to disrupt forwarding by causing other nodes to abort the protocol early. Or, the attacker may attempt to damage the reputation of a neighboring node by replaying old CTS messages, even if the neighbor is not currently responding.

In this experiment, for each ORTS message node $A3$ replays a legitimate CTS overheard from a neighbor. Figure 13 shows that this attack is much less damaging than the previous. Although IGF and SIGF-0-priority are fooled, SIGF-2 and SIGF-1-reputation are not. In between are the protocols that select relays randomly. These suffer from the attack, but still allow 42% or 71% of messages to be delivered.

6. Related Work

Although many secure routing protocols have been developed for ad-hoc networks, these are not directly applicable for several reasons. Some protocols (ARAN [21], SAODV [23], et al. [24, 16]) use public-key cryptography, which is not considered to be memory and energy efficient enough for frequent use in sensor networks. Recent implementations of elliptic-curve algorithms on sensor devices may allow for their infrequent use, but symmetric cryptography-based algorithms are still desirable for their greater efficiency [17].

Some protocols use symmetric cryptography or hashing, but require maintenance of routing tables by online distance-vector algorithms (SEAD [7]) or on-demand multi-hop route discovery and caching (Ariadne [8], SRP [19]). For large-scale networks, this requires non-trivial consumption of memory and energy for the storage and update of routes to remote nodes. It also increases the “surface area” for security attacks.

Other work (e.g., SPINS [20], TinySec [11]) provides secure channels for use by otherwise unsecured protocols. They may be used to establish basic shortest-path routing trees (as described in SPINS), but are inadequate defenses when nodes are compromised.

INSENS [3] is designed to tolerate node compromise and uses a variety of efficient mechanisms to establish routing. However, it is limited to routing upstream messages from nodes to base stations, using centralized topology collection and route computation.

Rather than maintain routing tables, SIGF chooses the next hop dynamically and non-deterministically. This contains the effect of compromise to a local neighborhood, increases robustness to node mobility and failure, and spreads energy drain more evenly across neighbors. Even without using symmetric cryptography, SIGF-0 and SIGF-1 achieve good performance under the attacks discussed in Section 3.

In addition to plain geographic forwarding (GF) [5] and IGF [1], on which SIGF is based, other geography-based routing algorithms have been proposed. GPSR and descendents [13, 14] extend GF to route around voids by traversing faces of a planar subgraph until greedy forwarding can resume. SIGF inherits a mechanism from IGF for handling forwarding failure, and many of the other techniques that have been proposed could be applied in the local or shared state contexts. ZRP [6] divides the network into variably-sized zones and allows different algorithms for intra- and inter-zone routing. These protocols are lightweight and efficient, but do not consider security.

7. Conclusion

We have presented SIGF (Secure Implicit Geographic Forwarding), a secure routing protocol family for wireless sensor networks that builds atop the inherently attack-containing, dynamic binding of IGF. Rather than maintain routing tables, SIGF chooses the next hop dynamically and non-deterministically. This contains the effect of compromise to a local neighborhood, increases robustness to node mobility and failure, and spreads energy drain more evenly across neighbors.

SIGF-0 keeps no state, but uses probabilistic means to avoid selecting an attacker for routing. SIGF-1 adds locally maintained reputations for dynamically discovered neighbors, using them to select well-behaved relays. SIGF-2 adds more traditional sequencing and cryptographic mechanisms for authentication, but at the greatest cost of resources.

We evaluated SIGF without attacks for base performance, and with black hole, selective forwarding, Sybil, and denial of service attacks. We showed that even without using symmetric cryptography, SIGF-0 is able to defend against many attacks with no state, and SIGF-1 achieves high PDRs by maintaining reputations of neighbors. This allows efficient operation when no attacks are present, and good enough security when they are.

References

[1] Brian Blum, Tian He, Sang Son, and John Stankovic. IGF: A state-free robust communication protocol for wireless sensor networks. Technical Report CS-2003-11, Univ. of Virginia, Charlottesville, VA, 2003.

[2] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Research in Security and Privacy*, 2003.

[3] Jing Deng, Richard Han, and Shivakant Mishra. A performance evaluation of intrusion-tolerant routing in wireless sensor networks. In *Proc. IEEE 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)*, 2003. implemented and timed RC5, RC4, AES, RSA on mica mote.

[4] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *9th ACM Conference on Computer and Communications Security*, 2002.

[5] G. G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISI/RR-87-180, ISI, March 1987.

[6] Zygmunt J. Haas, Marc R. Pearlm, and Prince Samar. *The Zone Routing Protocol (ZRP) for ad hoc networks*. IETF MANET Internet Draft, July 2002.

[7] Y.-C. Hu, D. B. Johnson, and A. Perrig. Secure efficient distance vector routing in mobile wireless ad hoc networks. In *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 3–13, June 2002.

[8] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: a secure on-demand routing protocol for ad hoc networks. In *Proceedings of the Eighth Annual International Conference on Mobile computing and Networking*, pages 12–23. ACM Press, 2002.

[9] IEEE Computer Society LAN MAN Standards Committee. *IEEE Std 802.11: Wireless LAN Medium Access Control and Physical Layer Specifications*, August 1999.

[10] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, March 1996.

[11] Chris Karlof, Naveen Sastry, and David Wagner. TinySec: a link layer security architecture for wireless sensor networks. In *Proc. of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04)*, pages 162–175, 2004.

[12] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 1–15, May 2003.

[13] Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM-00)*, pages 243–254, N. Y., August 2000. ACM Press.

[14] Young-Jin Kim, Ramesh Govindan, Brad Karp, and Scott Shenker. Geographic routing made practical. In *Proc. of the USENIX Symposium on Networked Systems Design and Implementation*, May 2005.

[15] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. *Wireless Networks*, 6(4):307–321, 2000.

[16] Jiejun Kong, Petros Zerfos, Haiyun Luo, Songwu Lu, and Lixia Zhang. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *9th International Conference on Network Protocols (ICNP'01)*, 2001.

[17] D. Malan, M. Welsh, and M. Smith. A public-key infrastructure for key distribution in Tinyos based on elliptic curve cryptography. In *1st IEEE International Conference on Sensor and Ad Hoc Communications and Networks*, pages 71–80, October 2004.

[18] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proc. of 3rd IEEE/ACM Information Processing in Sensor Networks (IPSN'04)*, pages 259–268, April 2004.

[19] P. Papadimitratos and Z.J. Haas. Secure routing for mobile ad hoc networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, January 2002.

[20] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *Proceedings of Seventh Annual International Conference on Mobile Computing and Networks MOBICOM 2001*, pages 189–199, July 2001.

[21] Kimaya Sanzgiri, Bridget Dahill, Brian N. Levine, Clay Shields, and Elizabeth M. Belding-Royer. A secure routing protocol for ad hoc networks. *IEEE International Conference on Network Protocols (ICNP)*, 99(99), November 2002.

[22] Anthony D. Wood and John A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, October 2002.

[23] M. Guerrero Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proc. ACM Workshop on Wireless Security (WiSe)*, pages 1–10. ACM Press, 2002.

[24] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.

[25] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. LEAP: efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communication Security*, pages 62–72. ACM Press, 2003.

Achieving Repeatability of Asynchronous Events in Wireless Sensor Networks with EnviroLog

Liqian Luo†, Tian He‡, Gang Zhou, Lin Gu, Tarek F. Abdelzaher†, John A. Stankovic
Computer Science Department, University of Virginia, Charlottesville, Virginia 22904
{ll4p, tianhe, gz5d, lg6e, zaher, stankovic}@cs.virginia.edu

Abstract— **Sensing events from dynamic environments are normally asynchronous and non-repeatable. This lack of repeatability makes it particularly difficult to statistically evaluate the performance of sensor network applications. Hence, it is essential to have the capability to capture and replay sensing events, providing a basis not only for system evaluation, but also for realistic protocol comparison and parameter tuning. To achieve that, we design and implement *EnviroLog*, a distributed service that improves repeatability of experimental testing of sensor networks via asynchronous event recording and replay. To use EnviroLog, an application programmer needs only to specify two types of simple annotations to the source code. Automatically, the preprocessor embeds EnviroLog into any desired level of an event-driven architecture. It records all events generated by lower layers and can replay them later to upper layers on demand. We validate the accuracy and performance of recording and replay through a set of microbenchmarks, using the latest XSM platforms. We further demonstrate the strength of EnviroLog in system tuning and performance evaluation for sensor network applications in an outdoor environment with 37 XSMs.**

I. Introduction

With the increase in maturity of sensor networks research and with recent solutions to several practical systems and deployment problems, sensor network applications are entering the real world. Representative experiences of such evolution are documented in recent literature such as military surveillance [7] [19], habitat monitoring [13] [12], and environmental monitoring [23] [2], just to name a few. Quite different from controlled lab settings, physical environments introduce a high degree of uncertainty that makes it hard to conduct reproducible experiments. Consequently it is hard for researchers to obtain statistically consistent empirical results. With the growing number of applications developed and deployed, there is an increasing need for tools and services to assist with system evaluation and debugging, as well as with performance tuning of applications in outdoor environments.

To address this issue, we propose *EnviroLog*, a tool to improve repeatability of experimental testing of distributed event-driven sensor network applications. Unlike time-driven applications, such as periodic sampling of environmental conditions, the state of event-driven systems can change depending on the particular sequence of events received and their timing. Matters such as tuning protocol parameters for particular event scenarios or comparing performance of different protocols typically require the same distributed event traces to be replayed (e.g., to ensure a common basis for comparison). To address this requirement, we provide an event recording and replay service that can capture and reproduce distributed events on demand. The service provides the abstraction of a completely repeatable environment as observed by the sensory subsystem for the sake of experimental testing. Communication properties remain stochastic. Hence, a separation is achieved between the effects of communication non-determinism and the effects of environmental non-repeatability in the study of sensor network protocols. The service is especially valuable in the study of rare, unsafe, or hard-to-reproduce events such as the motion of tracked animals through a sensor field.

Our service is geared for the final stages of testing, typically performed in-field, where the effects of environmental realities can be studied. Early testing can use simulators that may be good for initial debugging since they allow fully controlled and repeatable experiments to be conducted. However, simulators are notoriously inaccurate when it comes to sensor network applications, since: (i) certain practical issues (e.g., the distribution of radio irregularity) are not adequately captured in most available simulators, resulting in large discrepancies between simulation results and empirical measurements, and (ii) simulators do not faithfully mimic environmental event signatures which affects the performance of sensor network applications. The problem is especially severe in event-driven architectures, where application behavior is more sensitive to the sequence, timing, and parameters of events received.

In addition to improving the repeatability of field testing, our service significantly reduces experimentation cost. In the absence of a recording and replay service, the investigators would have to either physically reproduce or passively wait for environmental events of interest, which entails additional costs. For example, the authors of [7] have developed a surveillance system that tracks persons and vehicles in the field. The need for walking or driving through the field hundreds of times while tuning an array of protocol parameters has proved to be a major practical impediment imposing a significant limitation on the rate at which experiments could be conducted in practice. EnviroLog, the asynchronous event recording and replay service described in this paper, provides a comprehensive solution to this problem.

Most sensor platforms employ flash memory for persistent

The work reported in this paper was supported in part by the National Science Foundation grants EHS-0208769, CSR-0509233, and NETS-0435060, and by MURI grant N00014-01-1-0576.

†Liqian Luo and Tarek F. Abdelzaher are now with University of Illinois at Urbana-Champaign.

‡Tian He is now with University of Minnesota.

storage. For example, Mica, Mica2, Mica2Dot [15] and XSM [17] hardware platforms incorporate 128 KB internal flash for code storage and 512 KB external flash for other usage. EnviroLog logs environmental events into such persistent storage devices. Later on, in replay mode, EnviroLog replaces environmental inputs with retrieved logs and re-issues the logged events in their original time sequence as asynchronous inputs. In addition to environmental events, EnviroLog can also log system runtime status for future analysis by recording selected variable values at runtime as specified by the programmer using simple annotations.

EnviroLog has two unique features. First, EnviroLog can operate at any layer of an application. In other words, events recorded and replayed by EnviroLog are not limited to direct reflection of environmental events such as raw sensory readings. They can be any system-level events. This characteristic of EnviroLog enables the debugging or tuning of any specific layer using controlled and repeatable inputs from lower layers in an event-driven system. Second, EnviroLog provided a very friendly user interface. Users only need to add simple annotations before events or variables to be logged. Applications with EnviroLog annotations can be compiled either into production code that ignores all EnviroLog annotations or into development code that allows on-demand recording and replay.

The potential uses of EnviroLog include (i) in-field debugging and performance tuning of specific parameters of an application, (ii) collecting statistical results from a large number of repeated experiments, and (iii) generating traces for mixed simulation environments that accept experimental measurements as inputs.

The remainder of the paper is organized as follows. Section II reviews related work. Section III describes the design goals and system architecture for EnviroLog. Section IV describes the implementation details. Section V evaluates EnviroLog, using XSM platforms, through a series of in-field experiments based on several sample applications provided by TinyOS [8] and a surveillance system [7] built upon TinyOS. Section VI concludes the paper.

II. Related Work

In recent years, sensor network researchers have proposed several tools and middleware that aid the debugging and evaluation of sensor network applications. Generally, they can be divided into four categories: simulators, emulators, test-beds and services. The section compares EnviroLog with related work in each of these categories.

Simulators are popular tools in debugging and evaluation of sensor network applications since they don't usually require the deployment of sensor hardware. NS-2 [16], GloMoSim [24] and TOSSIM [11] are good examples. NS-2 is a discrete event simulator supporting various networking protocols over wired and wireless networks. GloMoSim focuses more on mobile, wireless networks. It allows comparison of multiple protocols at a given layer. TOSSIM is a simulator especially designed for TinyOS applications, which provides scalable simulations of sensor network software. Current simulators, however, do not adequately capture the real behavior of sensor networks. This is due to the difficulty in modeling practical imperfections such as radio irregularity as well as due to the lack of good models of environmental inputs. The ability of EnviroLog to record environmental events can presumably be utilized to improve these tools by importing recorded event data to simulate environmental inputs.

Another category of debugging and performance evaluation tools in sensor networks is emulators that mimic sensor devices either in software or hardware. AVR JTAG ICE [1], a real time in-circuit emulator, is a good representative of hardware emulators. It uses the JTAG interface to enable a user to do real-time emulation of the microcontroller of sensor devices. A drawback of such in-circuit emulators is that they have to be physically connected to emulated devices, which causes logistical difficulties in conducting experiments especially for large-scale applications covering a wide field. Atemu [18] is a software emulator for AVR-processor-based systems that emulates AVR processors as well as other peripheral devices on the MICA2 platform. Like TOSSIM, Atemu also simulates wireless communication. Such software emulators do not introduce the logistical difficulties exhibited in hardware emulators, but they are usually less realistic in reproducing network behavior.

The final stages of debugging and performance tuning typically use actual testbeds to evaluate sensor network applications. For example, Motelab [22] is a public testbed using MICA2 platforms, which allows users to upload executables and receive execution results via the Internet. Kansei [20] is another testbed. It employs XSM, MICA2, and Stargate platforms. EmStar [5] is a combination of emulators and testbeds for Linux-based sensor network applications, which runs applications using either a modeled radio channel or the channel of real nodes. EmTOS [6] extends EmStar to run TinyOS applications by compiling them into EmStar binaries. These testbeds ease the development and evaluation a lot without requiring full-scale deployment. However, they do not focus on repeatability of environmental inputs like EnviroLog does.

We categorize all other software facilitating field tests of sensor network applications as services. EnviroLog belongs to this category. Monitoring tools such as Message Center [21] aid field tests by capturing messages in the air, filtering and displaying them to users. Closest to EnviroLog is TOSHILT [10], a middleware for hardware-in-the-loop testing. TOSHILT defines emulated stimuli to replace the real environmental events, so that applications can be evaluated repeatedly before the final deployment. Since TOSHILT uses synthetic and parametric event profiles, the detail of accuracy is less than what can be captured by EnviroLog. In addition, TOSHILT doesn't provide abstractions similar to EnviroLog annotations to ease the integration of the middleware into user applications. All these difference make EnviroLog unique. In the following, EnviroLog is described in more detail.

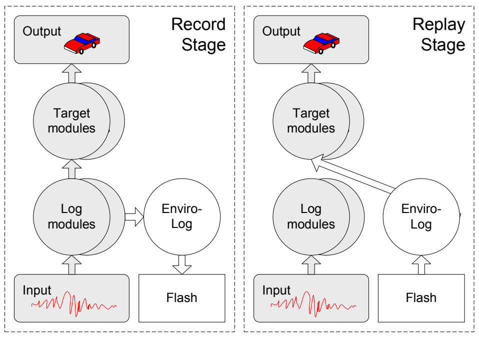


Fig. 1. Main idea of EnviroLog

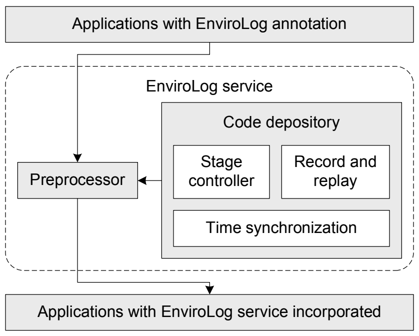


Fig. 2. System architecture of EnviroLog

III. Design

The major goal of EnviroLog is to enable experimental repeatability by recording and replaying environmental inputs. From a program's view, such inputs are essentially data streams that are transformed by each module of the program until they reach the modules at the topmost layer and become outputs. If we log the data stream generated by a module during the occurrence of an environmental event, and regenerate the same data stream later, from the perspective of the modules that consume the data, the environmental event is repeating itself.

EnviroLog makes the assumption that data is transferred from one module to another module only through function calls and their parameters. This assumption conforms to the event-driven sensor acquisition convention of TinyOS, where sensory data is usually obtained through asynchronous events (defined as a type of function calls in TinyOS) with parameters containing the data. Based on this assumption, the desired data stream generated by a module can be recorded by logging all its issued function calls and their parameters. That founds the main idea that EnviroLog is built upon.

Users of EnviroLog, through user interfaces, designate the modules that provide the data stream. These modules are called *log modules*. The modules that directly or indirectly consume the logged data stream are called *target modules*. As depicted in Figure 1, during the *record stage*, EnviroLog logs all function calls issued by the log modules into persistent storage devices such as a flash. During the *replay stage*, log modules are disabled. Instead, EnviroLog issues the previously recorded function calls at the right time and in the right sequence as recorded. Based on this main idea, the following subsections discuss the design of EnviroLog in more detail.

A. Design Goals

The design goals of EnviroLog are:

- **Effectiveness:** Effectiveness of EnviroLog is measured by its ability to perform: (i) accurate event record and event replay, and (ii) reliable runtime status record and retrieve.
- **Efficiency:** In sensor platforms, resource constraints (on both CPU and memory) are significant which makes it critical to use resources efficiently. EnviroLog is designed with efficiency in mind so that it can be applied to complex applications that consume a significant fraction of available resources.
- **Simplicity:** As a tool aimed to simplify the life of sensor network programmers, EnviroLog must be easy to use. In other words, simple user interfaces should be provided. Experiences tell us that simple tools tend to gain more popularity and persist longer while more complex tools sometimes run into usability barriers and become deserted.
- **Flexibility:** There are always tradeoffs between performance and overhead. Since EnviroLog cannot achieve optimal performance and minimum overhead at the same time, it should allow users to flexibly select their specific performance requirements (e.g., high logging throughput) at the expense of incurring a corresponding overhead. EnviroLog should also be flexible enough to allow users to record and replay any application behavior not limited to direct inputs of environmental events.

The above design goals provide guidelines throughout the design and implementation of EnviroLog.

B. System Architecture

Figure 2 illustrates the architecture of EnviroLog. Users indicate the set of events and variables to be recorded by inserting special annotations, called *EnviroLog annotations*, into applications. EnviroLog annotations are essentially the user interface of EnviroLog. Annotated applications are then processed by a *preprocessor* that translates EnviroLog annotations into real code, and, based on the annotations, integrates only the necessary modules from a *code repository* into the applications. The preprocessor and the code repository constitute *EnviroLog services*. As a result of the processing, application code is given the ability to record and replay specified events and to record and retrieve specified variable values that represent runtime status.

C. User Interface

Our design goal of simplicity calls for an easy user interface through which users are able to express the desired functionality of EnviroLog in a simple and intuitive manner.

The user specifies two main issues: asynchronous environmental event record and replay, and runtime status record and retrieve. The user interface design for the latter is easier since runtime status can be interpreted as values of variables at runtime. Such variables are simply annotated for logging. The former one is more complicated. As stated in the beginning of this section, EnviroLog supports environmental event record and replay by logging and regenerating data streams originating from environmental events. Since these data streams are transferred between modules through function calls and their parameters in most embedded systems such as TinyOS, we can log the behavior of a module by recording all issued function calls. Therefore, the only action an EnviroLog user is required to take is to specify a set of data-providing modules whose output data streams (issued function calls) are to be logged. The preprocessor takes the responsibility of enumerating function calls within the annotated modules and using APIs to log and replay each of them. To be more flexible, instead of specifying entire modules, advanced users are allowed to specify the exact function calls to be logged within a module.

Different from most services that usually provide function call APIs, EnviroLog provides EnviroLog annotations as the user interface. Table I lists the basic set of EnviroLog annotations.

The special characteristics of this user interface are:

1) EnviroLog annotations are simple to use. Users only need to insert annotations before function calls without worrying about details such as how these function calls are recorded, what data structures are used, and how and when to re-execute them when replaying. The preprocessor takes the responsibility to automatically generate code and integrate modules from the code depository to handle these details.

2) EnviroLog annotations take the form of comments that are ignored by the regular language compiler. This allows users to freely switch between original applications without EnviroLog integrated and EnviroLog-augmented applications. Annotated applications, when directly compiled, generate executables that do not include EnviroLog support. Alternatively, if they are processed by the EnviroLog preprocessor before compiling, the resulting executables are able to record and replay/retrieve the specified set of function calls and variable values based on user annotations.

To log environmental events, one option is to log the inputs of sensor drivers. However, users may focus on the evaluation and tuning of a specific layer higher than the layer of sensor drivers, thus requiring repeatable inputs to this layer. To fulfill such requirements, EnviroLog allows users to use EnviroLog annotations at any layer to enable repeatable input to that specific layer without interference from lower layers.

Though EnviroLog is geared for recording and replaying events of interest, it is also possible to choose whether to record the radio channel conditions using the user interface. If function calls to send messages or to cause the sending of messages are logged, the channel conditions are not captured and might be different between record and replay, depending mostly on the environment. Alternatively, if the programmers choose to log function calls that handle the reception of messages, the delivery of messages and their sequence should be the same between record and replay, no matter how the radio conditions change in reality. However, repeating the channel conditions is not always desired since in most experimental scenarios it is valuable to investigate how variances in channel conditions affect system behavior given the same sensory inputs.

D. Preprocessor

The simple user interface is supported by the preprocessor, which takes applications with EnviroLog annotations as input and outputs applications with EnviroLog code incorporated. The functionality of the preprocessor includes:

- Enumerating function calls and variables to be logged and assigning unique IDs, called *log IDs*, to them;
- Translating EnviroLog annotations into code that performs three functions. It defines the data structures for function parameters. It uses APIs provided by the record and replay module to record log IDs together with function parameters or variable values at the record stage. Finally, it re-executes at the replay stage the function calls upon the reception of logs from the record and replay module;
- Selecting necessary modules from the code repository and integrating them into applications.

To enhance robustness, logged data must be consistent with the current application during replay. Replaying data to the wrong application is not meaningful. A simple approach to ensure consistency is to use an ID, called the *application family ID* (or *application ID* for short) to denote the application tuned or the class of applications compared. The ID is logged as metadata at the record stage and is verified before replay. This ID is either specified by the application programmer who has the knowledge of which applications belong in the same family, or, if not specified, automatically created by the preprocessor by hashing the set of logged function calls into an application ID. The former solution allows for more flexibility while leaving the responsibility of ensuring consistency to the programmer. In the latter case, the same ID is generated as long as users don't change the set of function calls to be logged. This approach ensures consistency between logs and applications, making application IDs transparent to users since they are automatically handled by the preprocessor. Note that, both solutions enable repeatable environmental input to different application versions. Both the log modules and the target modules can be different between record and replay stage as long as data interfaces (in other words, the set of logged function calls) between them are the same.

Another challenge is to ensure *complete and consistent replay*, which means that:

- System outputs should be exactly the same during the record stage and replay stage, as long as target modules are not changed.
- Changes to target modules for purposes such as performance tuning should not affect data streams output by log

TABLE I
ENVIROLOG ANNOTATION LIST

Purpose	Annotation	Usage	Functionality
For event record and replay	/*LOG_MODULE*/	Insert the annotation in the beginning of the implementation of a module	To record all function calls issued in the module for future replay
	/*LOG_FUNCTION*/	Insert the annotation before a clause that makes a function call	To record the function call for future replay
For system status record and retrieve	/*LOG_VARIABLE: variable_name*/	Insert the annotation with specified variable name at a position within scope of the variable	To record current value of the variable for future retrieve

modules. If the target modules can affect the behavior of log modules, EnviroLog design may be unrealizable. For example, if power management services can dynamically select a subset of nodes and turn them off, a situation can arise where a node is turned off during the record stage but turned on during the replay. It is obviously impossible to decide on the correct value to be replayed since none was recorded. Another example is when a different (e.g., faster) sensor sampling rate is set by target modules during replay. Since data was recorded at a different rate, the information to be replayed is not available in the log. Both of the aforementioned cases are hard to accommodate, and are therefore not allowed.

With the prior guarantee that the two special cases above don't exist in a given application, the preprocessor can provide some consistency checks. To enable these checks, in addition to the set of log modules (L), users are required to specify the set of modules (I) that directly interact with the environment (e.g., senor drivers) and the set of modules (O) that provide system outputs (e.g., modules reporting final decisions to base stations). If one module u issues one or more function calls to another module v, we denote this relationship as $u \rightarrow v$. To formalize the initial check procedure, we further define that i represents system inputs, o represents system outputs and M represents the set of all application modules. Based on user inputs (L, I, O) and definitions (i, o, M), the preprocessor abstracts the application into a directed graph $G = (V, E)$, where:

$$V = \{u \mid u \in M\} \cup \{i\} \cup \{o\}$$
$$E = \{(u, v) \mid (u, v \in M \wedge u \rightarrow v) \vee (u = i \wedge v \in I) \vee (u \in O \wedge v = o)\}$$

Given the set of log modules (L) specified by the user and the calling graph G, the consistency check is done first by removing all edges that originate from vertices representing these log modules from the graph, which forms a new graph $G' = (V', E')$, where:

$$V' = V$$
$$E' = E - \{(u, v) \mid u \in L\}$$

If G' doesn't contain a directed path from i to o, the log module set is guaranteed to provide complete and consistent replay; otherwise, it may not be true.

Figure 3 gives an example to show more concretely the consistency check algorithm. Assume the application contains four modules, M_1 through M_4. Module M_1 directly consumes environmental inputs and issues function calls to M_2 and M_3. Both M_2 and M_3 further issue function calls to M_4. Finally, M_4 produces system outputs. The preprocessor builds the calling graph between application modules, adds output o and input i as virtual modules, connects input i to modules that directly consume environmental inputs (M_1), and connects modules producing system outputs (M_4) to output o. The result is the directed graph shown on the left rectangle of Figure 3. According to the algorithm, for each logged module, we then remove its outgoing edges from the graph. If the resulted graph doesn't contain any directed path from input i to output o, complete and consistent replay is guaranteed. As shown in the middle rectangle, logging of M_1 or logging of M_2 and M_3 ensures complete and consistent replay since no path exists from input i to output o. However, if only M_2 is logged, as shown in the right rectangle, a directed path traversing M_1, M_3 and M_4 exists between input i and output o, which fails in the consistency check and warns against a potentially incomplete or inconsistent replay.

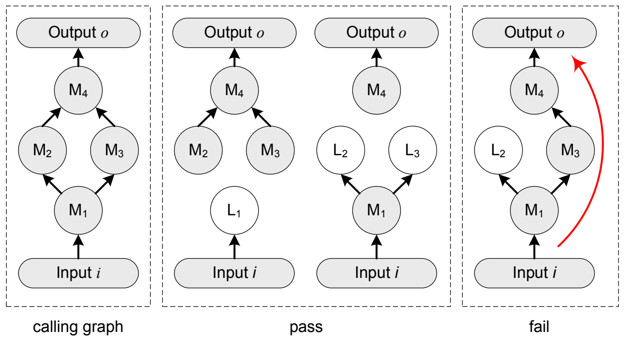


Fig. 3. Examples of consistency checks

Note that passing the initial checks is a sufficient, but not necessary condition for a complete and consistent replay guarantee, since not all function calls are issued due to environmental events. Thus, a directed path from i to o does not necessarily indicate that system outputs will indeed be affected by environmental inputs.

E. Stage Controller

EnviroLog needs runtime facilities to control (during execution) when to record environmental events or system status and when to replay events as well as to retrieve recorded system status. For this purpose, we provide the *stage controller* to interact with users during runtime, employing a client/server architecture. The server node, connected to a PC, receives *commands* from users through a command-line interface or GUI and disseminates them to client nodes. Upon the reception of commands, client nodes execute corresponding code for commands immediately or at a requested future time.

Each command must include a *command name* and a *stage name*. Command names include *start*, *stop*, *pause* and *continue*. Stage names can be *record*, *replay* or *retrieve*. The time period between the start of a stage and the stop of that stage is called a *run*. Other optional parameters in a command include a *replay speed* for the replay stage to request that logged events be replayed n times faster or slower than their original rate. For a retrieve stage, variable names and/or node IDs can be specified to denote the name of a variable whose recorded values and timestamps in the specified node are to be retrieved. Any stage can also specify a *run ID* to select the logs for the particular run the command will operate on.

To provide accurate replay for applications involving multiple nodes, one critical factor is that they have to be synchronized. In other words, these nodes have to execute the same command at the same time. If the same command is always delivered to all nodes at the same time, we can simply program nodes to execute the command right after its reception without worrying about synchronization. However, the assumption doesn't hold for multi-hop applications or single-hop applications with lossy links. We solve the problem by proposing *two-phase command execution*, which makes use of time synchronization and system-wide broadcast. When issuing commands, users are required to provide a future time as a command parameter, which specifies when the command is to be executed. Then, in the first phase, the command and synchronization beacons originated from the server node are propagated across the entire network to synchronize clocks of client nodes as well as to broadcast the command. When the specified future time comes, client nodes enter the second phase simultaneously to execute the command. Two-phase command execution is costly because of its time synchronization service and repeated system-wide broadcast. To be flexible, advanced users are allowed to configure the stage controller into its lightweight single-hop version as well as a version with two-phase command execution support.

Researchers on time synchronization [14] for current hardware platforms have observed a significant variance in clock frequency due to the instability of the used crystals. Although mechanisms like linear regression are able to compensate for clock drifts in the short term (e.g., within 30 seconds), periodic re-synchronization through messages is inevitable for long-term experiments to keep the error to the microsecond range. As a result, long experiments tend to introduce more inaccuracy if using two-phase command execution. In such a case, an alternative solution would be to keep the time synchronization service on throughout the two phases assuming that re-synchronization messages do not alter the behavior of the applications.

F. Record and Replay

The record and replay service is the core component of EnviroLog. It responds to stage control commands to switch between different stages, logs data into flash during recording, reads from flash the logged function calls to re-issue them in their original time sequence during replay, and reads from flash the logged variable values during status retrieving. Besides maintaining logs of function calls and variable values, it also maintains metadata such as application ID, run ID, and run length during recording, which are to be verified when replaying. The service also supports the replay of events at a speed different from recorded one, which can be used to emulate extremely fast or slow targets that are hard to generate physically.

1) Queue-based File System: If metadata and logs of one run are viewed as one file, we can easily design the service based on existing file systems such as Matchbox [4] and ELF [3]. To be comprehensive, these file systems usually support various file operations such as open, close, read, write, and append, consuming a lot of code as well as data memory. Our design goal of efficiency calls for a simpler solution. Hence, we propose a *queue-based file system*, where files are organized into one queue. At any point in time, only the file at the tail of the queue is writable and new data is always appended to this file. Only the file at the head of the queue can be deleted. It differs from typical file systems in that (i) each file occupies a continuous storage space, and (ii) the gap between two successive files is always smaller than one page. The queue-based file system brings about several benefits:

- It realizes the special characteristics of the logging behavior in EnviroLog: logs are sequentially written into flash, and oldest logs are usually most undesired.
- It exhibits low resource consumption. This file system only supports a minimum set of operations (file creation, sequential write, sequential read, file deletion) that is necessary in EnviroLog. The queue-based design eliminates the need for complicated storage space management such as free page maintenance and flash defragmentation. Hence, it consumes minimum code and data memory.
- It prolongs lifetime of flash memory by balancing writes to different pages. Each flash page has a write limit of about 10,000 times. In the queue-based file system, the sequential write access to flash pages ensures that the number of writes to different pages differs at most by 1.

2) Distribution of Data Structures: Although the ultimate storage space for logs is flash, during runtime, multiple memory levels are employed to improve efficiency and reliability. Figure 4 depicts the distribution of data structures in RAM, internal EEPROM and external flash.

Because flash access is relatively slow (e.g., for flash AT45DB041B used in MICA motes, erasing a page takes up to 8ms and writing a page takes up to 14ms), the service employs a buffer in RAM, called *log buffer*, to temporarily store logs before committing them into flash. *Log items* constructed for function calls or variable values to be recorded are queued in

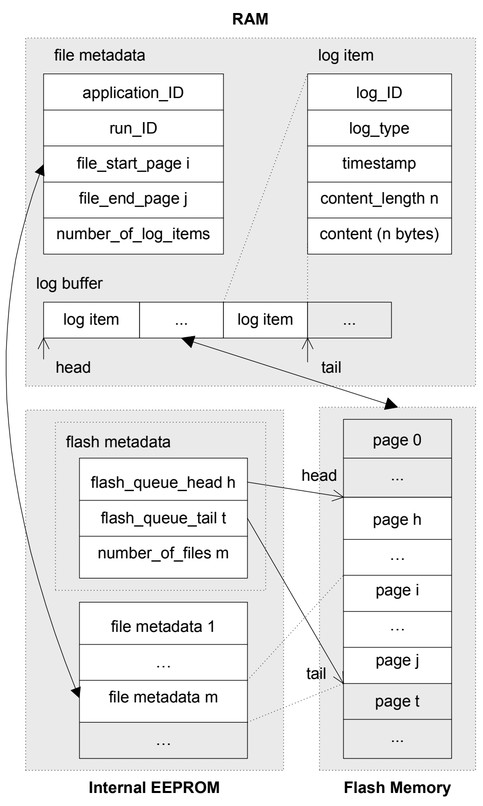


Fig. 4. Data distribution in different memory levels

this log buffer. Each log item consists of a log ID, a log type (`event` or `status`), a timestamp, log content (parameters of function calls or values of variables) and content length. We note that buffering can not only increase throughput of flash access, but also accommodate temporary data bursts typical in event detection or tracking applications. Another benefit of buffering is to support potential data compression. Many log items can be compressed before being written into flash to conserve space, including those containing the same content but different timestamps and those containing timestamps with fixed differences.

All log items of a run constitute a *file*, which is usually stored in flash. Access to a file is usually sequential, either reading from the beginning to the end or writing from the beginning to the end. Besides files, the metadata of files (*file metadata* for short) and the metadata of the queue (*flash metadata* for short) also require permanent storage. Different from access to files, access to metadata is much more frequent, that is why we store such data into the internal EEPROM of MICA motes which is smaller (4KB) but with a longer endurance (100,000 write/erase cycles) compared with the flash memory (512KB, 10,000 write/erase cycles). For other sensor devices without EEPROM, EnviroLog can be adapted to reserve several pages in flash for metadata storage. These pages are expected to be worn out earlier than other pages, which makes the usable flash size smaller.

3) Workflow of Different Stages: This section explains the execution flow of the record and replay service during different stages in more detail.

At the beginning of a record stage, upon the reception of a `start record` command, the service reacts by: (i) remembering the current time as the reference time, (ii) loading flash metadata into RAM, and (iii) constructing the metadata of a new file. During the record stage, whenever the application requests to record a function or a variable value, the service constructs a corresponding log item and enqueues it into the log buffer. The log item contains a relative timestamp, which is calculated by subtracting the reference time from the current time. If more than half of the buffer is filled up, all log items in the buffer are transferred into flash, which empties the buffer completely. An alternative choice would be to write everything in the buffer into flash whenever a new log item arrives. We decide on the former option, since higher throughput is observed for bigger block sizes in flash access (experiments on LogData component provided by TinyOS show that flash write speed is 12.99KB/s for a block size of 16B and 42.37KB/s for a block size of 128B). Finally, upon the reception of a `stop record` command, the service updates flash metadata and file metadata and commits them into EEPROM.

At the beginning of a replay stage, when a `start replay` command is received, the service takes several initial steps: (i) it loads into RAM the metadata of the corresponding file, whose run ID matches the one indicated in the command; (ii) to ensure data consistency, before replaying, it further verifies the application ID contained in file metadata against the one indicated by the user or produced by the preprocessor; (iii) it loads log items from flash to fill up the log buffer. After initialization, the service marks the current time as the reference time and starts to replay logged events. During replay, the service automatically loads data from flash to fill up the buffer whenever the buffer is half-empty. The service discards all `status` log items and replays `event` log items one by one. It dequeues the first `event` log item from the buffer, sets a timer based on the timestamp contained in the item, and upon the expiration of the timer, issues the corresponding function call. Then it proceeds with the next log item. The expiration time $T_{expiration}$ is calculated as follows:

$$T_{expiration} = \frac{T_{reference} + T_{timestamp} - T_{current}}{S_{replay}}$$

where $T_{reference}$, $T_{timestamp}$, and $T_{current}$ represent the reference time, the timestamp, and the current time respectively, and S_{replay} represents replay speed. As discussed before, S_{replay} is one of the parameters of `start replay` commands to speed up or slow down replay. Another way to calculate the expiration time is to take the difference between the timestamps of two successive events and divide it by S_{replay}. It is deserted because it makes time accuracy of the latter event always depend on the former one, and consequently accumulates errors over time. The replay stage ends when the entire file is processed or a `stop replay` command is received.

During the retrieve stage, the service simply discards all `event` log items. For `status` log items, it extracts the variable values and sends them to the stage controller, which then displays the data for end users.

IV. Implementation

In this section, we describe an implementation of EnviroLog, which has been fully tested on Mica2 and XSM hardware platforms. This implementation is expected to work on Mica and Mica2Dot as well. The common features of hardware platforms that this version of EnviroLog operates on are (i) 4KB EEPROM inside the micro-controller and (ii) 512KB external data flash. EnviroLog is implemented on TinyOS 1.x, a popular operating system for the aforementioned hardware platforms. Table II lists the implementation characteristics of different components in EnviroLog.

A. Preprocessor Implementation

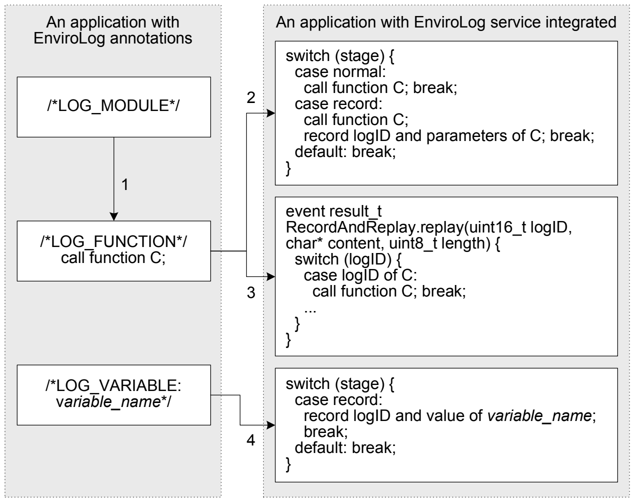
The preprocessor is essentially a translator that takes a TinyOS application annotated with EnviroLog annotations as input and outputs its corresponding version with the EnviroLog service integrated. Figure 5 depicts the main steps of this translation:

- Step 1: For modules annotated by `/*LOG_MODULE*/`, the preprocessor enumerates all function calls issued by these modules and annotates them by `/*LOG_FUNCTION*/`.
- Step 2: The preprocessor scans the entire application to search for all function calls that are annotated by `/*LOG_FUNCTION*/`. A unique log ID is assigned to address each of them. The clause that issues an annotated function call is replaced by a segment of code that (i) issues the function call only when the application is not at the replay or retrieve stage, and (ii) records the call's log ID and parameters during the record stage using APIs of the record and replay service.
- Step 3: The preprocessor also creates event handlers to handle replay requests from the record and replay service. For each replay request, it generates code to extract parameters and execute the corresponding function call.
- Step 4: after scanning the entire application, the preprocessor enumerates EnviroLog annotations in the form of `/*LOG_VARIABLE: variable_name*/`. The preprocessor assigns unique log IDs to them, and translates each of them into a segment of code that records the variable value and its log ID using APIs of the record and replay service.

The preprocessor then automatically wires necessary components (e.g., the record and replay service) into the resulting application to complete the integration of EnviroLog.

B. Stage Controller Implementation

Figure 6 depicts the field deployment of a system to use the stage controller. End users type-in stage control commands through the PC. The server node, which is connected to the PC through a serial cable, forwards the commands to the field. The client nodes in the field propagate the commands throughout


An application with EnviroLog annotations
An application with EnviroLog service integrated
/*LOG_MODULE*/
/*LOG_FUNCTION*/ call function C;
/*LOG_VARIABLE: variable_name*/
1
2
3
4

```
switch (stage) {
  case normal:
    call function C; break;
  case record:
    call function C;
    record logID and parameters of C; break;
  default: break;
}
```

```
event result_t
RecordAndReplay.replay(uint16_t logID,
char* content, uint8_t length) {
  switch (logID) {
    case logID of C:
      call function C; break;
    ...
  }
}
```

```
switch (stage) {
  case record:
    record logID and value of variable_name;
    break;
  default: break;
}
```

Fig. 5. Translation steps of the preprocessor

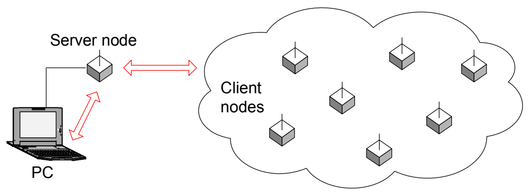


Fig. 6. A field deployment to use the stage controller

the field, and execute them immediately or in a two-phase manner.

We implement a simple Java tool on the PC to interact with end users. This tool has several functionalities such as: (i) encoding stage control commands into messages, (ii) injecting the messages into the server node through the serial port, (iii) receiving messages from the serial port, and (iv) displaying retrieved variable values to end users during the receive stage. The server node, running the `TOSBase` application provided by TinyOS, forwards messages between the PC and client nodes. The client nodes run the system with EnviroLog integrated, which includes a stage controller component.

The implementation of the single-hop version of the stage controller is simple. Commands are immediately executed upon reception and execution results, if any, are sent back to the server node through one-hop unicast. The multi-hop version that supports two-phase command execution needs more functionality. First, it contains a time synchronization service modified from multi-hop FTSP [14]. Multi-hop FTSP utilizes periodic flooding of synchronization beacons to perform continuous re-synchronization. We modify it to stop the periodic beacons at the end of the first phase; otherwise, these beacons may interfere with the system and change its behavior. Second, to conserve energy, commands are piggybacked onto periodic synchronization beacons. Client nodes remember the command when they receive the first synchronization beacon. The periodic nature of synchronization beacons also makes the dissemination of commands robust to sporadic message losses.

TABLE II
IMPLEMENTATION CHARACTERISTICS OF ENVIROLOG COMPONENTS

Component	Language	Code length (lines)	Data memory (bytes)
Preprocessor	Perl	873	
Stage controller	nesC	604	single-hop:46, multi-hop:137
Record and replay	nesC	758	54+buffer size

```
interface RecordAndReplay {
  command result_t record(uint16_t logID, uint8_t logType, char* content, uint8_t length); ①
  event result_t replay(uint16_t logID, char* content, uint8_t length);                   ②
  event result_t retrieve(uint16_t logID, char* content, uint8_t length);                 ③
  command result_t executeCommand(uint8_t name, uint8_t stage, char* parameters);        ④
}
```

Fig. 7. Interface of the record and replay component

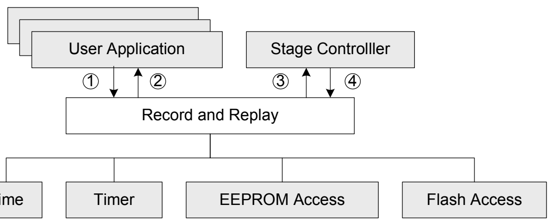


Fig. 8. Interactions between the record and replay component and other components

Third, the service incorporates a simple routing algorithm to collect execution results from client nodes. The routing service we implement is similar to directed diffusion [9]. Although the primary purpose of synchronization beacons is to synchronize clocks of client nodes, they also serve as interest beacons for client nodes to set up reverse paths to the server node. Later on, execution results can be sent back to the PC along those paths. Note that, the modified Multi-hop FTSP and the simple routing service become parts of the EnviroLog service only if users configure the stage controller as a multi-hop one before the application gets processed by the preprocessor. They are only invoked during the two-phase command execution and, therefore, are independent of any time synchronization or routing service used by user applications.

C. Record and Replay Implementation

The record and replay service provides a set of APIs to interact with application components and other EnviroLog components. It is implemented in one big component named `RecordAndReplayC`. Figure 7 illustrates the `RecordAndReplay` interface provided by the `RecordAndReplayC` component and Figure 8 depicts the interactions between this component and other components in the system. Application components (already processed by the preprocessor) call the command `record` to log function calls as well as variable values. During the replay stage, the record and replay component signals the event `replay` to request application components to execute the corresponding function calls. Another event `retrieve` is signaled to transfer retrieved variable values to the stage controller component, which then communicates the data back to the server node and, finally, to the PC. The stage controller component interacts with the record and replay component by issuing the command `executeCommand` to execute stage control commands from end users.

The record and replay component relies on several TinyOS modules: `clock`, `timer`, `EEPROM access`, and `flash access`. The clock component is used for the purpose of timestamp calculation. The timer component is utilized during replay to issue logged function calls in their original time sequence. Flash access and EEPROM access components are employed to read/write logged data and metadata.

V. PERFORMANCE EVALUATION

To evaluate the effectiveness and efficiency of the recording and replay service provided by EnviroLog, we integrate EnviroLog into several sample applications delivered with TinyOS, download them onto XSM motes, and carry out a set of empirical experiments. The XSM platform is an extended version of MICA2 motes, featuring improved peripheral circuitry, improved antenna and new types of sensors. The purpose of these microbenchmarks is to characterize the performance of EnviroLog by illustrating its maximum recording period, throughput, overhead, and replay accuracy.

Based on a complex surveillance system called Vigilnet [7] running on TinyOS and XSM platforms, we show the various functionalities of EnviroLog by using the recording and replay service to tune and evaluate performance of Vigilnet, to collect its runtime status and to replay targets with virtually increased or decreased velocities.

A. Microbenchmarks

In this section, we pick several sample applications provided by TinyOS and run a series of microbenchmarks to show how EnviroLog performs in terms of maximum recording period, throughput, overhead, and replay accuracy. These results, collected from simple applications, provide insights into the relevant basic aspects of EnviroLog's expected performance. Larger, more realistic applications are investigated later, highlighting higher-level performance aspects.

1) Maximum Recording Period: Due to limited storage space, EnviroLog cannot continuously record an infinite number of events. The term *maximum recording period* describes how long EnviroLog is able to continuously record. Maximum recording period RP_{max} depends on three factors: flash size S_{flash}, expected event interval $E\{eventInterval\}$ and expected log item length $E\{logItemLength\}$. The expected event interval indicates the average length of the time intervals between successive events. The expected log item length

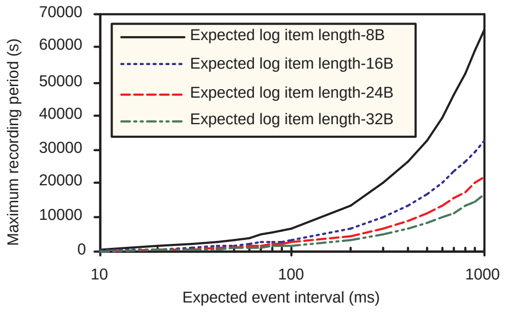


Fig. 9. Maximum recording period for different expected event interval and log item length

indicates the expected flash space a recorded event occupies. The maximum recording period is calculated as follows:

$$RP_{max} = \frac{S_{flash} E\{eventInterval\}}{E\{logItemLength\}}$$

Setting the flash size to be 512KB, Figure 9 depicts the maximum recording period for different expected event intervals and log item lengths. Typical raw sensing data, if generated at 10Hz, can be recorded for about 90 minutes, which is usually long enough for purposes of debugging or tuning sensor drivers.

2) Throughput: The throughput of EnviroLog (i.e., how fast EnviroLog is able to record) is evaluated based on the `Blink` application. `Blink` sets a periodic timer and toggles the red led when the timer fires. EnviroLog is used to record the toggling of the red led. The log item length equals the length of the log item header (7 bytes) since the toggling event contains no parameters. To make the scenario more realistic, the occurrences of toggling events are modeled as a Poisson process by making the time intervals between successive events exponentially distributed with the density function

$$f(x) = \lambda e^{-\lambda x}$$

where λ is the expected event interval.

To increase throughput, EnviroLog buffers events in memory before committing them to flash. We repeatedly change the expected event interval and buffer size, compile and download changed `Blink` onto a XSM mote and measure the success ratio of recording operations. Figure 10 illustrates the experimental results. Each point in this figure is the average of at least 10 runs to achieve a high confidence level. As shown in Figure 10, a higher success ratio is observed for lower event rates and bigger buffers. For events at 10Hz, a 128-byte buffer is enough to ensure a 100% success ratio.

3) Overhead: EnviroLog introduces a certain overhead, which may affect the runtime behavior of the original application. It must be verified that EnviroLog does not change the behavior of the original application dramatically during recording; otherwise, the replay of recorded behavior (which is dramatically different from the original behavior) becomes meaningless.

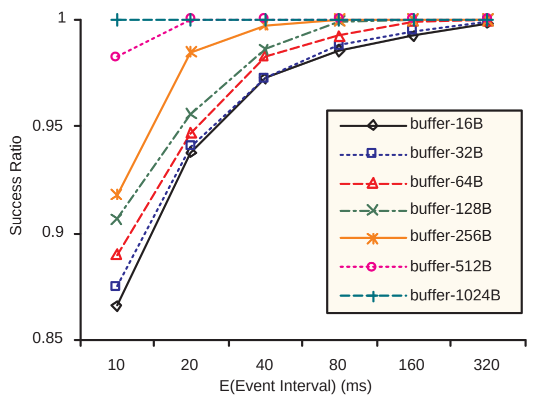


Fig. 10. Success ratio of recording operations for different expected event rate and buffer size

We try to quantify the overhead and its effect through an example application `CntToLedsAndRfm`. The application maintains a counter on a 4Hz timer and sends out the value of the counter by broadcast on each increment. The execution of the command that outputs the value of the counter by messages is recorded and replayed by EnviroLog. We change the message format to contain the `send` time of the current message and the `sendDone` time of the previous message in addition to the value of the counter, so that the time period to send a complete message, defined as *sending delay*, can be calculated by overhearing these messages. The modified `CntToLedsAndRfm` application is run on one XSM mote. Another XSM mote works as the server node of the stage controller to control the stage as well as overhear messages.

This experiment compares the sending delay during a normal stage with the one during a record stage to quantify the overhead of recording operations. Figure 11 depicts the cumulative distribution function of sending delay for 1000 messages during the normal stage and 1000 messages during the record stage. A longer sending delay is observed during the record stage compared with the normal stage, which indicates the overhead of recording operations. However, the effect of the overhead is trivial and acceptable. It is observed that the 95% confidence interval of the sending delay during the recording stage drifts only 0.4ms from the one during the normal stage.

4) Replay Accuracy: In this experiment, we use the same modified version of `CntToLedsAndRfm` as in the previous experiment. We first log about 100 commands by EnviroLog and overhear the messages to remember their send time. The remembered send time is actually the time when each logged command is executed. Then, we replay those commands 20 times. We calculate the difference between the average send time during the replay stage and the original send time during the record stage and depict its cumulative distribution in Figure 12. The results reflect how accurate the replay service is. As Figure 12 shows, the average error is less than 1ms.

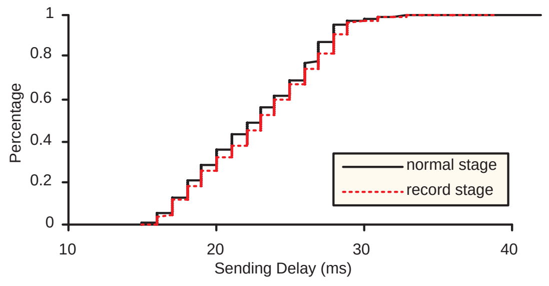


Fig. 11. Comparison of sending delay between normal stage and record stage

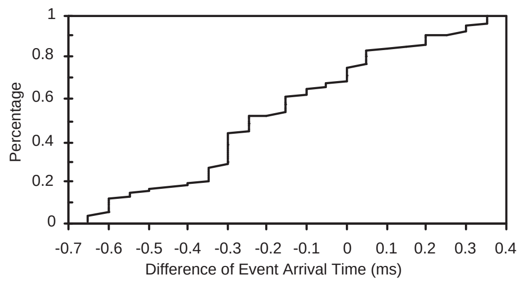


Fig. 12. Difference of send time between record stage and replay stage

B. Macrobenchmarks

Based on a surveillance system called Vigilnet [7], this section evaluates the effectiveness of EnviroLog in practice and showcases the variety of functionalities it provides. We first introduce the experimental methodology, including hardware, software and deployment scenarios. Then, we show the effectiveness of EnviroLog by recording and replaying different types of targets. Finally, we show how EnviroLog aids in-field tests of Vigilnet in various ways, including performance tuning and evaluation, runtime status collection and virtual velocity simulation.

1) Methodology: Vigilnet, implemented in TinyOS for XSM platforms, is targeted to detect, classify and track various events of interest in real-time through in-network processing. It takes environmental targets as inputs, applies multiple levels of processing before outputting results to end users. The lowest layer is sensor drivers which sample raw data from sensors and, if any target of interest is detected, signal detection results to higher layers. The layer above sensor drivers is a set of group management protocols that dynamically organize nodes in the vicinity of targets into local groups, collect detection results from individual nodes, elect leaders to aggregate these results and send aggregate data to nodes connected to base stations (base nodes). Target positions, as part of the aggregate data, are calculated by averaging locations of detection nodes. The highest layer is located in base nodes, where aggregate reports from leaders are further processed to extract properties such as target velocities. The highest layer outputs target types (vehicles or persons), trajectories and velocities to a GUI which displays results to end users.

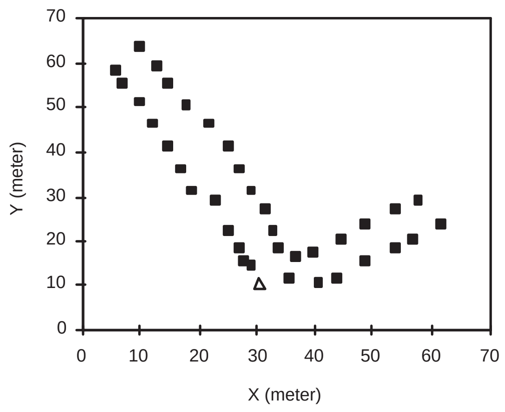


Fig. 13. System deployment

An obvious use of EnviroLog in this system would be to debug sensor drivers, to tune their sensor data processing parameters (e.g., various filters), or to compare their different versions by recording and replaying raw sensing data on individual nodes. However, what is more interesting is to see how EnviroLog aids the in-field tests of higher layers that involve coordination among multiple nodes in a multi-hop wireless network. Towards that end, we insert EnviroLog between the sensor driver layer and the group management layer, record outputs of sensor drivers rather than raw sensing data, and focus on the behavior of layers higher than the sensor drivers.

Vigilnet employs three types of sensors: magnetic, acoustic and motion sensors. Their drivers interact with higher layers by signaling instances of the following event:

```
event result_t detected(TargetConfidence
confidences);
```

where `TargetConfidence` is an array of integers, each representing the possibility of a certain target type.

To integrate EnviroLog into the system, we simply insert `/*LOG_FUNCTION*/` before each clause that signals the event, and process the code using the preprocessor before compiling the system. To collect empirical data, we download the system onto 37 XSMs. We deploy the XSMs approximately 5 meters apart on both sides of a driveway, as shown in Figure 13. The triangle marks the position of the base node connected to a laptop.

2) Effectiveness: To evaluate the effectiveness of recording and replay, we first set the system to be at record stage, and physically generate targets by jogging or driving through the driveway. The trajectories and calculated velocities for the jogging person and the vehicle are shown separately in Figure 14(a) and 14(d). Later on, we switch the system to its replay stage to virtually replay the jogging person and the vehicle. Figure 14(b) and 14(c) shows two different replays of the person, while Figure 14(e) and 14(f) shows two replays of

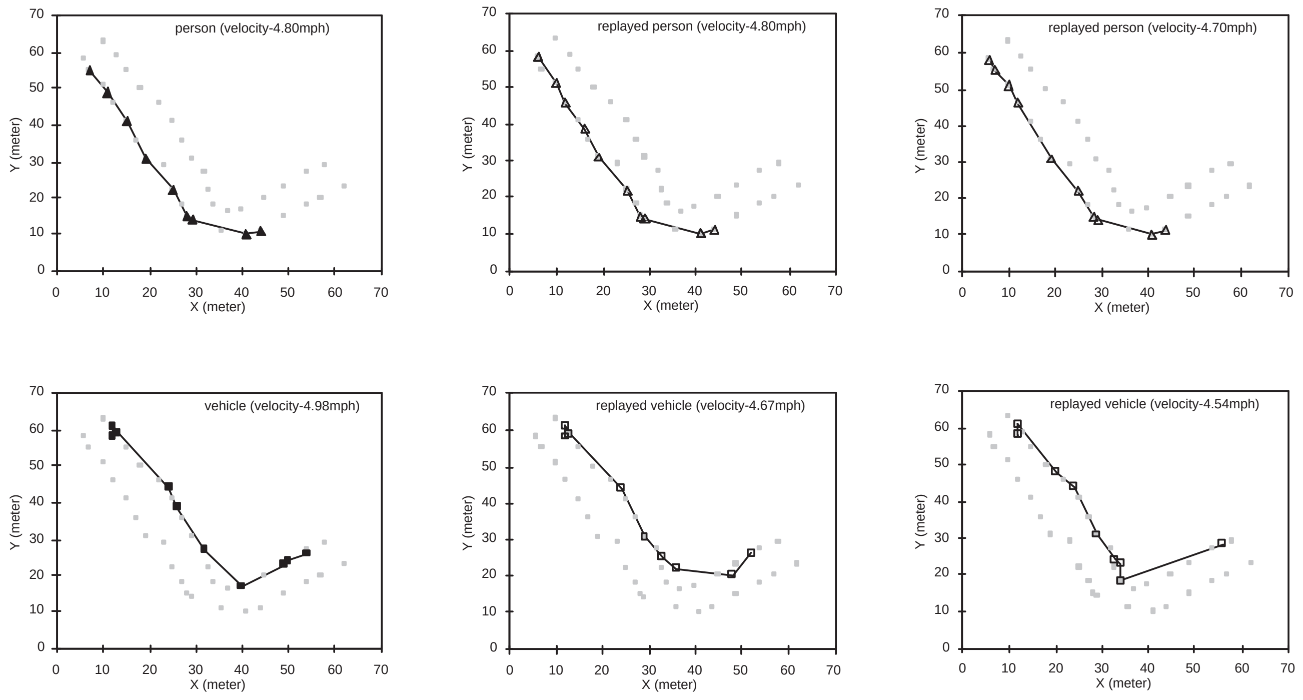


Fig. 14. Trajectories and calculated velocities for physical and replayed targets

the vehicle.

As is seen for both the person and the vehicle, the trajectories of real targets and replayed ones are very close, and the differences of their calculated velocities are below 0.5 mph. This observation verifies the effectiveness of EnviroLog. However, outputs are not exactly the same, which is expected considering the variability in layers above sensor drivers. For example, communication delays may change due to randomness in the MAC layer especially when multiple neighboring nodes request to send simultaneously. This also explains why replay of the person is more accurate than that of the vehicle. A person is usually detected by only one node while vehicles can be detected by multiple nodes simultaneously, which causes those nodes to send detection reports to leaders simultaneously. The possibility that these detection reports are always received by the leader in the same order during different runs is very low, which leads to the minor differences among observed target trajectories.

3) Potential Uses - Performance Tuning and Evaluation: A big portion of wireless sensor network applications are outdoor applications that detect targets or monitor environments. It is difficult to evaluate such applications by simulators, which either can't simulate environmental inputs or can't realistically simulate them. The environments are much more complex than what simulators can model. Even the modeling of the magnetic field near a vehicle is extremely challenging due to the uneven and unknown distribution of metals inside the vehicle. The lack of realistic sensing models makes in-field testing a necessary step before the real deployment of most applications involving target detection or environmental monitoring. EnviroLog makes in-field tuning and evaluation much easier as shown by the following experiments.

In Vigilnet, the group management layer used to group nodes that detect the same target has a tunable parameter called *DOA* (*degree of aggregation*). It is used to eliminate sporadic false positives in target detection. Group leaders do not report the detection of a target to the base station until the number of nearby nodes that detect the target reaches DOA. Higher DOA filters out more false positives, thus reducing the number of reports from leaders. However, too high DOA results in false negatives. To find out the proper DOA, we have to drive the vehicle or walk through the field multiple times while tuning this parameter. EnviroLog provides an alternative way to tune DOA with less overhead. We drive a vehicle or walk once to record the environmental inputs, then replay them multiple times with different DOA values. Figure 15 shows target trajectories for different DOA settings. As is seen, a higher DOA results in fewer trajectory points, thus more inaccurate calculated velocities. When DOA reaches 4, the calculated velocity (10.96 mph) is far away from the ground truth (5$\pm$1 mph).

We also use EnviroLog to log and retrieve the number of aggregate reports during runtime. Figure 16 depicts the cumulative distribution of the number of aggregate reports for different DOA values. As expected, a higher DOA leads to fewer aggregate reports. These results suggest that DOA values of 1, 2 and 3 are acceptable settings, though a higher value that leads to less communication overhead is more preferred.

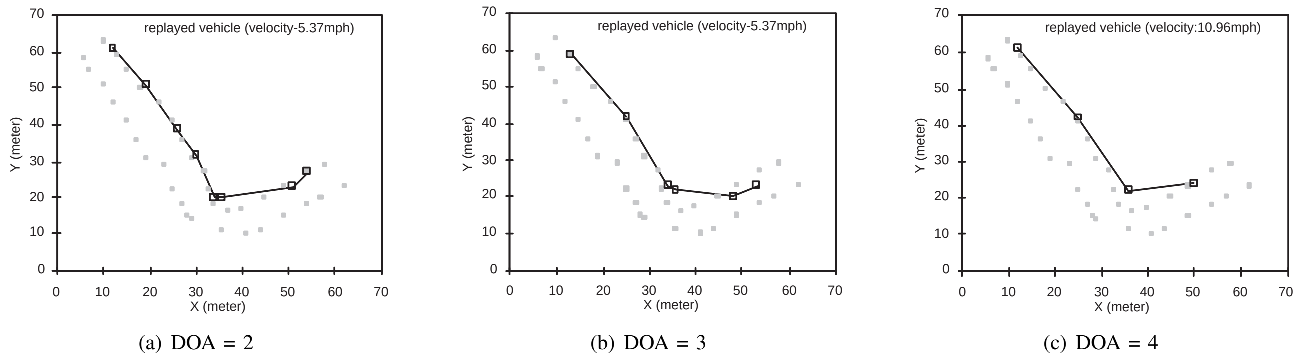


Fig. 15. Trajectories and calculated velocities for different DOA

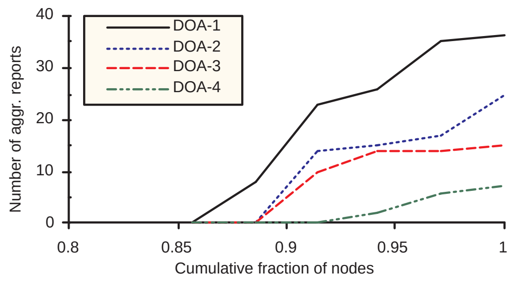


Fig. 16. Cumulative distribution of number of aggregate reports for different DOA

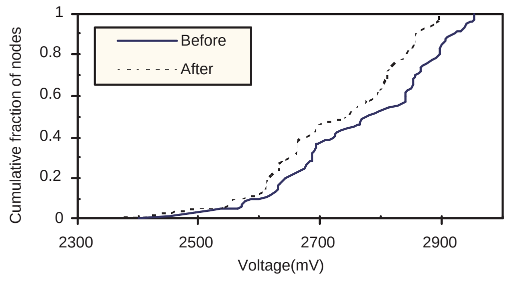


Fig. 17. Cumulative distribution of voltage values before and after experiments

4) Potential Uses - Runtime Status Collection: During development and testing, batteries are often depleted due to frequent experiments. One drawback of XSM motes is that batteries can not be measured or replaced without opening the package by unscrewing 4 screws. Vigilnet usually operates at the scale of hundreds of XSMs, which makes it extremely painful to check whether each node has a high enough voltage. Runtime status recording and retrieve supported by EnviroLog provides a simple solution for this problem. Voltage values of nodes can be logged and retrieved after each in-field test to find out those with low voltage, whose batteries then can be replaced before next test. Figure 17 shows the cumulative distribution of voltage values before and after the whole set of macrobenchmarks, which are collected through EnviroLog.

5) Potential Uses - Virtual Velocity Simulation: EnviroLog allows users to speed up or slow down the replay of events by setting a replay speed greater or less than 1. Note that changing replay speed is not always meaningful. For example, if sensor drivers pull data at a fixed rate and raw sensing data is logged, replaying at a different speed actually violates the logic of sensor drivers. In Vigilnet, EnviroLog records and replays the outputs of sensor drivers, which are detection events signaled by sensor drivers to higher layers. Replaying them at a higher speed can virtually simulate environmental targets with higher velocities. This experiment replays recorded events using different replay speeds to validate the effectiveness of virtual velocity simulation. Figure 18 depicts the calculated

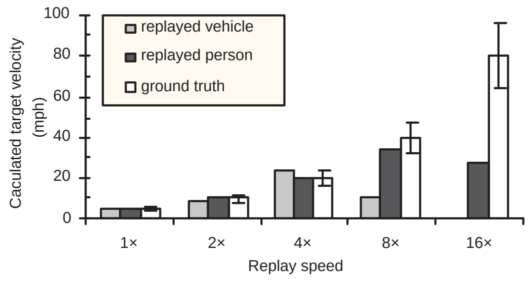


Fig. 18. Calculated velocities for different replay speed

velocities for different replay speeds and targets. When the replay speed doesn't exceed 4×, calculated velocities for both the person and the vehicle are close to the ground truth. Higher replay speeds lead to intolerable errors in velocity calculation.

C. Concluding Remarks

Results from the series of microbenchmarks and macrobenchmarks above validate the effectiveness of EnviroLog for both simple and complex applications. EnviroLog is able to record and replay high frequency events if assigned a big enough buffer (e.g., recording 10Hz events with a 128-byte buffer in the `Blink` application). Its recording operations bring little overhead (e.g., adding only 0.4ms delay in the `CntToLedsAndRfm` application). It also replays events accurately (e.g., the average difference between timestamps of recorded events and replayed events is less than 1ms). EnviroLog has various potential uses for in-field tests of large-scale systems, including performance tuning and evaluation, and runtime status collection. EnviroLog can replay events at different replay speed, which can be used to virtually simulate targets moving at different velocities. EnviroLog is available for download at http://www.cs.uiuc.edu/homes/lluo2/EnviroLog/.

VI. Conclusion

With the increasing popularity of wireless sensor networks, an increasing number of realistic applications employing large systems of sensor devices emerge. Although the initial development and debugging of these applications can be aided by simulators, in-field tests still have to be conducted at a later stage due to typical discrepancies between simulation results and empirical measurements. In this paper, we present the design, implementation and evaluations of EnviroLog, an asynchronous event record and replay service that improves repeatability of environmental events for in-field testing of distributed event-driven applications. The friendly user interface of EnviroLog allows users to integrate and utilize the service merely by inserting annotations into their applications and learning a few operation commands. Based on several sample applications of TinyOS and a complicated surveillance system, we validate the effectiveness of event recording and replay. We demonstrate the usefulness of EnviroLog in various aspects of in-field tests such as performance tuning without physically generating events, runtime status collection without extra hardware, and virtual velocity simulation. However, the potential uses of such service are not limited to what we have discussed in this paper. EnviroLog can be further extended to perform remote replay (recording events in environment A while replaying them remotely in environment B), and off-site replay (recording events on sensor devices while replaying them in simulators), which are on our agenda for future work on EnviroLog.

References

[1] Atmel Corporation. Mature AVR JTAG ICE. *http://www.atmel.com/dyn/products/tools-card.asp?tool-id=2737.*

[2] M. A. Batalin, M. Rahimi, Y. Yu, D. Liu, A. Kansal, G. S. Sukhatme, W. J. Kaiser, M. Hansen, G. J. Pottie, M. Srivastava, and D. Estrin. Call and response: experiments in sampling the environment. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 25–38, New York, NY, USA, 2004. ACM Press.

[3] H. Dai, M. Neufeld, and R. Han. Elf: an efficient log-structured flash file system for micro sensor nodes. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 176–187, New York, NY, USA, 2004. ACM Press.

[4] D. Gay. Matchbox: A simple filing system for motes. http://www.tinyos.net/ tinyos-1.x/doc/matchbox.pdf.

[5] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin. Emstar: a software environment for developing and deploying wireless sensor networks. In *Proceedings of the 2004 USENIX Technical Conference*, June 2004.

[6] L. Girod, T. Stathopoulos, N. Ramanathan, J. Elson, D. Estrin, E. Osterweil, and T. Schoellhammer. A system for simulation, emulation, and deployment of heterogeneous sensor networks. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, November 2004.

[7] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh. An energy-efficient surveillance system for sensor networks. In *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2004.

[8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *ASPLOS-IX: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pages 93–104, New York, NY, USA, 2000. ACM Press.

[9] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, 2003.

[10] D. Jia, B. H. Krogh, and C. Wong. *TOSHILT: Middleware for Hardware-in-the-loop Testing of Wireless Sensor Networks*. http://www.ece.cmu.edu/ w̄ebk/sensor-networks/toshilt/toshilt.html.

[11] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: Accurate and scalable simulation of entire tinyos applications. In *First International Conference on Embedded Networked Sensor Systems (SenSys'03)*, November 2003.

[12] T. Liu, C. M. Sadler, P. Zhang, and M. Martonosi. Implementing software on resource-constrained mobile sensors: experiences with impala and zebranet. In *MobiSYS '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 256–269, New York, NY, USA, 2004. ACM Press.

[13] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 88–97, New York, NY, USA, 2002. ACM Press.

[14] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49, New York, NY, USA, 2004. ACM Press.

[15] MICA motes. *http://www.tinyos.net/scoop/special/hardware/.*

[16] ns-2. The Network Simulator. *http://www.isi.edu/nsnam/ns/.*

[17] Ohio State University. XSM. *http://www.cast.cse.ohio-state.edu/exscal/index.php?page=main.*

[18] J. Polley, D. Blazakis, J. McGee, D. Rusk, and J. S. Baras. Atemu: A fine-grained sensor network simulator. In *First International Conference on Sensor and Ad Hoc Communications and Networks*, October 2004.

[19] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 1–12, New York, NY, USA, 2004. ACM Press.

[20] O. S. University. *Kansei: Sensor Testbed for At-Scale Experiments*, Feb 2005.

[21] Vanderbilt University. Message Center. *http://www.isis.vanderbilt.edu/projects/nest/msgctr.html.*

[22] G. Werner-Allen, P. Swieskowski, and M. Welsh. Motelab: A wireless sensor network testbed. In *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN'05), Special Track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS)*, April 2005.

[23] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A wireless sensor network for structural monitoring. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 13–24, New York, NY, USA, 2004. ACM Press.

[24] X. Zeng, R. Bagrodia, and M. Gerla. Glomosim: A library for theparallel simulation of large-scale wireless networks. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulation (PADS'98)*, page 154C161, May 1998.

MMSN: Multi-Frequency Media Access Control for Wireless Sensor Networks

Gang Zhou, Chengdu Huang[†], Ting Yan, Tian He[‡], John A. Stankovic and Tarek F. Abdelzaher[†]
Department of Computer Science, University of Virginia, Charlottesville 22903
{gzhou,ch4pp,ty4k,tianhe,stankovic,zaher}@cs.virginia.edu

***Abstract*— Multi-frequency media access control has been well understood in general wireless ad hoc networks, while in wireless sensor networks, researchers still focus on single frequency solutions. In wireless sensor networks, each device is typically equipped with a single radio transceiver and applications adopt much smaller packet sizes compared to those in general wireless ad hoc networks. Hence, the multi-frequency MAC protocols proposed for general wireless ad hoc networks are not suitable for wireless sensor network applications, which we further demonstrate through our simulation experiments. In this paper, we propose MMSN, which takes advantage of multi-frequency availability while, at the same time, takes into account the restrictions in wireless sensor networks. In MMSN, four frequency assignment options are provided to meet different application requirements. A scalable media access is designed with efficient broadcast support. Also, an optimal non-uniform backoff algorithm is derived and its lightweight approximation is implemented in MMSN, which significantly reduces congestion in the time synchronized media access design. Through extensive experiments, MMSN exhibits prominent ability to utilize parallel transmission among neighboring nodes. It also achieves increased energy efficiency when multiple physical frequencies are available.**

I. INTRODUCTION

As an emerging technology, wireless sensor networks (WSNs) have a wide range of potential applications [1] [2] [3], including environment monitoring, smart buildings, medical care, industry and military applications. Being an essential part of the communication stack, media access control (MAC) has received intense research attention, and a number of solutions have been proposed [4] [5] [6] [7] [8] [9]. While these solutions work well when one physical frequency is used, parallel data transmission when multiple frequencies are available is not considered. On one hand, the radio bandwidth in WSNs is very limited, 19.2Kbps in MICA2 [10] and 250Kbps in MICAz [11] and Telos [12]. On the other hand, the current WSN hardware, such as MICAz and Telos that use CC2420 radio [13], already provide multiple frequencies [10]. So it is imperative to design multi-frequency MAC protocols in wireless sensor networks to take full advantage of parallel transmission to improve network throughput.

In the state-of-the-art research, a significant number of multi-frequency MAC protocols have been proposed, for wireless networks in general. However, these protocols are not suitable for typical WSN applications. First, to save energy and reduce product cost, each sensor device is usually equipped with a single radio transceiver. This single transceiver can not transmit and receive at the same time, nor can it function on different frequencies simultaneously. This restricted hardware is quite different from more powerful hardware assumed in other wireless systems. For example, protocols [14] [15] are designed for frequency hopping spread spectrum (FHSS) wireless cards, and protocol [16] assumes the busy-tone functionality on the hardware. In protocols [17] [18] [19] [20], the hardware is assumed to have the ability to listen to multiple frequencies at the same time. Second, the network bandwidth in WSNs is very limited and the MAC layer packet size is very small, 30$\sim$50 bytes, compared to 512$+$ bytes used in general wireless ad hoc networks. Due to the small data packet size, the RTS/CTS control packets in IEEE 802.11 [21] no longer constitute a small overhead that can be ignored. So protocols [22] [23] [24] [25] [26] that are based on IEEE 802.11, and protocols [27] [28] [29] [14] that use RTS/CTS for frequency negotiation are not suitable for WSN applications, even though they exhibit good performance in general wireless ad hoc networks.

In this paper, we propose MMSN, abbreviation for Multi-frequency Media access control for wireless Sensor Networks. MMSN takes full advantage of multiple frequencies and is especially designed to meet WSN requirements. The detailed MMSN design is presented from two aspects: *frequency assignment* and *media access*, and its performance is evaluated through extensive simulation. The main contributions of this work can be summarized as follows:

- To the best of our knowledge, the MMSN protocol is the first multi-frequency MAC protocol especially designed for WSNs, in which each device is equipped with a single transceiver and the MAC layer packet size is very small.
- Instead of using pair-wise RTS/CTS frequency negotiation [27] [28] [29] [14], we propose lightweight frequency assignment, which takes advantage of the static property of many deployed wireless sensor networks [30] [31] [32] [33]. Even though pair-wise frequency negotiation is efficient when devices are highly mobile, it involves unnecessary overhead and is too costly when applied to static WSN applications.

  This paper gives a complete study of tradeoffs among physical frequency requirements, potential conflict re-

[†]Chengdu Huang and Tarek F. Abdelzaher are now with University of Illinois at Urbana-Champaign.
[‡]Tian He is now with University of Minnesota.

duction and communication overhead, during frequency assignment. Four optional frequency assignment schemes are proposed for MMSN, which exhibit distinguished advantages in different scenarios.

- We develop new toggle transmission and toggle snooping techniques to enable the single transceiver sensor device to achieve scalable performance, avoiding the non-scalable "one control channel + multiple data channels" design [34]. Also, MMSN has efficient broadcast support, which either is not addressed in [27] or is implemented by repeated link-layer retransmission of broadcast packets enqueued by higher layers in [22].

  Moreover, through strict theoretical analysis, an optimal non-uniform backoff algorithm is derived and its lightweight approximation is implemented in MMSN. Compared with a uniform backoff algorithm, this non-uniform scheme significantly reduces potential conflicts among neighboring nodes.

The rest of this paper is organized as follows: In Section II, we present the motivation of this work. In Section III, the design details of MMSN are explained. In Section IV, extensive experiments are conducted to evaluate MMSN's performance. Finally, in Section V, we give the conclusions and point out future work.

II. Motivation

To obtain a better understanding of the cost that RTS/CTS control packets incur in general wireless ad hoc networks versus WSNs, we choose a typical multi-frequency MAC protocol, the MMAC [27] protocol, proposed for general wireless ad hoc networks, as a case study. In MMAC, periodically transmitted beacons divide time into fixed-length beacon intervals. At the beginning of each beacon interval, there is a small window called the ATIM window, in which the nodes that have packets to send negotiate frequencies with destination nodes. After the ATIM window, nodes that have successfully negotiated frequencies with their destinations can send out data packets using the IEEE 802.11 protocol, i.e. exchanging RTS/CTS before sending out DATA packets. We implement MMAC in GloMoSim [35], a scalable discrete-event simulator developed by UCLA, and observe the performance. We adopt the same experiment set up as in [27]: 100 nodes are randomly placed in a 500m$\times$500m terrain. The transmission range of each node is 250m. Each node has 3 physical frequencies. Forty nodes are randomly chosen to be sources, and 40 nodes are randomly chosen to be destinations. Source nodes generate CBR traffic to destinations with a rate of 10 packets per second. Figure 1 plots the aggregate MAC throughput of the network with different packet sizes.

As can be observed in Figure 1, when the packet size is large, the MMAC protocol with 3 frequencies and a beacon interval of 100ms (the default configuration suggested in [27]) impressively enhances the aggregate MAC throughput by a factor of nearly 2 over IEEE 802.11. This result is consistent with that presented in [27]. However, the performance of both MMAC and IEEE 802.11 degrades when the packet size reduces. This is because the overhead of using RTS/CTS packets becomes more prominent when the data packet size is small. Moreover, the performance improvement of MMAC over IEEE 802.11 diminishes when the packet size becomes smaller. When the packet size is as small as 32 bytes, IEEE 802.11 has even a slightly higher throughput than MMAC. The reason is when the packet size reduces, more packets *could* be sent in a beacon interval. However, since nodes generally can not switch frequency during a beacon interval, the bandwidth wasted is more severe compared to the case when the packet size is large. Changing the length of the beacon interval could be beneficial, but the effect is two-sided. While lengthening the beacon interval can mitigate the overhead of having a fixed period of frequency negotiation, it deteriorates the bandwidth caused by the requirement that nodes have to stick to the frequency they have negotiated with some of their neighbors. In Figure 1, we also plot the cases with different beacon intervals. We can see that while using a shorter beacon interval (50ms) helps to some extent, MMAC with 3 frequencies still can not even outperform IEEE 802.11 with a single frequency, when the packet size is as small as 64 or 32 bytes. The main observation we make here is that while MMAC is a good multi-frequency MAC protocol for general wireless ad hoc networks where packets usually have large sizes, it is not suitable for WSNs where packets are much smaller.

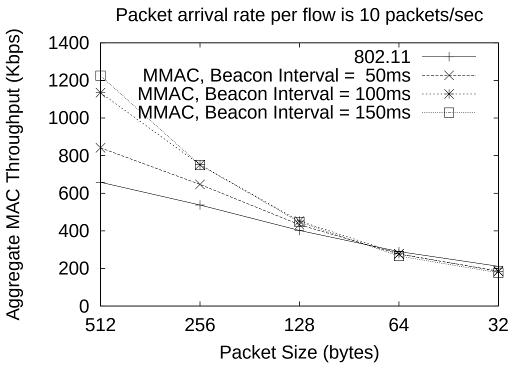


Fig. 1. Effect of Packet Size on MMAC

III. MMSN Protocol

This section presents the MMSN multi-frequency MAC protocol. MMSN is especially designed for WSNs, which is composed of hundreds of simple devices geographically dispersed in an ad hoc network over a large geographic area. Each device is equipped with a single transceiver and the packet size is very small, 30$\sim$50 bytes. The MMSN protocol consists of two aspects: frequency assignment and media access. The frequency assignment is used to assign different frequencies if enough frequencies exist, or evenly allocate available frequencies if there are more neighbors than available frequencies, to nodes that have potential communication conflicts. MMSN allows users to choose 1 of 4 available frequency assignment strategies. In media access design, nodes that have potential conflicts coordinate to access the shared physical frequencies, in a distributed way.

A. Frequency Assignment

In frequency assignment, each node is assigned a physical frequency for data reception. The assigned frequency is broadcast to its neighbors, so that each node knows what frequency to use to transmit unicast packets to each of its neighbors. We do not adopt RTS/CTS frequency negotiation, because it involves unnecessary overhead for many deployed wireless sensor networks [30] [31] [32] [33] where devices are generally not mobile. In WSNs, frequency assignment can either be done once at the beginning of the system deployment, or it can be done very infrequently just for adaptation to system aging. In order to reduce communication interference and hence reduce hidden terminal problems [21], nodes within two communication hops[1] are evenly assigned available physical frequencies.

In this section, four optional frequency assignment schemes are put forth: exclusive frequency assignment, even selection, eavesdropping and implicit-consensus. Among these four, exclusive frequency assignment guarantees that nodes within two hops are assigned different frequencies, when the number of frequencies is equal to or greater than the node number within two hops. Implicit-consensus also provides this guarantee, with less communication overhead, but requires more physical frequencies. Even selection and eavesdropping do not provide this guarantee and are designed for use when the number of available frequencies is smaller than the node number within two hops. Among these two, even selection leads to fewer potential conflicts while eavesdropping is more energy efficient. Users of MMSN can choose any one of the four options depending on their WSN attributes. Details of these four schemes are presented in the following subsections.

1) Exclusive Frequency Assignment: In exclusive frequency assignment, nodes first exchange their IDs among two communication hops, so that each node knows its two-hop neighbors' IDs. A simple way to implement this is for each node to broadcast twice. In the first broadcast, each node beacons its node ID, so that each node knows its neighbors' IDs within one communication hop. In the second broadcast, each node beacons all neighbors' IDs it has collected during the first broadcast period. Hence, after the second beacon period, each node gets its neighbors' IDs within two communication hops. Currently, we do not consider radio irregularity and link asymmetry [37] [38] [39] [40]. Readers can refer to [41] [42] [43] for more information about reliability issues in broadcast.

After nodes collect ID information of all neighbors within two hops, they make frequency decisions in the increasing order of their ID values. If a node has the smallest ID among its two communication hops, it chooses the smallest frequency among available ones, and then beacons the frequency choice within two hops. If a node's ID is not the smallest one among two hops, it waits for frequency decisions from other nodes within two hops that have smaller IDs. After decisions from all those nodes are received, the node chooses the smallest available (not chosen by any of its two-hop neighbors) frequency and broadcasts this choice among two hops.

This scheme guarantees to assign different frequencies to different nodes within any two-hop neighborhood, when the number of frequencies is at least as large as the two-hop node number.

2) Even-Selection: In exclusive frequency assignment, when there are not enough frequencies, it is possible that when a node makes its frequency decision, all physical frequencies have already been chosen by at least one node within two hops. In this case, the exclusive frequency assignment is extended by randomly choosing one of the least chosen frequencies. For convenience, we call this extension *even selection*, which makes an even allocation of available frequencies to all nodes within any two communication hops.

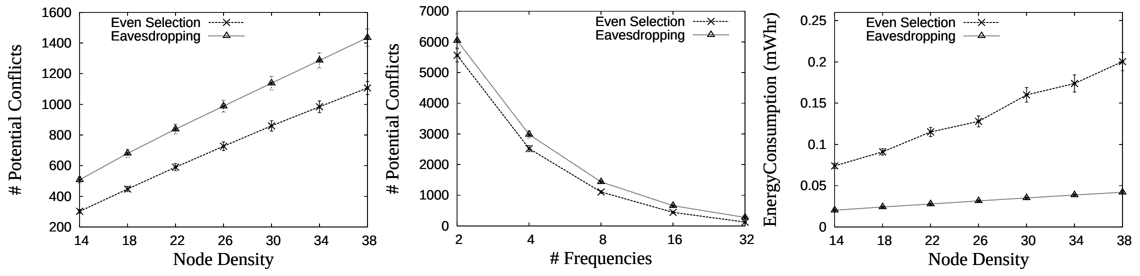
3) Eavesdropping: Even though the even selection scheme leads to even sharing of available frequencies among any two-hop neighborhood, it involves a number of two-hop broadcasts. To reduce the communication cost, we propose a lightweight eavesdropping scheme. In eavesdropping, each node takes a random backoff before it broadcasts its physical frequency decision. During the backoff period, each node records any physical frequency decision overheard. When a node's backoff timer fires, it randomly chooses one of the least chosen frequencies for data reception. Compared with even selection, eavesdropping has less communication overhead, but it also results in more potential conflicts, because it only collects information within one hop for frequency decisions.

4) Implicit-Consensus: When physical frequencies are abundant, the communication overhead in exclusive frequency assignment can be further reduced, while all nodes within any two-hop neighborhood can still be guaranteed to get assigned different frequencies. To achieve this performance, we propose the implicit-consensus scheme, which is inspired by the pseudo random number generator algorithms proposed in the NAMA [44] paper. In NAMA, the pseudo random number generators are used to design distributed time scheduling in TDMA. In this paper, we extend this basic pseudo random number generator idea, proposing a distributed frequency assignment algorithm for multi-frequency MAC designs.

In implicit-consensus, nodes' IDs need to be collected within two hops, in the same way as what is done in exclusive frequency assignment. Then, each node calculates its frequency number with a local computation. In the system, all nodes share the same pseudo random number generator, which is able to generate a unique random number sequence for each specified seed, the node ID here. Algorithm 1 presents the scheme for each node to calculate its frequency number. To assist explanation, node α is taken as an example.

As algorithm 1 states, for each frequency number, each node calculates a random number (Rnd_α) for itself and a

[1]In [36], it is pointed out that interference hops (connectivity based on interference relations), rather than communication hops, should be used for this purpose. For simplicity, we use two communication hops in this work. All algorithms proposed here can be easily extended by replacing the two communication hops with two interference hops.



(a) #Potential Conflicts vs. Node Density (b) #Potential Conflicts vs. #Frequencies (c) Energy Consumption vs. Node Density

Fig. 2. Performance Evaluation of Frequency Assignment

Algorithm 1 Frequency Number Computation

```
Input: Node α's ID ($ID_\alpha$), and node α's neighbors' IDs within two communication hops.
Output: The frequency number ($FreNum_\alpha$) node α gets assigned.
  $index$ = 0; $FreNum_\alpha$ = -1;
  repeat
    $Rnd_\alpha$ = Random($ID_\alpha$, $index$);
    $Found$ = $TRUE$;
    for each node $\beta$ in $\alpha$'s two communication hops do
      $Rnd_\beta$ = Random($ID_\beta$, $index$);
      if ($Rnd_\alpha < Rnd_\beta$) or ($Rnd_\alpha$ == $Rnd_\beta$ and $ID_\alpha < ID_\beta$) then
        $Found$ = $FALSE$; break;
      end if
    end for
    if $Found$ then
      $FreNum_\alpha$ = $index$;
    else
      $index$ ++;
    end if
  until $FreNum_\alpha > -1$
```

random number (Rnd_β) for each of its two-hop neighbors, with the same pseudo random number generator. A node wins the current frequency number if and only if its current random number is the highest among all current random numbers generated by all nodes within two hops. When two random numbers tie, the one with the larger node ID wins. In this way, each node explores all frequency numbers from zero to positive infinity until it finds the frequency for which it has the highest priority. By using the same pseudo random number generator, it is guaranteed that when a node decides that it wins frequency number $FreNum_i$, all nodes within two hops automatically agree with that decision and consensus is implicitly achieved, without any communication.

Here, a question may arise, since each node has a global ID. Why don't we just map nodes' IDs within two hops into a group of frequency numbers and assign those numbers to all nodes within two hops? Unfortunately, this scheme does not work, because a node's ID may get mapped to different frequency numbers in different two-hop neighborhoods. Also, it is not scalable to build a one-to-one mapping between nodes' IDs and all available frequencies, because this makes the frequency requirement depend on the network size, rather than the node density.

In implicit-consensus, when a node (node A) does not win the current frequency number ($FreNum_c$), because its random number is smaller than that of one of its two-hop neighbors (node B), it may happen that this neighbor (node B) has already won a previous frequency number. In this case, node B does not need the current frequency number. Node B should have already terminated its frequency number computation before it takes $FreNum_c$ into consideration, according to the repeat-until loop termination condition in algorithm 1. So, node A keeps trying larger frequency numbers until it finally finds one, while at the same time frequency number $FreNum_c$ is not chosen by any node within this two-hop neighborhood. Accordingly, the finally assigned frequency numbers among two communication hops may not be continuous. There may be holes, and some frequency numbers may not be assigned to any node, which is why the implicit-consensus scheme assumes that the available frequencies are abundant.

With the assigned frequency numbers, each node can easily calculate its physical frequency, with a local mapping. Let's put the available frequencies in a sorted list, $FreList = \{f_0, f_1, \ldots, f_N\}$, in increasing order. If the assigned frequency number is $FreNum_i$, the corresponding physical frequency is mapped to f_{FreNum_i}. After each node gets its physical frequency, it broadcasts this information to its one hop neighbors, so that each node knows what frequency to use to transmit packets to its neighbors.

B. Evaluation of Frequency Assignment

In this section, we compare the performance of even selection and eavesdropping, when available frequencies are not enough and potential conflicts exist. Performance comparison of exclusive frequency assignment and implicit-consensus are not presented, because both of them guarantee that there are no potential conflicts within any two-hop neighborhood.

In the experiments, performance is compared from three aspects. First, we compare the performance when the node density[2] increases while the number of available frequencies is fixed at 5. We use the number of potential conflicts as the performance metric, which is defined as the total number of node pairs in the system that satisfies the condition: the node pair is within two communication hops and both nodes share the same frequency. Since the two nodes are within two hops, two of their common neighbors may simultaneously transmit packets to them respectively. When they are assigned the same frequency, these two data transmissions interfere with each other, and packet loss may happen. So the number of potential conflicts measures the system's ability of full multi-frequency utilization. Second, besides node density, we also vary the number of available frequencies, to test the performance stability of even selection and eavesdropping. Third, we measure the communication energy consumption of all nodes within the system to compare the cost each scheme pays for its performance. We also explore the cost variation when different node densities are used.

The performance comparison is conducted in GloMoSim [35], in which 289 (17×17) nodes are uniformly deployed in a terrain of 200m×200m square. The radio type is set to RADIO-ACCNOISE [35] and the radio bandwidth is set to 250Kbps. The performance results are illustrated in Figure 2. For each data value we present, its 90% confidence interval is given as well.

As shown in Figure 2 (a), for all the node densities we set from 14 to 38, even selection always performs better than eavesdropping. For instance, when node density is 14, even selection has 302 potential conflicts, which is 40% less than the 507 potential conflicts eavesdropping has. When the node density increases to 38, even selection has 1106 potential conflicts and that is 23% less than the 1434 potential conflicts eavesdropping has. Even selection achieves this good performance because when a frequency decision is made, it is always the case that one of the least loaded frequencies is preferred within two hops. In this way, load is well distributed among all available frequencies within any two-hop neighborhood. However, in eavesdropping, nodes make frequency decisions based on overheard information within only one hop, which leads to a lower performance than even selection. From Figure 2 (a), it is also observed that the number of potential conflicts increases for both even selection and eavesdropping, when the node density increases. This is because the number of frequencies is fixed at 5, so the increased node density results in the increased number of nodes that share the same frequency within two hops.

Besides node density, we also vary the number of available frequencies to compare the performance of even selection and eavesdropping. In Figure 2 (b), the similar phenomenon is observed: even selection performs consistently better than eavesdropping, for all the numbers of frequencies we choose from 2 to 32.

With respect to energy consumption, Figure 2 (c) shows that even selection consumes more energy than eavesdropping. This is because even selection has two-hop neighbor discovery as well as two-hop broadcasts of frequency decisions, while eavesdropping only has one hop broadcasts.

However, this energy consumption is amortized during data transmission, because in many running sensor network applications [30] [31] [32] [33], sensor devices are generally static, so frequency assignment can either be done once at the beginning of the system deployment, or it can be done very infrequently just for adaptation to system aging. Accordingly, if the specific sensor network system is mostly static and the network congestion is a big issue, even selection is a better choice. On the other hand, if the system topology varies a lot with time and the network is lightly loaded, eavesdropping can be used to save more energy.

C. Media Access Design

After frequency assignment, each node gets a physical frequency for data reception. With the assigned frequencies, nodes cooperate to maximize parallel transmission among neighboring space in media access. To provide efficient broadcast support, nodes are time synchronized [45] during media access. A time slot consists of a broadcast contention period (T_{bc}) and a transmission period (T_{tran}). During the T_{bc} period, nodes compete for the same broadcast frequency and during the T_{tran} period, nodes compete for shared unicast frequencies. The T_{tran} period also provides enough time to actually transmit or receive a broadcast or unicast data packet. The time slot size depends on the number of nodes that compete for the same frequency and the data packet size. The regular time slot size is 3∼5ms.

Within one time slot, a node is able to either transmit or receive one packet. Each node first checks the broadcast frequency f_0[3] for receiving or transmitting a broadcast packet. If there is no broadcast packet to transmit or receive, unicast packet transmission and reception are considered. Each node's behavior differs depending on whether it has one packet to transmit or not, as well as whether it has a unicast packet or a broadcast packet to transmit. What follows explains the details.

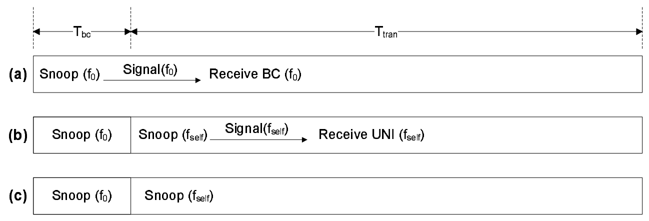


Fig. 3. When a Node Has no Packet to Transmit

1) Has No Packet to Transmit: If a node does not have any packet to transmit within a time slot, it behaves as Figure 3 presents. It first snoops on frequency f_0 during the time period

[2]The node density is defined as the number of nodes within one communication hop, and different node densities are configured by setting different radio ranges.

[3]One specific physical frequency is used for broadcast during the T_{bc} period, and this frequency can be reused during the T_{tran} phase for unicast transmission. So all frequencies are fully utilized.

T_{bc}. If the channel is busy, it becomes aware that another node is broadcasting a packet. So it receives the broadcast packet during the rest of the time slot, which is illustrated in case (a). On the other hand, if no signal is sensed during the time period T_{bc}, the node switches to snoop on frequency f_{self}, which is the frequency assigned to it for unicast packet reception. If a signal is sensed in frequency f_{self}, it receives the packet during the rest of the time slot, as shown in case (b). Here, we define $T_{Packet\,Transmission}$ as the time to deliver a packet after it gets the channel, which depends on the packet size and radio bandwidth. A nodes needs to keep on sensing the channel for a possibly incoming unicast packet, until the time left for the current time slot is shorter than $T_{Packet\,Transmission}$, as shown in case (c). When the time left for the current time slot is less than $T_{Packet\,Transmission}$, no neighboring nodes will send a packet to this node, so it turns off carry sensing until the start of the next time slot to save energy.

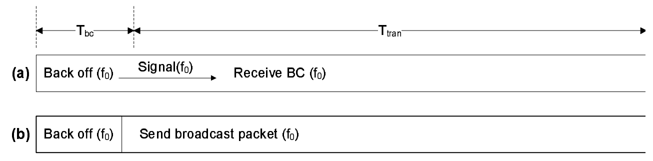


Fig. 4. When a Node Has a Broadcast Packet to Transmit

2) Has a Broadcast Packet to Transmit: If a node has a broadcast packet for transmission, it may have two different behaviors as illustrated in Figure 4. At the beginning of the time slot, the node uses frequency f_0, which is specified for transmitting and receiving broadcast packets. It first sets a random backoff within the time period T_{bc}. If it senses any signal during the backoff period, it becomes aware that another node is broadcasting a packet. In this case (case (a)), the node spends the rest of the time slot receiving the broadcast packet. There is another case, case (b), in which the node does not sense any signal in frequency f_0, during its backoff time period. In this scenario, a broadcast packet is sent out from this node, after the backoff timer fires.

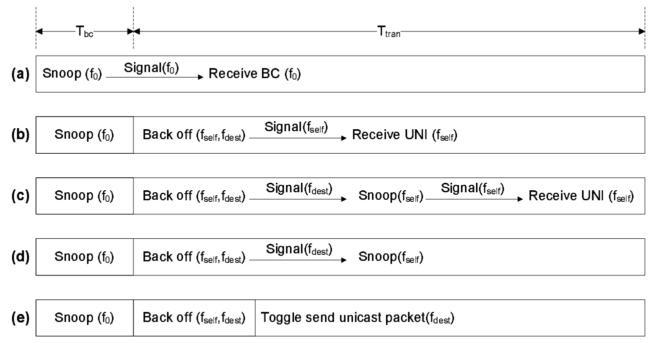


Fig. 5. When a Node Has a Unicast Packet to Transmit

3) Has a Unicast Packet to Transmit: Figure 5 illustrates the different behaviors a node may take, if it has a unicast packet for transmission. The node first listens to the broadcast frequency f_0 during time period T_{bc}. If it senses any signal during T_{bc}, which must be a broadcast packet, the node spends the rest of the time slot receiving the broadcast packet, as shown in case (a).

Cases (b)(c)(d)(e) illustrate the other four scenarios in which the node does not sense any broadcast signal during the time period T_{bc}. In these cases, the node takes a random backoff within the time period $T_{tran} - T_{Pakcet\,Transmission}$. During the backoff time period, the node snoops on two frequencies. On one hand, it snoops on frequency f_{self}, which is assigned to it for data reception, to get prepared for a possibly incoming unicast packet. On the other hand, it also snoops on frequency f_{dest}, which is assigned to the destination node of its unicast packet for data reception. If frequency f_{dest} is sensed busy, it can be aware that another node is transmitting a unicast packet to the same destination node, and it can choose not to transmit the unicast packet in the current time slot to avoid collisions. The node snoops on these two frequencies alternatingly, and we call this scheme *toggle snooping*, which is discussed in detail in subsection III-C.4.

During toggle snooping, if the node senses any signal on frequency f_{self}, it gets to know that it itself is the destination of an incoming unicast packet. So it stops toggle snooping to receive the data packet, which is illustrated in case (b). During the toggle snooping, the node may also sense a signal on frequency f_{dest}. When frequency f_{dest} is sensed busy, the node gets to know that another node is competing for the shared frequency, by sending a unicast packet to the same destination node. In this case, the node stops toggle snooping and switches to snoop on frequency f_{self} only. It gives up transmitting a unicast packet in this time slot and prepares to receive possible data packet transmitted to it. So if any signal is sensed in frequency f_{self}, as shown in case (c), it receives the unicast packet during the rest of the time slot. Before the node senses any signal in frequency f_{self}, it keeps sensing the frequency until the time left for the current time slot is $T_{Packet\,Transmission}$, as shown in case (d). When the left time for the current time slot is shorter than $T_{Packet\,Transmission}$, it turns off carry sensing to save energy.

If the node does not sense any signal in both frequency f_{self} and f_{dest} during the backoff time period, as shown in case (e), it sends out a unicast packet with the toggle transmission technique, which is illustrated in Figure 6.

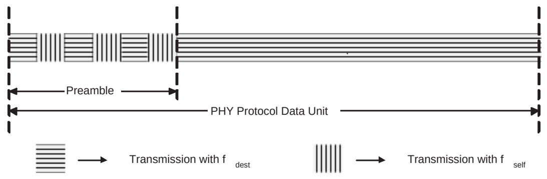


Fig. 6. Toggle Transmission

As Figure 6 illustrates, the preamble bytes of the physical layer protocol data unit (PPDU) is transmitted with two frequencies, f_{self} and f_{dest}, in an alternating way. The rest of the PPDU is transmitted to the destination node in frequency f_{dest}. The toggle transmission scheme is useful to reduce collisions.

As shown in Figure 7 (a), when node B is transmitting a unicast packet to node C with the toggle transmission technique, the preamble transmitted in frequency f_{self} informs other nodes that this channel is busy, so that any node that wants to send a packet to node B can back off. On the other hand, the preamble transmitted in frequency f_{dest} informs any node that wants to send a data packet to node C to back off and avoid possible collisions. The relation of toggle transmission and toggle snooping is analyzed in the following subsection.

4) Toggle Snooping and Toggle Transmission: When a node has a unicast packet for transmission, toggle snooping is used during the T_{tran} period and the node snoops on two frequencies alternatingly: the frequency it uses for data reception (f_{self}), and the frequency the destination node of its unicast packet uses for data reception (f_{dest}). The time a node takes to snoop on both of the two frequencies for one round is called the toggle snooping period, represented by parameter T_{TS}. In toggle transmission, a node transmits the preamble bytes of the PPDU with two frequencies, the frequency the node itself uses for data reception (f_{self}) and the frequency the destination node of the unicast packet uses for data reception (f_{dest}). The transmitter switches between these two frequencies alternatingly and the time the node sweeps the two frequencies for one round is called the toggle transmission period, represented by parameter T_{TT}.

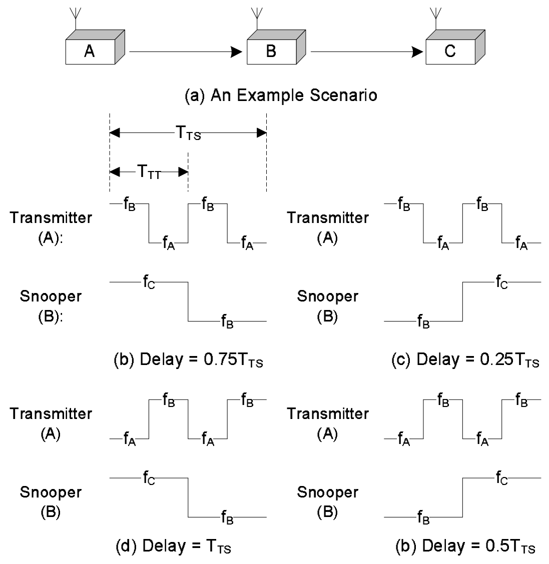


Fig. 7. Toggle Snooping

In the MMSN protocol, we let $T_{TS} = 2 \times T_{TT}$, so that when one node sends out a packet using the toggle transmission scheme, any other node that is snooping using the toggle snooping scheme is able to detect this transmission within a maximal delay of T_{TT}, if toggle transmission and toggle snooping have any shared frequency. With the help of Figure 7, we make this point more clear. In Figure 7 (a), node A uses frequency f_A for packet reception and it has a unicast packet to send to node B. Node B uses frequency f_B for packet reception and it has a unicast packet to send to node C, which uses frequency f_C for packet reception. During the T_{tran} time period, both A and B set up backoff timers and snoop on two frequencies. Node A snoops on frequency f_A and f_B and node B snoops on frequency f_B and f_C. Let's suppose that node A's timer fires first. So node A switches from toggle snooping to toggle transmission, while node B is still in the toggle snooping state. In different application scenarios (not only limited to the case in Figure 7 (a)), node B may take different time delays to become aware that node A is transmitting, as shown in Figure 7 (b)(c)(d)(e). In the scenario presented in case (b), node B is able to detect node A's transmission in frequency f_B after the time delay of $0.75\,T_{TS}$. In case (c), the delay to detect node A's transmission is $0.25\,T_{TS}$. In case (d) and (e), the delays are T_{TS} and $0.5\,T_{TS}$, respectively.

According to the above analysis, it is guaranteed that when one node transmits a packet using the toggle transmission scheme, the maximum time delay for another node, which uses the toggle snooping scheme, to detect the transmission is T_{TS}. Accordingly, if the backoff timer used in the slotted time period in Figure 5 is only allowed to fire at the end of a toggle snooping period, a node whose backoff timer fires after the previous one can have enough time to detect the previous node's transmission, and hence abandon its transmission in the current time slot to reduce congestion.

D. *IMPLICATION OF BACKOFF ALGORITHMS*

In media access, neighboring nodes may compete for the same physical frequency in both the broadcast contention period (T_{bc}) and the transmission period (T_{tran}), as explained in Section III-C. To reduce congestion, random backoff is needed for both broadcast and unicast transmission. Taking unicast backoff as an example, we give theoretical analysis to prove that a uniform backoff algorithm is not a good choice for the time synchronized media access in MMSN, and a non-uniform backoff algorithm achieves better performance. We derive an optimal non-uniform backoff algorithm, and choose its lightweight approximation for implementation in MMSN. All results derived here also apply for the broadcast transmission in MMSN.

During the backoff in the T_{tran} period in Figure 5, the time slot is further divided into small time slices. As explained in the previous section, each time slice has the length of T_{TS} and each backoff timer is only allowed to fire at the end of a time slice. If any two nodes choose the same backoff time slice, there is a collision. In order to minimize the probability of collision, we derive an optimal bound and a simple suboptimal distribution of the backoff time slices.

First we derive the optimal probability distribution $P(t)$ of backoff time slice t to minimize the probability of collisions when two nodes attempt to grab the same time slice after backoff. $P(t)$, $t = 0, 1, ..., T$, denotes the probability that a node attempts to grab time slice t and T is the maximum backoff time slice. Obviously $0 \leq P(t) \leq 1$ and $\sum_{t=0}^{T} P(t) = 1$. We assume that each node independently selects the backoff time slice conforming to the same distribution.

According to the analysis in Section III-C, in a time slot, the node that selects the earliest backoff time slice gets the physical channel, and all nodes whose backoff timers fire later should abandon their transmission. Hence, a node successfully grabs time slice t if all other nodes attempt to grab time slices after t. If at least two nodes in the same neighborhood attempt to grab the same earliest time slice, there is a collision. We need to find the probability distribution $P(t)$ to maximize P_{nc}, the probability that there is only one node that grabs the earliest time slice, to avoid collisions as much as possible. Assuming the earliest time slice is i, $0 \leq i \leq T-1$, and there are N nodes in the neighborhood, the probability that one and only one node attempts to grab this time slice and all other nodes attempts to grab later time slices is $N \cdot P(i) \cdot S_{i+1}^{N-1}$, where $S_{i+1} = \sum_{t=i+1}^{T} P(t)$. Considering all possible earliest time slices, we have

$$P_{nc} = \sum_{i=0}^{T-1} N \cdot P(i) \cdot S_{i+1}^{N-1}.$$

Now we apply a recursive approach to decide the optimal probability distribution $P(t)$. First we assume that the values for $P(t)$, $t = 0, ..., T-2$, are already known. From the constraints that the sum of all $P(t)$'s is 1, $S_{T-1} = P(T-1) + P(T)$ is also known. The question is how to divide S_{T-1} between $P(T-1)$ and $P(T)$ to maximize P_{nc}. This division only affects the term $N \cdot P(T-1) \cdot P(T)^{N-1}$ in the calculation of P_{nc}. The other terms are not affected by the way S_{T-1} is divided. For simplicity we denote $P(T)$ as a and $P(T-1)$ as b. The first order condition for maximizing is

$$\frac{d}{da}(Nba^{N-1}) = N(N-1)S_{T-1}a^{N-2} - N^2 a^{N-1} = 0,$$

and we have

$$S_T = P(T) = a = k_T S_{T-1},$$

where $k_T = \frac{N-1}{N}$.

We omit the validation of the second order condition for brevity, but for $N \geq 2$, the above equation does give a maximized result

$$N \cdot P(T-1) \cdot P(T)^{N-1} = k_T^{N-1} S_{T-1}^{N}.$$

Then we consider the division of probability S_{T-2} between $P(T-2)$ and S_{T-1} assuming that the values for $P(t)$, $t = 0, ..., T-3$ are known. For simplicity we denote $P(T-2)$ as c. The terms affected by this division are only

$$NcS_{T-1}^{N-1} + k_T^{N-1} S_{T-1}^{N}.$$

The first order condition is

$$N(N-1)S_{T-2}S_{T-1}^{N-2} + (k_T^{N-1} - N)NS_{T-1}^{N-1} = 0,$$

and therefore

$$S_{T-1} = k_{T-1} S_{T-2},$$

where $k_{T-1} = \frac{N-1}{N-k_T^{N-1}}$. Then we obtain the optimal value of the sum of the two terms:

$$NcS_{T-1}^{N-1} + k_T^{N-1} S_{T-1}^{N} = k_{T-1}^{N-1} s_{T-2}^{N}.$$

With the similar approach we get the recursive formulas for S_t and k_t as follows.

$$S_{t+1} = k_{t+1} S_t,$$

where $t = 0, ..., T-1$, and $S_0 = 1$.

$$k_{T-t-1} = \frac{N-1}{N - k_{T-t}^{N-1}},$$

where $t = 0, ..., T-2$, and $k_T = \frac{N-1}{N}$.

Therefore, the optimal distribution $P(t)$ is

$$P(t) = S_t - S_{t+1},$$

where $t = 0, ..., T-1$, and $P(T) = S_T$.

The optimal distribution gives an optimal bound of the non-collision probability. However, the distribution depends on the number of competing nodes, which may vary from neighborhood to neighborhood in deployed systems. Also, the process of computing the distribution is complicated and hence too costly for power-limited sensor devices. Accordingly, if a simple solution can provide a non-collision probability close to the optimal bound, it is more favorable. We propose a suboptimal distribution to be used by each node, which is easy to compute and does not depend on the number of competing nodes. A natural candidate is an increasing geometric sequence, in which

$$P(t) = \frac{b^{\frac{t+1}{T+1}} - b^{\frac{t}{T+1}}}{b-1}, \tag{1}$$

where $t = 0, ..., T$, and b is a number greater than 1.

The problem is which value of b should be chosen. We choose various b values to calculate the corresponding non-collision probabilities and compare them with that from the optimal $P(t)$. To be consistent with the evaluation section, we choose the same number of time slices and node densities. The results are shown in Figure 8. From the figure we can see that if we choose $b = 1000$, the difference between the simple solution's non-collision probability and that of the optimal $P(t)$ is smaller than 6% for the node densities we choose and $T = 33$, which is the number we use in the simulation.

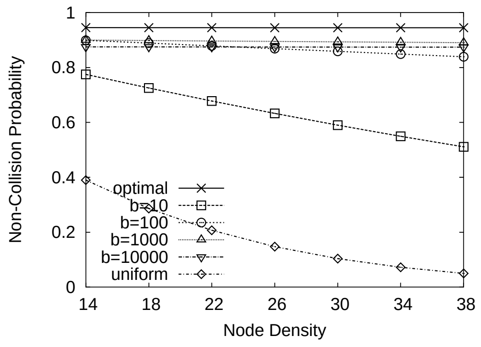


Fig. 8. Non-Collision Probability with Various b Values

A similar approach is also proposed in [46], which minimizes collisions in slotted CSMA. We deem that the optimal solution is more relevant for our MAC than for slotted CSMA. The slice time for slotted CSMA can be chosen as small as the sum of propagation delay, detection time and other processing delays, which are in microseconds typically. Compared with the maximum backoff time, the slice time is orders of magnitudes smaller, so the number of slices is large. When the slice number approaches infinity, the non-collision probability for a uniform distribution is

$$\lim_{T\to\infty} P_{nc} = \lim_{T\to\infty} (N \sum_{i=0}^{T-1} \frac{1}{T+1} (\frac{T-i}{T+1})^{N-1}).$$

Let $a = \frac{T-i}{T+1}$, from the definition of the Riemann integral, we have

$$\lim_{T\to\infty} P_{nc} = N \int_0^1 a^{N-1} da = N \cdot \frac{1}{N} = 1.$$

Therefore, when the slice number $T+1$ approaches positive infinity, the non-collision probability approaches 1, which means even the uniform distribution gives a very small chance of collision. Calculation shows that if we have 1000 time slices, even when 200 nodes compete, the non-collision probability for a uniform distribution is still above 90%. Since the slice number we use in MMSN is much smaller ($T+1 = 34$ in our simulation), to reduce protocol overhead, the suboptimal approach shows a significant performance improvement over the uniform distribution, as shown in Figure 8.

In our algorithm, we use the suboptimal approach for simplicity and generality. We need to make the distribution of the selected backoff time slice at each node conform to that shown in Equation (1). It is implemented as follows: first a random variable α with a uniform distribution within the interval $[0, 1)$ is generated on each node; then time slice i is selected according to the following equation:

$$i = \lfloor (T+1)\log_b[\alpha(b-1)+1] \rfloor.$$

It can be easily proven that the distribution of i conforms to Equation (1).

IV. PERFORMANCE EVALUATION

We implement MMSN in GloMoSim [35] and conduct extensive experiments to evaluate its performance and compare it with CSMA as well. In this evaluation, MMSN uses even selection for frequency assignment, since it results in fewer potential conflicts. For this performance evaluation, three groups of experiments are designed. In the first group, different traffic patterns are used. In the second group, different system loads are considered, and in the third group of experiments, the node density is varied.

For all the three groups of experiments, four performance metrics are adopted: aggregate MAC throughput, packet delivery ratio, channel access delay, and energy consumption. The aggregate MAC throughput measures the performance gain and is calculated as the total amount of useful data successfully delivered through the MAC layer in the system per unit time. The packet delivery ratio is calculated as the ratio of the total number of data packets successfully delivered by the MAC layer over the total number of data packets the network layer requests the MAC to transmit. The channel access delay measures the time delay a data packet from the network layer waits for the channel before it gets sent out. The energy consumption reflects the cost each protocol pays to achieve its performance, which is calculated as the energy consumed to successfully deliver a useful data byte. Since we have measured the cost for each frequency assignment scheme in Section III-B and this energy consumption is amortized during data transmission, it is no longer counted here.

TABLE I
SIMULATION CONFIGURATION

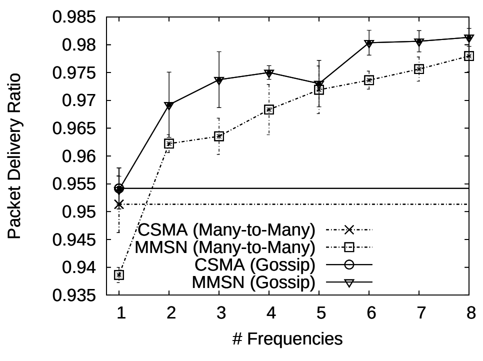
TERRAIN	(200m×200m) Square
Node Number	289
Node Placement	Uniform
Application	Many-to-Many/Gossip CBR Streams
Payload Size	32 Bytes
Routing Layer	GF
MAC Layer	CSMA/MMSN
Radio Layer	RADIO-ACCNOISE
Radio Bandwidth	250 Kbps
Radio Range	20m∼45m

During all the experiments, the Geographic Forwarding (GF) [47] routing protocol is used, and simulation is configured according to the settings in Table I. For each data value we present in the results, we also give its 90% confidence interval.

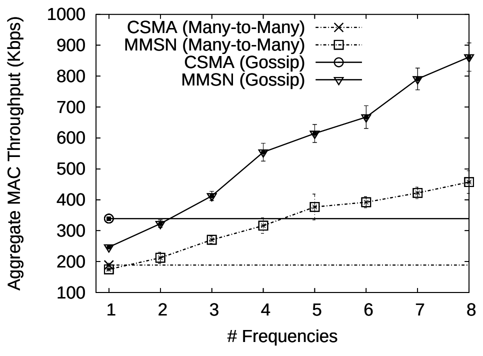
A. Performance Evaluation with Different Traffic Patterns

In the first group of experiments, two different traffic patterns are used, many-to-many and gossip traffic patterns. The many-to-many traffic pattern is used to simulate the typical sensor network application scenario: multiple sensor nodes report their readings to multiple base stations over multiple hops. Since the routing design affects the contention level at the MAC layer (e.g., hot spots), the MAC performance is more statistically valid when a simulation can isolate the effect from the routing layer. Therefore, we also evaluate the MAC performance with the gossip traffic pattern, in which each node only communicates with its neighbors. For both of these two traffic patterns, we increase the number of available frequencies, to observe the performance variation. In this group of experiments, 50 CBR streams are used and the node density is set to 38, by configuring the radio range to 40m. To achieve meaningful results, we evaluate the performance when the packet delivery ratio in the MAC layer is reasonably high, higher than 93%. The small amount of packet loss is due to hidden terminal problems [21].

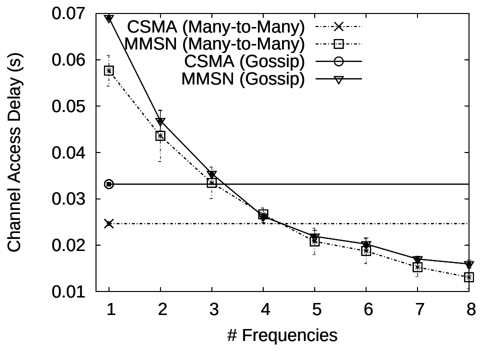
The performance results illustrated in Figure 9 confirm MMSN's scalability. When the number of frequencies increases from 1 to 8 and the gossip traffic is used, (a) illustrates that the packet delivery ratio increases from 95.4% to 98.1%, (b) shows that the aggregate MAC throughput increases from 246.9 Kbps to 861.8 Kbps, (c) informs that the average channel access delay decreases from 0.069s to



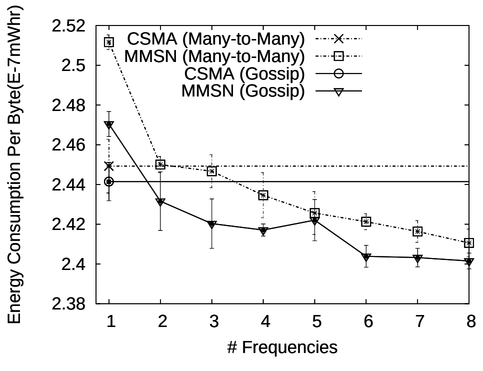
(a) Packet Delivery Ratio in MAC



(b) Aggregate Throughput in MAC



(c) Average Channel Access Delay



(d) Energy Consumption Per Delivered Data Byte

Fig. 9. Performance Evaluation with Different #Physical Frequencies

0.016s, and (d) states that MMSN becomes more energy efficient: the energy consumption per byte of successfully delivered data decreases from 2.47×10^{-7} mWhr to 2.40×10^{-7} mWhr. Similar performance increase is also exhibited when many-to-many traffic pattern is used. MMSN's performance increases, because available physical frequencies are evenly shared within two-hop neighboring nodes, and the increase of available frequencies leads to a higher degree of parallel data transmission within each neighborhood. When more physical frequencies are used, more nodes are able to conduct simultaneous transmission in the deployed system without collisions, so the aggregate MAC throughput increases. Plus, fewer nodes are assigned to use the same frequency within two hops. So communication interference decreases, which leads to less backoff and decreased channel access delay. Also, the decreased communication interference leads to less packet loss, and more useful data bytes are successfully delivered with the same amount of energy. On the other hand, MMSN does not achieve 8 times performance improvement when 8 frequencies are used compare to the case when one frequency is used. This is due to the fundamental hardware limitation of using a single transceiver in each sensor device.

Compared with CSMA, MMSN has similar or a little lower performance when the number of frequencies is small. This is because MMSN has a fixed backoff time period allocated within each time slot, while CSMA can fire the backoff timers at any time within the backoff window. However, when the number of frequencies increases, more parallel transmission within each neighborhood occurs and it results in more gains than the cost paid due to the fixed backoff period, and MMSN outperforms CSMA.

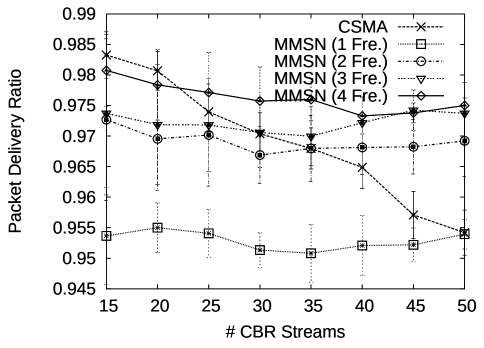
We are also aware that MMSN has constantly increasing aggregate MAC throughput when the gossip traffic pattern is used, while the speed of throughput increase slows down when the many-to-many traffic pattern is used. This is because the many-to-many traffic consists of a number of many-to-one traffic, in which multiple nodes transmit data packets to the same destination node. In this case, all these transmitters use the same physical frequency that the destination node gets assigned, and hence there is no potential parallel transmission that can be utilized. This is also one major difference between the single-transceiver[4] multi-frequency MMSN protocol and the multi-transceiver multi-frequency protocols proposed in [17] [18] [19].

B. Performance Evaluation with Different System Loads

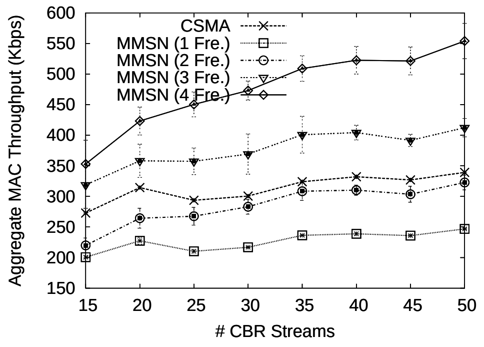
In the second group of experiments, we explore MMSN's performance when different system loads are used, which are generated by different numbers of CBR streams. To analyze performance scalability, we conduct all experiments with different numbers of frequencies as well. In the experiments, the node density is set to 38, and the gossip traffic pattern is used.

As Figure 10 shows, for all the system loads we configure from 15 CBR streams to 50 CBR streams, it is observed that MMSN always exhibits better performance when more

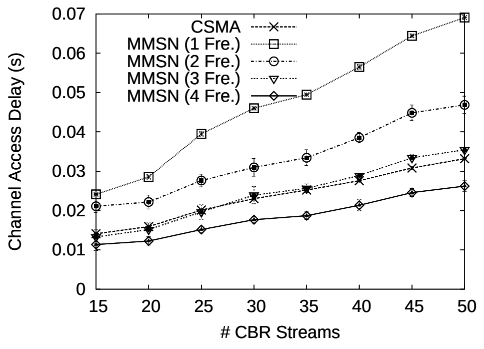
[4]One solution is for each base state to have multiple transceivers. The multiple transceivers snoop on different frequencies, so that the base station can receive simultaneous data reporting from multiple nodes.



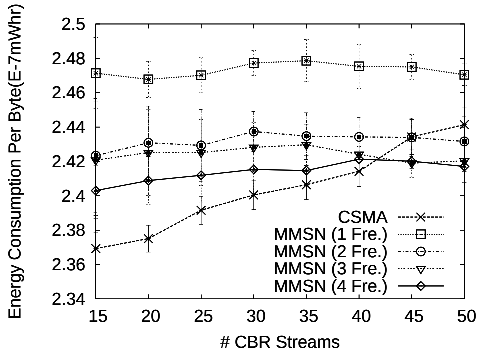
(a) Packet Delivery Ratio in MAC



(b) Aggregate Throughput in MAC



(c) Average Channel Access Delay



(d) Energy Consumption Per Delivered Data Byte

Fig. 10. Performance Evaluation with Different System Loads

frequencies are used, which is consistent with the result presented in the previous group of experiments. For example, as shown in Figure 10, when the number of frequencies increases from 1 to 4 and 40 CBR streams are used, MMSN's packet delivery ratio increases from 95.2% to 97.3% in (a). At the same time, MMSN's aggregate MAC throughput increases by 119% from 239Kbps to 523Kbps as shown in (b), and the channel access delay decreases to 0.021s, which is 37.5% of the delay when only 1 frequency is available, as shown in (c). In such a case, (d) also informs that MMSN's energy consumption for each successfully delivered data byte decreases from 2.48×10^{-7}mWhr to 2.42×10^{-7}mWhr. MMSN achieves improved performance when the number of frequencies increases, because the increased frequencies lead to increased parallel transmission within the same neighboring space and to decreased congestion for the same physical frequency.

Figure 10 (a) also shows that CSMA has a decreased packet delivery ratio from 98.3% to 95.4%, while MMSN does not have such an obvious packet loss. This is because the non-uniform backoff algorithm design is more tolerant to the system load variation than the uniform backoff algorithm. The sharply increased system load, from 15 CBR streams to 50 CBR streams, leads to more congestion and more packet loss in CSMA while the slotted backoff is not impacted as much. In (b), the aggregate MAC throughput increases with the increase of system load, because more nodes get involved in communication and more parallel data transmission occurs. In addition, the increased nodes becoming involved in communication result in increased congestion and hence increased channel access delay increases in (c). Since CSMA is more sensitive to system load and has lower packet delivery ratio, it is less energy efficient when the system load increases, while MMSN' packet delivery ratio is more tolerant to system load and hence does not exhibit apparent decrease of energy efficiency.

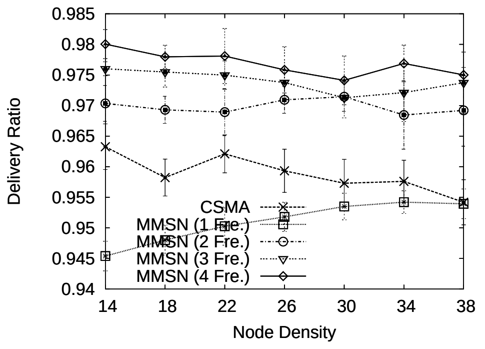
For similar reasons as explained in the previous experiments, MMSN is observed to have a lower performance than CSMA when there is only one, or two in some cases, physical frequencies available as shown in Figure 10. However, MMSN outperforms CSMA when three or more frequencies are used, which is also exhibited in Figure 10.

C. *Performance Evaluation with Different Node Densities*

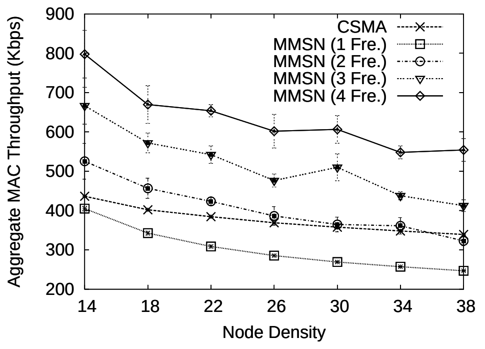
In many deployed sensor network systems [30] [31] [32] [33], providing node redundancy is an efficient and effective method to increase the system lifetime. So, in the third group of experiments, we evaluate MMSN's performance when different node densities are utilized. The node density is increased from 14 to 38, by configuring different radio ranges, and a gossip traffic pattern is used that consists of 50 CBR streams. We also measure the performance difference when different numbers of frequencies are used as well.

Once again, the experimental results confirm that MMSN always achieves a higher performance when more frequencies are available, which can be observed in Figure 11 (a)$\sim$(d). The corresponding reasons can be found in the first two groups of experiments and are not repeated here.

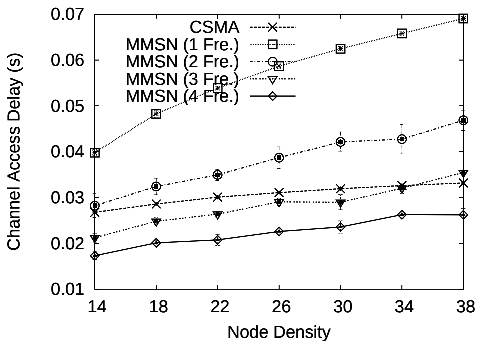
From Figure 11 (b), it is observed that the aggregate MAC throughputs in both CSMA and MMSN decrease with the



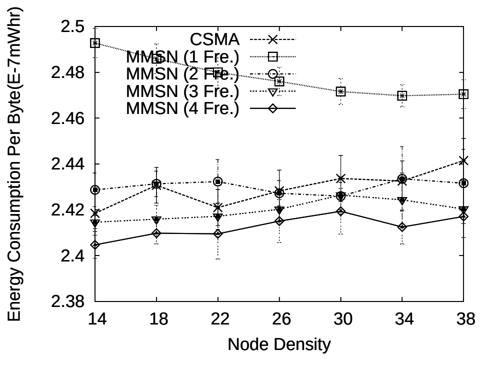
(a) Packet Delivery Ratio in MAC



(b) Aggregate Throughput in MAC



(c) Average Channel Access Delay



(d) Energy Consumption Per Delivered Data Byte

Fig. 11. Performance Evaluation with Different Node Density

increase of node density. This is because when node density increases, the same number of frequencies are shared by more nodes within two hops. When the same percentage of nodes participate in communication, congestion is increased and hence backoff and channel access delay are increased, as shown in Figure 11 (c). We do not observe consistent trends for packet delivery ratio variation in (a) and energy consumption variation in (d), when the number of frequencies is greater than one and the node density is increased from 14 to 38. But we do notice that when there is only one frequency, the packet delivery ratio of MMSN increases in (a), with the increase of node density. We think this is because of decreased hidden terminal problems, when the radio range gets increased to increase node density, while at the same time the system topology is fixed to be 200m×200m. When the number of frequencies increases, this effect becomes very small and no similar trend is observed. Also, because of the increased packet delivery ratio, the energy consumption becomes more efficient as shown in (d), when MMSN uses one frequency and the node density increases from 14 to 38.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we propose the first effort to design a multi-frequency MAC protocol for wireless sensor network applications, in which each device adopts a single radio transceiver. The different MAC design requirements for wireless sensor networks and general wireless ad hoc networks are compared, and a complete WSN multi-frequency MAC design (MMSN) is put forth. During the MMSN design, we analyze and evaluate different choices for frequency assignment, and also discuss the non-uniform backoff algorithms for the slotted media access design. Finally, we evaluate MMSN's performance through extensive experiments, and the performance results show that MMSN exhibits prominent ability to utilize parallel transmission among neighboring nodes. MMSN also achieves increased energy efficiency when multiple physical frequencies are available.

In the future, we plan to implement MMSN in a large scale running sensor network system and evaluate its performance with different sensor devices. In addition, the current work assumes that we have access to multiple well separated channels. In future work, we also plan to extend MMSN to the case when the multiple channels have partially overlapping [48] frequency bandwidths.

ACKNOWLEDGEMENT

This work is supported by the National Science Foundation, under grant CNS-0435060, grant CCR-0325197 and grant EN-CS-0329609. The authors would like to thank the anonymous reviewers of INFOCOM for their valuable comments for the camera-ready version of this work.

REFERENCES

[1] D. Culler, D. Estrin, and M. Srivastava, "Overview of Sensor Networks," in *IEEE Computer, Special Issue on Sensor Networks*, August 2004.

[2] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," in *ACM MobiCom 1999*, August 1999.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: a Survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[4] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Median Access for Wireless Sensor Networks," in *ACM SenSys 2004*, November 2004.

[5] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *IEEE INFOCOM 2002*, June 2002, pp. 1567–1576.

[6] V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves, "Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks," in *ACM SenSys 2003*, November 2003.

[7] T. Dam and k. Langendoen, "An Adaptive Evergy-Sufficient MAC Protocol for Wireless Sensor Networks," in *ACM SenSys 2003*, November 2003.

[8] A. Woo and D. Culler, "A Transmission Control Scheme for Media Access in Sensor Networks," in *ACM MobiCom 2001*, July 2001.

[9] A. El-Hoiyi, J.-D. Decotignie, and J. Hernandez, "Low Power MAC Protocols for Infrastructure Wireless Sensor Networks," in *The Fifth European Wireless Conference*, February 2004.

[10] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System Architecture Directions for Networked Sensors," in *The Nineth International Conference on Architectural Support for Programming Languages and Operating Systems*, November 2000, pp. 93–104.

[11] "XBOW MICA2 Mote Specifications," http://www.xbow.com.

[12] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling Ultra-Low Power Wireless Research," in *ACM/IEEE IPSN/SPOTS 2005)*, April 2005.

[13] "CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver," http://www.chipcon.com.

[14] Z. Tang and J.J. Garcia-Luna-Aceves, "Hop-Reservation Multiple Access (HRMA) for Ad-Hoc Networks," in *IEEE INFOCOM 1999*, March 1999.

[15] A. Tyamaloukas and J. J. Garcia-Luna-Aceves, "Channel-Hopping Multiple Access," in *IEEE ICC 2000*, 2000.

[16] J. Deng and Z. Haas, "Dual Busy Tone Multiple Access (DBTMA): A New Medium Access Control for Packet Radio Networks," in *IEEE ICUPC*, October 1998.

[17] A. Nasipuri, J. Zhuang, and S. R. Das, "A Multichannel CSMA MAC Protocol for Multihop Wireless Networks," in *IEEE WCNC*, September 1999.

[18] S.-L. Wu, C.-Y. Liu, Y.-C. Tseng, and J.-P. Shen, "A New Multi-Channel MAC Protocol with On-Demand Channel Assignment for Multi-Hop Mobile Ad Hoc Networks," in *I-SPAN*, 2000.

[19] A. Nasipuri and S. R. Das, "Multichannel CSMA with Signal Power-based Channel Selection for Multihop Wireless Networks," in *IEEE Vehicular Technology Conference*, September 2000.

[20] M. Caccaco, L. Y. Zhang, L. Sha, and G. Buttazzo, "An Implicit Prioritized Access Protocol for Wireless Sensor Networks," in *IEEE RTSS 2002*, December 2002.

[21] "IEEE 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification," 1999, ANSI/IEEE Std. 802.11, 1999.

[22] P. Bahl, R. Chancre, and J. Dungeon, "SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks," in *ACM MobiCom 2004*, September 2004.

[23] A. Raniwala and T. Chiueh, "Architecture and Algorithm for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network," in *IEEE INFOCOM 2005*, March 2005.

[24] A. Adya, P. Bahl, J. Padhye, A.Wolman, and L. Zhou, "A Multi-Radio Unification Protocol for IEEE 802.11 Wireless Networks," in *IEEE Broadnets 2004*, 2004.

[25] F. Fitzek, D. Angelini, G. Mazzini, , and M. Zorzi, "Design and Performance of an Enhanced IEEE 802.11 MAC Protocol for Multihop Coverage Extension," in *IEEE Wireless Communications*, 2003.

[26] J. Li, Z. J. Haas, M. Sheng, , and Y. Chen, "Performance Evaluation of Modified IEEE 802.11 MAC for Multi-Channel Multi-Hop Ad Hoc Network," in *IEEE AINA 2003*, 2003.

[27] J. So and N. Vaidya, "Multi-Channel MAC for Ad-Hoc Networks: Handling Multi-Channel Hidden Terminal Using A Single Transceiver," in *ACM MobiHoc 2004*, May 2004.

[28] N. Jain and S. R. Das, "A Multichannel CSMA MAC Protocol with Receiver-Based Channel Selection for Multihop Wireless Networks," in *IEEE IC3N*, October 2001.

[29] A. Tzamaloukas and J.J. Garcia-Luna-Aceves, "A Receiver-Initiated Collision-Avoidance Protocol for Multi-Channel Networks," in *IEEE INFOCOM 2001*, April 2001.

[30] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson., "Wireless Sensor Networks for Habitat Monitoring," in *ACM WSNA 2002*, September 2002.

[31] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A Wireless Sensor Network For Structural Monitoring," in *ACM SenSys 2004*, November 2004.

[32] T. He, S. Krishnamurthy, J. A. Stankovic, T. F. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh, "Energy-Efficient Surveillance System Using Wireless Sensor Networks," in *ACM MobiSys 2004*, June 2004, pp. 270–283.

[33] G. Simon, M. Maróti, Á. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton, "Sensor Network-Based Countersniper System," in *ACM SenSys 2004*, November 2004.

[34] Y. Li, H. Wu, D. Perkins, N.-F. Tzeng, and M. Bayoumi, "MAC-SCC: Medium Access Control with a Separate Control Channel for Multihop Wireless Networks," in *IEEE ICDCSW 2003*, 2003.

[35] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks," in *The 12th Workshop on Parallel and Distributed Simulations*, May 1998.

[36] G. Zhou, T. He, J. A. Stankovic, and T. F. Abdelzaher, "Radio Interference Detection in Wireless Sensor Networks," in *IEEE INFOCOM 2005*, March 2005.

[37] J. Zhao and R. Govindan, "Understanding Packet Delivery Performance in Dense Wireless Sensor Networks," in *ACM SenSys 2003*, November 2003.

[38] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Impact of Radio Irregularity on Wireless Sensor Networks," in *ACM MobiSys 2004*, June 2004, pp. 125–138.

[39] A. Woo, T. Tong, and D. Culler, "Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks," in *ACM SenSys 2003*, November 2003.

[40] A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, and D. Estrin, "Statistical Model of Lossy Links in Wireless Sensor Networks," in *ACM/IEEE IPSN'05*, April 2005.

[41] J. Chang and N. F. Maxemchuk, "Reliable Broadcast Protocols," in *ACM Transactions on Computer Systems (TOCS)*, 1984.

[42] J. Tourrilhes, "Robust Broadcast: Improving the Reliability of Broadcast Transmissions on CSMA/CA," in *IEEE PIMRC 1998*.

[43] E. Vollset and P. Ezhilchelvan, "A Survey of Reliable Broadcast Protocols for Mobile Ad-Hoc Networks," in *Technical Report CS-TR-792, University of Newcastle upon Tyne*, 2003.

[44] L. Bao and J. J. Garcia-Luna-aceves, "A New Approach to Channel Access Scheduling for Ad Hoc Networks," in *ACM MobiCom 2001*, July 2001, pp. 210–221.

[45] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The Flooding Time Synchronization Protocol," in *ACM SenSys 2004*, November 2004.

[46] Y. C. Tay, K. Jamieson, and H. Balakrishnan, "Collision-Minimizing CSMA and Its Applications to Wireless Sensor Networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 1048–1057, 2004.

[47] B. Karp, *Geographic Routing for Wireless Networks*, Ph.D. thesis, Harvard University, Cambridge, MA, 2000.

[48] A. Mishra, E. Rozner, S. Banerjee, and W. Arbaugh, "Exploiting Partially Overlapping Channels in Wireless Networks: Turning a Peril into an Advantage," in *ACM/USENIX IMC 2005*, October 2005.

Efficiency Centric Communication Model for Wireless Sensor Networks

Qing Cao[†], Tian He[‡], Lei Fang, Tarek Abdelzaher[†], John Stankovic, Sang Son
Department of Computer Science, University of Virginia, Charlottesville VA 22904, USA

***Abstract*— Recent studies on radio reality provided strong evidence that radio links between low-power sensor devices are extremely unreliable. In this paper, we study how to improve energy efficiency for reliable communication using such unreliable links. We identify an optimal bound on energy efficiency for reliable communication, and propose a new communication model in the link layer that asymptotically approaches this bound. This new model indicates a better path metric compared to previous path metrics, and we validate this by establishing a routing infrastructure based on this metric, which indeed achieves a higher energy efficiency compared to other state-of-the-art approaches. We present results from a systematic analysis, simulations and prototype experiments based on the MicaZ platform. The results give us fundamental insights on communication efficiency over unreliable links.**

I. Introduction

The evolution of radio models has consistently shaped the upper layers of the communication stack in wireless sensor networks. Early communication stacks were usually designed based on over-simplified radio assumptions, such as the unit disk graph model. Recently, this model has been repeatedly challenged by empirical measurements. For example, studies in [31], [26], [8] suggested that wireless links are irregular and unreliable. These studies also observed highly different packet delivery ratios for the same link in reverse directions. The design of communication stacks must take into account these radio layer realities.

Motivated by these observations, we focus on how to minimize overhead while providing reliable end-to-end communication over these unreliable links. Reliable communication is important for sensor networks despite the wide use of aggregation techniques, because realistic applications often use alarms and other one-time event notifications that need to be communicated reliably. The successful delivery of such information has a direct effect on the overall performance of the system.

Formally, we model an unreliable link between nodes A and B as (p, q), where p represents the packet delivery ratio from A to B, and q represents the packet delivery ratio from B to A. When p and q are less than 100%, we must use a packet recovery mechanism, such as retransmissions and acknowledgements, to achieve reliability. The problem of minimizing overhead, therefore, is equivalent to minimizing the additional traffic compared to the *ideal* scenario where every link is perfect. To model this traffic, we introduce a new parameter, Energy Per Bit, or EPB, to characterize the energy efficiency aspect of communication. EPB represents the average energy consumption for each *delivered* bit from the source to the destination. EPB is decided by several factors, such as the link layer packet recovery mechanism, the routing layer path selection, the relative positions of the source and the destination, and the network topology, among others. Some of these factors, such as the relative positioning of the source and the destination, are unique to a particular transport task. Therefore, we do not aim to optimize EPB across different transport tasks. Rather, we focus on optimizing EPB for a given transport task, for example, node A sends 1000 bytes to B through multiple hops.

The optimization presented in this paper is a joint optimization process in two layers: in the link layer, it optimizes how lost packets are detected; in the routing layer, it optimizes how paths are selected. More specifically, we present two corresponding techniques: 1) the *lazy* packet loss detection in the link layer and 2) the use of a stream based path metric in the routing layer. The first optimization technique applies to a particular chosen path, while the second one applies to the path selection process. These two optimization techniques are unified by their consistency: we first analyze the first optimization technique and demonstrate its effectiveness in reducing EPB given a particular path. Based on the analysis, we distill a path metric that features a stream nature, interacts with the commonly used spanning tree routing structure, and leads to the second optimization technique. Therefore, by jointly applying these two techniques, we further reduce EPB, as validated by an extensive evaluation.

Our optimization techniques have applications in a wide variety of scenarios. One of them is surveillance [27], [3]. In this type of applications, sensed data, such as temperature, light or pressure readings, are periodically sampled and relayed to a central data collection node, referred to as the base station. Normally, the data reporting rate is relatively low, and timing requirements are not strict. Since data are generated periodically, traffic naturally exhibits a streaming nature (i.e., a data flow to the base via multiple hops for a long period of time). Our optimization techniques can therefore be readily applied to such applications.

The rest of this paper is organized as follows. Section II presents the lazy lost packet recovery mechanism. We then derive a general *stream path metric* based on analysis of this mechanism. Section III presents a systematic performance

† Qing Cao and Tarek Abdelzaher are now with the University of Illinois, Urbana-Champaign.
‡ Tian He is now with the University of Minnesota.

evaluation of the mechanism from two aspects: the end-to-end delay and the buffer requirements. Section IV integrates the stream metric into the routing layer design, leading to the second optimization technique. We compare its performance to two state-of-the-art protocols that take into account the unreliable nature of sensor network communication. We demonstrate that, by using the joint optimization, our protocol stack achieves a considerably better EPB value than either of them. At last, we outline related work in Section V and conclude this paper with Section VI.

II. Lazy Lost Packet Recovery

We now describe the first optimization technique, called lazy lost packet recovery. This section is organized into three parts. First, we describe different link quality metrics based on empirical experiments. Next, we present the design of the optimization algorithm, and demonstrate that it approaches the *optimal* efficiency bound. At last, we discuss applications of reliable packet delivery in sensor networks.

A. Link Quality Metrics Overview

Link quality metrics are used to classify and select links. Prior measurements on link quality reveal that different links have considerably different packet reception properties [30]. One popular model is to treat a link as a bi-directional packet reception probability vector (p, q). However, reception probability is not the only representation of link quality. Recently, another metric, LQI, or Link Quality Indicator, was defined by IEEE standard 802.15.4 [1], and was implemented on the Chipcon CC2420 radio component [7]. The CC2420 radio has been used on MicaZ and Telos nodes. Another interesting link quality indicator is $RSSI$ (Received Signal Strength Indicator), also implemented on the CC2420 radio. In this section, we demonstrate that LQI is closely correlated with packet reception probability, and either of them can be used as the underlying metric of our optimization model. On the other hand, our experiments demonstrate that $RSSI$ is not a satisfactory indicator of link quality, and should not be adopted for link classification.

The implementation of LQI, as defined in [1], was that it "may be implemented using receiver ED (energy detection), a signal-to-noise ratio estimation, or a combination of these methods." The CC2420 radio module implemented LQI based on a sampling of the error rate for the first eight symbols of each incoming packet. This sampling generates a correlation value in the range of $[50, 110]$, followed by a linear conversion of this value to a range of $[0, 255]$, which is the value provided to the user.

$RSSI$ is not detailed in the 802.15.4 standard, but it is provided by the CC2420 radio module together with LQI. $RSSI$ is also based on eight symbols, but instead of using the error rate, it uses the average energy level to calculate its value.

We carried out a series of experiments with both indoor and outdoor environments during different periods of the day. Due to space limitations, we only present one representative set of data in Figure 1.

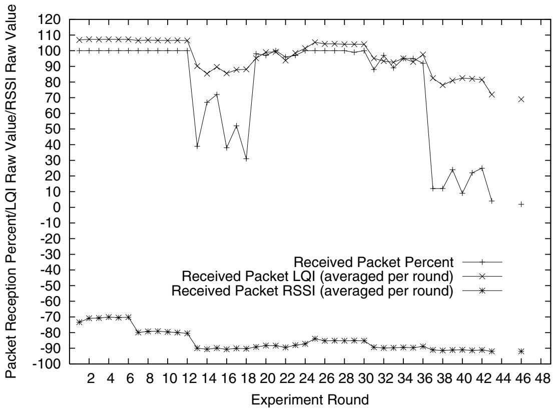


Fig. 1. Correlation between the packet receiving ratio and LQI of CC2420 radio at receiver side. At each distance, six rounds of packets are sent from the sender to the receiver.

In this particular experiment, we use a pair of MicaZ nodes, one as the sender and the other as the receiver. We vary the distance from the receiver to the sender from $5ft$ to $40ft$, in steps of $5ft$. At each distance, the receiver sends six rounds of packets, with 100 packets in each round. The packet reception ratio, the average LQI and the average $RSSI$ are plotted for comparison. Both LQI and $RSSI$ are calculated in every round.

We have three observations. First, we observe a strong correlation between the averaged LQI values and packet reception probabilities at the receiver. Statistical analysis shows that the Pearson's correlation coefficient is 0.90 between these two variables. Of course, there are still some inconsistencies observed, especially when the received signal is weak. These inconsistencies explain why the Pearson' s coefficient is not 1.0. Nevertheless, the observed correlation is still quite interesting, since LQI is calculated only from those packets that are *received*, whereas the packet reception probability takes into account those packets that are *dropped*. This correlation implies that LQI is a good measurable indicator of the packet reception probability. Second, at each distance, the packet reception probability has a narrow range of variation that is less than 20%. Our additional experiments also exhibit the same property. This observation is consistent with results in the literature [26]. We, therefore, consider the packet reception probability as a relatively stable parameter to classify links. Third, observe that there is a much smaller correlation between $RSSI$ and the packet reception probability. The Pearson's correlation coefficient is only 0.56 between the packet reception probability and the $RSSI$ value. Furthermore, observe that when the signal is weak, even though there is a considerable variation in the packet loss rate, $RSSI$ does not change appreciably. Therefore, we do not recommend using $RSSI$ as a reliable link quality indicator.

B. The Design of the Lazy Packet Loss Detection

We now describe the lazy packet loss detection algorithm. The optimization goal is to minimize EPB. Since this algorithm works in the link layer, we assume a chosen path. For example, we consider the scenario where the source sends 1000 packets to the destination via certain hops. Observe that the amount of meaningful traffic is fixed. Therefore, the optimization goal is equivalent to minimizing the overhead. This goal can also be expressed as maximizing the fraction of non-redundant data packets.

We now introduce the path efficiency parameter, η. Formally, for a path with N hops, let U be the total useful traffic delivered in bits, and S be the total amount of bits transmitted from *all* nodes on this path. We have:

$$\eta = \frac{UN}{S} \tag{1}$$

Clearly, for a fixed path, η and EPB are inversely proportional. Therefore, *for a given path*, we want to maximize η. Note that, however, η can not replace EPB when the path is not fixed. For example, one way to maximize η without a given path is to select only *strong* links (links with high reception probabilities), so that very few packets will be lost. In this case, we obtain a high η value. On the other hand, by only choosing strong links, we are essentially accepting an excessively large number of hops, which increases EPB.

The rest of this section optimizes η. First, we consider the upper bound of η. Next, we present the lazy lost packet detection algorithm. At last, we show that the reliable communication model, assumed in this paper, generally achieves a better η than unreliable models.

1) Path Efficiency Upper Bound: We now derive the upper bound of η. A simplified model is shown in Figure 2.

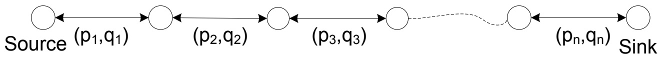


Fig. 2. The simplified link model

In this model, we assume a path of n links. Suppose link L_i has a forward packet reception probability of p_i and a backward packet reception probability of q_i. Observe that for a single link, we have:

The path efficiency η over a single link cannot be higher than its forward packet reception probability p.

The reasoning is intuitive: since useful traffic can only flow *forward*, if the sender sends N packets, only pN packets can be delivered successfully. Therefore, even if (though this is not practical) the sender knows, without any additional cost, which packets are lost and re-transmits them, an upper bound of p nevertheless holds for η.

Next, we consider the path efficiency over multiple links. Suppose the sender intends to send K bits to the destination over n hops. Suppose the total traffic flowing over link L_i in both directions is M_i bits. For link L_i, we have:

$$\eta_i = \frac{K}{M_i} \leq p_i \tag{2}$$

On the other hand, we know the path efficiency η_{path} over n hops is (according to Equation 1):

$$\eta_{path} = \frac{nK}{\sum_{i=1}^{n} M_i} \tag{3}$$

Let the maximal p_i for $1 \leq i \leq n$ be p_{max}, we have:

$$\eta_{path} = \frac{n}{\sum_{i=1}^{n} \frac{1}{\eta_i}} \leq \frac{n}{\sum_{i=1}^{n} \frac{1}{p_i}} \leq p_{max} \tag{4}$$

A tight upper bound on path efficiency η over multiple hops therefore is $\frac{n}{\sum_{i=1}^{n} \frac{1}{p_i}}$. Using this bound, we can investigate the available reliable communication models. Surprisingly, few have considered the aspect of efficiency and some intuitive solutions can *never* approach this bound. Such is the case with the popular timeout-based solution.

In a timeout-based solution, the sender generally relies on a timer to control retransmissions. More specifically, if the sender does not receive an acknowledgement from the receiver when the timer fires, it assumes that either the data packet or the acknowledgement is lost. Therefore, it sends the data packet again. This solution is intuitive, elegant and has been proved to be robust in practice. However, despite its merits, we argue that this timeout-based design is inherently less advantageous for sensor networks for efficiency reasons. Specifically, let us consider the number of packets it takes for transmitting one data packet reliably for one hop over a link of reception probabilities of (p, q) using the timeout-based protocol. Since the combined packet delivery success ratio for a round of packet exchanges (i.e., for a pair of data and acknowledgement packets), is pq, the sender is expected to send the data packet $1/pq$ times before both the data packet and the corresponding acknowledgement packet are delivered successfully. Since the receiver only acknowledges those data packets it receives, it is expected to send $1/pq \times p$, or $1/q$ acknowledgements. Let the packet length ratio between the acknowledgement packet and the data packet be λ. The path efficiency η over this link becomes $\frac{pq}{1+p\lambda}$. As $\lambda \to 0$, or, if the data packet is sufficiently large compared to the acknowledgement packet, η approaches pq. Compare this result with Equation 2, as long as the backward link is not perfect, this efficiency value is always smaller than its upper bound.

2) Lazy Lost Packet Detection based Communication: So why does timeout-based design fail to approach the upper bound? Observe that in this design, the key mechanism the sender relies on to detect a packet loss is the acknowledgment packets from the receiver. Unfortunately, the backward link is not perfect, therefore, the sender wastes bandwidth by retransmitting data packets that have been successful. To avoid this problem, therefore, the sender must use additional information, other than the acknowledgements from the receiver, to detect packet losses. In the design described in this section, we use a combination of two techniques, overhearing and sequence number counting, to achieve this purpose.

We present three aspects of the new communication model: lazy loss detection, implicit acknowledgement, and path efficiency analysis. The basic mechanism is shown in Figure 3.

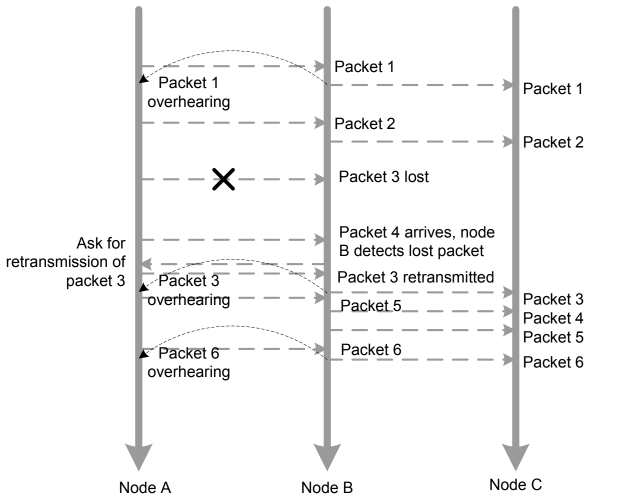


Fig. 3. This example shows the basic principle of lazy loss detection. At each node, packets are sent out in order, although packets may not be *received* in order. In this example, A sends out 6 packets in total, and packet 3 is lost when first sent from A to B. B then detects this packet loss after receiving a following packet using sequence number counting, and asks A to retransmit it. Packet 3 is then retransmitted.

Lazy Loss Detection In lazy loss detection, the sender does not employ any timeout mechanism. Rather, the detection of lost packets is delayed to the moment when the receiver gets another packet from the sender. For example, in Figure 3, the loss of packet 3 is not detected until B receives packet 4 from node A. We therefore call this loss detection *lazy*. The receiver then recovers the lost packets by sending retransmission request packets (RRPs). Since the link from B to A is also not perfect, it may take multiple RRPs to inform A of the lost packet. Therefore, B should periodically resend RRPs, with an interval larger than the round-trip time, until it recovers the lost packet. Obviously, to maintain the correct sequence of packets, we enforce that new packets received by B during this period should be temporarily buffered.

Implicit Acknowledgement To respond RRPs, a node must buffer the lost packets. Hence, a node deletes a packet it sent out earlier only when it decides that this packet has indeed been received by the receiver. In practice, the sender relies on *implicit* acknowledgements. These acknowledgements come from two sources. First, based on the broadcasting nature of wireless links, A can usually overhear the packets sent out by B to downstream nodes. Such overhearing is the first source of implicit acknowledgement That is, once A overhears a packet it sent out earlier being relayed, it can safely delete it from its buffer. For example, in Figure 3, node A can delete packet 1 after it overhears it sent by node B. Second, observe that RRPs can also serve as acknowledgements, as long as RRPs for different lost packets are kept in order. That is, once B decides that a packet has been lost, B should stop relaying new packets, and immediately switch to sending RRPs. When A receives a RRP, it can decide that 1) all packets prior to this lost one must have been successfully received and 2) it should resend the requested (lost) packet. RRPs therefore serve as implicit acknowledgements. For example, in Figure 3, when B sends RRPs to A regarding the lost packet 3, A decides that previous packets must have been received by B, and can safely delete them from the buffer.

Observe that some techniques presented here, such as the packet sequence number counting and the implicit acknowledgement, work only if A continues to send packets to B. For the last few packets in a stream, the sender should either switch to sender-timeout mechanism for packet-loss detection, or it can send additional dummy packets to the receiver. The dummy packets should assume higher sequence numbers, so that the receiver can correctly diagnose potential packet losses.

Path Efficiency Analysis We now analyze path efficiency η in the lazy communication model. We define λ as the length ratio between a RRP packet and a data packet. For one link, if the first packet transmission is successful, which has a probability of $1-p$, there are no retransmissions needed. Otherwise, when the receiver detects a packet loss, it sends a request for retransmission. For a link with a delivery ratio of (p,q), the number of RRPs, after a packet loss, conforms to a geometric distribution with parameter pq, and the average number of RRPs sent is $1/pq$. On the sender side, since it only responds to those RRPs it receives, the average number of retransmissions for a lost data packet is $1/p$. Therefore, the path efficiency is:

$$\eta = \frac{1}{1/p + (1-p)\lambda/pq} = \frac{pq}{q + (1-p)\lambda} \tag{5}$$

The interesting fact regarding Equation 5 is that if the length of data packets is sufficiently large compared to the length of RRPs, or, if $\lambda \to 0$, $\eta \to p$. Remember that p is the upper bound for efficiency over a single link. Therefore, we have shown that lazy packet loss detection overcomes the disadvantage of timeout-based mechanisms and approaches the path efficiency upper bound. Additionally, this result indicates that it is beneficial to use various techniques, such as data aggregation, to increase the length of data packets and decrease λ.

We can also easily get the expected EPB value over this link:

$$EPB = p \times 1 + (1-p) \times (1 + \frac{1}{p} + \frac{\lambda}{pq}) = \frac{1}{p} + \frac{1-p}{pq}\lambda \tag{6}$$

EPB represents energy consumption. We demonstrate that once this metric is incorporated into the routing layer design, it can significantly improve the path energy efficiency of data streams (which we call the *stream model*.

C. Why Reliable Packet Delivery is Important

So far, we have assumed our communication model to be reliable. In this section we explain why. We present a comparison between three communication models on path efficiency over multiple hops: our new model (denoted *stream*), the timeout-resend model (denoted *timeout*), and the best effort, non-reliable data communication model (denoted *noack*). For the end-to-end path efficiency, we have:

$$\eta_{stream} = \frac{N}{\sum_{i=1}^{N} \frac{q_i + (1-p_i)\lambda}{p_i q_i}} \tag{7}$$

$$\eta_{timeout} = \frac{N}{\sum_{i=1}^{N} \frac{1+p_i\lambda}{p_i q_i}} \tag{8}$$

$$\eta_{noack} = \frac{p_1 \cdot p_2 \cdots p_n \cdot N}{1 + p_1 + p_1 \cdot p_2 + \cdots + p_1 \cdot p_2 \cdots p_{n-1}} \tag{9}$$

For an intuitive understanding of their differences, assume the link quality p over different links can be approximated by $\tilde{p}$, and q approximated by $\tilde{q}$. Indeed, such an approximation is highly simplified, and we shall take into account quality reception variations in Section III-A. In the simplified case, we have:

$$\eta_{stream} = \frac{\tilde{p}\tilde{q}}{\tilde{q} + (1-\tilde{p})\lambda} \tag{10}$$

$$\eta_{timeout} = \frac{\tilde{p}\tilde{q}}{1 + \tilde{p}\lambda} \tag{11}$$

$$\eta_{noack} = \frac{(1-\tilde{p}) \cdot \tilde{p}^N \cdot N}{1 - \tilde{p}^N} \tag{12}$$

These results show one critical difference between the best effort communication model (Equation 12) and the reliable communication model (Equation 10,11): the path efficiency of the best effort model is relevant to the number of hops, whereas the path efficiency of the reliable model is not. Therefore, the best effort communication model is *not scalable* to long paths. This fact is even more intuitive if we make a further simplification by enforcing that $\lambda \to 0$ and $N \to \infty$. We get:

$$\eta_{stream} \to \tilde{p} \ \ (\lambda \to 0) \tag{13}$$

$$\eta_{timeout} \to \tilde{p}\tilde{q} \ \ (\lambda \to 0) \tag{14}$$

$$\eta_{noack} \to 0 \ \ (N \to \infty) \tag{15}$$

This result shows that as the packet travels over many hops, the communication efficiency of the best effort communication model approaches 0! This fact also implies that virtually any conventional design that primarily relies on end-to-end loss recovery techniques can not be ported to sensor networks, due to scalability issues.

Arguably, sensor networks represent a new paradigm for communication where sometimes it is difficult to motivate reliable communication. For example, data may be aggregated along the path. The network may indeed tolerate a certain amount of packet losses without significantly affecting the aggregate. However, for the system to scale, the fraction of delivered packets should remain finite and representative of the original pool, which means that a certain degree of reliability is needed to prevent end-to-end communication efficiency from dropping below a certain minimum with an increased hop count.

III. Understanding the Performance of the Stream Communication Model

We now present a detailed investigation on the performance of communication model, which we call the *stream communication model*. We analyze performance from three aspects. First, we study path efficiency through empirical experiments, and validate the performance advantages of the new approach. Next, we analyze end-to-end delay, and validate our results using simulations. At last, we analyze buffer size requirements.

A. Empirical Validation of the Communication Model

To compare the path efficiency in realistic settings, we used MicaZ nodes to test our communication model. In the experiment, sixteen nodes (labeled 1 to 16) were placed in an indoor hallway. The experiments were carried out at midnight to avoid external interference. The output power level of each node was set to be $-7dBm$. In the experiment, each node first sent periodic beacons to neighbors to determine the packet reception probability. This process took 2000 packets for each node over a period of 100 seconds. Then, node 1 sent out 3200 packets, hop by hop, to node 16. We enforced that nodes relay the packets sequentially, that is, node 1 sent packets to 2, 2 to 3, etc, at a rate of 2 packets per second. We deliberately chose a low rate to avoid any potential interference, so that the effect of unreliable links can be isolated from that of congestion. Each packet had a payload of 29 bytes, so the overall useful traffic was approximately $100K$ bytes. We implemented the three different communication models mentioned previously and ran this experiment using each of them separately. Each node logged the number of transmissions, retransmissions, requests for retransmissions, and acknowledgements into its flash. These data were collected after the experiments. The results presented here are based on the analysis of these data.

First, we determine the link qualities connecting these nodes. The result is plotted in Figure 4. Note that we only plot link qualities between nodes adjacent to each other.

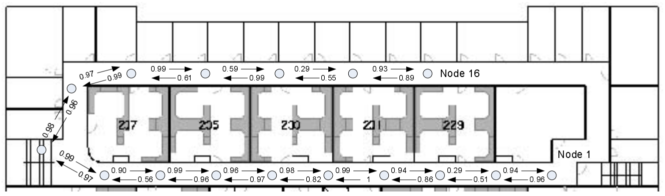


Fig. 4. This figure shows the positioning of 16 nodes and the links that connect them, represented by the packet reception probabilities. Links connecting non-neighboring nodes are not plotted for clarity.

We observe several well-known phenomena in wireless communication, such as the existence of link asymmetry and weak links. These phenomena have been reported in prior literature. Based on link quality, we can calculate the expected packet transmission rounds for the stream communication model. The actual packet numbers fit the predictions very well, as shown in Figure 5.

There are slight differences between the predictions and the actual measurements in Figure 5, for two reasons. First,

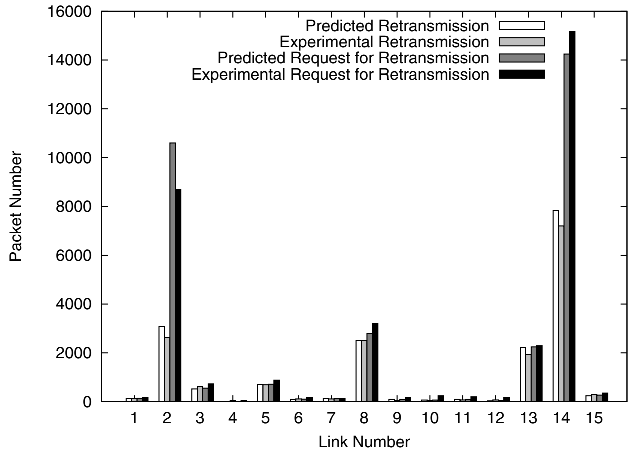


Fig. 5. This figure compares the predicted and experimental data traffic. The former is calculated based on the analysis of the communication model, and the latter is measured in the experiment.

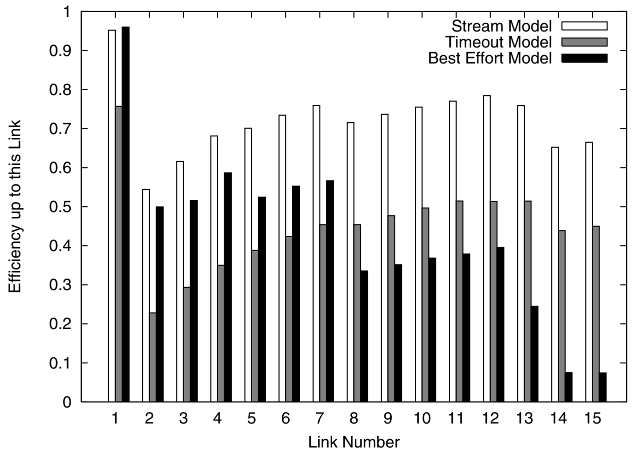


Fig. 6. This figure compares the efficiency of three different transport services: stream model, timeout-resend model and best-effort model.

link quality may change slightly over time. Second, in our implementation, the sender switches to timeout-based packet delivery to ensure reliability at the end of the stream. Both factors lead to slight performance deviations, but Figure 5 demonstrates that these deviations are small and the predictions are still quite accurate.

We also compare the efficiency of the stream communication model to the other two models. We plot the results in Figure 6. The path efficiency is calculated based on experimental data. The efficiency for a particular link refers to the *accumulated* path efficiency from link 1 to this link, as calculated based on the logged data. For example, observe that link 10 corresponds to an efficiency of 0.75 for the stream model, meaning that as data flow from link 1 to 10, the overall efficiency of these ten links is 0.75. Also observe that because of the poor quality of link 2 ($(0.51, 0.29)$ in the experiment), all three communication models experience a considerable decrease in terms of efficiency at link 2. The third observation is that we have similar results as the analysis in Section II-C: the best-effort communication model does not scale with path length and the stream communication model outperforms the other two models considerably.

B. End-to-End Delay Analysis

To derive the end-to-end delay, observe that this delay contains multiple random variables conforming to different distributions. For example, both the time it takes to detect a packet loss and the number of $RRPs$ sent after the detection of a lost packet conform to geometric distributions, whereas the number of hop-wise delivery failures for a packet from the source to the sink conforms to a binomial distribution. Despite the fact that it is quite difficult to obtain the accurate statistical distribution of the end-to-end delay, it is relatively easy to give an estimate of the *expected end-to-end delay*. In practice, this parameter is very useful. We derive an estimate of end-to-end delay for two types of traffic: periodic traffic and Poisson traffic. We also validate our analysis using simulations. We do not use empirical data due to the fact that in order to maintain precise time synchronization, which is necessary for calculating the end-to-end delay, current sensor networks protocols employ periodic data exchanges to cope with clock drifts. Such periodic packet exchanges incur a non-trivial traffic overhead and lead to unexpected interference, both of which we want to avoid. Therefore, it is hard to measure precise packet delays without having side-effects on link quality due to interference. Moreover, our simulations also allow us to use a much larger traffic volume to get more precise distribution curves.

We first analyze the end-to-end delay for periodic traffic. Consider a path with N hops. Let the delivery latency at each hop be T_l. Let the source generate data packets at a fixed interval, T_d. We assume $T_l \ll T_d$. Suppose the timeout for $RRPs$ is T_a. A packet sent over a link with a reception probability (p, q), has a probability of $1-p$ of being lost. One must then wait for a new packet to be successfully transmitted, so that the lost packet can be detected. On average, it takes $1/p$ new packets before one packet can be successfully delivered. After the lost packet is detected, it takes on average $1/pq$ $RRPs$ to recover the packet. Therefore, the total expected delay over one hop can be approximated by $T_l + (1-p)(\frac{T_d}{p} + \frac{T_a}{pq})$. For a path with N links, the end-to-end delay can be estimated as:

$$Delay = \sum_{i=1}^{N} [T_l + (1-p_i)(\frac{T_d}{p_i} + \frac{T_a}{p_i q_i})] \qquad (16)$$

The analysis for Poisson traffic is quite similar. We model the packet flow as a Poisson process with parameter λ. Therefore, the expected time period for $1/p$ new packets to be generated is $1/p\lambda$. The end-to-end delay can then be estimated as:

$$Delay = \sum_{i=1}^{N} [T_l + (1-p_i)(\frac{1}{p_i \lambda} + \frac{T_a}{p_i q_i})] \qquad (17)$$

Both Equations 18 and 17 only model the expected hop delay of the first hop, and then extend this approximation to the remaining hops. Therefore, these results are somewhat sketchy. However, through simulations with different parameters, we find that Equations 18 and 17 indeed provide quite accurate predictions regarding the average end-to-end delay.

In the simulation validation, we use the same link quality data set as in Figure 4. All other parameters are set strictly according to MicaZ's technical specifications. The bandwidth of MicaZ's CC2420 radio is $250kbps$, therefore, the time to transmit one packet is roughly $0.001s$. We set the timeout parameter for generating RRP packets to be $0.01s$, which is sufficient for one round of packet exchanges. The end-to-end delay distribution from the simulation is plotted in Figure 7. We also plot the theoretical predication of the expected delay, $2.877s$, based on Equation 18, along with the average end-to-end delay calculated from simulation results.

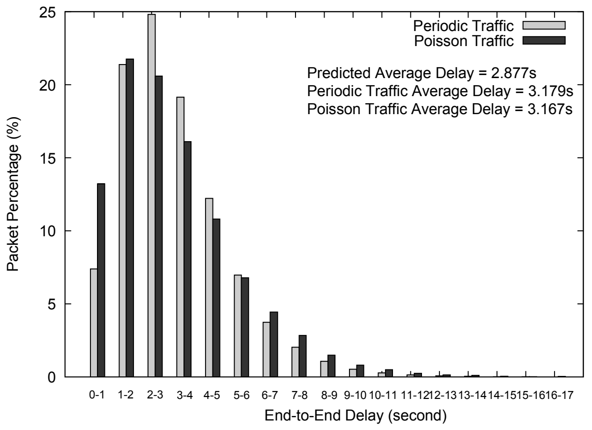


Fig. 7. The Distribution of End-to-End Delay

As observed in Figure 7, our earlier theoretical prediction on the average end-to-end delay fits the simulation results with an error of approximately 10%. This error is attributed to the analysis procedure earlier: we simplified the problem by extending the average delay for the first hop to the whole path, while in practice, a packet loss could cause cascading effects, therefore increases the end-to-end delay.

We also briefly compare the end-to-end delay of the stream communication model with other communication models. For the best-effort model, it is trivial in that each successfully delivered packet has the minimal delay. For the timeout-based model, we assume the most generic case in which the sender uses a timer to decide whether a packet it sent has been successfully received. If the timer expires at T_a and no acknowledgement packet is received, the sender retransmits the packet. Therefore, we can easily get the expected end-to-end delay for the timeout-based model as:

$$Delay = \sum_{i=1}^{N} [T_l + (\frac{1}{p_i} - 1)T_a] \tag{18}$$

Following the settings in Figure 4, the expected end-to-end delay is approximately $70.54ms$, which is much smaller than the stream model. The reason is that in the stream model, the sender relies on subsequent packets to detect potential packet losses. The experiment settings in Figure 4 feature a relatively low data rate, therefore introducing a longer end-to-end delay.

C. Buffer Requirement Analysis

We now analyze buffer requirements. We consider this problem: given a set of links, what is the appropriate buffer size to avoid packet losses? Although it is much more complex to model the exact relationship between the buffer size and the loss rate, it is usually sufficient to provide a lower bound (i.e., how large the buffer should be to avoid packet losses with high probability). We shall answer this question in this section.

There are two requirements that must be satisfied to ensure that packet losses should not occur. First, packet losses should be detected in time before the packet is deleted from the buffer, and second, lost packets should be recovered. Each issue turns out to be a constraint in the model.

First, in order to detect lost packets in time, we consider the instant when one packet gets lost. This packet is buffered by the sender until the buffer is full. Then, the oldest packets in the buffer are dropped since we enforce that the buffer is FIFO. During this period, if at least one packet sent after the lost packet is received, then the receiver can detect the packet loss based on sequence number counting. Therefore, as long as the buffer holds enough packets such that at least one packet following the lost packet can be received, the lost packet should be detected. Put in another way, the probability that a packet loss is left undetected is equivalent to the probability that none of the sequence of packets following the lost packet is successfully received before the lost packet is deleted from the buffer. Consequently, for a buffer size of N and a link with a reception probability of (p, q), the probability of a packet loss detection failure is $(1-p)^N$. In practice, we want this probability to be sufficiently small, for example, less than 10^{-3}. Therefore, we have:

$$(1-p)^N < 10^{-3} \quad \Rightarrow N > -\frac{3}{log_{10}(1-p)} \tag{19}$$

Second, we also need to enforce that lost packets can be recovered. Consider a period of T and a fixed data generation interval of T_d, we know $\frac{T}{T_d}$ packets are to be transmitted. Among them, $\frac{(1-p)T}{T_d}$ packets are expected to be lost. Since recovering one packet requires on average $1/pq$ rounds of $RRPs$, therefore, recovering one packet takes an expected period of $\frac{T_a}{pq}$. Over the period of T, at most $\frac{T}{T_a/pq}$ packets can be recovered. Since the number of recovered packets must not be smaller than the number of lost packets, therefore, we have:

$$\frac{T}{T_a/pq} > \frac{(1-p)T}{T_d} \quad \Rightarrow \frac{T_d}{T_a} > \frac{1-p}{pq} \tag{20}$$

Interestingly, the buffer size does not appear in this constraint, instead, this result implies that both p and q should not be too small. For example, in our earlier experiment, the data rate was 2 packets per second and the timeout was 10 milliseconds. Therefore, $\frac{T_d}{T_a} = 50$. This constraint therefore enforces that $\frac{1-p}{pq} < 50$. All communication links in the earlier experiment satisfied this constraint.

Above all, Equation 19 appears to be the most relevant constraint that the buffer size of each node should satisfy. We use simulations to validate this claim. For simplicity, we enforce that $p = q$ in all cases, and simulate different p values. We plot the relationship between the packet loss ratio and the buffer size from simulations in Figure 8.

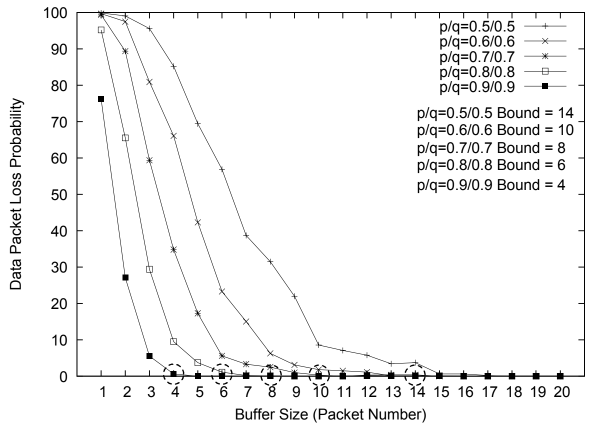


Fig. 8. The Relationship between the Buffer Size and the Data Loss Rate

In this simulation, we still use 15 links. We calculate the theoretical prediction (we call them *bounds* in Figure 8) on buffer size for each link quality setting. These predictions are circled on the X axis at their corresponding positions in this figure. Observe that generally the predictions work quite well. We indeed observe $1 - 5\%$ packet loss even when the predicted buffer size is allocated. Part of the reason is that when multiple lost packets are detected simultaneously, it may take too many rounds of $RRPs$ to recover them, causing the following packets from upstream nodes not to be buffered. To make the problem tractable, our model does not take this effect into account, therefore, slightly larger buffer size should be allocated to reduce packet losses in practice.

An interesting observation concerning Figure 8 is that normally, the required buffer size is indeed low. For example, even with quite unreliable links (0.5/0.5), the buffer size requirement is merely 16 packets. Hence, our design is space-efficient, making it suitable for resource constrained sensor nodes.

We also briefly compare the buffer requirement of the stream communication model to other communication models. For the best-effort model, it trivially requires a buffer space of only one packet. For the timeout model, if the timer has a much higher firing rate than the data packet arriving rate, as is the case in our experiment, one packet is almost guaranteed to be received by the next node before the next data packet arrives. Therefore, the buffer requirement is also negligible.

IV. Routing Layer Optimization

In this section, we present the routing layer optimization technique. Specifically, we integrate the stream communication model presented in Section II-B.2 with the spanning tree routing protocol by taking into account link quality variations. The rest of this section is organized as follows. First, we describe in detail the optimization procedure. Next, as a preparation for the comparison and analysis, we present a brief overview of two related path selection methods that appeared in recent literature. Both of them also have taken into account the effect of path quality, though their path quality metrics are different. At last, we compare these three approaches and analyze our results. We use EPB as the primary comparison metric. The comparison results show that the stream communication model achieves a considerably higher energy efficiency than the other two approaches. We believe the performance improvement is primarily attributed to our joint optimization technique across the link layer and the routing layer.

A. Spanning Tree Routing Structure Optimization

The optimization of spanning tree routing has two phases: a *link estimation phase* where each node independently estimates its link quality to immediate neighbors by exchanging packets, and a *selective flooding phase* where each node obtains its parent node as well as its *path cost* to the base station.

In the first phase, each node broadcasts a fixed number of packets and records the number of successfully received packets from its neighbors. Each node then exchanges this information with neighbors, thereby filling its neighbor table with both outward and inward links with link quality indicators (p, q).

In the second phase, the base station initializes a selective flooding procedure to build a weighted routing tree. In the beginning, the base sets the *path cost* to itself to 0. All other nodes consider the path cost to the base as infinitely large. The base sends out its current path cost to its neighbors. To handle possible packet losses, it rebroadcasts this information multiple rounds. Once one neighbor receives such a packet, if the path cost contained in the packet is smaller than its current path cost, it recalculates its path cost to the base by adding the received path cost, for example, from a node V with a link quality (p, q), with a stream metric cost corresponding to this link, calculated from Equation 6. This node also records V as its *parent* node. It rebroadcasts its updated path cost after a short waiting period, to avoid congestion. Similarly, it rebroadcasts the update multiple times. Meanwhile, this node records all packets sent by its neighbors, thereby maintaining the path cost information of its neighbors.

After the second phase, each node maintains a path cost to the base station through its parent node. By following the parents, one node can reach the base station. We shall next compare the effectiveness of this approach to related approaches in the literature.

B. Related Approaches

In this section, we give a brief overview of two related path selection approaches that appeared in the recent literature. They are the geographic routing based path selection as

TABLE I

SIMULATION SETTINGS

Radio			
Modulation	FSK	Encoding	Manchester
Output Power	-7 dBm	Frame	50 bytes
Transmission Medium			
Path Loss Exponent	3	PL_{D_0}	55 dBm
Noise Floor	-105 dBm	D_0	1m
Deployment Configuration			
Area Height	200 m	Area Width	200 m
Node Number	1000	Range	10-25m

presented in [18] and a high-throughput metric based path selection as presented in [8].

The authors of [18] compared multiple path selection indicators in the context of geographical forwarding and concluded the metric $PRR \times distance$ product achieves the best performance. Here, for a pair of nodes U and V, PRR stands for the packet reception probability of V for packets from U. $Distance$ stands for the distance advanced towards the destination by node V. Node U computes the metric $PRR \times distance$ among all its neighbors and selects the maximal one as the best next neighbor. For complete derivations, please refer to [18].

The authors of [8] proposed another metric ETX (the expected number of data transmissions required to send a packet over a link), defined as $1/pq$ for a link with quality (p, q). [8] did not optimize a spanning tree, instead, it integrated ETX with the DSR routing algorithm. If we consider the base station in the spanning tree as the *source node* in the DSR routing algorithm, we are able to optimize the spanning tree using the ETX metric based on the approach provided in [8].

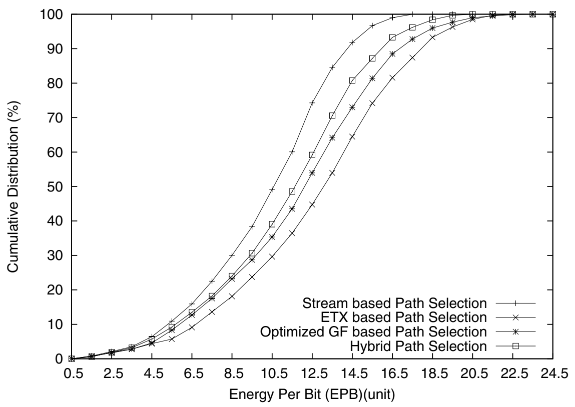


Fig. 9. The CDF of EPBs for different metrics. The CDF is plotted based on the analysis of all 1000 nodes.

C. Performance Comparison

We set up the simulation to reflect realistic communication behavior of sensor networks. We use the radio model in [31], which models many realistic radio features, such as the existence of the transitional region, radio irregularity, and antenna directionality, among others. In our simulation, we

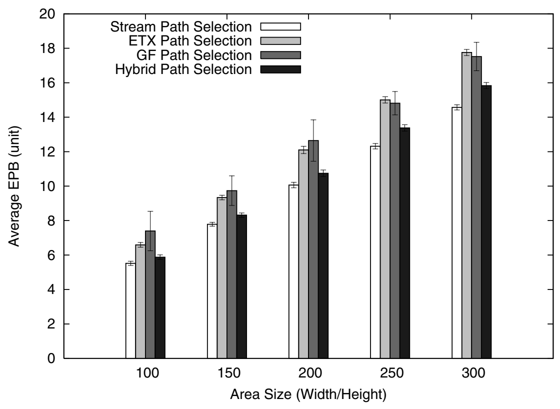


Fig. 10. Impact of Area Size

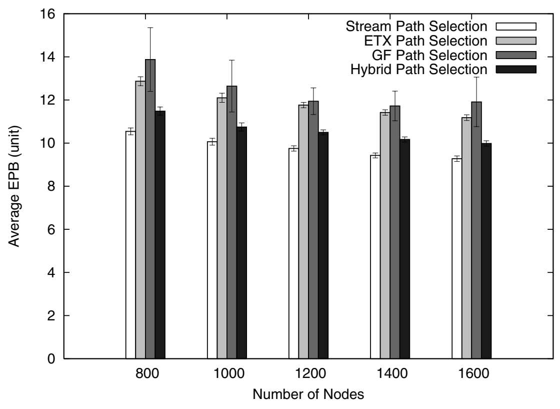


Fig. 11. Impact of Density

use this model as the radio layer. For a detailed description of the radio model, please refer to [31]. We set the parameters of the radio strictly according to the technical specifications of the MicaZ radio module. There are, of course, several adjustable parameters of the $CC2420$ radio on the MicaZ nodes, such as the output power. In this case, we adjust the parameters consistently with our earlier experiments. The complete simulation setup is shown in Table I.

In the simulation, we assume that nodes are randomly deployed in a square area. The default number of nodes is 1000, and the default area size is $200m \times 200m$. One node is positioned at $(0, 0)$ to serve as the base station. After nodes form a spanning tree structure, we apply different optimization procedures, and calculate the EPB value for each node. We assume the energy consumed for transmitting one bit over one hop at the default power level $-7dBm$ is one *unit*. We plot the results for four different mechanisms: ETX based Path Selection in [8]; Optimized GF Path Selection in [18]; Stream based Path Selection as proposed in this paper; Hybrid Path Selection, which uses the ETX metric to select paths and uses the stream communication model to deliver packets. The reasoning behind the last path selection is that we want to isolate the optimization effect of the routing layer.

In each of the following experiments, fifty rounds of sim-

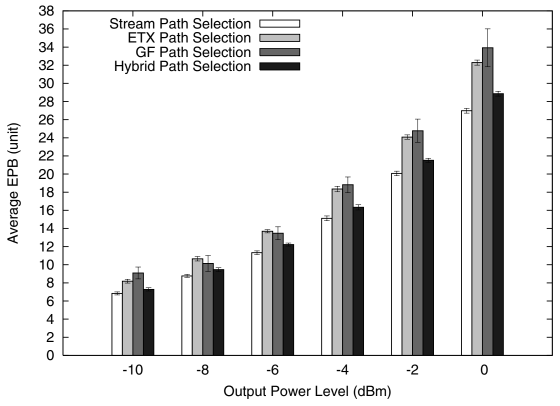


Fig. 12. Impact of Output Power Level

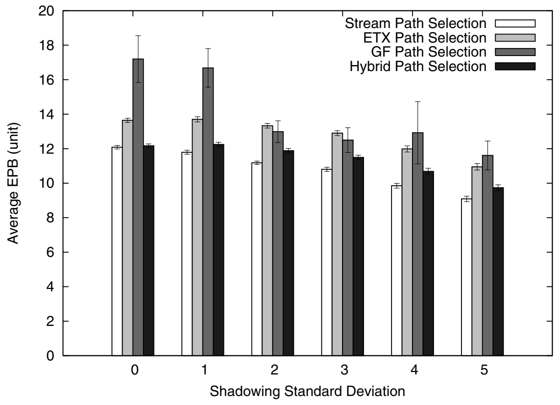


Fig. 13. Impact of the Shadow Standard Deviation

ulations are carried out and the averaged values from these rounds are used. A confidence interval of 95% is used where applicable.

Figure 9 compares the EPB distributions of the four aforementioned path selection mechanisms. A lower EPB means a better energy efficiency. We observe that the stream based path selection performs the best, followed by the hybrid path selection, then the optimized GF path selection, and at last, the ETX based path selection. As described earlier, one EPB unit stands for the energy consumption for a transmission of one bit over one hop at the output power level of $-7dBm$.

We also study the impact of two adjustable parameters: the area size and the node density. Figure 10 plots the impact of the area size, where node density is the same across different settings. The average EPB for all nodes is used for comparison. Observe that the stream based path selection consistently performs the best. This figure and the three following figures also have the confidence level plotted.

In Figure 11, we vary the node density. The number of nodes varies from 800 to 1600, and the area is kept at $200m \times 200m$. An interesting observation is that as the density increases, EPB slightly decreases. This is quite intuitive since increased density implies better paths may exist, which leads to a decreased energy cost. Another consistent observation is that the stream based path selection performs the best.

Figure 12 considers the effects of different output power levels. The area size is $200m \times 200m$. Notice that the $unit$ in this figure still means the energy consumption for transmitting one bit using an output power level of $-7dBm$. The energy consumption values of other output power levels are scaled to this unit. Interestingly, we observe that by increasing the output power level, the energy consumption grows considerably, even though we did observe reduced path length in terms of hops and better connectivity.

Figure 13 considers the effect of a transitional region, in which the link quality changes abruptly. The existence of a transitional region has been repeatedly reported in the recent literature [26], [30], [31]. In the radio model we use, we are able to tune the parameter σ, the shadowing standard deviation, to adjust the width of the transitional region: a smaller σ leads to a narrower transitional region. In this experiment, we change the value of σ from 0, where the transitional region does not exist, to 5, where a very wide transitional region appears. The σ value of MicaZ nodes is approximately 3.8 [31]. We observe from the results that, generally, a larger transitional region leads to better performance in terms of EPB. This indicates that, interestingly, a smart use of the highly varied links in the transitional zone can improve the performance. In fact, this is because a wider transitional region provides more links to choose from. However, we cannot make the assertion that choosing more links in the transitional region *always* leads to better performance. A counterexample is observed in the following experiment.

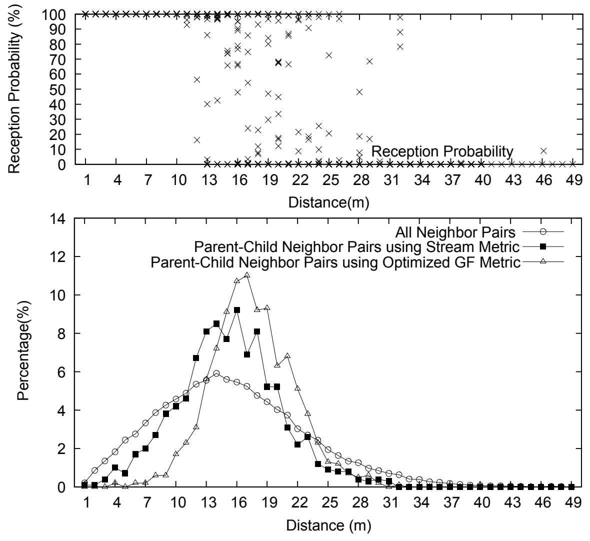


Fig. 14. Neighbor Selection Comparison

In the next experiment, we consider this interesting problem: exactly what is the relationship between the link choice and the energy efficiency? For example, does choosing more links in the transitional region generally lead to better performance? To answer this question, we study three distributions: the distribution of distances of all neighbor pairs, the distribution of distances of neighbor pairs that have parent-child relationship using the stream based path selection, and the distribution of distances between neighbors that have parent-child relationship using the optimized GF path selection. We know that the stream based path selection performs better than the optimized GF path selection, so is this a result of choosing more links in the transitional region?

As a comparison, we also plot the transitional region based on the radio model. Observe that the transitional zone spans roughly from $10m$ to $30m$. Comparing the two sub-graphs in Figure 14, the first important observation is that links inside the transitional zone are obviously preferred compared to other links in both path selection policies. This is consistent with the observations made in [18]. The second observation is even more interesting: the optimized GF path selection uses considerably more links in the transitional zone than the stream based path selection. We believe the reason is that the optimized GF path selection is too aggressive in terms of selecting far away links with poor connectivity. Combine this fact with our earlier results, we know that choosing links in the transitional region too aggressively actually degrades the overall energy efficiency.

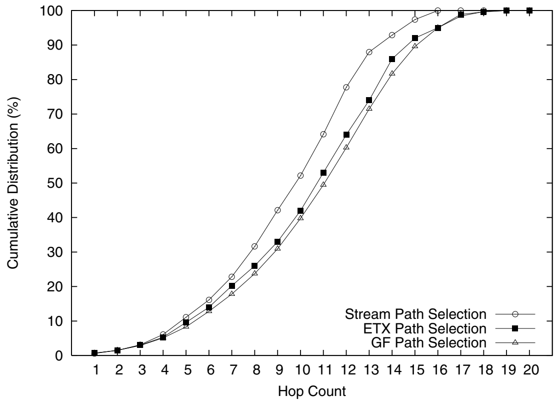


Fig. 15. Distribution of Hop Counts for Different Path Selection Mechanisms

The last experiment studies the distribution of hop count numbers using different path selection metrics. The hop count metric is the conventional *path length* parameter. Since the hybrid path selection leads to the same path length as the ETX based path selection, we only need to compare three metrics. The results are shown in Figure 15. Observe that the stream based path selection generally leads to the shortest paths, followed by the ETX based path selection and the optimized GF path selection.

One very interesting, yet somewhat counter intuitive observation about Figure 15 and Figure 14 is that the optimized GF path selection selects more links in the transitional region (which are expected to be more distant) than the stream based path selection, yet it still has longer paths. The reason is that the stream based path selection incorporates a flooding process to build the routing structure, yet GF only uses *local* information to make path selection choices. Therefore, although the optimized GF approach does choose distant links more aggressively, such links may not lead the packets in the globally right direction. Therefore, the paths in the optimized GF path selection are relatively longer.

D. Conclusion

Based on the comparison results, we can decrease EPB by using the stream metric in the routing layer. Of course, one serious limitation of the spanning tree routing structure is that it can only support a limited number of base stations, while the optimized GF mechanism presented in [18] can handle more generalized peer-to-peer communication patterns. Nevertheless, since many applications indeed use only a few base stations, we believe our study is quite applicable to these real systems. Furthermore, we plan to extend the stream based communication model to more generalized, peer-to-peer communication patterns in our future work.

V. Related Work

Research on radio properties [6], [30] indicates that the wireless links between low power sensor devices are extremely unreliable. Specifically, Woo [26] points out the existence of three distinct data reception regions within a radio range: full connected, transitional and disconnected regions. In the transitional region, the reception of data becomes highly varied. Meanwhile, it is observed that, in realistic systems, the radio qualities are severely affected by the multi-path effect, reflection, diffusion, scattering and ground attenuation [10], [22], [12].

To achieve reliability over unreliable links, many protocols have been designed, evaluated and implemented [8], [18], [17], [21], [27]. Some protocols regard the reliability in data delivery as a main design goal. For example, RMST [21] (Reliable Multi-Segment Transport) tracks packet fragments so that receiver-initiated requests, using NACK control packets, can be satisfied when individual pieces of an application payload get lost. Another work is the transport layer design of Wisden [27]. The transport layer of Wisden shares some features of RMST and uses overhearing in the same way as our stream communication model. In addition, robust data delivery [9] simultaneously sends packets along multiple paths at the expense of increased communication overhead. Our work is different from these approaches in the sense that we achieve communication efficiency and reliability, simultaneously.

Recently there also have been some protocols designed to address both reliability and congestion, simultaneously. Among them are CODA [24] and ESRT [17]. Both protocols are more focused on how to reduce congestion through various techniques. Specifically, CODA uses a sampling of the channel to determine whether the channel is currently congested, and if it is, nodes decrease the traffic allowed. Alteratively, ESRT monitors the current network state based on the congestion conditions in the network. Such conditions guide ESRT to adjust the reporting frequency of the source node to maintain event-to-sink reliability dynamically.

The topic of efficiency has been considered less frequently in the recent literature. Representative protocols include Fusion [11] and a revised geographical forwarding proposed in [18]. Both protocols discussed possible enhancements on efficiency. In Fusion, the main topic is how to leverage various congestion-control mechanisms to increase efficiency. In [18], the authors studied how to minimize the energy spent in geographical routing, therefore increasing efficiency as well. Neither of them, however, presented a systematic study of the relationship between efficiency and link quality, nor did they propose any new communication models. Therefore, our work is quite novel in this aspect and provides significant insight on communication efficiency.

The study of communication efficiency falls into a larger research topic: energy efficiency, where solutions are even more diversified. Lower power microcontrollers [2], radio circuits and antenna designs [7], [16] significantly reduce energy consumption in data delivery at the hardware level. Media Access Control [15], [23], [29] reduces the network contention, hence reduces the number of retransmissions. Sensing coverage protocols [25], [28], [20], [5] provide surveillance to a certain geographic area with a minimal energy budget. Data aggregation techniques [14], [19] reduce the number of messages sent while still distributing information an application requires. Data cache techniques [4] are also designed for applications where multiple sinks coexist and use caching to update and distribute data to leaf nodes at the minimally requested rate. Cross-layer designs [13], [20] coordinate the operations of the physical, medium access control (MAC), and routing layers together to achieve a better efficiency than what can be achieved by individual protocols. Since these protocols achieve energy efficiency from different perspectives, we consider them orthogonal and complimentary to our work.

VI. Conclusions

In this paper, we have studied the problem of communication efficiency based on a quantitative modeling of wireless link unreliability. We followed a joint-optimization process that considers both the link layer lost packet recovery and the routing layer path selection. Our contributions are two-fold. First, in the link layer, we present a lazy lost packet detection and recovery technique that proves to be especially useful in terms of improving efficiency. Second, we develop and use a stream metric for path selection in the routing layer. Compared with other state of the art path selection approaches, our joint optimization considerably decreases Energy Per Bit. Our simulation and implementation results demonstrate the correctness of our approach, and give us fundamental insight into the efficiency aspect of sensor network communication over unreliable links.

Acknowledgments

We would like to thank Marco Zuniga at the ANRG group of USC for sending us the source code of their radio model. We also would like to thank the anonymous reviewers for their insightful comments. The work reported in this paper was supported in part by the National Science Foundation grants EHS-0208769, CSR-0509233, and NETS-0435060, and by MURI grant N00014-01-1-0576.

References

[1] IEEE standard 802.15.4 2003.
[2] *ATmega128 Specification*, Atmel Corporation, available at www.atmel.com/acrobat/doc2467.pdf.
[3] M. Batalin *et al.*, “Call and response: Experiments in sampling the environment,” in *ACM Sensys*, 2004.
[4] S. Bhattacharya *et al.*, “Energy-Conserving Data Placement and Asynchronous Multicast in Wireless Sensor Networks,” in *MobiSys*, 2003.
[5] M. Cardei *et al.*, “Energy-efficient target coverage in wireless sensor networks,” in *IEEE Infocom*, 2005.
[6] A. Cerpa *et al.*, “Statistical model of lossy links in wireless sensor networks,” in *IPSN*, 2005.
[7] *CC2420 Product Information and Data Sheet*, chipcon, avaiable at http://www.chipcon.com/.
[8] D. D. Couto *et al.*, “A high-throughput path metric for multi-hop wireless routing,” in *ACM Mobicom*, 2003.
[9] D. Ganesan *et al.*, “Highly resilient, energy efficient multipath routing in wireless sensor networks,” in *Mobile Computing and Communications Review*, 2002.
[10] T. He *et al.*, “An Energy-Efficient Surveillance System Using Wireless Sensor Networks,” in *MobiSys*, 2004.
[11] B. Hull *et al.*, “Techniques for mitigating congestion in sensor networks,” in *ACM Sensys*, 2004.
[12] V. Kottapalli *et al.*, “Two-tiered Wireless Sensor Network Architecture for Structural Health Monitoring,” in *Proc. of the Intl. Symp. on Smart Structures and Materials*, 2003.
[13] R. Madan *et al.*, “Cross-layer design for lifetime maximization in interference-limited wireless sensor networks,” in *IEEE Infocom*, 2005.
[14] S. Madden *et al.*, “TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks,” in *Operating Systems Design and Implementation*, 2002.
[15] J. Polastre and D. Culler, “Versatile Low Power Media Access for Wireless Sensor Networks,” in *SenSys*, 2004.
[16] J. M. Rabaey *et al.*, “PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking,” *Computer Magazine*, vol. 33, no. 7, 2000.
[17] Y. Sankarasubramaniam *et al.*, “Esrt: Event-to-sink reliable transport in wireless sensor networks,” in *ACM Mobihoc*, 2003.
[18] K. Seada *et al.*, “Energy efficient forwarding strategies for geographic routing in lossy wireless sensor networks,” in *ACM Sensys*, 2004.
[19] N. Shrivastava *et al.*, “Medians and Beyond: New Aggregation Techniques for Sensor Networks,” in *SenSys*, 2004.
[20] M. L. Sichitiu, “Cross-layer scheduling for power efficiency in wireless sensor networks,” in *IEEE Infocom*, 2004.
[21] F. Stann and J. Heidemann, “Rmst: Reliable data transport in sensor networks,” in *IEEE SNPA*, 2003.
[22] R. Szewczyk *et al.*, “An Analysis of a Large Scale Habit Monitoring Application,” in *SenSys*, 2004.
[23] T. van Dam and K. Langendoen, “An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks,” in *SenSys*, 2003.
[24] C.-Y. Wan *et al.*, “Coda: Congestion detection and avoidance in sensor networks,” in *ACM Sensys*, 2003.
[25] X. Wang *et al.*, “Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks,” in *SenSys*, 2003.
[26] A. Woo *et al.*, “Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks,” in *SenSys*, 2003.
[27] N. Xu *et al.*, “A wireless sensor network for strucutral monitoring,” in *ACM Sensys*, 2004.
[28] T. Yan *et al.*, “Differentiated Surveillance Service for Sensor Networks,” in *SenSys*, 2003.
[29] W. Ye *et al.*, “An Energy-Efficient MAC Protocol for Wireless Sensor Networks,” in *IEEE INFOCOM*, 2002.
[30] J. Zhao and R. Govindan, “Understanding packet delivery performance in dense wireless sensor networks,” in *ACM Sensys*, 2003.
[31] M. Zuniga and B. Krishnamachari, “Analyzing the transitional region in low power wireless links,” in *IEEE SECON*, 2004.

Achieving Long-Term Surveillance in VigilNet

Tian He†, Pascal Vicaire, Ting Yan, Qing Cao‡, Gang Zhou, Lin Gu,
Liqian Luo‡, Radu Stoleru, John A. Stankovic, Tarek F. Abdelzaher‡
Department of Computer Science
University of Virginia, Charlottesville 22903

***Abstract*— Energy efficiency is a fundamental issue for outdoor sensor network systems. This paper presents the design and implementation of multi-dimensional power management strategies in VigilNet, a major recent effort to support long-term surveillance using power-constrained sensor devices. We integrate a novel tripwire service with an effective sentry and duty cycle scheduling in order to increase the system lifetime, collaboratively. Through extensive system implementation, we demonstrate the feasibility to achieve high surveillance performance and energy efficiency, simultaneously. We invest a fair amount of effort to evaluate our architecture with a network of 200 XSM motes in an outdoor environment, an extensive simulation with 10,000 nodes, as well as an analytical probabilistic model. These evaluations demonstrate the effectiveness of our integrated approach and identify many interesting lessons and guidelines, useful for the future development of energy-efficient sensor systems.**

I. Introduction

VigilNet is a recent major effort to support long-term military surveillance, using large-scale micro-sensor networks. Besides requirements of accurate target tracking and classification [1], one of the key design goals of VigilNet is to achieve long-term surveillance in a realistic mission deployment. Due to the small form factor and low-cost requirements, sensor devices such as the XSM motes [2] are normally equipped with limited power sources (e.g., two AA batteries). Moreover, because of the hostile environment and a large number of nodes deployed, currently it is not operationally and economically feasible to replace the power source without introducing enormous effort and elements of risk to the military personnel. In addition, the static nature of the nodes in the field prevents the scavenging of the power from ambient motion or vibration [3][4]. The small form factor and possible lack of the line of sight (e.g., deployment in the forest) make it difficult to harvest solar power. On the other hand, a 3$\sim$6 month system life span is essential to guarantee the effectiveness of normal military operations, which necessitates a 12$\sim$24 fold extension of the normal lifetime of active sensor nodes. Consequently, it is critical to investigate practical approaches of spending the power budget effectively.

Many solutions have been proposed for energy efficiency at various levels of the system architecture, ranging from the hardware design [5][2], coverage [6][7][8][9] MAC [10][11][12], routing [13][14][15], data dissemination [16], data gathering [17][18], data aggregation [19][20], data caching [21], topology management [22], clustering [23], placement [24] [25] to energy-aware applications [26][27]. Instead of focusing on a single protocol, our answer to energy efficiency is an integrated multi-dimensional power management system. Our contributions, presented in this paper, are identified in the following aspects: 1) Our design is validated through an extensive system implementation: VigilNet – a large-scale sensor network system delivered to military agencies. 2) VigilNet takes a systematic approach, and the energy efficiency is not narrowly accounted for within a single protocol. We propose a novel tripwire service, integrated with an effective sentry and duty cycle scheduling to increase the system lifetime, collaboratively. 3) Tradeoffs are investigated to meet requirements of both surveillance performance and the network lifetime. We present a complete system with 40,000 lines of code, running on motes, that achieves performance and energy efficiency simultaneously. 4) We devote considerable effort to evaluate the system with 200 XSM motes in an outdoor environment and an extensive simulation of 10,000 nodes, in order to identify a set of useful lessons and guidelines for future research.

The remainder of the paper is organized as follows: Section II categorizes power management features for different application scenarios. Section III describes the power management requirements in VigilNet. Section IV introduces three power management strategies utilized in VigilNet namely, sentry service, tripwire service and duty cycle scheduling. Section V describes the integrated power management architecture in VigilNet. Section VI briefly discusses some additional energy efficient techniques applied in VigilNet. Section VII addresses the tradeoff between energy efficiency and network performance. Section VIII details the system implementation. Section IX provides the evaluation of a network of 200 XSM motes as well as an extensive hybrid simulation with 10,000 nodes. Finally, Section X concludes the paper.

II. Background

Power management is by no means a stand-alone research issue. It can be dramatically affected by the underlying system configuration and by the application requirements. These include the form factor [28], hardware capability [5], possibility of energy scavenging [4][29], network/sensing topology and density [6], link quality [30], event patterns, node mobility, availability and accuracy of time synchronization [31], real-time requirements and the nature of the applications [26]. At the hardware level, multi-level sleep modes in the low power

†Tian He is now with University of Minnesota tianhe@cs.umn.edu

‡Qing Cao, Liqian Luo and Tarek Abdelzaher are now with University of Illinois, Urbana-Champaign

microcontroller [5] enable software to control the rate of power dissipation. Fine-grained power control [2] allows applications to activate hardware modules incrementally. Radio wakeup circuits [32] achieve passive vigilance with a minimal power draw. Energy scavenging [3] is also possible for some application scenarios, where ambient energy can be harvested. Sensing coverage schemes [6][7] exploit redundancy in the node deployment to activate only a subset of nodes. The coordinated scheduling of the sensor duty cycle [33] increases the probability of detection and reduces the detection delay with a minimal power consumption. Communication protocols turn off the radio when a node is not the intended receiver [12]. Though many individual solutions are proposed, few real systems actually achieve power efficiency comprehensively, which makes the integrated approach in VigilNet novel and practically useful. Considering the diversity of the different approaches, we categorize power management strategies in the context of two types of systems: sampling systems and surveillance systems.

A. Power Management in Sampling Systems

Great Duck Island [26] and Structural Monitoring [27] are typical sampling systems, which are deployed as distributed large-scale data acquisition instruments. Power management strategies in these systems normally make use of the following techniques:

- **Predefined sampling schedules:** Most environmental phenomena, such as temperature, exist ubiquitously over space and continuously over time. The static nature of these phenomena makes it sufficient to construct the data profile by sampling the environment within discrete time and space. Nodes can conserve energy by turning themselves off, according to a predefined schedule.
- **Synchronized and coordinated operations:** Once the sampling interval is defined *a priori*, nodes can communicate in a synchronized fashion. With a precise time synchronization [31], a receiver can turn on the radio module right before the message payload arrives. Consequently, we can avoid low-power listening over radio [10] during a non-active period. In addition, with the knowledge about the sending rate of individual nodes, we are able to estimate the link quality without control messages [34].
- **Data aggregation and compression:** Since channel media access is costly, especially when the receiver is in a deep-sleep state [10], it is beneficial to send out one aggregate containing multiple sensing readings [19][20]. In addition, due to the value locality of the sensed data, we can compress the total number of bits to be sent over the air. Since both aggregation and compression need to buffer a relatively large number of readings, which introduces a certain delay, they are not quite suitable for time-critical surveillance systems. However, they match most sampling systems very well.

B. Power Management in Surveillance System

On the other hand, operations in surveillance systems [35][36][37] [38], such as VigilNet, are event-driven in nature. In surveillance systems, we are more interested in the data profile between inception and conclusion of the transient events. These systems should remain dormant in the absence of the events of interests, and switch to an active state to obtain high fidelity in detection. Normally, the surveillance systems improve the system lifetime through the following approaches:

- **Coverage control:** Surveillance systems are normally deployed with a high density (For instance, the default configuration of VigilNet [38] has 28 nodes per nominal radio range (30 m)) for the sake of robustness in detection and fine-grained sensing during tracking. We can increase the system lifetime by activating only a subset of nodes at a given point of time, waiting for potential targets.
- **Duty cycle scheduling:** The duration of transient events within the area of surveillance is normally non-negligible. By coordinating nodes' sleep schedules, we can conserve energy without noticeably reducing the chance of detection. Duty cycle scheduling is different from sample scheduling in the sense that duty cycle scheduling is at the micro-scale (milliseconds vs. minutes) and it is strongly affected by the dynamics of the events (e.g., target velocity).
- **Incremental activation:** The sampling systems are normally designed for data logging. At each sample instance, all sensors should be activated to obtain a complete data profile. In contrast, surveillance systems are designed to detect transient events of interest. It is sufficient to activate only a subset of sensors for the initial detection. After the initial detection, we can activate other sensors to achieve a higher sensing fidelity and to perform classification.

III. Power Management Requirements in VigilNet

Our power management strategies are motivated by a typical military surveillance application. The mission objective of such a system is to conduct remote, persistent, clandestine surveillance to a certain geographic region to acquire and verify enemy capabilities and transmit summarized intelligence worldwide in a near-real time manner. Several system requirements affect our power management design within VigilNet:

- **Continuous surveillance:** Due to the dynamic/transient nature of the event, VigilNet is required to provide continuous surveillance. This requirement significantly affects the overall architecture of power management strategies and the degree of energy conservation VigilNet can achieve.
- **Real-time:** As a real-time online system for target tracking, VigilNet is required to cope with fast changing events in a responsive manner. The delays introduced by the power management directly affect the maximum target speed our VigilNet can track. It is an essential

design tradeoff to balance between network longevity and responsiveness.

- **Rare and critical event detection:** Due to the nature of military surveillance, VigilNet deals with the rare event model. In this model, the total duration of events is small, compared to the overall system lifetime. On the other hand, events are so critical that the power management becomes a secondary consideration in the presence of events.
- **Stealthiness:** Deployed in hostile environments, it is vital for VigilNet to have a very low profile. Miniaturization makes nodes hard to detect, physically; however, radio messages can be easily intercepted if nodes frequently communicate. Power management protocols designed for VigilNet should maintain silence during surveillance in the absence of significant events.
- **Flexibility:** We envision the deployment of VigilNet under different densities, topologies, sensing and communication capabilities. Therefore, it is essential to design a power management architecture that is flexible enough to accommodate various system scenarios.

IV. Key Power Management Strategies in VigilNet

In order to achieve long-term surveillance that meets the military requirement (e.g., 3∼6 months), an aggressive 12∼24 fold life-time extension is essential. Our initial investigation [38] indicates that a single power management strategy is neither sufficient nor flexible. Therefore we restructure our prototype system described in [38] by adding a new combination of tripwire service and duty cycle scheduling. We believe this is the right direction to pursue. In this section, we detail three main strategies, namely the tripwire service, sentry service and duty cycle scheduling, before presenting an overarching architecture in the next section. In order to support these strategies, all nodes within VigilNet find their positions with an accuracy of 1∼2 meters and they synchronize with each other within 1∼10 milliseconds using the techniques described in [39] and [31], respectively. Long-range communication devices are deployed as bases to relay sensor reports outside of the sensor field.

A. Tripwire Services

This section proposes a novel network-wide power management strategy called *Tripwire Service*. This service divides the sensor field into multiple sections, called *tripwire sections*, and applies different working schedules to each tripwire section. A tripwire section can be either in an active or a dormant state, at a given point of time. When a tripwire section is dormant, all nodes within this section are put into a deep-sleep state to save energy. Surveillance in active tripwire sections can be done by either turning all nodes on or applying coverage algorithms such as the sentry service discussed later in Section IV-B. The rationale behind the tripwire service is the existence of roads in the area of interest. By deploying the tripwire along the road, we can guarantee the detection without activating all sensors in the area.

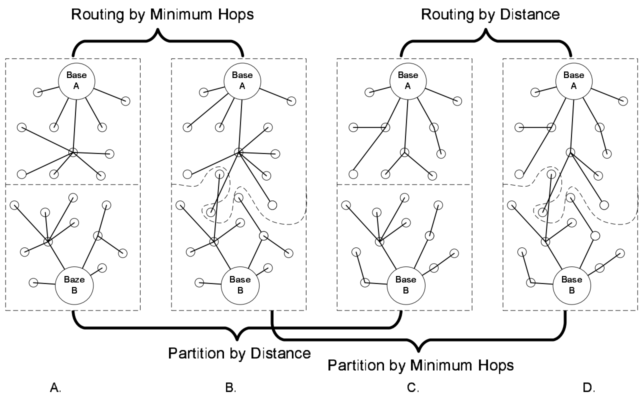


Fig. 1. Four Different Partition Methods

1) Tripwire partition: VigilNet implements its tripwire partition policy based on the Voronoi diagram. A network with n bases is partitioned into n tripwire sections such that each tripwire section contains exactly one base i and every node in that tripwire section is closer to its base i than to any other base inside the sensor field. Every node in the network uniquely belongs to one and only one tripwire section. The rational behind Voronoi partition is to reduce the energy consumption and the end-to-end delay in data delivery.

The positions of bases directly determine the layout of tripwire sections and affect the routing path length for individual nodes. The optimal base placement method to minimize the average path length to the nearest base can be found at [40]. In practice, the base placement strategy is normally determined by the mission plan and topology.

2) Tripwire partition mechanism: This section describes the mechanism to enforce the tripwire partition policy. At the beginning of the tripwire partition operation, each base broadcasts one initialization beacon, to its neighbors with a hop-count parameter initialized to one. Link Symmetry Detection [41] is used to ensure beacons can only be received through high quality symmetric links. Each receiving node maintains the minimum hop-count value of all beacons it received from the nearest base, in terms of the physical distance, and ignores beacons with higher hop-count values and those beacons from other bases. Beacons are flooded outward with hop-count values incremented at every intermediate hop. Through this mechanism, all nodes in the network get the shortest high quality path, in *hops*, to the nearest base, in *physical distance*. While the above mechanism is intuitive, the design deserves some further clarification. First, the boundaries between partitions are well delimited if we partition the network according to the physical distance between sensor nodes and bases (Figure 1A and 1C). If the communication hop is used instead, the radio irregularity and the interference cause partitions to interleave with each other (Figure 1B and 1D). This brings complexity and uncertainty to the design of optimal tripwire placement strategies. Second, it is beneficial to use hop counts to build diffusion trees within each partition, because 1) the normal geographic-based routing does not guarantee high-quality shortest path to the root. 2) Due to the existence of high-quality long links, a smaller number of nodes become active backbone nodes in the hop-based routing than in the geographic-based routing. Finally, this design provides certain robustness to the base failure. If a base fails, the sensor field can be easily repartitioned without this base.

3) Tripwire scheduling: A tripwire section can be either in an active or a dormant state. We configure the state of each tripwire section by setting a 16 bits schedule at the corresponding base. Each bit in the schedule denotes the state of this tripwire section in each round (rotation) up to 16 rounds. After 16 rounds, the pattern is repeated. With this design, we can assign 65536 different schedules to each tripwire and assign 65536^N (N is the number of tripwires.) different schedules to the network. The schedule can be predetermined or randomly generated. Random scheduling is done by setting the Tripwire Duty Cycle (TDC), which is the percentage of active rounds in the schedule.

B. Sentry Services

In order to exploit the high node density within the sections, we design and implement a section-wide power management strategy, called *sentry service*. The main purpose of the sentry service is to select a subset of nodes, which we define as *sentries*, in charge of surveillance. Sentry selection contains two phases. Nodes first exchange neighboring information through hello messages. In each hello message, a sender attaches its node ID, position, number of neighbors and its own energy readings. After the first phase, each node builds up a one-hop neighbor table. In the second phase, each node sets a delay timer. The duration of the timer is calculated based on the weighted Energy rank R_{energy} and weighted Cover rank R_{cover} as shown in Equation 1. The energy rank R_{energy} is assigned according to energy readings among neighboring nodes (e.g., the node with the highest energy reading within a neighborhood has a rank of 1.) Similarly, the cover rank R_{cover} is assigned according to the number of neighbors within a node's sensing range. As for current implementation, we assign equal weights to both ranks.

$$T_{timer} = \frac{W_e \times R_{energy} + W_c \times R_{cover}}{(W_e + W_c) \times \#Neighbors} MaxDelay + Jitter \tag{1}$$

After the delay timer fires in one node, this node announces itself as *sentry* by sending out a declaration message. While other nodes, in the vicinity of the declaring node, cancel their timers and become dormant *non-sentry* nodes. The effective range, in physical distance, of a sentry's declaration message is defined as the Range of Vicinity (ROV). While the sentry selection can be straightforwardly implemented, the challenging part is to *choose* and to *enforce* the appropriate range of vicinity (ROV). This parameter directly affects the sentry density, hence affects the lifetime of the network.

1) How to choose ROV: The appropriate ROV value can be chosen by the analytical intrusion detection model detailed in Appendix. This model describes the relationship between the detection probability, the sensing range and the sentry density.

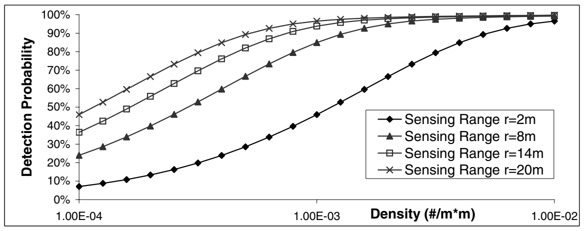


Fig. 2. Detection Prob. Vs. Sentry Density

Since, theoretically, there is at most one sentry within each ROV range, according to the circle covering theorem [42], the sentry density is upper bounded by $\frac{2\pi}{\sqrt{27}ROV^2}$. Given the area size, sensing range and sentry density, we get the detection probability (Figures 2) according to the derived model. For a typical deployment with 1000 nodes in 100×1000 m^2 area, Figure 2 indicates how to choose the right combination of system configurations. For example, in order to achieve a 99% detection probability, we can choose either a sentry density of 0.008 nodes/m^2 (ROV= 6 meters) with 8 meter sensing range or a lower density of 0.004 nodes/m^2 (ROV=8.5 meters) with 14 meters sensing range. We note that when ROV is set to the sensing range, we can guarantee 100% detection, assuming no voids.

2) How to enforce ROV: After we choose a ROV value, we need to enforce it during the sentry selection phase. Since the sensing range is normally smaller than the radio range, directly using the radio range as the ROV cannot guarantee an effective coverage of the area. For example, the HMC1002 dual-axis magnetometer used by MICA2 has only 30-feet effective range for a moving car. If we use the Chipcon radio (>100 feet) to define the range of vicinity, less than 10% of area is sensing covered. There are two approaches to address these issues. The first approach is to reduce the radio sending power to emulate the ROV range. The power setting can be chosen in such a way that there is about one sentry within each sensing range. The second approach is to discard declaration messages from any sentry beyond the distance of ROV. The first approach achieves sensing coverage, without the location information of the nodes [43], while the second approach provides a more predictable sentry distribution, because the emulated ROV would be affected by the radio irregularity in the environment. Consequently, we adopt the second solution in our system, given the fact that localization [39] is supported in VigilNet.

C. Sentry Duty Cycle Scheduling

The requirement for continuous sensing coverage in the sentry service imposes a theoretical upper bound on the system lifetime. This upper bound is decided by the total number of nodes deployed. Since a target normally stays in the sensing area of a sentry node for a non-negligible period of time, it is not necessary to turn sentry nodes on all the time. By using duty cycle scheduling, we are able to break the theoretical upperbound imposed by the full coverage algorithms [7]. Let T$_{\text{on}}$ be the active duration and T$_{\text{off}}$ be the inactive duration, then

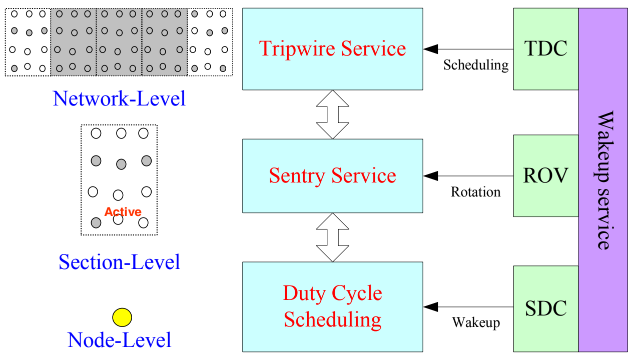


Fig. 3. Integrated PM Architecture

the Sentry Toggle Period (STP) is defined as $(T_{on} + T_{off})$, and the Sentry Duty Cycle (SDC) is defined as $\frac{T_{on}}{STP}$. Theoretically, the duty cycle scheduling can achieve unbounded energy conservation by lowering the SDC value. The paramount concern of this technique is that lowering the SDC value increases the detection delay and reduces the detection probability. We can either effectively implement random duty cycle scheduling or more sophisticated scheduling algorithms to coordinate node activities to maximize performance. In [33], we demonstrate a local optimal scheduling coordination algorithm to reduce the detection delay and increase the detection probability. We prove that, at relatively large SDC (e.g 5% <SDC), the difference between random scheduling and optimal scheduling can be practically ignored. Since the random scheduling does not need control messages for coordination (more stealthy) and it is not affected by time drift, we choose random scheduling over the coordinated one in the system implementation.

V. Integrated Solution: Tripwire-based Power Management with Sentry Scheduling

To achieve an aggressive network lifetime extension, the VigilNet power management subsystem integrates the three strategies mentioned in previous sections into a multi-level architecture, as shown in Figure 3. At the top level, the tripwire service controls the network-wide distribution of power consumption among sections; the uniform discharge of energy across sections is achieved through the scheduling mechanism we discussed in section IV-A.3. We use a Tripwire Duty Cycle (TDC), which is the percentage of active time for each tripwire section, to control the network-wide energy-burning rate. There are two special cases: when TDC equals 100%, the whole network becomes active and the tripwire service is merely a network partition service. When TDC equals 0%, the whole network is in dormant status and it can only be awaken by external sources. At the second level, the sentry service controls the power distribution within each section. The uniform discharge of energy in a section is achieved through automatic rotation strategies according to the remaining power within individual nodes. We use the Range of Vicinity (ROV) parameter to control the energy-burning rate of active sections. When ROV equals the sensing range of nodes, the section is fully covered. A higher ROV value than the sensing range leads to a partial coverage and a lower ROV value than the sensing range leads to redundancy in the coverage. When ROV equals 0 meter, the sentry service is actually disabled and all nodes with the section are awake, providing the highest degree of coverage. At the third-level, duty cycle scheduling controls the energy-burning rate of individual sentry nodes by manipulating their wakeup/sleep schedule. The Sentry Duty Cycle (SDC) parameter is used to control the awareness of sentry nodes, which is the percentage of active time. The duty cycle scheduling can be disabled by setting SDC to 100%. By adopting different values for TDC, ROV and SDC, we can flexibly adjust our power management to accommodate different system scenarios.

VI. Other Energy Conservation Techniques

Besides the three main power management strategies, several other techniques have been integrated into various aspects of the VigilNet system. Similar techniques [20][44][27][10] have been proposed in the literature and we provide this section for the completeness of the VigilNet power management design and implementation.

- **Minimum connected dominating tree:** To ensure a swift delivery of messages, VigilNet requires an active diffusion tree over active tripwire section. Since the communication range is normally much larger than the sensing range [5][2], it is possible to build a diffusion tree on top of sentry nodes. To reduce the energy spent during idle listening, VigilNet desires a tree with the minimum connected dominating set (a tree with minimum non-leaf nodes). Since it is a NP-Complete problem to find the minimum connected dominating set of a graph, we adopt a localized approximation as follows: during the building process, each node rebroadcasts the hop-count beacon after a certain time delay. The delay in one node is inversely proportional to the number of neighbors and the energy remaining. By doing so, the node with more neighbors and more energy left has a higher chance to become the parent node within the diffusion tree.
- **Data aggregation:** The channel media access in wireless sensor network is relatively expensive. For example, in the Chipcon radio implementation for MICA2, to deliver a default payload size of 29 bytes, the total overhead is 17 bytes (37%!), including 8 bytes preamble, 2 bytes synchronization, 5 bytes header and 2 bytes CRC. This motivates us to utilize various kinds of aggregation techniques. The first technique we use is called Application-Independent Aggregation, which concatenate data from different modules into one aggregate, regardless of their semantics. For example, system-wide parameters can be sent with time synchronization messages. The second technique we use is called Application-Dependent Aggregation. The tracking subsystem in VigilNet performs the in-network aggregation by organizing the nodes into groups. Instead of each node reporting its position separately, a leader node calculates the weighted center

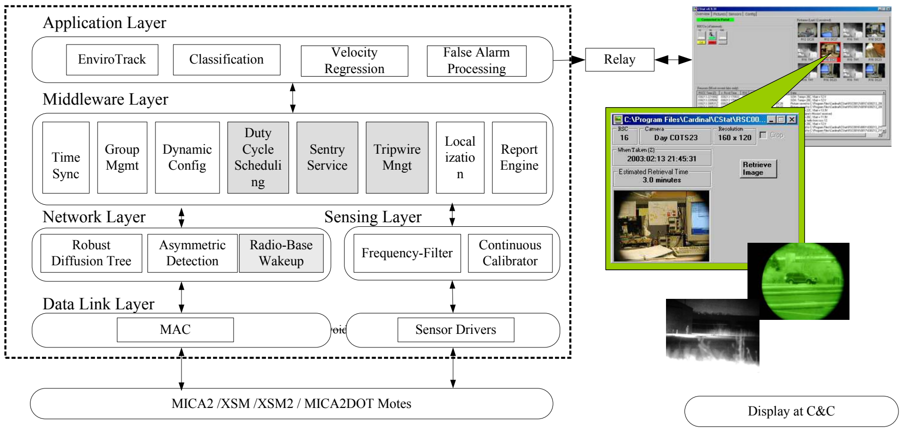


Fig. 4. The VigilNet System Architecture

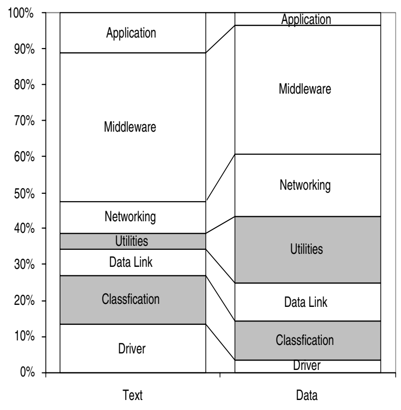


Fig. 5. Memory Layout

of gravity from multiple inputs and reports only one aggregate back to the base.

- **Implicit acknowledgement:** Given that the sensor payload is very small, it might not be energy efficient to acknowledge every packet explicitly. Implicit acknowledgement can be achieved through several approaches. They differ in functionality and overhead. B-MAC [10] provides an efficient implementation of the CSMA protocol with radio-layer acknowledgement support. Observing that most of the packets need to be forwarded for routing, we alternatively implemented the acknowledgement as a special field in outgoing packets. When there are no outgoing packets for a period of time, a special acknowledgement packet is sent.
- **Incremental detection:** Multi-sensing modalities are desired for achieving target classification. However, it is not necessary to activate all sensors only for detection. Among the three types of sensors in XSM motes, the optic TR230 PIR sensor has the longest detection range and a relatively low power consumption,i.e., 0.88mW. We use this sensor to support the initial detection and to incrementally wakeup other sensors for classification purposes.
- **Passive wakeup circuitry :** Several efforts [2][45][32] have been made to support low-power passive wakeup by using an acoustic detector [45], infrared sensor [2] or radio [32]. Currently, the hardware-event-driven design [2] of XSM motes is not mature enough for VigilNet to exploit this aspect. However, this is a very promising direction.

VII. Tradeoff: Performance vs. Energy Efficiency

One key research challenge for VigilNet is to reconcile the need for network longevity with the need for fast and accurate target detection and classification. The former requires most sensor nodes to remain inactive, while the later desires many active sensor nodes. As we mentioned before, the event model directly affects the design of the power management. Energy efficiency can be comparatively easy to achieve if events of interests are ubiquitously present. The data quality of some events, such as temperature and humidity, are not directly correlated with the responsiveness of the system. While in the surveillance system, *responsiveness* and *awareness* directly affect the system performance including tracking and classification. The former can be measured in terms of the detection probability and delay, and the later can be measured in terms of the number of nodes detecting external events, simultaneously. We have investigated responsiveness in previous sections IV-B and IV-C. This section focuses on how to improve the system awareness. In VigilNet, awareness is supported by the *on-demand wakeup service*. The on-demand control is stealthier compared to the periodic control [38], because wakeup beacons are sent only when events occur. To support the on-demand control, we need to guarantee the delivery of wakeup beacons. Because of the special stealthiness requirement, the non-sentries cannot synchronize their clocks with their sentries by exchanging messages. Therefore, neighboring non-sentry motes may no longer have a sleep-wakeup cycle synchronized with each other due to the clock drift, and a sentry cannot keep track which of its neighbors are awake. To guarantee delivery, a non-sentry periodically wakes up and checks radio activity (detects preamble bytes) once per checking period (e.g., every second). If no radio activity is detected, this node goes back to sleep, otherwise it remains active for a period of time, preparing for incoming targets. If a sentry node wants to wake up all neighboring nodes, it only needs to send out a message with a long preamble with a length equal to or longer than the checking period of non-sentry nodes. Since in the rare event model, the wakeup operations are done very infrequently, the long preamble doesn't introduce much energy consumption in sentry nodes. On the other hand, since the amount of time taken to check the radio activity is constant for a specific radio hardware, the length of checking period determines the energy consumption in non-sentry nodes. In general, a long checking period leads to a lower energy consumption. However, to ensure that a sentry node wakes up neighboring non-sentry nodes before a target moves out of their sensing range, the checking period can not be arbitrarily long. Theoretically, the upperbound of checking period is $\frac{\sqrt{R^2-r^2}}{S}$, where R is radio

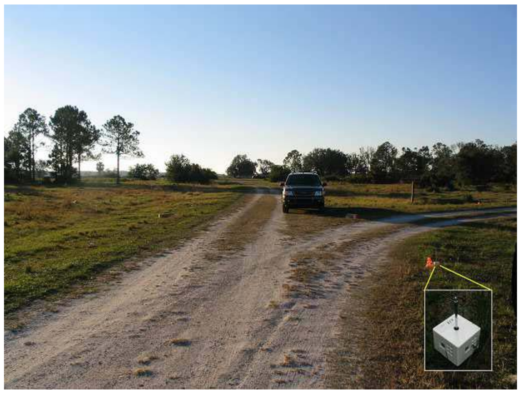
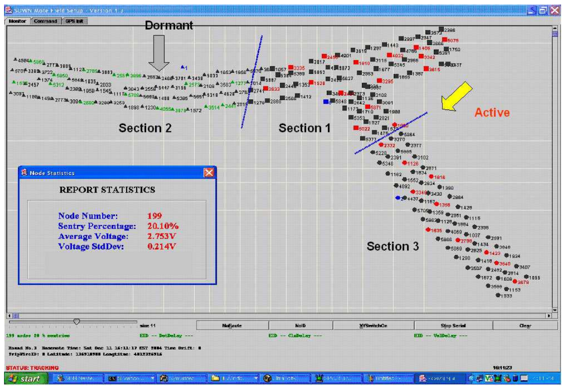


Fig. 6. Location of Deployment and A Deployed XSM Mote



REPORT STATISTICS

Node Number: 199
Sentry Percentage: 20.10%
Average Voltage: 2.753V
Voltage StdDev: 0.214V

```
199 nodes 20 % sentries
END -- DetDelay ---
Road No.3  Basenote Time: Sat Dec 11 16:11:17 EST 2004 Time Drift: 0
TripWireID: 0 Latitude: 136918980 Longitude: 4013326916
STATUS: TRACKING
```

Fig. 7. Tripwire Partition

range, r is sensing range of sentries and S is the speed of target. Due to the other delays, such as sensor warm-up time, the checking period should be smaller than this theoretical bound. In our implementation, non-sentry nodes have 1% duty cycle with 1 second checking period.

VIII. Implementation

The power management architecture described in Section V has been integrated into the VigilNet system. We have successfully transferred VigilNet to a military agency for deployment by the end of 2004. The overarching architecture of VigilNet is shown in Figure 4. The components in gray are specially designed for the power management purpose. Other components provide extra energy-aware features, as mentioned in section VI.

VigilNet is built on top of the TinyOS operating system. TinyOS supports a lightweight event driven computation model with two-level scheduling. VigilNet is mostly written in NesC, a derivative language from C specially designed for embedded programming. This language enables the programmers to define the interfaces, functions of components and the relations (dependencies) among them. The size of VigilNet is about 40,000 lines of code, supporting multiple existing mote platforms including MICA2 and XSM. The compiled image occupies 83,963 bytes of code memory and 3,586 bytes of data memory. The code and data memory maps are shown in Figure 5.

We categorize the system components into seven groups; Data link and sensor driver layers use default components in TinyOS; Network layer consists of three major components: robustness diffusion tree, asymmetric link detection [41] and radio-based wakeup service. The sensing layer provides detection and classification with continuous calibrator and frequency filters [1]. We note that it is very critical to have a sensing subsystem with minimal false alarms in an outdoor environment. Otherwise, the network lifetime is severely reduced due to unnecessary wakeup operations. The application layer focuses on tracking and high-level classification [46]. The middleware layer occupies most code (40%) and data memory (35%). Among all the middleware services, the tripwire service, sentry selection, duty cycle scheduling and wakeup service form the basis for power management subsystem. Their functionalities are supported by other services. For instances, the localization service provides the basis for the tripwire partition and sentry selection. The group management service allows power-efficient data aggregation. The configuration service facilitates the online tuning of the power management parameters. Multilevel sleep modes in the ATmega128 permit a high-granularity control of power dissipation. Selectable transmission power settings (255 levels) in CC1000 enable us to adjust the effective range of sentry declaration messages dynamically.

IX. System Evaluation

This section presents experimental results that evaluate the performance of the power management subsystem. The experimental results in Section IX-A are obtained through an actual deployment of 200 XSM motes, focusing on the sentry selection, tripwire partition and tracking delays. Other experiments in Section IX-B, especially those related to the system lifetime, require a significant amount of time. Unfortunately, we currently can not afford to deploy such a large system unattended for a long time. We have to conduct those evaluations through a hybrid approach, which uses basic measurements from a smaller number of motes as input to a simulator. By doing so, we can investigate the impact of different system configurations on the performance of power management.

A. Field Evaluation

The field evaluation was done as part of a technical transition on December 2004, when we deployed 200 XSM motes on a dirt T-shape road (200 meters by 300 meters). The XSM mote is designed by the joint efforts of Ohio State University [2] and CrossBow Inc, which features an Atmel ATmega128L microcontroller and a Chipcon 433MHz CC1000 radio. Its sensing suite includes magnetic, acoustic, photo, temperature and passive infrared sensors (PIR). Figure 6 displays the environment where our system was located and the picture of one of the XSM motes. Nodes are randomly placed roughly 10 meters apart, covering one 300-meter road and one 200-meter road.

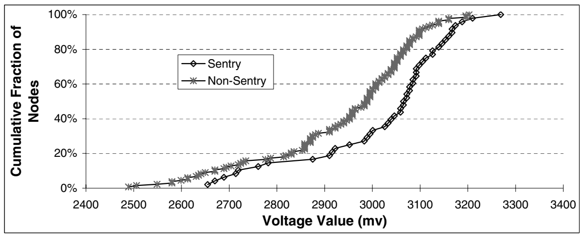


Fig. 8. Effectiveness of Sentry Selection

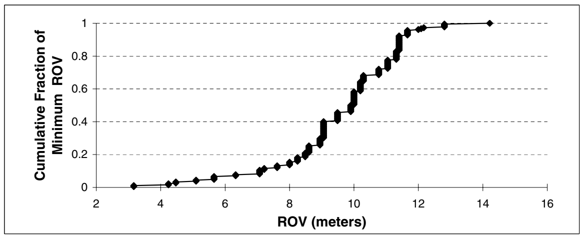


Fig. 9. ROV Enforce Results

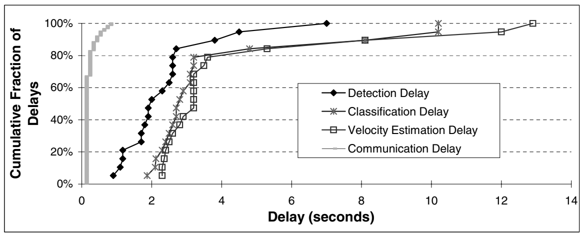


Fig. 10. Distribution of Different Delays

1) *Effectiveness of the tripwire partition:* One snapshot of the network layout collected by our GUI is shown in Figure 7. We placed 200 XSM field motes and 3 mica2dot base motes in the field. Accordingly, the network is divided into three sections. The layout indicates that the Voronoi-based tripwire partitioning is very effective and that all nodes attach to the nearest base nodes through the shortest path.

2) *Effectiveness of the sentry selection:* In this experiment, we evaluate the effectiveness of sentry selection. Figure 8 plots the cumulative distribution function of the voltages of nodes within the network. The left curve is the voltage CDF of non-sentry nodes and the right curve is the voltage CDF for sentry nodes. It confirms that our sentry selection process is effective and that nodes with high remaining energy have a high probability to be chosen as sentries. For instance, none of nodes with voltage below 2.65V is chosen as a sentry. Figure 8 further confirms that it is not the case that nodes with high voltages are always selected as sentries, due to the random jitter introduced in Equation 1 and to the localized selection process on a non-uniform distribution of XSM motes.

3) *Effectiveness of ROV enforcement:* We also investigate the effectiveness of enforcing Range of Vicinity (ROV), when we set the system parameter ROV as 10 meters. Figure 9 shows the cumulative distribution function of minimum distances between sentry-pairs. The average minimum is 9.57 meters with 1.88 meters standard deviation. We note that due to the radio irregularity introduced by the ground effect in the outdoor environments, a small percentage of sentry nodes can not reach each other, even when they are very close (<5 meters) to each other.

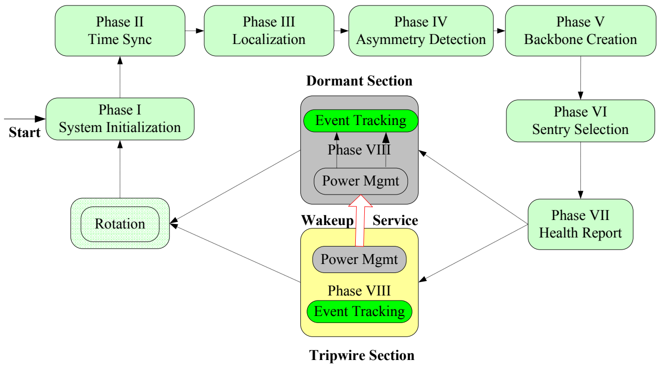


Fig. 11. Phase Transition and Rotation

TABLE I
POWER CONSUMPTION ACCORDING TO THE MOTE STATE.

Node state	Radio State (Messages per second)	Processor State	Sensors State	Total Power
Init	receive (2)	active	off	$49.449mW$
SentrySleep	off (0)	sleep	off	$42\mu W$
NonSentrySleep	LPL (0)	sleep	off	$450\mu W$
AwakeComm	receive (2)	active	off	$49.449mW$
AwakeCommSensing	receive (2)	active	on	$71.45mW$
AwakeSensing	receive (0)	active	on	$70.01mW$

4) *Delays under power management:* In this experiment, we investigate various delays under power management. When a target enters the surveillance area, a detection report is issued first, followed by classification reports. Finally, after sufficient information is gathered, velocity reports are issued. Figure 10 illustrates the cumulative distribution of different delays. The communication delay (leftmost curve) is much smaller compared with other delays. About 80% of detections are done within 2 seconds. Over 80% of the classification and velocity estimations are made within 4 seconds. This empirical result indicates that our power management does not degrade the tracking performance significantly.

TABLE II
KEY SYSTEM PARAMETERS

Parameter	Defi nition	Default Value
SDC	Sentry duty cycle (see IV-C)	25%
STP	Sentry toggle period (see IV-C)	1 second
SSA	Sentry service activation	True
TN	Number of tripwire partitions in the network	1
TDC	Tripwire duty cycle percentage (see IV-A.3)	100%
VS	Target Speed	4 m/s
RN	Number of system rotations per day	1
SR	Sensing Range	10 m
RR	Radio Range	30 m

B. Hybrid Evaluation

In the hybrid evaluation, we use the experimental measurements from the XSM platform as inputs to a discrete event

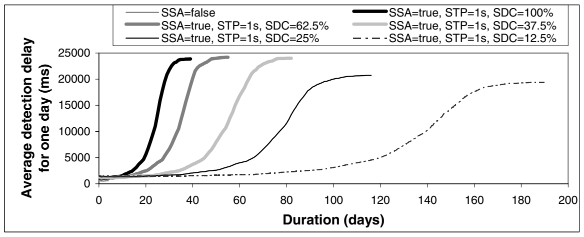


Fig. 12. Influence of sentry duty cycle (SDC) on average detection delay(ADD).

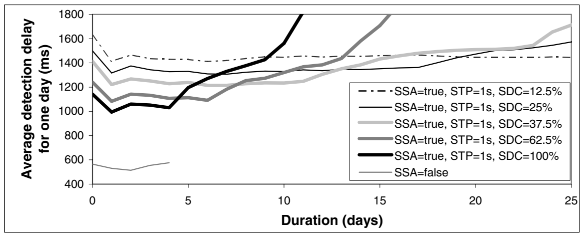


Fig. 13. Influence of sentry duty cycle (SDC) on average detection delay (ADD) (the second view)

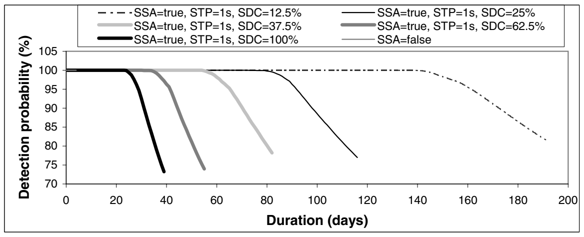


Fig. 14. Influence of sentry duty cycle (SDC) on detection probability (DP).

simulator. This simulator emulates the multi-phase VigilNet operations as shown in Figure 11. We distribute 10,000 nodes randomly within a 1,000,000 m^2 square. VigilNet initializes in three minutes with a sequence of phases (from Phase I to VII). After that, VigilNet enters the surveillance phase (Phase VIII). The system rotates periodically to refresh system-wide soft-states and balance the power consumption. The number of rotations per day is defined as RN as shown in Table II. A target enters the network randomly at one of the edges and exits randomly at the opposite edge of the area. To emulate the sensing delay we experienced in the real testbed, we consider that a target is detected when it is within the sensing range of an active node for at least 5 milliseconds and when that node can reach its tripwire base station to report the event.

1) Battery model: We obtained similar empirical power consumption results as reported in [2], which provides very complete analysis of XSM motes. XSM motes use two standard AA (A91) batteries. Each battery has an energy capacity uniformly chosen between 2,848mAh and 2,852mAh [47]. However, to model reality better [48], we suppose that a mote dies when it has used 85% of the available energy.

The sensor nodes are in one of six power consumption states at any time. We list and detail the power consumption of these six states in Table I. When a message is transmitted, the radio switches to the transmit state for 30ms (a typical time required by XSM nodes to send a message under the MAC contention). The indicated number of messages per second in Table I is an upper bound result from the empirical observations.

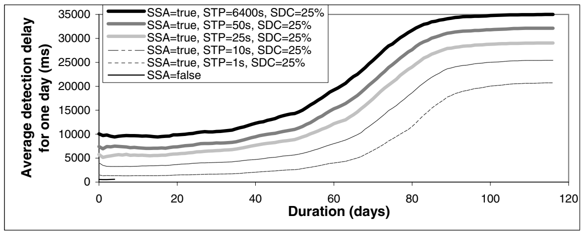


Fig. 15. Influence of sentry toggle period (STP) on average detection delay (ADD).

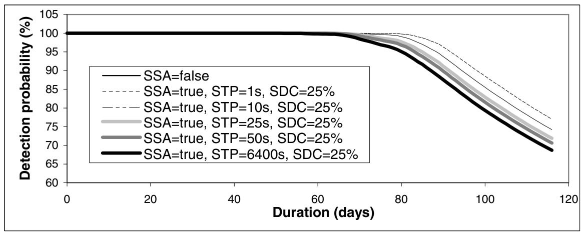


Fig. 16. Influence of sentry toggle period (STP) on detection delay (DP).

2) Performance metrics and system parameters: We investigate three major performance metrics under different system configurations. 1) Detection Probability (**DP**), which is the percentage of successful detections among all targets that enter into the system during one day. 2) Average detection Delay (**ADD**), which is the average time elapsed between the entrance of a target into the area and its detection by one of sensor nodes. 3) Network lifetime (**NL**), which is defined as the number of days for which the detection probability of a target remains greater than 90%. The key system parameters are listed in Table II. Unless mentioned otherwise, the default values in Table II are used in all experiments. The baseline for comparison is VigilNet without any power management.

3) Impact of the sentry service and duty cycle scheduling: In this section, we evaluate the energy savings achieved by the sentry service and the duty cycle scheduling. In particular, we study the influence of the activation of the sentry service (SSA), of the sentry duty cycle (SDC), and of the sentry toggle period (STP) on energy consumption. One hundred targets are simulated during each rotation to obtain statistics in detection probability, but we take into consideration the power consumption of only ten of them (VN=10) as the real workload. As previously mentioned, we use a network of 10,000 nodes randomly distributed within a square of 1km edge length. Each node has a radio range of 30 meters. This configuration matches our real system requirements dictated by the military: nodes have an average of 27.5 neighbors within their communication range, and an average of 3.1 neighbors within their sensing range.

Figures 12, 13 and 14 show the variations of the average

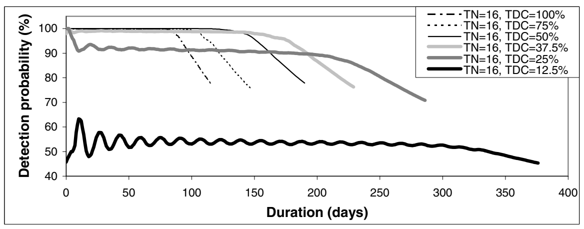


Fig. 17. Influence of tripwire duty cycle (TDC) on detection probability (DP).

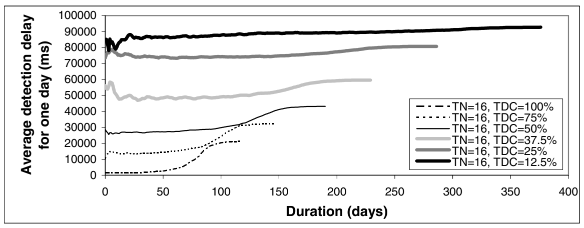


Fig. 18. Influence of tripwire duty cycle (TDC) on average detection delay (ADD).

detection delay, detection probability, and network lifetime, according to the sentry duty cycle. Figure 13 takes a closer look at a particular section of Figure 12. We first observe that without the sentry service (SSA = false), the lifetime of the network (NL) is short: all the nodes run out of energy after only four days. The activation of the sentry service increases the lifetime of the network by approximately seven times. However, it also slightly increases the average detection delay of a target by 1 second. This could be expected as the first nodes that the target encounters may be dormant. We note that the delay is relatively small (e.g., 1∼2 seconds), as shown in Figure 13.

The use of duty cycle scheduling (SDC≠100%) significantly improves the network lifetime. For instance, with a duty cycle of 12.5%, the lifetime of the network is multiplied by about five times. This may be surprising: we would expect the network lifetime when SDC=12.5% to be approximately eight times the network lifetime when SDC=100%. The observed values are due to the energy consumed during the rotation phase and when target detection occurs. These tasks consume a non-negligible amount of energy and therefore impose a limit on the network lifetime.

We remark that during the first four days of network operation, the average detection delay is shorter when the sentry duty cycle is higher. This could be expected when a target enters the sensing range of a sentry node, this node may be in a dormant state. We note that the difference between the average detection delays for different values of SDC (Figure 13) is no more that one second. This can be explained by the short sentry toggle period (1 second).

Figure 14 shows the influence of the sentry duty cycle (SDC) on the detection probability. We observe that, for all configurations, the initial detection probability is 100%. As nodes start to run out of power, the detection probability decreases until all the nodes become dysfunctional. On average, during the network lifetime, the successful detections reported in Figure 13 occur between 0.5 second and 2 second after the target entered the square area. Beyond the network lifetime, VigilNet can still detect the targets, however the delay increase gradually as shown in Figure 12.

In Figure 15 and Figure 16, we study the effect of the sentry toggle period (STP) on the average detection delay and the detection probability. We fix the sentry duty cycle at 25%. We observe that a greater toggle period negatively impacts the average detection delay. Indeed, if the toggle period is small (e.g., 1 second), a dormant sentry, having a target entering its sensing range, wakes up with a high probability before this target exits the sensing range. Conversely, if the toggle period is big (e.g., 6400 second), a dormant sentry has a low probability of being waken up before the target leaves its sensing range.

Guidelines: From the analysis of this section, we can conclude the following. First, to reduce the detection delay, we must choose a sentry toggle period as small as possible. Second, to increase the network lifetime, we advise to select a small sentry duty cycle. However, note that the time during which a sentry remains awake cannot be arbitrarily small, because it is limited by the time necessary to warm up the sensors and by the time necessary to gather enough sensor data to infer whether there is a target or not. Consequently, rapid sensor wake up and quick target detection algorithms are features that can significantly extend the lifetime of a sensor network. Efforts in this direction are worthwhile.

4) Impact of the tripwire service: We investigate both the grid and the random placement of tripwire bases. In the case of a Tripwire Number (TN) greater than 16, the two placement strategies generate similar results. For TN smaller than 16, the grid topology performs better. Due to space constraints, we report here only the results concerning the grid tripwire topology.

We configure the wireless sensor network as in Table II. In Figures 17 and 18, we study the impact of the tripwire duty cycle on the performance of the network. We use sixteen tripwires. As we would expect, the smaller the tripwire duty cycle, the longer the lifetime of the network. For instance, when the tripwire duty cycle equals 25%, the network lifetime is about twice the lifetime obtained when the tripwire duty cycle equals 100%. One could expect a multiplication of the lifetime by four times but this would not take into consideration the energy consumed during the rotation phase and when target detection occurs. Additionally, we observe that the average detection delay is significantly longer when the tripwire duty cycle is small. Indeed, when the tripwire duty cycle is small, a relatively small portion of the network is awake at any given time, and the target may cover a bigger part of the network without being detected. Finally, we notice that a tripwire detection cycle of less than 25% seriously impacts the detection probability during the first weeks of network operation. This is due to the fact that, with such low levels of tripwire activity, a large part of the network may

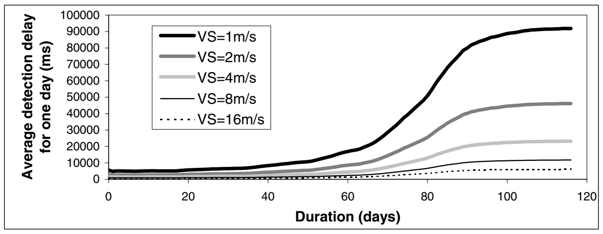


Fig. 19. Influence of target speed (VS) on average detection delay (ADD).

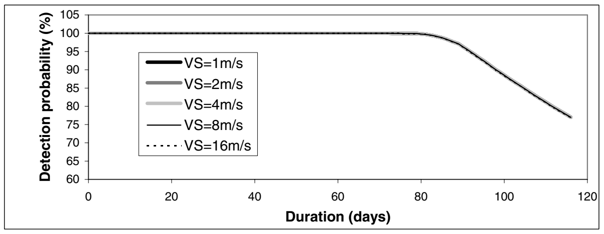


Fig. 20. Influence of target speed (VS) on detection probability (DP).

remain dormant for an extended period of time, producing the possibility that the target crosses the network exclusively through such zones.

Guidelines: From this section, we can conclude the following: a low tripwire duty cycle increases the network lifetime, but increases noteworthily the detection delay and decreases the detection probability. In a grid deployment, a tripwire duty cycle below 25% is detrimental to the performance. This, on the other hand, indicates the importance of the smart tripwire placement strategy under a low tripwire duty cycle.

5) Impact of the targets speed: In this section, we study the effect of the target speed on performance. The configuration of the network is the same as in Table II. Figures 19 and 20 show the influence of target speed on average detection delay and detection probability. We observe that a high target speed decreases the detection delay. This may be surprising as when the target speed increases, it spends less time within the sensing range of a given sensor, thereby decreasing the probability of being detected. However, as the target speed increases, it covers more motes in a shorter amount of time. The effect of target speed on detection probability is insignificant for VS$\leq$16m/s. We recall that the sensing range of a sensor is 10 meters in this experiment. At a speed of 16m/s, the target spends a maximum of 1250ms within the sensing range of a given sensor. This duration is sufficient for a sensor to differentiate the target from false alarms.

Guidelines: To summarize the results from this section, we can conclude the following. First, it takes more time to detect slow targets than faster ones. Second, a network with characteristics similar to the one defined in this experiment can handle targets with speeds typical of moving terrestrial objects (up to at least 16 meters per second i.e. 35.8 miles per hour).

X. Conclusion

This paper presents a recent major effort to address the energy efficiency for outdoor long-term surveillance. It is a comprehensive case study on power management in a realistic environment with a large testbed. We investigate the power management at the network, section and node level by using a novel tripwire service, sentry service and duty cycle scheduling, respectively. We invest a significant amount of effort to validate our system with a network of 200 XSM motes in an outdoor environment, an extensive simulation with 10,000 nodes, as well as an analytical probabilistic model. These demonstrate the effectiveness of our approach and identify several useful guidelines and lessons for the future development of energy-efficient sensor systems.

XI. Acknowledgements

This work was supported in part by the DAPRPA IXO offices under the NEST project (grant number F336615-01-C-1905), the MURI award N00014-01-1-0576 from ONR and NSF grant CCR-0098269. The authors specially thank the NEST program manager Vijay Raghavan for his valuable contributions.

Appendix

Figure 21A shows a rectangular deployment area with sides a and b. N sensor nodes are uniformly deployed in the area, therefore, the node density d is N/ab. Assuming that the area is considerably large, therefore, the number of nodes in an area of A ($A \ll ab$) can be approximated by a Poisson distribution with parameter $\lambda = dA$. We assume that the entry point of the intruding target (intruder) is uniformly distributed along all sides, and to make the problem tractable, we assume that the intruder moves along a straight line. The angle between the target direction and the side where the entry point is located is θ, which is also considered uniformly distributed and $\theta \in [0, \pi]$. The whole picture of the intruding scenario is shown in Figure 21A. Observe that the area of nodes that can detect the intruder contains all points whose distances to the intruder's locus are no larger than sensing range r. If the length of the intruder's locus in the deployment area is L , the detection area can be approximated by $2Lr$, without considering the edge effect. Based on Poisson distribution, the probability that there is at least one node in this area is $1 - e^{-2Lrd}$.

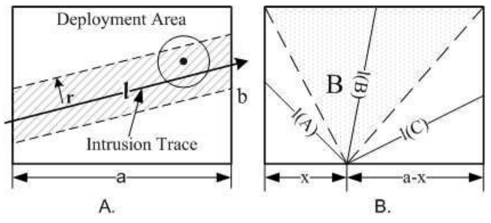


Fig. 21. Intrusion Model

As shown in Figure 21B, the deployment area is divided into three regions. The length of the intrusion trace is:

$$L(\theta, x) = \begin{cases} x/\cos\theta & Locus \in A \\ b/\sin\theta & Locus \in B \\ (x-a)/\cos\theta & Locus \in C \end{cases} \quad (2)$$

We can calculate the expected detection probability $Expected(P_{detection})$ that an intruder is detected by at least one node by integrating over all entry points on the four adjacent sides of the area.

$$Expected(P_{detection}) = 1 - \frac{F(a,b,r,d) + F(b,a,r,d)}{\pi(a+b)} \tag{3}$$

where $F(m,n,r,d) =$

$$\int_0^m \begin{array}{l} [\int_0^{\arctan(\frac{n}{x})} e^{-\frac{2rxd}{\cos\theta}} d\theta + \int_{\arctan(\frac{n}{x})}^{\pi - \arctan\frac{n}{m-x}} e^{-\frac{2rnd}{\sin\theta}} d\theta \\ + \int_{\pi-\arctan(\frac{n}{m-x})}^{\pi} e^{-\frac{2r(x-m)d}{\cos\theta}} d\theta] dx \end{array} \tag{4}$$

REFERENCES

[1] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A.Tirumala, Q. Cao, J. A. S. T. He, T.Abdelzaher, and B. Krogh., "Lightweight Detection and Classification for Wireless Sensor Networks in Realistic Environments," in *SenSys 2005*, 2005.

[2] P. Dutta, M. Grimmer, A. Arora, S. Biby, and D. Culler, "Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events," in *IPSN'05*, 2005.

[3] J. A. Paradiso and T. Starner, "Energy Scavenging for Mobile and Wireless Electronics," *IEEE Pervasive Computing*, vol. 4, no. 1, 2005.

[4] S. Roundy, P. K. Wright, and J. Rabaey, "a Study of Low Level Vibrations as a Power Source for Wireless Sensor Nodes," *Computer Communications*, vol. 26, no. 11, 2003.

[5] *Mica2 data sheet*, CrossBow, available at http://www.xbow.com.

[6] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks," in *SenSys 2003*, November 2003.

[7] T. Yan, T. He, and J. Stankovic, "Differentiated Surveillance Service for Sensor Networks," in *SenSys 2003*, November 2003.

[8] M. L. Sichitiu, "Cross-layer scheduling for power efficiency in wireless sensor networks," in *IEEE Infocom*, march 2004.

[9] M. Cardei, M. Thai, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in *IEEE Infocom*, march 2005.

[10] J. Polastre and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks," in *SenSys 2004*, November 2004.

[11] T. van Dam and K. Langendoen, "An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *SenSys 2003*, November 2003.

[12] W. Ye, J. Heidemann, and D. Estrin, "An Energy-Efficient MAC Protocol for Wireless Sensor Networks," in *INFOCOM*, 2002.

[13] K. Seada, M. Zuniga, A. Helmy, and B. Krishnamachari, "Energy Efficient Forwarding Strategies for Geographic Routing," in *SenSys 2004*, November 2004.

[14] Y. Xu, J. Heidemann, and D. Estrin, "Geography-informed Energy Conservation for Ad Hoc Routing," in *MobiCom*, 2001.

[15] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Ad Hoc Sensor Networks," in *ICDCS'03*, May 2003.

[16] M. Agarwal, J. H. Cho, L. Gao, and J. Wu, "Energy-efficient broadcast in wireless ad hoc networks with hitch-hiking," in *IEEE Infocom*, march 2004.

[17] Y. Yu, B. Krishnamachari, and V. K. Prasanna, "Energy-Latency Tradeoffs for Data Gathering in Wireless Sensor Networks," in *IEEE INFOCOM*, 2004.

[18] W. Choi and S. Das, "A novel framework for energy-conserving data gathering in wireless sensor networks," in *IEEE Infocom*, march 2005.

[19] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks," in *OSDI'02*, December 2002.

[20] N. Shrivastava, C. Buragohain, S. Suri, and D. Agrawal, "Medians and Beyond: New Aggregation Techniques for Sensor Networks," in *SenSys 2004*, November 2004.

[21] S. Bhattacharya, H. Kim, S. Prabh, and T. Abdelzaher, "Energy-Conserving Data Placement and Asynchronous Multicast in Wireless Sensor Networks," in *MobiSys'03*, May 2003.

[22] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," in *MOBICOM'01*, 2001.

[23] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," in *HICSS*, 2000.

[24] D. Ganesan, R. Cristescu, and B. Berefull-Lozano, "Power-Efficient Sensor Placement and Transmission Structure for Data Gathering under Distortion Constraints," in *IPSN'04*, 2004.

[25] A. Bogdanov, E. Maneva, and S. Riesenfeld, "Power-aware base station positioning for sensor networks," in *IEEE Infocom*, march 2004.

[26] R. Szewczyk, A. Mainwaring, J. Anderson, and D. Culler, "An Analysis of a Large Scale Habit Monitoring Application," in *SenSys 2004*, November 2004.

[27] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin, "A Wireless Sensor Network for Structural Monitoring," in *SenSys 2004*, November 2004.

[28] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next Century Challenges: Mobile Networking for Smart Dust," in *MOBICOM'99*, August 1999.

[29] K. Kar, A. Krishnamurthy, and N. Jaggi, "Dynamic node activation in networks of rechargeable sensors," in *IEEE Infocom*, march 2005.

[30] A. Keshavarzian, E. Uysal-Biyikoglu, F. Herrmann, and A. Manjeshwar, "Energy-efficient link assessment in wireless sensor networks," in *IEEE Infocom*, march 2004.

[31] M.Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The Flooding Time Synchronization Protocol," in *SenSys 2004*, November 2004.

[32] L. Gu and J. A. Stankovic, "Radio-Triggered Wake-Up Capability for Sensor Networks," in *RTAS'04*, 2004.

[33] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic, "Towards Optimal Sleep Scheduling in Sensor Networks for Rare Event Detection ," in *IPSN'05*, 2005.

[34] A. Woo, T. Tong, and D. Culler, "Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks," in *SenSys 2003*, November 2003.

[35] J. Liu, J. Reich, and F. Zhao, "Collaborative In-Network Processing for Target Tracking," *J. on Applied Signal Processing*, March 2003.

[36] A. Arora, P. Dutta, and S. B. et.al, "A Wireless Sensor Network for Target Detection, Classification, and Tracking," *Computer Networks (Elsevier) Systems*, 2003.

[37] G.Simon and et. al., "Sensor Network-Based Countersniper System," in *SenSys 2004*, November 2004.

[38] T. He, S. Krishnamurthy, J. A. Stankovic, and T. Abdelzaher, "An Energy-Efficient Surveillance System Using Wireless Sensor Networks," in *MobiSys'04*, June 2004.

[39] R. Stoleru, T. He, and J. A. Stankovic, "Walking GPS: A Practical Solution for Localization in Manually Deployed Wireless Sensor Networks," in *EmNetS-I*, October 2004.

[40] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial Tessellations:Concepts and Applications of Voronoi Diagrams*. Wiley, 2000.

[41] G. Zhou, T. He, and J. A. Stankovic, "Impact of Radio Irregularity on Wireless Sensor Networks," in *MobiSys'04*, June 2004.

[42] R. Williams, *Circle Coverings*. New York: Dover, 1979.

[43] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-Free Localization Schemes in Large-Scale Sensor Networks," in *MOBICOM'03*, September 2003.

[44] T. He, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher, "AIDA: Adaptive Application Independent Data Aggregation in Wireless Sensor Networks," *ACM Transactions on Embedded Computing System, Special issue on Dynamically Adaptable Embedded Systems*, 2004.

[45] D. H. Goldberg, P. J. Andreas G. Andreou, P. O. Pouliquen, L. Riddle, and R. Rosasco, "A Wake-up Detector for an Acoustic Surveillance Sensor Network: Algorithm and VLSI Implementation," in *IPSN'04*, 2004.

[46] T. F. Abdelzaher, B. M. Blum, Q. Cao, D. Evans, J. George, S. George, T. He, L. Luo, S. H. Son, R. Stoleru, J. A. Stankovic, and A. Wood., "EnviroTrack: An Environmental Programming Model for Tracking Applications in Distributed Sensor Networks," in *ICDCS'04*, March 2004.

[47] *Energizer current batteries datasheets*, Energizer, available at http://www.energizer.com.

[48] *Mica2 aa battery pack service life test*, CrossBow, available at http://www.xbow.com/Support/.

Achieving Real-Time Target Tracking Using Wireless Sensor Networks

Tian He[§], Pascal Vicaire[†], Ting Yan[†], Liqian Luo[*], Lin Gu[†], Gang Zhou[†], Radu Stoleru[†], Qing Cao[*], John A. Stankovic[†] and Tarek Abdelzaher[*]

[†]Department of Computer Science, University of Virginia
[§]Department of Computer Science and Engineering, University of Minnesota

Abstract

Target tracking systems, consisting of thousands of low-cost sensor nodes, have been used in many application domains such as battlefield surveillance, wildlife monitoring and border security. These applications need to meet certain real-time constraints in response to transient events, such as fast-moving targets. While the real-time performance is a major concern in these applications, it should be compatible with other important system properties such as energy consumption and accuracy. Hence, it is desirable to have the ability to exploit the tradeoffs among them. This work presents the real-time design and analysis of VigilNet, a large-scale sensor network system which tracks, detects and classifies targets in a timely and energy efficient manner. Based on a deadline partition method and theoretical derivations of each sub-deadline, we are able to make guided engineering decisions to meet the end-to-end tracking deadline. To confirm our design and obtain an empirical understanding of these tradeoffs, we invest significant efforts to perform large-scale simulations with 10,000 nodes as well as a field test with 200 XSM motes, running VigilNet. The results from both analysis and evaluation can serve as general design guidelines to build similar real-time systems.

1 Introduction

Recent developments in sensor techniques make wireless sensor networks (WSNs) available to many application domains [6, 12, 17, 26, 32]. Most of these applications, such as battlefield surveillance, disaster and emergency response, deal with various kinds of real-time constraints in response to the physical world. For example, surveillance may require a sensor node to detect and classify a fast moving target within 1 second before it moves out of the sensing range. Compared with the traditional distributed systems, the real-time guarantee for sensor networks is more challenging due to the following reasons. First, sensor networks directly interact with the real world, in which the physical events may exhibit unpredictable *spatiotemporal* properties. These properties are hard to characterize with the traditional methods. Second, although the real-time performance is a key concern, it should be performance compatible with many other critical issues such as energy efficiency and system robustness. For example, it is not efficient to activate the sensors all the time only for the benefit of a fast response. This naive approach severely reduces the system lifetime [12]. Third, the resource constraints restrict the design space we could trade off. For example, the limited computation power in sensor nodes makes the Fast Fourier Transformation not quite suitable for real-time detection. All these issues challenge us with two questions. *How to make the design of a large-scale real-time sensor network system manageable?* And *how to trade off among system metrics while maintaining the real-time guarantee?* Our answer to these questions, presented in this paper, is a case study of the VigilNet system, a real-time outdoor tracking system using a large-scale wireless sensor network.

Our contribution lies in the following aspects: 1) This work addresses a real-world application with a running real-time system, designed and implemented over last few years. 2) We investigate multi-dimensional tradeoffs between the real-time performance and other system properties. Such investigation provides the guidance for the future development of similar systems. 3) The real-time design and tradeoffs are validated by a large-scale field evaluation with 200 XSM motes and an extensive simulation with 10,000 nodes. These evaluations reveal quite a few practical design suggestions that can be applied to other real-time sensor systems.

The remainder of the paper is organized as follows: Section 2 introduces the tracking process in VigilNet. Section 3 identifies the real-time requirements. Section 4 provides a real-time analysis of VigilNet's tracking performance and its tradeoffs. In Section 5, we evaluate the real-time performance of VigilNet in an outdoor field test. In Section 6, we conduct a large-scale simulation to further validate and analyze the real-time issues in VigilNet. Section 7 discusses the related work. Section 8 concludes the paper.

[*]Liqian Luo, Qing Cao and Tarek Abdelzaher are now with University of Illinois, Urbana-Champaign

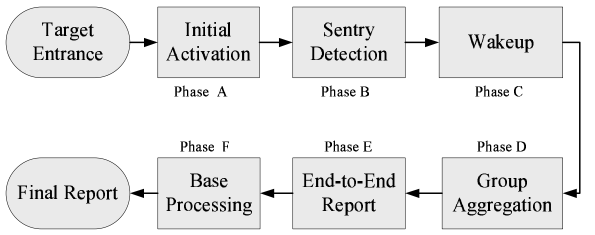


Figure 1: The Delay Breakdown in Tracking Operation

2 Overview of VigilNet Tracking Operations

VigilNet is an energy-efficient surveillance and tracking system, designed for spontaneous military operations in remote areas. In these areas, the events of interest happen at a relatively low rate, i.e. the duration of significant events (e.g., intruders) is very short, compared with the overall system lifetime (e.g., 5-minute tracking per day). According to our empirical results [13], nearly 99% of energy is consumed during the idle-waiting period for potential targets. Therefore to conserve energy, the most effective approach is to selectively turn a subset of nodes off, and wake them up on demand in the presence of significant events. This power management technique fundamentally shapes the VigilNet tracking process. It introduces a set of new delays that traditional tracking systems do not experience.

In this section, we give a brief overview of the VigilNet tracking operation, serving as a background for the real-time design and analysis in the following sections. As shown in Figure 1, after a target enters the area, it activates the first sensor node that can confirm the detection, then other nodes nearby are waken up to form a group to deliver the aggregated reports to the base. More specifically, the VigilNet tracking operation has six phases:

A. **Initial Activation:** VigilNet stays in the power management state when there are no targets. The power management protocol puts nodes into either one of two states: *sentry* and *non-sentry*. In brief, a node becomes a sentry node if it is a part of the routing infrastructure or it needs to provide the sensing coverage. Otherwise, it becomes an inactive non-sentry node. The details of sentry selection can be found in [12]. If the sentry nodes are active 100% of time (i.e. the deployed area is always covered), any incoming target is covered by at least one sentry node immediately. On the other hand, if the sentry nodes have a certain duty cycle (i.e. they go to sleep and wake up periodically to save energy), there will be an initial activation delay, denoted as $T_{initial}$, before the first sentry node starts to sense the incoming target.

B. **Initial Target Detection:** After the initial activation, it takes a certain delay, defined as $T_{detection}$, for the first sentry node to *confirm* the detection. This delay consists of the hardware response delay, the discrete sampling delay and the delay to accumulate a sufficient number of samples before a detection algorithm recognizes the target.

C. **Wake-up:** Normally, the detection from a single sentry node does not provide sufficient confidence in detection and classification, therefore a group-based tracking is designed in VigilNet. In order to form a group with a reasonable size, non-sentry nodes need to be waken up after the initial target detection by a sentry node in Phase B. We define the wake-up delay T_{wakeup} as the time required for a sentry node to wake up other sleeping non-sentry nodes. This delay is determined by the time to broadcast the wake-up messages.

D. **Group Aggregation:** Once awaken, all nodes that detect the same target join the same logic group to establish a unique one-to-one mapping between this logic group and the detected target. Each group is represented by a leader to maintain the identity of the group as the target moves through the area. Group members (who by definition can sense the target) periodically report to the group leader. A leader starts to report detection to the base after the number of member reports exceeds a certain threshold, defined as the degree of aggregation (DOA). We use $T_{aggregation}$ to denote the group aggregation delay, which is the time required to collect and process the detection reports from the member nodes.

E. **End-to-End Report:** After the group aggregation, the leader node reports the event to the nearest base. Multiple bases are used to partition a network into several sections, in order to bound the end-to-end delivery delay T_{e2e}.

F. **Base Processing** (T_{base})**:** A base is in charge of processing the reports from the leaders of different logic groups. Since the reports from the same logic group are spatiotemporal correlated, a string of consecutive reports can be further analyzed and summarized for end users. For example, taking the time stamps and the locations of targets as the inputs, a base uses the least-square estimation to obtain the velocity of each target.

3 Real-Time Requirement in VigilNet

To ensure the effectiveness of the target tracking, VigilNet must meet a certain real-time constraint. Specifically, VigilNet should detect, classify and analyze the incoming targets within a certain end-to-end deadline (e.g.,5 seconds from Phase A to F). As shown in Section 2, this deadline involves complex analysis of the whole tracking process. It is not scalable for us to identify a system-wide feasible region within such a high-dimensional design space. Therefore, we adopt the deadline partition method to divide the end-to-end deadline into multiple sub-deadlines. The sub-deadline partition varies with the system configurations. As a concrete example, supposing a target enters the field with a speed up to 20 mph, to guarantee that this target can be detected by the first sentry node with a probability higher than 90% , we need to design a detection algorithm with a sub-deadline less than 1 second, assuming the detection range is 10 meters.

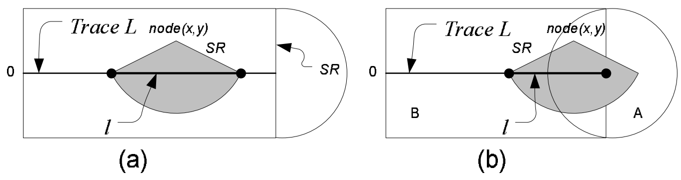


Figure 2: Detection Probability

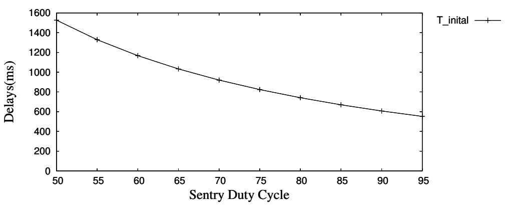


Figure 3: Initial Delay vs. SDC

By confining the real-time decisions within each phase, we make the end-to-end analysis manageable in a lower-dimensional design space. As long as the individual sub-deadlines are met, we have a certain guarantee on the end-to-end delay. To achieve this, we present a set of real-time designs in next section.

4 VigilNet Real-Time Tracking Analysis

The description of this section follows the natural order of VigilNet's tracking operations presented in Section 2. Such design and analysis is validated later with a real system implementation consisting of 200 XSM nodes as well as a large-scale simulation in Section 5 and Sections 6, respectively.

4.1 Initial Activation Delay and Its Tradeoffs

In a duty-cycle-based power management scheme, the sentry nodes go to sleep and wake up periodically. In this case, the initial activation delay $T_{initial}$ may not be zero, because sentry nodes near the target's entry point may be asleep when the target enters the field. In this section, we identify a quantitative relationship between the energy savings and the $T_{initial}$, which helps us make decisions to guarantee that the initial activation finishes within a given sub-deadline $D_{initial}$.

In our VigilNet design, all sentry nodes agree on a common sentry toggle period P and a common sentry duty cycle SDC. For each period, a sentry wakes up randomly and stays awake for $P \cdot SDC$, then goes to sleep. Assuming a target enters the tracking area from point 0 for L meters as shown in Figure 2(a), we first derive P_r, the probability that a single sentry node detects this target. Obviously, the nodes that may detect the target must be in the rectangle or the semi-circle shown in the Figure 2(a). The size of the area is $2SR \cdot L + \pi \cdot SR^2/2$, where SR is the Sensing Range. For a single node located at (x, y) in this area, the probability that the node detects the target $P(x, y)$ is $SDC + l(x, y)/(P \cdot TS)$, where $l(x, y)$ is the overlapping length of the node's sensing range and the target's trace, and TS is the Target Speed. If we consider all possible locations in this area, we can get P_r in Equation 1 by integrating and normalizing the $P(x, y)$ over the area. We note that when x, y is in the circle (area A) as shown in Figure 2(b), $l(x, y) = \sqrt{SR^2 - y^2} + L - x$. When (x, y) is in area B, $l(x, y) = 2\sqrt{SR^2 - y^2}$.

$$P_r = \frac{\int\limits_A (SDC + \frac{\sqrt{SR^2-y^2}+L-x}{P \cdot TS})ds + \int\limits_B (SDC + \frac{2\sqrt{SR^2-y^2}}{P \cdot TS})ds}{(2SR \cdot L + \pi SR^2/2)} \tag{1}$$
$$= SDC + \frac{\pi \cdot SR \cdot L}{(2L + \pi \cdot SR/2) \cdot TS \cdot P}$$

We note that P_r calculated by Equation 1 is valid only when the target speed is faster than $2SR/(P - P \cdot SDC)$. We have also derived a slower-target case, which is of less interest to the real-time tracking. Therefore, we omit it here due to the space constraint, please refer [3] for more details.

Now we are ready to provide a statistical real-time guarantee for the initial activation process, i.e. we need to ensure a target is detected before the sub-deadline $D_{initial}$. Equivalently, a target should be detected before it enters for $L = TS \cdot D_{initial}$ meters. Obviously, $P(T_{initial} < D_{initial})$ equals $P(T_{initial} \cdot TS < L)$, where $P(T_{initial} \cdot TS < L)$ is the probability that at least one of nodes in the area (A+B) detects the target. If there are n nodes in the area, the probability that at least one of them detects the target is $1 - (1 - P_r)^n$. Suppose the sentry density is D_s and n conforms to a Poisson distribution with parameter $\lambda = (2SR \cdot L + \pi \cdot SR^2/2)D_s$, therefore, the probability that the initial activation finishes before sub-deadline $D_{initial}$ is:

$$P(T_{initial} < D_{initial}) = P(T_{initial} \cdot TS < L) = 1 - e^{-P_r \cdot \lambda} \tag{2}$$

Equation 2 identifies a feasible region for us to decide the system parameters such as sentry duty cycle (SDC) and sensing range (SR) to ensure the real-time property in Phase A. In addition, we can obtain the expected value of $T_{initial}$ from the formula $E(T_{initial}) = \int_0^\infty (1 - P(T_{initial} < t))dt = \int_0^\infty (1 - P(SD < TS \cdot t))dt$. According to Equations 1 and 2, we have the expected delay for a fast target:

$$E(T_{initial}) = \frac{e^{-SDC \cdot \pi \cdot SR^2 \cdot D_S/2}}{(2SR \cdot SDC \cdot TS + \pi SR^2/P)d_S} \tag{3}$$

One caveat in the analysis needs some attention. Above we derive the expected detection delay for a duty cycle based system with *random deployment*. However, sentry nodes are located more evenly than totally randomly case [12]. Fortunately, we can prove that the random deployment case provides a theoretical upper bound for the sentry-based deployment case. It can be easily proved that if for all t, $P(T_1 < t) > P(T_2 < t)$, we must have $E(T_1) < E(T_2)$. For $0 < P_r < 1$, $1 - (1 - P_r)^n$ is a

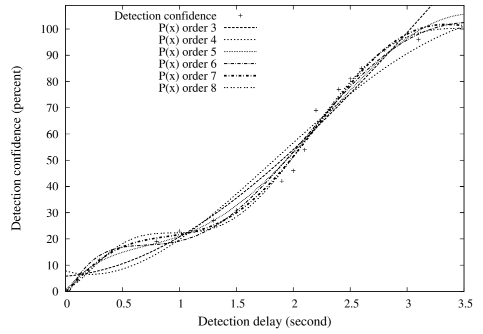


Figure 4: Detection Confidence vs. Detection Delay

strictly concave function of n. Therefore, $E(1-(1-P_r)^n) \le 1-(1-P_r)^{E(n)}$, and the left side of the equation equals the right side if and only if n is a constant. Given the same $E(n)$, the more scattered the distribution of n is, the smaller the value of $E(1-(1-P_r)^n)$ is. Since the sentry nodes are selected more uniformly than the random case, $P(T_{initial} < D_{initial})$ for the sentry based system is greater than a totally randomly distributed system, and therefore the expected delay is smaller. The expected delay for the random case can be used as an upper bound for the expected detection delay for a more evenly distributed system. Later, we will see from the simulation that the analytical result overestimates the $T_{initial}$ by 15%.

We can further take the detection delay $T_{detection}$ into account, since a successful detection in Phase B activates a full tracking process. In this case, we establish an equivalent model for $T_{initial}$. Specifically, in Equation 3, we substitute SDC with the effective sentry duty cycle $SDC_{eff} = SDC - T_{detection}/P$ and substitute SR with the effective sensing range $SR_{eff} = \sqrt{SR^2 - (T_{detection} \cdot TS/2)^2}$. Figure 3 gives a more concrete view of the tradeoff between SDC and expected $T_{initial}$. We take parameters from the real VigilNet implementation: $D_S = 0.01 node/m^2$, $P = 10s$, $SR = 10m$, $TS = 10m/s$ and $T_{detection} = 1000ms$. This result is consistent with what we obtained from the real experiments and simulations.

4.2 Sentry Detection Delay and Its Tradeoffs

After the initial delay in Phase A, a target approaches the vicinity of a sensor, which begins to observe a different signal pattern than that without a target. With the current sensing algorithms, the signal pattern can be amplitude, frequency, or a combination of the two. We call the signal pattern corresponding to a target *a target signature*. The recognition of a target signature indicates a sensor-level detection, and produces data for higher-level detection and classification algorithms.

As defined before, $T_{detection}$ is the time for a detection algorithm to recognize a target signature. This delay must be smaller than a certain sub-deadline $D_{detection}$. Multiple reasons contribute to this delay. First, the sensor hardware has a response delay for the physical signals that the target generates. Second, the sensing circuitry requires special operations with a further delay. For example, the magnetometer in MICA2 node [5] takes about $35ms$ to stabilize after the potentiometer adjustment. Third, the sampling is discrete and periodic, not continuous, which leads to sampling delay. Finally, the target signature itself may be time related (e.g., a certain frequency), which can not be recognized by just one sample.

Now we describe how to decide the sub-deadline $D_{detection}$. Obviously, a detection algorithm must finish before a target moves out of the sensing range of a node. Suppose that the nominal sensing area is a circle with a fix sensing range SR, the amount of time a target stays in a node's sensing range can be derived from the speed of the target, TS, and the minimum distance from the target's trajectory line to the sensor node. Since the target trajectory intersects with the sensing circle randomly, we assume this minimum distance is uniformly distributed within $[0, R)$, therefore the probability of a target stays in one sensing circle for at least $D_{detection}$ seconds can be calculated as

$$\begin{array}{ll} P(t > D_{detection}) = \sqrt{1 - \frac{(TS \cdot D_{detection})^2}{4SR^2}} & D_{detection} < \frac{2SR}{TS} \\ P(t > D_{detection}) = 0 & D_{detection} \ge \frac{2SR}{TS} \end{array} \qquad (4)$$

According Equation 4, the sub-deadline $D_{detection}$ can be decided by choosing a desired $P(t > D_{detection})$ value.

In addition, we desire to know how a detection algorithm performs under a given sub-deadline $D_{detection}$. We define the *Detection Confidence* (DC), as the confidence on the target detection, i.e. 100% DC indicates this sensor has no doubt about the existence of the target. Normally, the longer $D_{detection}$ is, the more information about target signature a sensing algorithm can obtain, and therefore, it can achieve a higher detection confidence DC. Such relationship depends on the type of sensors. In order to quantitatively analyze the relation between DC and $D_{detection}$ as a case study, we performed experiments on XSM motes with the magnetic sensing algorithm detecting a moving vehicle in an outdoor environment. We approximate the sensing range as 7 meters around the sensor node, according to empirical data. Figure 4 plots the relation between the detection confidence and the detection delay, based on the experiments. As we can see from the figure, DC does not have a linear relation to $D_{detection}$. Based on experimental measurements, we use a polynomial to characterize DC versus $D_{detection}$. Figure 4 shows a series of polynomials of different orders that fit the points representing the relation between the detection confidence and the detection delay. The plotting indicates that the polynomials of an order higher than 5 are fairly close to each other and fit the points well. Hence, we choose the polynomial of order 5 to characterize the relation, as shown below.

$$DC = f(D_{detection}) = \sum_{i=0}^{5} a_i D_{detection}^i \qquad (5)$$

The coefficients of the polynomial calculated from the curve fitting are $a_5 = 1.0999, a_4 = -13.1138, a_3 = 51.3443, a_2 = -73.2343, a_1 = 54.6671, a_0 = 0.2402$. The polynomial

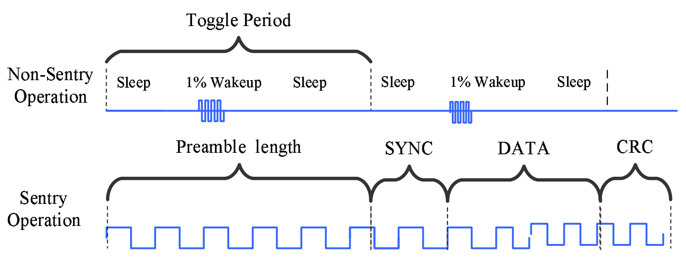


Figure 5: The Wake-up Operation

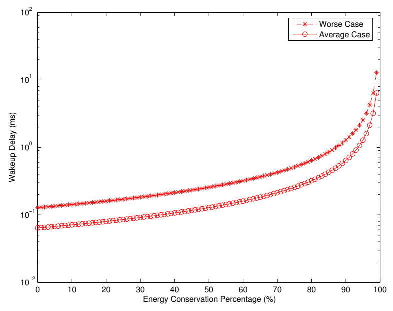


Figure 6: Wakeup Delay Vs. Non-Sentry Energy Saving

$f(D_{detection})$ characterizes the relation of the detection confidence and the imposed sub-deadline $D_{detection}$ when the vehicle is moving at a relatively low speed. In the scenarios where the vehicles move faster, the detection delay tends to be shorter and the detection confidence will be higher because the targets impose a faster change to the sensor readings. Hence, $f(D_{detection})$ represents a conservative estimation of the detection confidence, given a certain amount of time available to the sensor node to capture and process the target signals.

We note that in the analysis of the time-related properties of the sensing algorithms, we choose such a conservative-case approach instead of a worst-case approach. In many cases, the worst-case scenario is a rare event that the system is not designed to handle well. For example, with the magnetic sensing algorithm, the worst-case of detection delay is infinity – if a vehicle moves extremely slowly, it provides a low-frequency signal just as the back-ground noise, resulting in a non-detection for that target. We note that an analysis with such a worst-case scenario provides little insight into the system. To represent a reasonably practical scenario, we study a conservative case in which a target can be detected.

In conclusion, according to Equation 4 and 5, running a detection algorithm with a sub-deadline $D_{detection}$, one node can detect $P(t > D_{detection})$ percent of targets with DC percent of the confidence in detection. This analysis justifies the benefits of fast detection algorithms and the need for group aggregation to improve the detection confidence.

4.3 Wake-up Delay and Its Tradeoff

Once a target is detected in Phase B, we need more nodes to join in order to increase the confidence in detection. We design a wake-up service to activate the non-sentry nodes after the sentry nodes detect the incoming targets. Different target speeds impose different sub-deadlines D_{wakeup} to the wake-up services.

Normally the wake-up service can be supported either through hardware or software. Several hardware solutions have been proposed in [6, 9]. Since the wake-up circuits accumulate the ambient energy slowly, the current hardware solutions are not fast enough for the real-time target tracking. Therefore, we propose a software-based wake-up strategy, which has a short average delay and a predictable worse-case delay. The wake-up operation goes as shown in Figure 5. A non-sentry actually does not sleep all the time. It periodically wakes itself up, quickly senses the radio activity at a particular frequency. If no radio activity is detected, this node goes back to sleep, otherwise it remains active and starts to sample the environment. We control the non-sentry operation through two parameters: *Toggle Period* (TP) and *Channel Clear Access duration* (CCA). The toggle period is defined as the time interval between two consecutive wake-up instances. The CCA is defined as the minimal time for a radio module to detect the existence of the radio signal. For example, the CC1000 radio transceiver takes at least $2ms$ (8 symbol periods, as specified by 802.15.4 [16]) to access the radio activity. Based on TP and CCA, we can get the Non-Sentry Duty Cycle ($NSDC$) as $\frac{CCA}{TP}$. At the sentry side, once a sentry detects a target, it broadcasts a radio message with a long preamble. This long preamble is guaranteed to be sensed by neighboring non-sentry nodes as long as this preamble has a length equal or longer than the toggle period of non-sentry nodes. The worst-case wake-up delay WC_{Delay} equals TP. In another word, the sub-deadline D_{wakeup} can be ensured trivially in our design by setting $TP = D_{wakeup}$. Let the power consumption for an active node during a unit of time be E, the energy consumption for a non-sentry node is $\frac{E \times CCA}{TP}$. Since the amount of time to check the radio activity (CCA) is constant for a specific radio hardware, the length of the toggle period determines the energy consumption rate in non-sentry nodes. In general, a long toggle period TP leads to a low energy consumption, however also leads to a long delay in waking up the non-sentry nodes. Figure 6 shows such a tradeoff, using the CC1000 radio transceiver for MICA2/XSM motes as an example. As shown in Figure 6, a sub-deadline of 200ms lead to a 99% energy saving for the non-sentry nodes.

4.4 Aggregation Delay and Its Tradeoffs

Once all nodes near the target are awaken in Phase C, the group-based tracking begins. To avoid an excessive power consumption, instead of relaying every detection message back, VigilNet sends only aggregates to the base stations for further processing. Such online aggregation process is subject to a certain sub-deadline $D_{aggregation}$ determined by the target speed and the node density.

Specifically, we organize nodes in the vicinity of a target into one group. We use semi-dynamic leader election [21] to minimize the delay. Nodes that detect the target become the group members, which, upon detection, immediately report their own locations and sensing data to a leader. The leader then averages the locations of members as the estimates of the target positions, and sends such estimates to a base station. To filter out the sporadic false alarms of individual nodes, we introduce a configurable parameter, DOA (Degree of Aggregation), which forces the leader to withhold reports to a base station until the number of received member reports reaches DOA. To achieve a high confidence in target detection, one should set a high DOA value (e.g., 4). On the other hand, a higher DOA value inevitably introduces a longer group aggregation delay since the leader waits longer to expect more member reports. This tradeoff allows us to choose appropriate DOA to meet the sub-deadline $D_{aggregation}$.

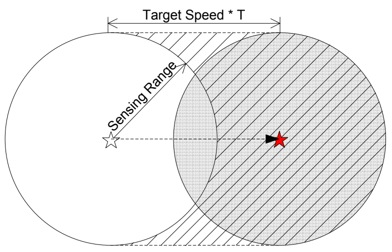


Figure 7: The Detection Areas Before and After Movement

The relation between DOA and the group aggregation delay is complicated by various factors, e.g., the sensing range, the target speed, and the node density. Therefore, we make several assumptions to simplify the analysis, including a circular sensing range, a straight target trajectory and randomly distributed nodes. Based on these assumptions, Figure 7 depicts the movement of a target with a speed TS for a time period T. Again, the sensing range of the target is SR. The white circle and the grey circle denote the detection area of the target before and after movement, respectively. Nodes located in the diagonally lined area are the new detectors of the target, which contribute to DOA by sending reports to the leader. To guarantee a certain sub-deadline $D_{aggregation}$, the number of new detectors must exceed or equal DOA before the sub-deadline $D_{aggregation}$:

$$D_{aggregation} \geq T_{aggregation} = \frac{DOA}{2 \cdot SR \cdot TS \cdot D} \qquad (6)$$

where D represents the node density. Note that after the wake-up process, not only the sentry nodes but also the non-sentry nodes participate in the tracking. Equation 6 quantitatively reveals a feasible region for us to guarantee the sub-deadline $D_{aggregation}$. For example, if the network density (D) and the sensing range (SR) are fixed, we can exploit a feasible solution, using different DOA values under different target speeds. Figure 8 gives a more concrete design space by depicting the group aggregation delay for varied DOA values and target speeds when the sensing range is 10m, the node density is 1 per 100 m^2. We note that this result is consistent with the results obtained form large-scale simulation shown in Section 6.

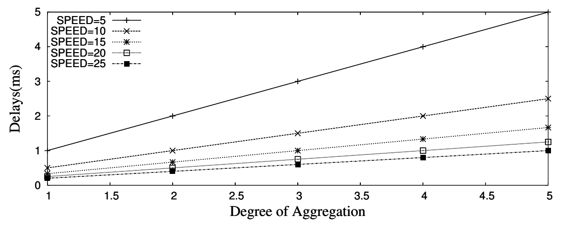


Figure 8: Minimal Group Aggregation Delay for Varying DOA and Target Speed

4.5 Communication Delay and Its Tradeoff

After group aggregation in Phase D, the leader delivers the aggregated tracking reports to a nearby base. Suppose the end-to-end communication sub-deadline is D_{e2e} and one-hop worst case communication delay is T_{WC_MAC}, we need to ensure that the number of hops is smaller than D_{e2e}/T_{WC_MAC}. For a given node density, the hop length L_{hop} can be estimated through Kleinrock-Silvester formula [19], which gives the correlation between the hop length L_{hop}, the communication range CR and the number of neighbors N as:

$$L_{hop} = CR \times (1 + e^{-N} - \int_{-1}^{1} e^{-\frac{N}{\pi}(arccos(t) - t\sqrt{1-t^2})} dt) \qquad (7)$$

Therefore, to guarantee a sub-deadline D_{e2e}, when we deploy the network, we should ensure that every node can reach a base within a radius of L_{e2e}:

$$L_{e2e} = \frac{D_{e2e} \cdot L_{hop}}{T_{WC\_MAC}} \qquad (8)$$

In VigilNet, the sub-deadline D_{e2e} is guaranteed by partitioning the whole network into multiple sections based on the Voronoi diagram [24]. Specifically, a network with n bases is partitioned into n Voronoi sections such that each section contains exactly one base and every node in that Voronoi section is closer to its base than to any other base inside the network.

4.6 Base Processing Delay and Its Tradeoffs

After a base receives the reports delivered in phase E, it performs the high-level processing such as the velocity estimation. In order to do so, a base node needs to accumulate several reports from the network. The delay to accumulate the reports T_{base} is subject to its sub-deadline D_{base}. We defined the minimal number of reports needed by the base as K. This value can be one, if the in-networking processing is sufficient. The frequency of reports depends on the speed of the target and the aggregation of locations from nodes at different locations. From the analysis in the section 4, we know that after the target enters the system for time t, the expected number of nodes can sense the target is $(\pi \cdot SR^2/2 + 2SR \cdot TS \cdot t)D$. Obviously, if the target goes further for Δt, the expected number is increased by $2SR \cdot TS \cdot \Delta t$. Considering the detection delay $T_{detection}$, only nodes that are $\sqrt{SR^2 - (T_{detection} \cdot TS/2)^2}$ meters away from the target trajectory can recognize the target. Therefore, we can estimate the number of report (NR) generated before the sub-deadline D_{base} as:

$$NR = (2TS \cdot D \cdot \sqrt{SR^2 - (T_{detection} \cdot TS/2)^2}) \cdot D_{base} \quad (9)$$

Alteratively, to guarantee D_{base}, we need to select the K, the minimal number of reports needed by the base, a value smaller than NR.

Now we consider how the selection of K impacts the accuracy in velocity estimation. Since each report only approximates the target location, there is an error in the result of velocity estimated using the least square method. Without loss of generality, we first consider the velocity along the x-axis. Statistics has established the variance of the estimated slope in a two-variable least square linear regression as

$$\frac{\sigma^2}{\sum_{i=1}^{K}(x_i - \bar{x}^2)},$$

where σ is the standard deviation of the disturbance, which in our case is the detection error of a single report; x_i in our case is a timestamp. It is hard to get the distribution of $\sum_{i=1}^{K}(x_i - \bar{x})^2$, but a rough estimation can be obtained by a simplification so that the values of x_i are evenly distributed and $x_i = i/(2D \cdot SR \cdot TS \cdot P_R)$. Thus we can get an estimation of the standard deviation of the velocity:

$$\frac{4\sigma \cdot D \cdot SR \cdot TS \cdot P_R}{\sqrt{3K(K+1)(K-1)}}, \quad (10)$$

where σ is the standard deviation of the location error of a single report. Equation 10 reveals the tradeoff between the accuracy in tracking and the delay in base processing. In brief, T_{base} increases linearly with the number of reports required and the standard deviation of the velocity estimation reduces approximately linearly with $K^{-3/2}$.

4.7 Summary of the Analysis and Tradeoffs

Dealing with the physical world, many sensor-based systems must respond to external stimuli within certain time constraints. Such constraints could change overtime with the changes of the application objectives. For example, a surveillance system should be able to track fast vehicles at a high-energy budget as well as slow personnel at a smaller budget. So it is desirable for a system designer to have the ability to trade off the system parameters to satisfy certain real-time constraints. In this section, we use the deadline partition method to guarantee the sub-deadline of each phase, consequently guarantee the end-to-end deadline. This approach makes the real-time design for a complex sensor network manageable. Since VigilNet aims at various tracking scenarios, for a given end-to-end deadline, the actually partition among the phases would vary significantly. Our analysis is independent of how the sub-deadlines are assigned, which give the designer more flexibility to exploit the feasible regions until the end-to-end real-time requirement is met.

We note that this analysis can be generally applied to other tracking systems with or without certain features. For example, the tracking system presented in [2] does not consider the power management, which makes the analytical results of $T_{initial}$ and T_{wakeup} trivially zero, while other analytical results are still applicable. We also note due to the unpredictable and statistical nature of environmental inputs (e.g., a target could move infinitely slowly), VigilNet is not quite amenable to the traditional worst-case real-time analysis. Nevertheless, the analytical results we provide can assist the designer to provide soft real-time guarantee and make guided decisions on the system configurations. In the next section, we validate our real-time design and analysis through a physical test-bed with 200 XSM motes as well as a large-scale simulator with 10,000 nodes.

5 Evaluation on Real System Performance

In the evaluation, we validate the analytical results as well as provide more insights on the timing issues from the real system and simulation perspectives.

5.1 System Configurations

A large portion of code of VigilNet is written in NesC [7], an modulized extension to the C programming language. Since the concept of traditional OS kernels does not exist in TinyOS [14], a NesC programmer can directly access the hardware devices including the sensors and flash memory, which facilitates the time analysis within a single node [23]. The network infrastructure in VigilNet is a multi-path diffusion tree rooted at bases. The contention-based B-MAC protocol [25] is the default media access control protocol, which has certain uncertainty in the communication delay. Three detection algorithms are designed separately for acoustic, magnetic and motion sensors. They identify the target signatures through a lightweight classification scheme as described in [8]. VigilNet consists 40,000 lines of code, supporting multiple existing mote platforms including MICA2, MICA2dot and XSM. The compiled image occupies 83,963 bytes of code memory and 3,586 bytes of data memory.

As a real-time online tracking system, VigilNet is designed to complete detection, classification and velocity estimation within 4 seconds. The field test was done on a T-shape dirt road in Florida as shown in Figure 9 from the aerial view. We deployed

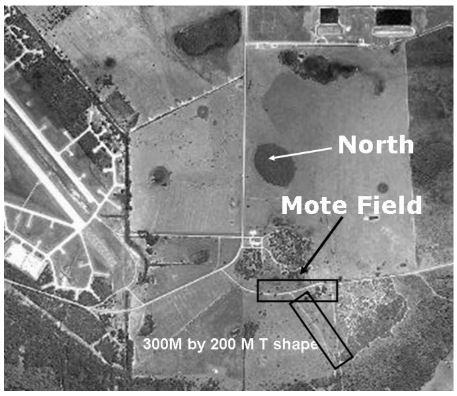


Figure 9: Deployment Site

200 XSM motes which are equipped with CC1000 radio, magnetic, acoustic, photo, temperature and passive infrared sensors (PIR). Along the road, nodes were randomly placed roughly 10 meters apart, covering one 300-meter road and one 200-meter road. Through localization [28, 10], nodes were aware of their positions. In order to measure various kinds of delay, all nodes within VigilNet synchronized with the base within 1∼10 milliseconds using the techniques described in [22]. The time stamps of various actions such as initial detection were sent back to the base, so that we can calculate the delay. We used a Ford Explorer that weighted about 4000 lbs. as the target.

5.2 Delay Measurements

When a car enters the surveillance area at about 10 meters per second (22 mph), a detection report is issued first, followed by classification reports. Finally, after sufficient information is gathered, velocity reports are issued. Figure 10 illustrates the cumulative distribution of different delays. The communication delay (leftmost curve) is much smaller compared with other delays. About 80% of detections are done within 2 seconds. Over 80% of the classification and velocity estimations are made within 4 seconds. The empirical results from most runs are consistent with our analysis in Section 4 and the simulation results in Section 6.

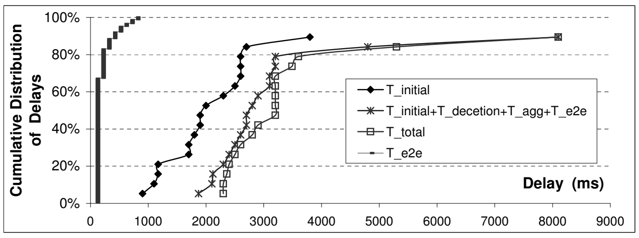


Figure 10: Various Delays Measurements from Field Test

We emphasize here that field experiments indicate that VigilNet meet its real-time requirement and our real-time analysis can approach the reality with a reasonable precision, despite the amount of complexity within the VigilNet (30 protocols integrated). On the other hand, we acknowledge that due to various physical constraints, field experiments can only exploit a very limited design space and obtain a limited amount of data. Therefore, to understand the real-time properties in VigilNet at scale with a much large context, we provide a large-scale simulation in the next section.

6 Large-Scale Simulation

Our simulator emulates the tracking operations as shown in Figure 1. We distribute 10,000 nodes randomly within a 100,000 m^2 rectangle area. We run each experiment 30 times with different random numbers. The figures are plotted with the average value as well as the 95% confidence interval.

6.1 Experiment Setup

We note that our evaluation does not choose deadline/ sub-deadline miss ratios as the major metrics, because such an approach reveals less information about the tradeoff between actual delays and other system performance parameters. Since the mean value and 95% confidence intervals of the delays are plotted in the figures, one can determine the appropriate system settings for a given deadline requirement.

In our experiments, we study several system-wide parameters that directly affect the real-time properties of VigilNet. These parameters are: 1) the target speed (TS), 2) the physical delay in detection ($T_{detection}$), 3) the sentry duty cycle (SDC), 4) the non-sentry duty cycle (NSDC), 5) the required degree of aggregation (DOA), 6) the sensing range (SR) and 7) the required number of reports for base processing (K). We match the simulations with the analysis to see how well they fit with each other.

We use the settings from the VigilNet system as the default values for these system parameters, which are listed in Table 1. Unless mentioned otherwise, the default values in Table 1 are used in all experiments. The metrics used to measure the system performance are mainly the six types of delays discussed in Section 2, the end-to-end delay and the energy consumption per day per node.

Table 1: Key System Parameters

Parameter	Definition	Default Value
TS	Target Speed	10 m/s
SDC	Sentry Duty Cycle	50%
NSDC	Non-Sentry Duty Cycle	1%
DOA	Degree of Aggregation	1%
SR	Sensing Range	10
K	Reports required by the base	1
D	Node Density	0.01 m^2

6.2 Performance vs. Target Speed

The target speed determines the spatiotemporal distribution of events over a certain time period. It is crucial to understand

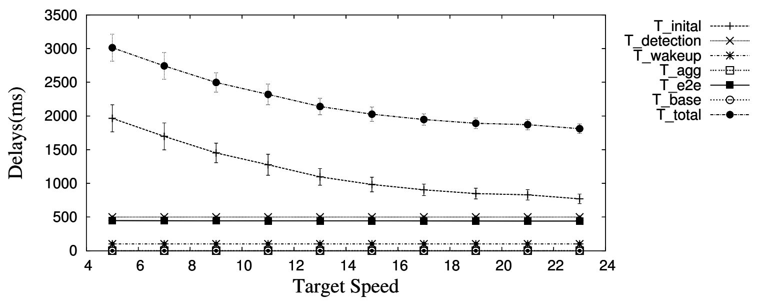


Figure 11: Delays vs. Target Speed

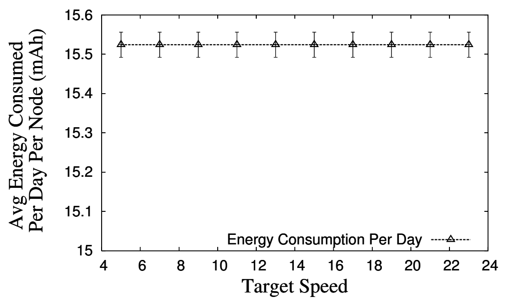


Figure 12: Energy Consumption vs. Target Speed

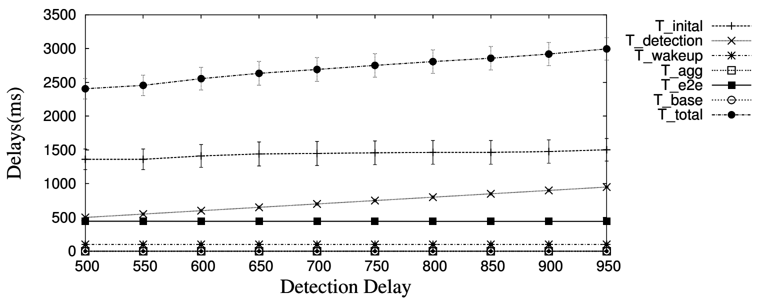


Figure 13: Delays under Varying Detection Delay

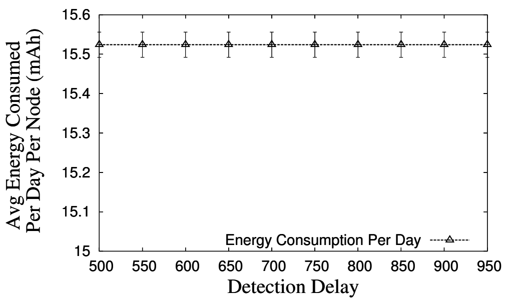


Figure 14: Energy Consumption vs. Detection Delay

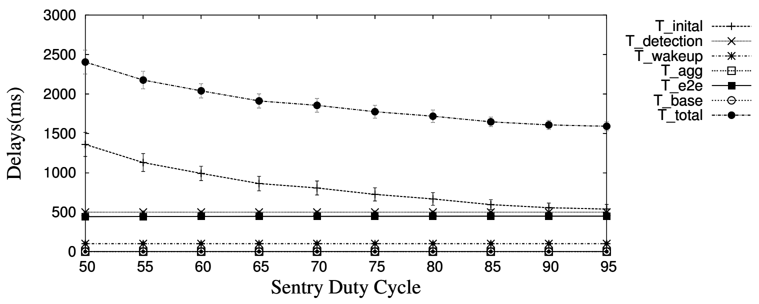


Figure 15: Delays vs. Sentry Duty Cycle

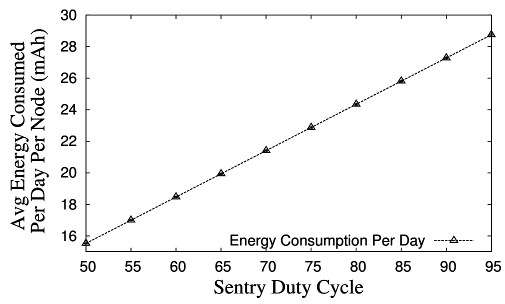


Figure 16: Energy Consumption vs. Sentry Duty Cycle

its impacts on the tracking performance. In this experiment, we incrementally increase the target speed (TS) form 5m/s to 15m/s in steps of 1 meter. As expected from our analysis in Section 4, $T_{initial}$ and $T_{aggregation}$ decrease with the target speed as shown in Figure 11. One interesting observation is that the descend rate of $T_{initial}$ diminishes when TS becomes larger. This is because that a node needs a sufficient sensing time to ensure detection. It is possible that a quick target passes one sensor without detection, which negatively affects the $T_{initial}$. Since VigilNet deals with a rare event model, the energy consumed during the tracking is not perceptibly affected by the target speeds as shown in Figure 12.

6.3 Performance vs. Detection Delay

Different tracking systems use different sensing devices and detection algorithms, which have various detection delays $T_{detection}$. In this experiment, we increase the delay in the detection algorithm $T_{detection}$ from 500 ms to 1000 ms in steps of 50 ms. It is interesting to observe in Figure 13 that at a speed of 10m/s, the detection delay has a small impact on the initial delay, however it contributes most significantly to the overall increase of the total tracking delay. Again, since the detection time is relatively small, this system parameter does not noticeably affect the overall energy consumption as shown in Figure 14.

6.4 Performance vs. Sentry Duty Cycle

From the analytical results in Section 4, we obtain an analytical delay curve between $T_{initial}$ and SDC in Figure 3. In this experiment, we obtain another curves (Figure 15) through the simulation. By comparing these two results, we conclude that they are consistent with each other. For example, at a default 50% duty cycle, $T_{initial}$ obtained from the analysis in Figure 3 is 1600ms , while $T_{initial}$ obtained form the simulation (Figure 15) is 1360ms (Note that our analysis is relatively conservative). In addition, Figure 16 reveals that the energy consumption escalates linearly with the SDC, which indicates an efficient sentry scheduling algorithm is beneficial.

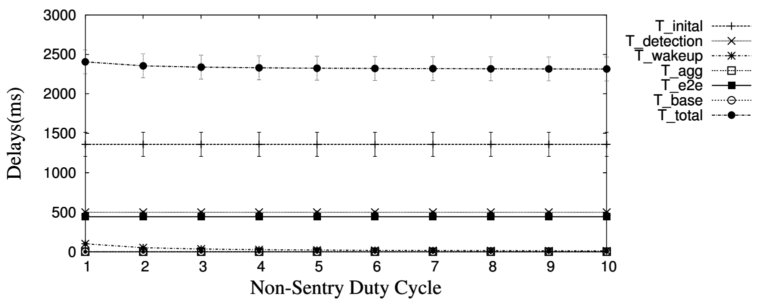


Figure 17: Delays vs. Non-Sentry Duty Cycle

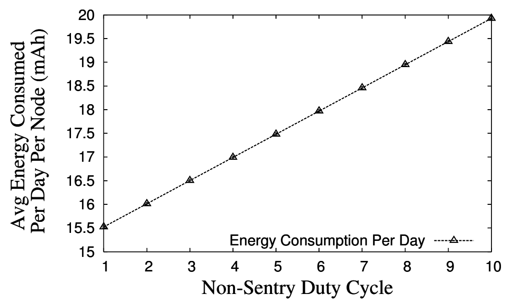


Figure 18: Energy Consumption vs. Non-Sentry Duty Cycle

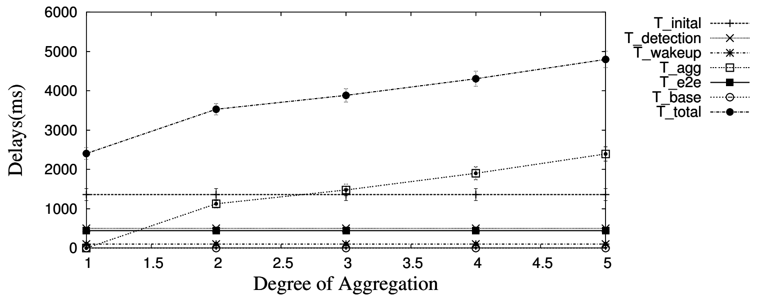


Figure 19: Delays vs. DOA

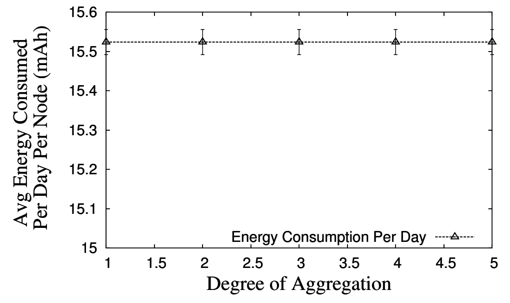


Figure 20: Energy Consumption vs. DOA

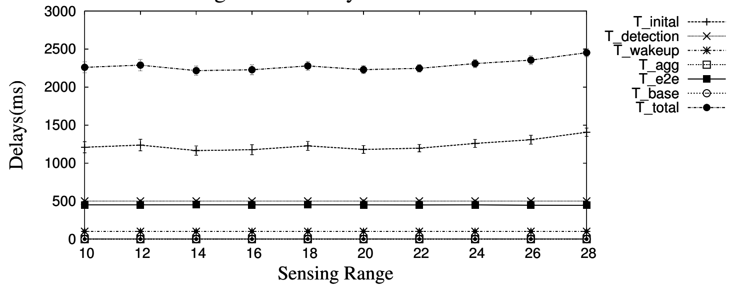


Figure 21: Delays vs. Sensing Range

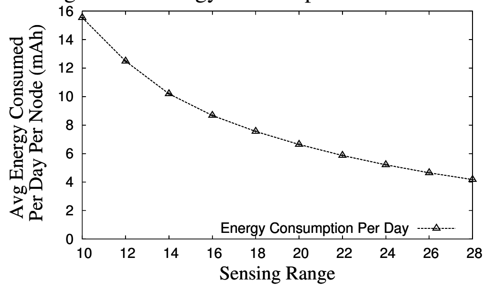


Figure 22: Energy Consumption vs. Sensing Range

6.5 Performance vs. Non-sentry Duty cycle

Here, we evaluate the impact of the wake-up operation on the delay and energy consumption. First, the simulation results confirm that the average wake-up delay is approximately half of the toggle period as predicted in Section 4.3. Since the wake-up delay T_{wakeup} is an order of magnitude smaller than other delays such as $T_{initial}$, a slight decrease in the wake-up delay shown in Figure 17 does not noticeably impact the overall delay. However, interestingly a slight increase of the Non-Sentry Duty Cycle leads to a significant increase of energy consumption as shown in Figure 18. This is because that the non-sentry nodes are by far the majority, so an duty-cycle increase of the non-sentry nodes leads to a quick increase in the total energy. This result indicates that it is beneficial to increase the wake-up delay, when possible, in exchange of the energy saving.

6.6 Performance vs. DOA

In-network processing through data aggregation can reduce the amount of data transmit over the network and increase the confidence in target detection. However to accumulate enough report, it inevitably introduces a certain delay. This experiment studies the effects of data aggregation. We gradually increase the DOA threshold for a leader to report to base. Since the DOA value only affects the tracking phase, which has a small energy consumption, DOA's impact on the energy consumption is not noticeable. On the other hand, with a larger DOA value, it takes more time for a leader to collect the member reports. For example as shown in Figure 19, it takes as long as 2.39 seconds to achieve DOA value of 5. We note that this simulation result is again consistent with the analytical results shown in Figure 8, which has an estimated delay of 2.5 seconds.

6.7 Performance vs. Sensing Range

To accommodate various requirements in detection and classification, different tracking systems use sensors with different ranges. Figure 21 and Figure 22 investigate the impact of sensing range to the tracking performance and energy consumption. With a large sensing range, a smaller number of sentry is required. Therefore, the total energy consumption decreases quickly. For example in Figure 21, the energy reduces by 75%

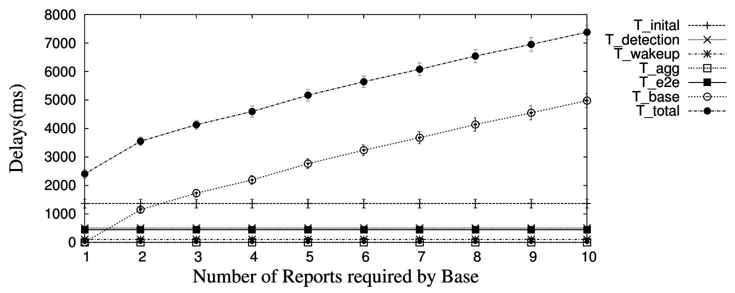


Figure 23: Delays vs. Num of Required Reports

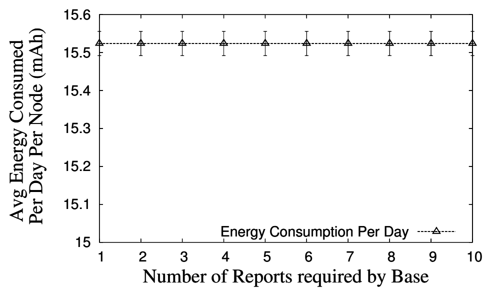


Figure 24: Energy Consumption vs. Num of Required Report

when the sensing range increases from 10m to 28m. It is interesting to see that the initial delay $T_{initial}$ actually slightly increases. This is because the number of sentry nodes reduces while the coverage per sensor increases, the total coverage by all sentry nodes remains the same. We can derive from Equation 3 that expected $T_{initial}$ is higher when the sensing range is smaller, given the same coverage in both cases. This analytic result is confirmed by the simulation results shown in the Figure 21. Due to the space constraint, we omit the detailed derivation here.

6.8 Performance vs. Number of Reports

To improve the estimation of target velocity and to classify targets with a high confidence, a base node normally needs to accumulate a certain number of spatiotemporal related reports from the same logic tracking group. This experiment investigates the impact of the number of reports required by a base to the tracking delays. Obviously, this only affects T_{base}. Figure 23 shows that T_{base} approximately increases linearly with the number of reports, which is expected by our analytical results in Section 4.6. Since the operation is done at the base, there is no energy impact to the sensor network as shown in Figure 24.

7 Related Work

Real-time protocols play an important role to guarantee the effectiveness of the interactions between wireless sensor networks and the physical world. RAP [20] uses a novel velocity monotonic scheduling to prioritize the real-time traffic and enforce such prioritization through a differentiated MAC Layer. Woo and Culler [31] propose an adaptive rate control scheme to achieve fairness among the nodes with different distances to a base station. Huang [15] et al. propose the Mobicast protocol to provide just-in-time information dissemination to nodes in a mobile delivery zone. Given the complete knowledge of traffic pattern, Li [18] proposes a SLF message scheduling algorithm to exploit spatial channel reuse, so that deadline misses can be reduced. The Lightning protocol [30] localizes the acoustic source with a bounded delay regardless of the node density. Carley [4] designs a periodic message scheduler to provide a contention-free predicable medium access control. Somasundara [27] proposes a mobile agent scheduling algorithm to collect the buffered sensor data, before the buffer overflow occurs at the sensor nodes.

Besides the real-time protocol design, several research focuses on the time analysis for sensor networks. In [23], Mohan et al. provides a cycle-accurate WCET analysis tool for the applications running on the Atmega Processor Family. Abdelzaher [1] derives a real-time capacity bound for multi-hop wireless sensor networks. It is a sufficient schedulability condition for a class of fixed priority packet scheduling algorithms. Using this bound, one can determine whether a certain traffic pattern can meet its real-time requirement before hand.

With advances in the sensor techniques, several large-scale sensor systems have been built recently. The GDI Project [29] provides an environmental monitoring system to record animal behaviors for a long period of time. The shooter localization system [26] collects the time-stamps of the acoustic detection from different nodes within the network to localize the positions of the snipers. These systems mention some timing issues, however they do not treat real-time as a major concern. Our previous publications on VigilNet [12, 11] focus on the middleware services and overarching system integration. To the best of our knowledge, this work is the first to analyze the real-time performance and its tradeoffs in a real-world large-scale wireless sensor system.

8 Conclusion

In this paper, we demonstrate the feasibility to design a complex real-time sensor network, using the deadline partition method, which guarantees an end-to-end tracking deadline by satisfying a set of sub-deadlines. We also analytically identify the tradeoffs among system properties while meeting the real-time requirements. We validate our design and analysis through both a large-scale simulation with 10,000 nodes as well as a field test with 200 XSM nodes. We contribute a set of tradeoffs that are useful for the future development of real-time sensor systems. Given real-time constraints, a system designer can make guided engineering judgements on the system parameters such as the network density, the appropriate detection algorithm and the duty-cycle settings for the sensor nodes.

Finally, we acknowledge that although it is amenable to provide the worst-case real-time analysis for a certain protocol such as the wake-up protocol in Section 4.3. However, due to the dynamic and unpredictable nature of the sensor networks, it is a long-term research goal for us to achieve precise worst-case real-time analysis across the whole system.

Acknowledgements

This work was supported in part by the DAPRPA IXO offices under the NEST project (grant number F336615-01-C-1905), the MURI award N00014-01-1-0576 from ONR and NSF grant CCR-0098269. The authors specially thank the NEST program manager Vijay Raghavan for his valuable contributions.

References

[1] T. F. Abdelzaher, S. Prabh, and R. Kiran. On Real-Time Capacity Limits of Multihop Wireless Sensor Networks. In *IEEE RTSS*, 2004.

[2] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A Wireless Sensor Network for Target Detection, Classification, and Tracking. *Computer Networks (Elsevier)*, 2004.

[3] Q. Cao, T. Yan, T. Abdelzaher, and J. Stankovic. Analysis of target detection performance for wireless sensor networks. In *DCOSS'05*, 2005.

[4] T. W. Carley, M. A. Ba, R. Barua, and D. B. Stewart. Contention-free periodic message scheduler medium access control in wireless sensor/actuator networks. In *IEEE RTSS*, 2003.

[5] CrossBow. *Mica2 data sheet*, 2003. Available at `http://www.xbow.com`.

[6] P. Dutta, M. Grimmer, A. Arora, S. Biby, and D. Culler. Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events. In *IPSN'05*, 2005.

[7] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC Language: A Holistic Approach to Networked Embedded Systems. In *Proceedings of Programming Language Design and Implementation (PLDI) 2003*, 2000.

[8] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A.Tirumala, Q. Cao, J. A. S. T. He, T.Abdelzaher, and B. Krogh. Lightweight Detection and Classification for Wireless Sensor Networks in Realistic Environments. In *SenSys'05*, 2005.

[9] L. Gu and J. A. Stankovic. Radio-Triggered Wake-Up Capability for Sensor Networks. In *Proceedings of RTAS*, 2004.

[10] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-Free Localization Schemes in Large-Scale Sensor Networks. In *MOBICOM'03*, September 2003.

[11] T. He, S. Krishnamurthy, L. Luo, T. Yan, R. Stoleru, G. Zhou, Q. Cao, P. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, and B. Krogh. VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance. *ACM Transaction on Sensor Networks*, To appear.

[12] T. He, S. Krishnamurthy, J. A. Stankovic, and T. Abdelzaher. An Energy-Efficient Surveillance System Using Wireless Sensor Networks. In *MobiSys'04*, June 2004.

[13] T. He, P. Vicaire, T. Yan, Q. Cao, G. Zhou, L. Gu, L. Luo, R. Stoleru, J. A. Stankovic, , and T. Abdelzaher. Achieving Long-Term Surveillance in VigilNet. In *IEEE Infocom*, 2006.

[14] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System Architecture Directions for Networked Sensors. In *Proc. of Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 93–104, 2000.

[15] Q. Huang, C. Lu, and G.-C. Roman. Spatiotemporal Multicast in Sensor Networks. In *SenSys 2003*, November 2003.

[16] IEEE. IEEE Wireless Medium Access Control(MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs).

[17] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In *Proc. of ASPLOS-X*, October 2002.

[18] H. Li, P. Shenoy, and K. Ramamritham. Scheduling Messages with Deadlines in Multi-hop Real-time Sensor Networks. In *RTAS'05*, 2005.

[19] L.Kleinrock and J.Slivester. Optimum transmission radii for packet radio networks or why six is a magic number. In *national Telecomm conference*, 1978.

[20] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He. Rap: A real-time communication architecture for large-scale wireless sensor networks. In *IEEE RTAS*, 2002.

[21] L. Luo, T. He, T. Abdelzaher, and J. Stankovic. Design and comparison of lightweight group management strategies in envirosuite. In *DCOSS '05: International Conference on Distributed Computing in Sensor Networks*, June 2005.

[22] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The Flooding Time Synchronization Protocol. In *SenSys'04*, pages 39–49, Nov. 2004.

[23] S. Mohan, F. Mueller, D. Whalley, and C. Healy. Timing Analysis for Sensor Network Nodes of the Atmega Processor Family. In *IEEE RTSS*, 2004.

[24] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations:Concepts and Applications of Voronoi Diagrams*. Wiley, 2000.

[25] J. Polastre and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.

[26] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton. Sensor Network-Based Countersniper System. In *SenSys'04*, November 2004.

[27] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava. Mobile Element Scheduling for Efficient Data Collection in Wireless Sensor Networks with Dynamic Deadlines. In *IEEE RTSS*, 2004.

[28] R. Stoleru, T. He, and J. A. Stankovic. Walking GPS: A Practical Solution for Localization in Manually Deployed Wireless Sensor Networks. In *EmNetS-I*, October 2004.

[29] R. Szewczyk, A. Mainwaring, J. Anderson, and D. Culler. An Analysis of a Large Scale Habit Monitoring Application. In *SenSys'04*, 2004.

[30] Q. Wang, R. Zheng, A. Tirumal, X. Liu, and L. Sha. Lightning: A Fast and Lightweight Acoustic Localization Protocol Using Low-End Wireless Micro-Sensors. In *IEEE RTSS*, 2004.

[31] A. Woo and D. Culler. A Transmission Control Scheme for Media Access in Sensor Networks. In *Proc. of Mobile Computing and Networking (Mobicom)*, 2001.

[32] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A Wireless Sensor Network for Structural Monitoring. In *SenSys 2004*, 2004.

Towards Optimal Sleep Scheduling in Sensor Networks for Rare-Event Detection

Qing Cao, Tarek Abdelzaher, Tian He, John Stankovic
Department of Computer Science, University of Virginia, Charlottesville, VA 22904
email:{qingcao, zaher, tianhe, stankovic}@cs.virginia.edu

Abstract— **Lifetime maximization is one key element in the design of sensor-network-based surveillance applications. We propose a protocol for node sleep scheduling that guarantees a bounded-delay sensing coverage while maximizing network lifetime. Our sleep scheduling ensures that coverage rotates such that each point in the environment is sensed within some finite interval of time, called the detection delay. The framework is optimized for rare event detection and allows favorable compromises to be achieved between event detection delay and lifetime without sacrificing (eventual) coverage for each point. We compare different sleep scheduling policies in terms of average detection delay, and show that ours is closest to the detection delay lower bound for stationary event surveillance. We also explain the inherent relationship between detection delay, which applies to persistent events, and detection probability, which applies to temporary events. Finally, a connectivity maintenance protocol is proposed to minimize the delay of multi-hop delivery to a base-station. The resulting sleep schedule achieves the lowest overall target surveillance delay given constraints on energy consumption.**

I. Introduction

Sensor networks promise surveillance of large areas with possibly unprecedented accuracy. Currently, energy supply is one fundamental bottleneck. It is very expensive to replace sensor node batteries once they are deployed, both because of the large number of sensing nodes and because of the typically hazardous or unfriendly environment in which these nodes are deployed. Hence, prolonging battery life is a prime consideration in network design. Current literature advocates employing redundancy to allow some nodes to go to sleep without jeopardizing sensory coverage. These approaches imply that a minimum number of nodes must remain awake for the right degree of coverage to remain satisfied. A trade-off exists between energy savings and coverage. For example, in [1], [2] partial coverage schemes are investigated to increase energy saving gains. In these efforts, both random and synchronized sleep schedules are proposed and studied for certain scenarios. The former refers to the case where each node independently chooses random sleep and wakeup times. The latter refers to the case where all nodes go to sleep and wake up together in a synchronized fashion.

In contrast, we develop a near-optimal deterministically rotating sensory coverage. In this scheme, the area is only partially covered at any point in time. However, any point is eventually sensed within a finite delay bound. The energy/coverage trade-off is thus more meaningfully expressed as one between energy savings and the average *detection delay*, defined as the average time elapsed between event occurrence at a point and its detection by a nearby sensor. It is desired to minimize average detection delay subject to a constraint on energy consumption (expressed as a duty-cycle constraint). The goal of this paper is to develop a localized distributed protocol for (near-optimally) solving the aforementioned constrained optimization problem while ensuring upper bounds on the worst-case detection delay.

The paper also addresses sleep scheduling schemes for minimizing packet delivery latency to a common base-station. Observe that at very low duty cycles, it is likely that sensor nodes that are awake at any given time do not form a connected graph unless their wakeup times are appropriately synchronized. Such synchronization, however, may deviate from the optimal sleep schedule from the perspective of minimizing average detection delay. We develop a heuristic that provides partial synchronization to reduce delivery latency without significantly impacting the average detection delay.

The combination of detection delay and packet delivery latency is the perceived surveillance delay, which refers to the time elapsed from the occurrence of an event in the system to the time the event is reported to a base-station. Hence, the overall contribution of this paper is to develop a protocol for minimizing the surveillance delay subject to energy (namely, duty cycle) constraints.

Our protocol is optimized for detection of rare (but urgent) events. In such applications, network longevity is especially important, since mission lifetime must be appropriately large. Nodes operate at very low duty cycles and do not communicate unless an event is detected. Therefore, we consider sensing power as the predominant energy drain over the system lifetime. Once detection occurs, a prompt reaction may be needed (e.g., activating a camera or reporting an emergency). Consider, for example, the detection of forest fires. There are two natural concerns with this application: first, how long will the network last once deployed? Second, how responsive will it be in reacting to fire events? Our design translates these two questions into two related design parameters; namely, the energy consumption rate (i.e., the duty cycle which determines lifetime) and the surveillance delay. Our protocol offers a design space in which the designer can trade-off these parameters in a near-optimal fashion.

The rest of the paper is organized as follows. Section II presents the general framework and assumptions underlying our approach. A localized distributed optimization algorithm is presented in Section III to produce a sleep schedule that approaches the optimal on detection delay. This algorithm is subsequently enhanced to reduce delivery latency as well. Simulation results are presented in Section IV. We survey related work in Section V, and conclude this paper in Section VI with our summary and directions for future work.

II. General Framework

We consider an area covered by sensing nodes. Let some event (e.g., a fire) occur at one point in the area. The *maximal* detection delay for an event occurring at this point is defined as the longest time that may elapse before the event is detected by a nearby node. The *average* detection delay for this point is defined as the average time elapsed until the event is detected. The maximal detection delay for the entire area is the largest value of all maximal detection delays at points in the area. Similarly, the average detection delay for the area, denoted γ, is the average value for all detection delays of all points. Trivially, when the area is sensing covered, both the maximal detection delay and the average detection delay for the area are 0, since all events are detected immediately.

Sensors in the area are duty-cycled. Most sensors have a finite "warm-up" time T_w upon startup before reliable readings can be reported. Following the warm-up time, a sensor takes a sample of the environment, which itself takes time T_s (possibly including repeated sensor readings). This may be followed by other necessary processing (such as data logging) which takes time T_p. Hence, from the instant a node is powered on, a minimum time interval, $T_{on} = T_w + T_s + T_p$, must elapse before the node can go to sleep again. Given a duty cycle constraint β which defines the maximum percentage of time a node can be awake, the node must sleep for at least a duration T_d, where $T_{on}/(T_{on} + T_d) = \beta$. Hence, any event is detected in at most $T_d + T_{on}$ time units. It is desired to minimize the *average* event detection time. We are especially interested in very low duty-cycle operation where $T_{on} << T_d$.

We propose a two-level sleep scheduling framework. The first level selects a minimal subset of all deployed nodes, called the *primary* subset, such that sensing coverage is maintained using the fewest primary nodes. We assume that there are enough nodes in the network for sensory coverage to be achieved. The remaining nodes are turned off. This process is repeated periodically at a fairly large period (e.g., of the order of tens of hours) to change the set of primary nodes so that their energy is not depleted. Algorithms for such rotation have been proposed in prior literature and are not considered in this paper. The second level focuses on the current primary nodes. It contributes further energy savings by duty-cycling these nodes at a higher frequency (e.g., seconds or minutes). That is to say, each node in the primary subset sleeps for T_d then wakes up for T_{on}, where $T_{on}/(T_{on} + T_d) = \beta$, the desired duty cycle. Our purpose, in this paper, is to coordinate the duty cycles of primary nodes such that the average detection delay in the area is minimized.

One interesting remark is that although the maximum energy savings by first level scheduling are bounded by the need to maintain sensory coverage, the second level savings can be made arbitrarily large by decreasing the duty cycle of primary nodes. In principle, there is no lower bound on energy consumption after the second level scheduling. The only consideration is that lowering the duty cycle increases average detection delay.

If the average number of primary nodes within a sensory radius is α, any point in the environment is sensed by α nodes on average. Since each node sleeps for T_d and wakes up for T_{on}, at low duty cycles (i.e., when $T_{on} << T_d$), a point is sensed on average no more than once every T_d/α time units. An event arriving randomly between sense instants will thus suffer an average detection delay no lower than $T_d/2\alpha$. This value establishes a lower bound on detection delay given the sensor wakeup period, T_{on}, and the chosen duty cycle, $\beta = T_{on}/(T_{on} + T_d)$, which uniquely determine the minimum T_d, and hence the minimum $T_d/2\alpha$.

On the other extreme, if all primary nodes sleep and wake up in unison, each point is sensed only once every T_d, and the average detection delay for a randomly arriving event is $T_d/2$. Our purpose is to design a sleep scheduling protocol that approaches the lower bound, $T_d/2\alpha$, on the average detection delay.

It can be shown that minimizing detection delay leads to minimizing the variance in detection delay as well. Intuitively, this is because the sum of the squares (or higher powers) of numbers that add up to a constant is minimized when these numbers are equal. Hence, equally spacing sensor wakeup times within an interval T_d leads to minimizing both the mean and variance of detection delay. The complete proof is omitted for space limitations.

Finally, observe a relationship between detection delay and detection probability. An event with a short lifespan can be detected as long as its lifespan intersects any of the waking periods of neighboring sensor nodes. It is easy to show that the probability of such intersection is maximized when the wakeup periods are equally spaced. Thus, the sleep scheduling that optimizes the detection delay also maximizes the detection probability of short-lived events. Next we present a protocol that produces a near-optimal sleep schedule.

III. Sleep Schedule Optimization

In this section, we describe a sleep scheduling protocol that outperforms both random and synchronized scheduling in terms of average detection delay. The protocol is distributed, and has the favorable feature that it guarantees *local optimality* in that every node ends up with a wakeup point that cannot be further improved in terms of the average detection delay within its sensing range. We also present a protocol for optimizing end-to-end delivery latency. The combination of these two protocols is explored to reduce overall surveillance delay.

A. Detection Delay Optimization

Our overall algorithm for minimizing detection delay is a three stage transition process, shown in Figure 1.

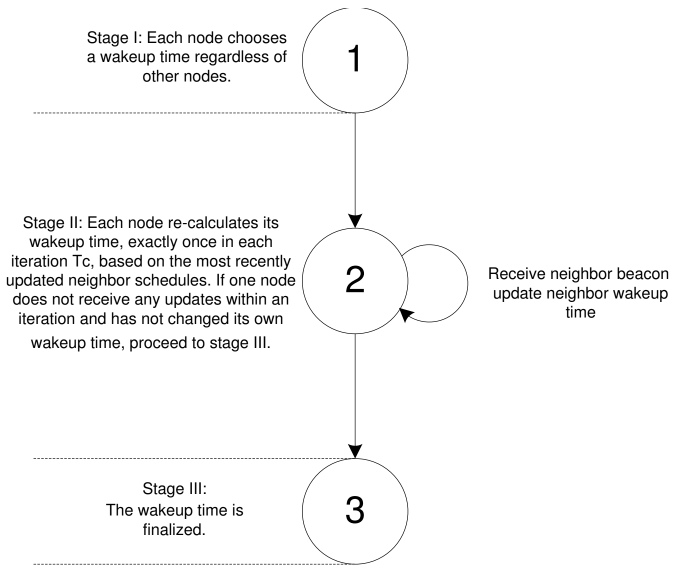

Stage I: Each node chooses a wakeup time regardless of other nodes.
1
Stage II: Each node re-calculates its wakeup time, exactly once in each iteration Tc, based on the most recently updated neighbor schedules. If one node does not receive any updates within an iteration and has not changed its own wakeup time, proceed to stage III.
2
Receive neighbor beacon update neighbor wakeup time
Stage III: The wakeup time is finalized.
3

Fig. 1. State Transition of Optimization Algorithm

We assume that neighboring nodes have approximately synchronized clocks. Protocols for clock synchronization in sensor networks can be found in [3]. Each node i starts at Stage 1, where it randomly picks an initial wakeup time, $t_i[0]$ for itself on a common timeline in the cyclic interval $[0, T_d + T_{on})$. For the purposes of this analysis, the wakeup time denotes the instant at which the node's wakeup interval T_{on} starts. The initial selection of the wakeup times of different nodes is completely uncoordinated. Each node communicates its randomly chosen wakeup time to its neighbors, sets up an iteration timer to fire at a period T_c, and enters Stage 2. Observe that in this stage all primary nodes are still awake (i.e., have not yet started their duty-cycling). The period T_c is called the *schedule iteration period*, which is different from the period $T_{on} + T_d$ of the would-be duty cycles.

In Stage 2, each node undergoes multiple schedule iterations. Within a single iteration, a node makes at most one adjustment to its wakeup time to reduce the average detection delay. Ultimately, a local minimum is reached where no more reductions can be obtained. More specifically, when the iteration timer of node i fires, denoting the beginning of a new schedule iteration, k, the node considers adjusting its wakeup time from $t_i[k-1]$ (the value chosen in the previous iteration) to a new value, $t_i[k]$. This new value should minimize the average detection delay in the area within node i's sensing range, denoted $\gamma_i[k]$, given the updated wakeup times received from i's neighbors in the last iteration. Note that by neighbors, we are only referring to those nodes that have overlapping sensing ranges with the current node, since for the the current node, only the waking times of these *sensing neighbors* are relevant. We will use *communication neighbor* to specifically refer to the nodes within communication range of the current node, and without further explanation, use neighbor to denote sensing neighbors.

In our discussion, we assume that each node knows its sensing range. This assumption is supported by our observations with current sensor nodes. For example, in XSM2 [4] motes developed by OSU and CrossBow, a roughly circular sensing range can be measured for the set of PIR sensors before deployment. Each node can use this knowledge to determine whether or not a given point is located within its sensing range.

If the difference between the old and new detection delays ($\gamma_i[k]-\gamma_i[k-1]$) is larger than a preset threshold, h, the new wakeup time, $t_i[k]$, is adopted and the node reports this new wakeup time to all its neighbors. Otherwise, the old wakeup time, $t_i[k-1]$, remains in place and no updates are sent. The node then waits for the next invocation of the iteration timer T_c to start a new iteration. If the node does not receive any updates within an iteration and has not changed its own wakeup time, it enters Stage 3 in which it starts duty-cycling, phased in accordance with its computed wakeup time. Once all nodes reach Stage 3, we consider the detection delay optimization complete. Note that, since clocks drift over time, the duty cycle period T_d+T_{on} must be large enough to accommodate a fair amount of phase drift without the need for clock re-synchronization. This constraint is met naturally, since we are interested in very low duty cycles ($T_d >> T_{on}$) in which T_d must be reasonably large (of the order of seconds or minutes).

The critical part of the above optimization process lies in the localized computation of the optimal wakeup time of an individual node at Stage 2 as a function of those of its neighbors. The problem is formulated as follows. Given a node, i, that is informed of all the current wakeup times of its neighbors, what wakeup time, $t_i[k]$, should it choose to minimize the average detection delay, $\gamma_i[k]$, in the area within its sensing range?

To answer this question, in the following, we first derive an expression for the average detection delay within the sensory range of node i as a function $f_i(t)$ of the node's unknown wakeup time t (and the known wakeup times of its neighbors). We then find the wakeup time t that minimizes this expression (i.e., for which $f_i(t)$ is minimum). Finally, we present an implementation that computes $f_i(t)$ and the corresponding wakeup time efficiently at run-time.

1) Derivation of an Optimal Wakeup Time: To derive $f_i(t)$, consider an arbitrary point A in node i's sensing range. Let point A be located within the intersection of the sensing ranges of n nodes (including node i). The average detection delay at point A is the average time elapsed from the occurrence of an event at A to the next time some neighboring node wakes up and samples the environment. It depends on the relative spacing of the respective sampling times of the n neighbors. Since each node will sample the environment once every duty cycle period, there will be a total of exactly n samples within each interval T_d+T_{on}. Let the samples of different nodes be separated by time intervals x_1, ..., x_n, where $x_j \geq 0$ for $1 \leq j \leq n$.

Figure 2 shows an example of a duty cycle of length 1, with nodes $N1$, $N2$ and O, sampling the environment at times 0.25, 0.6 and t respectively. The intervals x_1, ..., x_3 between successive samples are indicated. The circle in this figure depicts a repeated duty-cycle. The arrows indicate the direction of the passage of time. Observe that while a node might be awake for a finite period of time, T_{on} (which includes sensor warm-up and data post-processing times), its sampling time, for purposes of this analysis, refers to the time instant at which the node completes its environmental reading. In our model, this instant occurs at a fixed offset from the node's wakeup time (namely, at offset $T_w + T_s$ defined in Section I). However, it is straightforward to extend our analysis to the case where nodes continue sampling the environment for some contiguous finite duration.

Given inter-sample separations x_1, ..., x_n, the average detection delay D at point A is given by the sum of the average detection delays for event arrivals in an interval x_j (given by, $x_j/2$), each multiplied by the probability of arriving within that respective interval, which is $x_j/(T_d+T_{on})$. Hence, D equals the sum of $(x_j/2)x_j/(T_d+T_{on})$, $1 \leq j \leq n$, which gives:

$$D = \frac{x_1^2 + ... + x_n^2}{2(T_d+T_{on})} \tag{1}$$

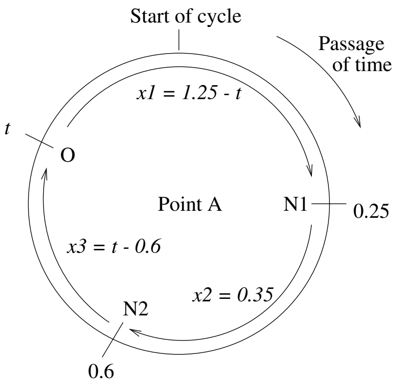


Fig. 2. A Cyclic Sleep Schedule

Since node i knows the wakeup times of all its neighbors, substituting in Equation (1) we get a quadratic expression that is a function only of node i's own wakeup time. For example, substituting with intervals x_1, x_2 and x_3, shown in Figure 2, into Equation (1) we get a quadratic function of t that represents the average detection delay at point A. Observe that this quadratic function depends on the ordering of the unknown wakeup time t with respect to the wakeup times of the neighboring nodes. For example, Figure 2 shows t to be in the range $0.6 \leq t < 1$. Substituting in Equation (1) gives an expression that is valid only for the corresponding range. Similar expressions can be derived for the other ranges. Putting the expressions for different ranges of t together, we obtain a continuous piecewise quadratic equation that yields the average detection delay at point A as a function of the unknown wakeup time t anywhere in the duty cycle. We call it the *optimality curve* for point A. The optimality curve for point A shown in Figure 2 is given in Figure 3.

To minimize the average detection delay across the entire sensing range of some node i, the quadratic optimality curves of all points in i's sensing range are added. The resulting piecewise quadratic function is the sought function $f_i(t)$ that is then solved for a global

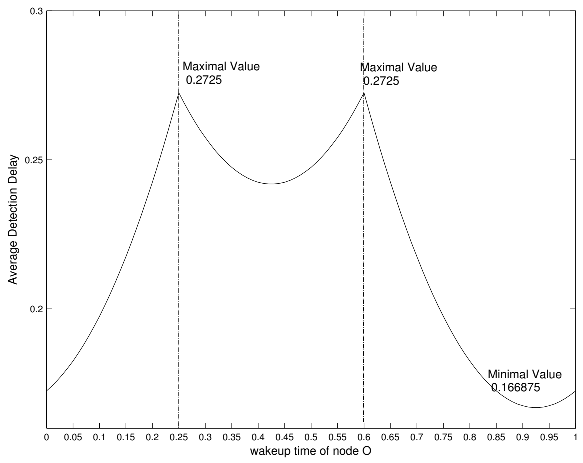


Fig. 3. Optimality Curve for node O at point A

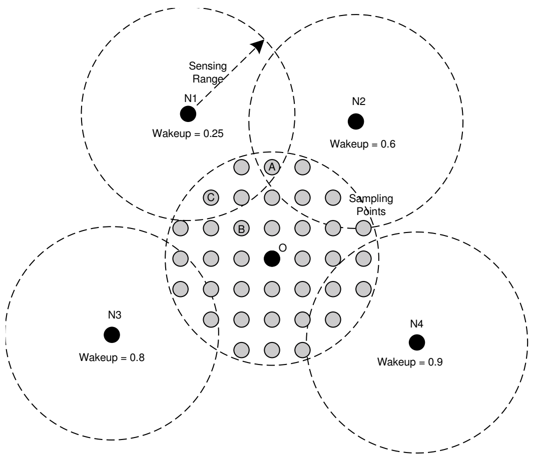


Fig. 4. An Optimization Example

minimum. This conceptual procedure lends itself to an efficient implementation in view of the following two observations.

First, note that points covered only by node i (and no other nodes) will always have the same average detection delay regardless of when i chooses to wake up. At low duty cycles, this delay is well approximated by $T_d/2$. Such points need not be considered in the aforementioned summation as they do not change the optimization result. Second, note that all points that lie at the intersection of sensing ranges of the same nodes lead to the same quadratic optimality curves. Hence, it is enough to compute such curves only once. For example, node O in Figure 4 needs to consider only five distinct optimality curves corresponding to the five intersection regions between its sensory range and that of other nodes. The equation for each curve is weighted by the area of the corresponding intersection and the results added up to obtain $f_i(t)$.

It can be shown that the resulting overall function is piecewise quadratic with a number of segments that depends only on the total number of neighbors, M, of node i. Its global minimum can only occur at one of the local minima of the individual segments or at the points at which these segments are joined. Inspecting these points is an $O(M)$ operation. The algorithm can therefore efficiently determine the position of the global minimum and hence the new wakeup time. Next, we present a detailed example of computing an optimality curve, and our actual implementation of the entire algorithm.

2) Example: Computing the Optimality Curve: Consider again node O in Figure 4. Node O has four neighbors denoted $N1$ to $N4$. In this example, there are five distinct sensor range intersection regions within O's sensing range that need to be considered. Figure 4 depicts these regions and the wakeup times of all neighboring nodes. Point A exemplifies one region that lies at the intersection of the sensing ranges of nodes $N1$, $N2$ and O. In the duty cycle $[0, 1)$, there are three cases to consider for the wakeup time t of node O, namely. $0 \leq t < 0.25$, $0.25 \leq t < 0.6$ and $0.6 \leq t < 1$, where 0.25 and 0.6 are the known wakeup times of neighbors $N1$ and $N2$. Figure 2 depicts the case where $0.6 \leq t < 1$. As seen in Figure 2, the intervals between successive wakeup times are $1.25-t$, 0.35 and $t-0.6$ respectively. Substituting in Equation (1), the average detection delay in this case is $\frac{(1.25-t)^2+0.35^2+(t-0.6)^2}{2}$, which evaluates to $t^2-1.85t+1.0225$. Similarly, we can determine that for $0 \leq t < 0.25$ the average detection delay is given by $t^2 - 1.85t + 0.4225$, and that for $0.25 \leq t < 0.6$ it is given by $t^2 + 0.15t + 0.1725$. Together, the above three segments constitute the optimality curve for point A (shown in Figure 3).

3) Efficient Implementation: In our implementation, a node builds a polynomial function table for each optimality curve, in which each segment of the function is stored as a three-element tuple (a, b, c), denoting the function as $f(t) = at^2+bt+c$, It also stores the starting and ending point of each segment. For the optimality curve computed above, the polynomial function table is shown in Table I.

A node also sorts the wakeup times of its M neighbors to determine the $M + 1$ intervals between these wakeup times within a duty cycle. It then initializes a new polynomial function table that will hold the final function $f_i(t)$ for the area covered by node i. We call it the *result* table.

To simplify computation, a node then considers a virtual grid within its sensing range. Points on this grid are considered sequentially, each with the same weight. For each point, the algorithm classifies this point based on which nodes are less than one sensing range away from it. Then, the coefficients of all segments of its optimality curve are fetched from the corresponding polynomial function table and added to the coefficients of the corresponding segments in the result table, which generates an intermediate segmented quadratic polynomial function. When all points have been considered, the result function is complete. For example, the result table for point O in Figure 4, is shown in Table II. The corresponding aggregated function is plotted in Figure 5. The optimal wakeup time can be decided by finding the lowest value on the aggregated function (which turns out to be 4.4166 at $t = 0.385$). This can be done by inspecting function values at segment boundaries and local minima (a local minimum of a function $at^2 + bt + c$ occurs at $t = -b/2a$). The time at which the lowest value occurs is the sought wakeup time $t_i[k]$ of node i at iteration k. Once the wakeup time is determined, the node sends out its decision. We now briefly explain the content of the decision packet. Each node keeps an incrementing counter as the current version of its wakeup time. It also keeps the latest versions of its neighbors. Once it makes a new adjustment, it sends out its ID, its new wakeup time, the version counter, as well as the version counters of its neighbors. The last piece of information is necessary to avoid non-serializable modifications of wakeup times of neighboring nodes. Such modifications may lead to endless loops in the adjustment. Therefore, once two nodes find that they have each adjusted their sleeping times independently, the node with lower ID

TABLE I

POLYNOMIAL FUNCTION TABLE FOR POINT A

	Range	Tuple	Function
1	$[0, 0.25]$	$(1, 0.15, 0.1725)$	$t^2 + 0.15t + 0.1725$
2	$[0.25, 0.6]$	$(1.0, -0.85, 0.4225)$	$t^2 - 0.85t + 0.4225$
3	$[0.6, 1]$	$(1.0, -1.85, 1.0225)$	$t^2 - 1.85t + 1.0225$

TABLE II

RESULT TABLE

	Range	Tuple	Function
1	$[0, 0.25]$	$(15, -4.55, 4.89)$	$15t^2 - 4.55t + 4.89$
2	$[0.25, 0.6]$	$(15, -11.55, 6.64)$	$15t^2 - 11.55t + 6.64$
3	$[0.6, 0.8]$	$(15, -18.55, 10.84)$	$15t^2 - 18.55t + 10.84$
4	$[0.8, 0.9]$	$(15, -26.55, 17.24)$	$15t^2 - 26.55t + 17.24$
5	$[0.9, 1]$	$(15, -34.55, 24.44)$	$15t^2 - 34.55t + 24.44$

revokes its prior decision and rolls back to its last version. The same rule applies to more than two nodes as well. One node also needs to roll back if its packet is lost in transmission. Therefore, we use an acknowledgement based MAC layer. If one node cannot receive the acknowledgements from all neighbors, it should either revoke its prior decision, if it receives a parallel adjustment from one of its neighbors during the time, or resend its decision to all its neighbors. Observe the fact that communication range in sensor networks is typically much larger than sensing range. Therefore, we expect that the sensing neighbors are typically located sufficiently nearby, and connected via relatively reliable links to the current node.

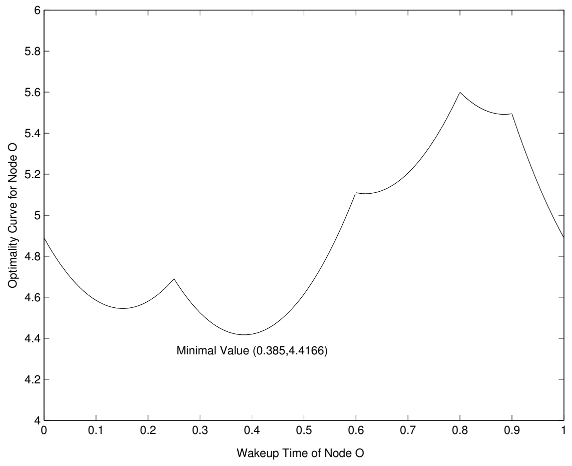


Fig. 5. Aggregated Optimality Curve for Node O

4) Algorithm Analysis: **Cost Analysis** We now consider the computational cost and requirements on storage of the algorithm. We consider storage requirements first. For each sampling point covered by n neighbors, the maximal number of segments is $n + 1$ (there is $n+1$ because we treat the first region and the last region in Figure 3 as different functions). Therefore, for a node with M neighbors, the number of segments for the aggregated function is at most $M+1$, due to the fact that points in the same partition share the same segments. Once we have aggregated a segment function, the storage it occupies can be freed, therefore, at most $M + 1$ entries are needed in the global result table, which is not memory intensive. Second, as far as computation cost goes, the overall cost is proportional to the product of grid resolution and the number of neighbors, M. We can easily control the former factor to reduce overall cost to an acceptable value. Our experiments on MICA2 nodes show that comparable computation load can be well afforded.

On the Convergence of the Algorithm We now show that the overall optimization process terminates in a finite number of steps. First, note that each adjustment of the wakeup time by one node in Stage 2 decreases the average detection delay within the sensing range of this node, but does not affect the average detection delay outside its sensing range. Hence, the average detection delay for the area also decreases with individual node adjustment. Also note that, in our design, we have avoided non-serializable adjustments of neighboring nodes. Therefore, the whole process exhibits a contractive property. Since the initial average detection delay for the whole area must be finite, and since the algorithm makes adjustments only if they decrease the average detection delay (in some node's sensing range) by some minimum finite amount, the algorithm must terminate after a finite number of adjustments. Note that during the process, it is possible that the adjustment of one node's schedule may propagate to its neighbors, however, such propagation will only decrease the overall detection delay, which obviously will terminate after a finite number of steps.

To estimate the convergence time of the algorithm in area, S, suppose each node has a sensing range, r, and communication range, $R > r$. For each adjustment of one node in Stage 2, the average detection delay decreases by at least h in the sensing area of this node. Thus, each adjustment decreases the average detection delay for the whole area by at least $\frac{\pi r^2}{S} \times h$. Remember that the average detection delay is upper-bounded by approximately $T_d/2$ and lower-bounded by approximately $T_d/2\alpha$. The maximum number of adjustments is therefore bounded by the difference between the two bounds divided by the adjustment per step, which yields $\frac{T_d}{2}(1 - 1/\alpha)\frac{S}{\pi r^2 h}$. Now assume that nodes outside each other's communication range (and hence outside each other's sensing range) can perform adjustments in parallel. There are roughly $\frac{S}{\pi R^2}$ such nodes. Hence, the number of rounds of adjustment is roughly $\frac{T_d}{2}(1 - 1/\alpha)\frac{R^2}{r^2 h}$, which takes $\frac{T_d}{2}(1 - 1/\alpha)\frac{R^2}{r^2 h}T_c$ time units to complete. This estimate, of course, is a quite relaxed bound: each adjustment may decrease the average detection delay within one node's sensing area well beyond the lower bound h. In practice, our simulations show that the system always converges within twenty rounds.

B. End-to-End Delay Optimization

Next, we propose an optimization for end-to-end delivery delay. Observe that at low duty cycles, the fraction of nodes that are awake at any given time do not necessarily form a connected network. Delivering sensed events to the base-station requires synchronization of waking times between communication neighbors along the path. We consider networks where the communication range is relatively large compared to the sensing range. Hence, after first-level scheduling (which determines the minimum number of nodes needed for full sensory coverage), the resulting primary nodes have many neighbors within their communication range. The problem, of course, is that after the ensuing second-level scheduling, not all neighbors will be awake at the same time. From the perspective of minimizing event delivery time to a base-station, it is desired to synchronize duty cycles of nodes into a streamlined sequence to pipe the data efficiently. This idea is not unlike the common practice of synchronizing traffic lights to turn green (wake up) just in time for the arrival of vehicles (packets) from previous intersections (hops). Observe that it is enough for each node to synchronize its duty cycle with only one neighbor within its communication range that is closer to the basestation. Consequently, synchronized routes are formed to expedite data delivery from any node.

An example of this type of coordination is shown in Figure 6. As shown in this example, packets delivered from node 0 to 9 have minimum delay. We call this technique *streamlined wakeup*. In the following, we propose an optimization of delivery delay based on the streamlined wakeup technique. We focus on the most common case where each sensor reports to only one base-station (although different parts of the network might report to different local base-stations). Our algorithm works as follows:

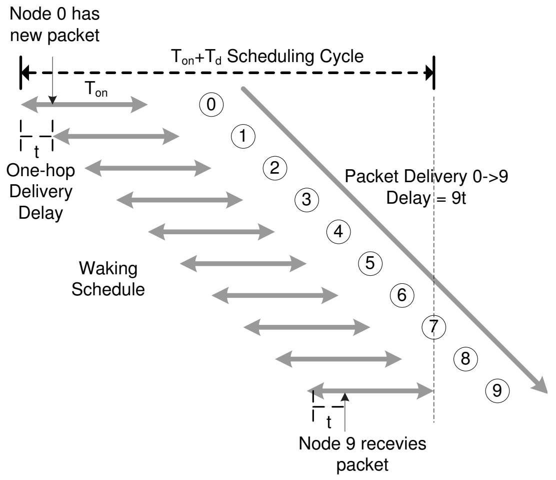


Fig. 6. Scheduling Example for Node Pipe

1) After first-level scheduling is complete, the base-station floods the network with a message containing a hop count that is incremented at each hop (interest propagation). Each node keeps track of the lowest hop-count received and maintains that number as its hop count from the base-station. Since the base-station is assumed to be always up, nodes one hop from the base-station (i.e., its direct neighbors) set a *pipe* flag indicating that they have a valid streamlined path to the destination. Any node that sets this flag communicates this fact to its neighbors.
2) Nodes run the detection delay minimization algorithm described earlier to compute their wakeup times. In Stage 3 of this algorithm, instead of actually implementing the duty cycle, they execute the step below.
3) Any node whose neighbors with shorter hop counts to the base-station have set their *pipe* flag, finds the one such neighbor with the closest wakeup time to its own. The node then overlaps its wakeup interval with that neighbor's, effectively appending itself to an established streamlined data pipe that is closest to its ideal wakeup time. Observe that the number of such pipes that may be established in the network is of the order of the number of the immediate communication neighbors of the base-station. The larger is this number, the lower (on average) is the adjustment needed to a node's wakeup time to join a pipe. For example, a base-station with a sensitive enough antenna to hear all sensors will enable each node to be its own data pipe with no additional synchronization or adjustment needed. Having joined a pipe, a node sets its *pipe* flag and communicates this fact along with its new wakeup time.
4) Any node that has set its *pipe* flag and communicated this information now enters the duty-cycling phase in according with its updated wakeup schedule.

The above algorithm ensures that a synchronization wave propagates outwards from the base-station. When the wave reaches the outer perimeter of the network, all nodes will have routes to the base-station with appropriately overlapped wakeup times. All nodes will have entered the duty-cycle mode. The initialization is thus complete. If the communication range is large enough, it is easy to find neighbors with close wakeup times to your own. The algorithm therefore does not have much impact on the optimality of average detection delay in networks with a large communication range, as will be demonstrated in the next section.

IV. Evaluation

In this section, we verify the theoretical results and optimizations given in the previous section via extensive simulations.

A. Simulation Setup

We simulate a two-level scheduling framework. By default, the area is $100m \times 100m$. Each node has a sensing range of $10m$. Initially more than enough nodes are deployed to guarantee sensing coverage. The first level scheduling is then applied where as many nodes as possible are put to sleep without compromising overall sensing coverage. The remaining nodes form the basis for evaluating the protocols designed in this paper, where different approaches are compared.

In practice, we deploy 300 nodes, followed by a first-level scheduling protocol to turn off redundant nodes. An average of 76 nodes remain awake, so we generate ten scenarios with 76 nodes remaining as the basis for second level scheduling evaluation. Each of these deployment scenarios guarantees full sensing coverage and no node is redundant. We simulated a simple MAC layer with packet acknowledgement. In the simulations, packet loss and retransmissions appear to have very limited effect on the overall performance, since we can adjust the pace of schedule readjustment sufficiently to accommodate packet retransmissions.

B. Detection Delay Optimization

In this section, we focus on the optimization of average detection delay. For each scenario, we compare the optimized and random energy saving schedules to the theoretical lower bound and the upper bound (the case of a synchronized schedule). The results are shown in Figure 7. The horizontal axis varies the ratio of the sleep interval to the waking interval, T_d/T_{on}, on a logarithmic scale, over two orders of magnitude. The vertical axis shows the normalized average detection delay over T_{on}.

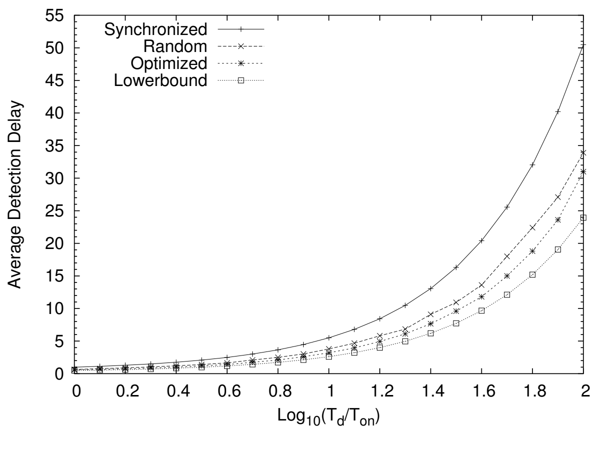


Fig. 7. Average Detection Delay

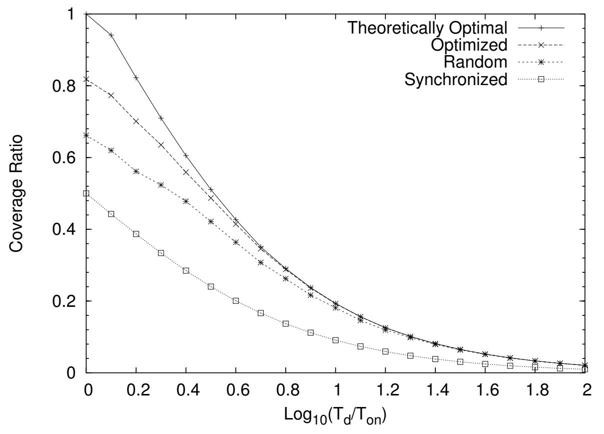


Fig. 8. Rotating Coverage Ratio

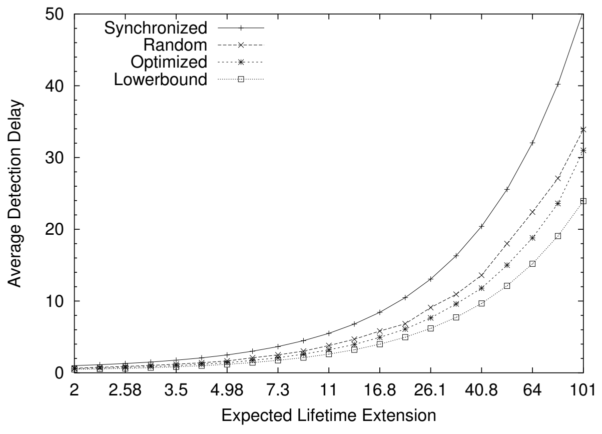


Fig. 10. The Lifetime vs. Delay Trade-off

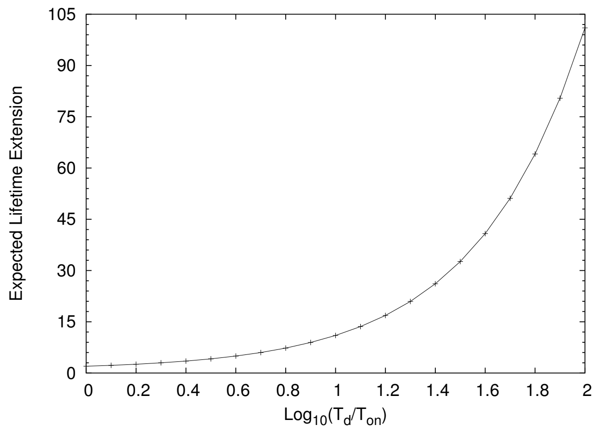


Fig. 9. Expected Lifetime Extension

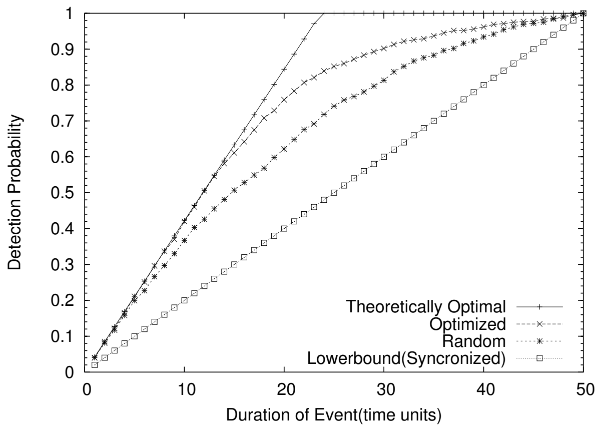


Fig. 11. Event Detection Probability

Notice that the theoretical lower bound to which we compare the results is over optimistic. No scheduling approach can achieve this bound because different nodes in an irregular network generally cannot achieve perfectly equal wakeup time spacing simultaneously. Thus, optimizing the average detection delay for one point usually leads to sub-optimal scheduling for neighboring points. While no algorithm can achieve the optimistic lower bound, we observe that ours demonstrates considerable performance enhancement compared with both random and synchronized sleep scheduling.

For example, from Figure 7, when $T_d/T_{on} = 10$ (or $log(T_d/T_{on}) = 1$), the theoretical average detection delay lower bound is 2.6, our algorithm achieves 3.2, random sleep scheduling achieves 3.8, while synchronized sleep scheduling is as high as 5.5. More generally, our protocol can reduce the gap between random scheduling and the optimal bound in terms of average detection delay by 30% to 50%, and has a absolute average detection delay reduction over random scheduling up to 15%.

We also evaluate the notion of coverage ratio defined as the percentage of covered area in time and space. For the purposes of this experiment, covered area refers to area in the range of at least one sensor that is awake at the time. Since each node is awake during T_{on}, the aggregation of such coverage intervals reflects a measure of vigilance of the network. The results are presented in Figure 8. As shown, as the duty cycle decreases (by increasing T_d/T_{on}), the coverage ratio of random scheduling and optimized scheduling converges quickly to the optimal. This is expected because both random scheduling and optimized scheduling are not likely to overlap the wakeup periods of neighboring nodes. Since the coverage ratio is only relevant to the aggregated waking period, these two sleep scheduling policies eventually lead to the same (optimal) ratio.

Another important factor is the expected extension in lifetime. Figure 9 plots the relationship between the ratio T_d/T_{on} and the expected lifetime extension of the sensor network in multiples of its original lifetime (the one when all primary nodes are always on).

Combining Figure 9 with Figure 7, we quantify the trade-off relationship between the expected lifetime extension and the corresponding increase in the average detection delay achieved by different sleep scheduling algorithms. This trade-off is expressed in Figure 10. As observed, our optimization algorithm clearly outperforms both synchronized and random scheduling in the sense of achieving a longer lifetime for the same average detection delay, or achieving a lower average detection delay for the same lifetime. This figure clearly demonstrates the advantage of our approach from an application's perspective.

At last, we present the performance of different sleep scheduling policies in detecting temporary events. If events persist for a short time duration, sleep scheduling has a profound impact on their probability of detection. Figure 11 plots the relationship between

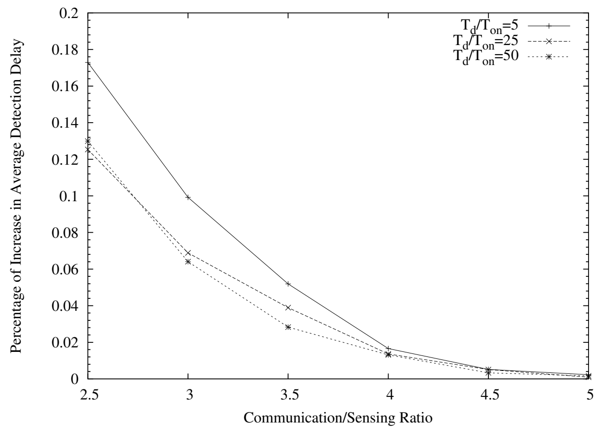


Fig. 12. Effect of Pipe Synchronization for Multihop Delivery

detection probability and event lifetime. The horizontal axis plots the event duration normalized to T_{on}, where T_{on} is assumed to be 1 time unit. In this experiment, $Td = 50T_{on}$. It is shown that our optimized sleep scheduling algorithm performs considerably better than both synchronized and random scheduling in terms of improving the probability of short event detection. This result is due to the more even spread of wakeup times under our approach.

C. End-to-end Surveillance Delivery Latency

Figure 12 characterizes the impact of optimizing packet delivery latency on average detection delay. The main factor that characterizes that impact is the ratio between the communication and sensing radius. Since nodes on each path to the base-station must be synchronized, their synchronization increases the average detection delay. However, as the ratio between communication range and sensing ranges increases, the number of primary nodes within one's communication range increases, which makes it easier to find a neighbor to synchronize with. The negative effect of such synchronization on average detection delay is thus reduced.

At last, we want to emphasize that while our algorithm is locally optimal, it has left a gap between itself and the theoretical global optimal. More global coordination of sleep schedules may improve performance further. We believe, however, that it would be difficult to beat this performance with other *localized* algorithms.

V. Related Work

Minimizing energy consumption has been a central topic in many papers in recent years. Effective techniques have been proposed to put nodes to sleep while maintaining full coverage at a specified degree of redundancy [5], [6], [7], [8]. These solutions can be conveniently integrated as first level scheduling algorithms in our framework.

Research on partial coverage based protocols has received less attention. Among the first publications are [2] and [1], which study the problem of tracking moving targets. Our work differs in that (i) we focus on stationary event detection, and (ii) we aim at finding a localized algorithm that approaches the minimum average delay bound.

In studying the impact of partial sensing coverage, we inevitably face the problem of connectivity. The work of [9] proposes remote radio triggered hardware, which extracts energy from specific radio signals without using an internal energy source, to provide wakeup signals. This service can be used in our framework to forcefully wake up additional nodes to form a connected network when an event of interest is detected. Since we focus on rare (but important) events, wasting energy when an event occurs is permissible. A similar hardware is reported in [10], where a low-power VLSI wake-up detector is designed in an acoustic surveillance sensor network.

Also relevant to our analysis for delivery latency is [11], which addresses this issue through an extension of first passage percolation theory for completely uncoordinated scheduling. Similarly, [12] addresses this issue through a Markov model based approach, where distribution of the data delivery delay is analytically determined. [13] uses a slotted approach for communication scheduling, where nodes determine their wakeup times based on their relative positions in the aggregation tree. Our work is different from these efforts primarily in our streamlined wakeup scheduling, as discussed in Section III-B, where the delivery latency is considerably lower.

VI. Conclusion and Future Work

In this paper, we have outlined, studied and evaluated the problem of minimizing surveillance delay subject to energy constraints. We consider this delay to be composed of detection delay and delivery delay, and propose optimizations for both. The final outcome is a flexible framework in which application designers can trade-off energy versus latency of event detection. We focus on detection of rare events, where the network is normally silent, except when events occur. This is in contrast to data collection networks that continuously stream periodic data to a collection center. The study reported in this paper is a first step towards more general models that optimize performance in the presence of communication as well. A general study of optimizing detection delay for moving targets is another worthy extension. We expect to address these issues in future publications.

Acknowledgements

The work reported in this paper was supported in part by the National Science Foundation under grants EHS-0208769 and ITR EIA-0205327.

References

[1] S. Ren, Q. Li, H.N.Wang, X. Chen, and X. Zhang, "Probabilistic coverage for object tracking in sensor networks," in *Mobicom 2004 Poster Session*, 2004.
[2] C. Gui and P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks," in *ACM Mobicom*, 2004.
[3] M.Maroti, B.Kusy, G.Simon, and A.Ledeczi, "The flooding time synchronization protocol," in *ACM Sensys*, 2004.
[4] "Xsm website http://cast.cse.ohio-state.edu/exscal/."
[5] X. W. et al., "Integrated coverage and connectivity configuration in wireless sensor networks," in *ACM SenSys*, 2003.
[6] T. Yan, T. He, and J. A. Stankovic, "Differentiated surveillance for sensor networks," in *ACM SenSys*, 2003.
[7] D. Tian and N.D.Georganas, "A node scheduling scheme for energy conservation in large wireless sensor networks," in *Wireless Communications and Mobile Computing Journal*, 2003.
[8] T.He, S.Krishnamurthy, J.A.Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, G. Zhou, J. Hui, and B. Krogh, "Vigilnet:an integrated sensor network system for energy-efficient surveillance," in *In submission to ACM Transaction on Sensor Networks*, 2004.
[9] L. Gu and J. Stankovic, "Radio-triggered wake-up capability for sensor networks," in *IEEE RTAS*, 2004.
[10] D. Goldberg, A.Andreou, P.Julian, P.Pouliquen, L.Riddle, and R.Rosasco, "A wake-up detector for an acoustic surveillance sensor network: Algorithm and vlsi implementation," in *IEEE IPSN*, 2004.
[11] O. Dousse, P. Mannersalo, and P. Thiran, "Latency of wireless sensor networks with uncoordinated power saving mechanisms," in *MobiHoc*, 2004.
[12] C.F.Chiasserini and M.Garetto, "Modeling the performance of wireless sensor networks," in *IEEE Infocom*, 2004.
[13] "Tinydb project http://telegraph.cs.berkeley.edu/tinydb/."

RID: Radio Interference Detection in Wireless Sensor Networks

Gang Zhou, Tian He, John A. Stankovic,Tarek Abdelzaher
Department of Computer Science
University of Virginia, Charlottesville 22903
{gzhou,tianhe,stankovic,zaher}@cs.virginia.edu

Abstract— **In wireless sensor networks, many protocols assume that if node A is able to interfere with node B's packet reception, node B is within node A's communication range. It is also assumed that if node B is within node A's communication range, node A is able to interfere with node B's packet reception from any transmitter. While these assumptions may be useful in protocol design, they are not valid, according to the real experiments we conducted in MICA2 platform. For a strong link that has a high packet delivery ratio, the interference range is observed smaller than the communication range, while for a weak link that has a low packet delivery ratio, the interference range is larger than the communication range. So using communication range information alone is not enough to design real collision-free media access control protocols. This paper presents a radio interference detection protocol (RID) and its variation (RID-B) to detect run-time radio interference relations among nodes. The interference detection results are used to design real collision-free TDMA protocols. With extensive simulations in GlomoSim, and with sensor network application scenarios, we observe that the TDMA which uses the interference detection results has 100% packet delivery ratio, while the traditional TDMA has packet loss up to 60%, in heavy load. In addition to the scheduling-based TDMA protocols, we also explore the application of interference detection on contention-based MAC protocols.**

I. INTRODUCTION

Wireless sensor networks (WSN) is an emerging technology that has a wide range of potential applications [1], including environment monitoring, smart houses, remote medical systems, sheep shepherding, and intrusion detection. Recent work [2][3][4][5][6] found that radio communication in wireless sensor networks (WSN) differs significantly from traditional Mobile Ad-Hoc Networks (MANET). For example, when node C's signal can interfere with node A's signal, preventing A's signal from being received at node B, it's usually assumed that node B must be within node C's communication range and there is communication connectivity from C to B. We name this assumption the interference-connectivity assumption, which is widely used to design collision-free Media Access Control (MAC) protocols [7][8].

However, from our experiments on the MICA2 platform, we find that this interference-connectivity assumption is not valid. In actuality, a node can interfere with another node even if it is beyond its communication range. In our experiments we show that when the receiver is within, but close to the edge of the transmitter's radio range, (referred to as a weak link), the transmitter's signal is easily interfered with at the receiver by another node which has no connectivity with the receiver. Such experiments are repeated several times and it is always found that the interference-connectivity assumption is violated.

The interference-connectivity assumption assumes that interference always comes from connectivity. Another assumption, the connectivity-interference assumption, assumes that connectivity always leads to interference. The connectivity-interference assumption was firstly addressed in [9] and is described as follows: when two transmitters, A and B, both have connectivity to a receiver C and transmit simultaneously, a collision occurs, the data is corrupted and neither packet is received correctly. Our experiments confirm the observation in [9] that the connectivity-interference assumption is not always maintained. In other words in spite of the logical interference, one packet may be received while the other is corrupted. We observe that this assumption is usually violated in the case of strong links, that is, when the transmitter is close to the receiver and has a very strong signal that dominates the interfering signal.

Without these assumptions between connectivity and interference, it's extremely challenging to design collision-free MAC protocols. In this paper, the idea of radio interference detection at run-time is put forth, for the first time, to design real collision free MAC protocols that don't depend on these non-realistic assumptions. Our solutions obtain the interference relations among nodes to assist in achieving real collision-free packet delivery. The design of the radio interference detection protocol, RID, is presented. A lightweight version, the RID-Basic (RID-B), is also presented. Extensive simulations using sensor network scenarios are conducted to compare the performance of TDMA and TDMA-RID-B (TDMA with RID-B support). The performance evaluation shows that traditional TDMA can have up to 60% packet loss in heavy-loaded networks, while TDMA-RID-B can maintain 100% packet delivery ratio.

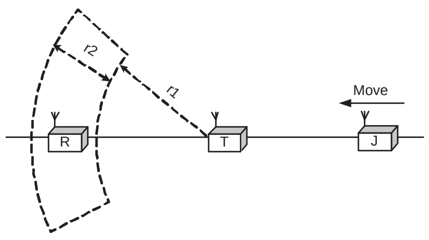
The rest of this paper is organized as follows: in Section II, experimental observations of radio interferences in the MACA2 platform are presented. Then in Section III, the radio interference detection protocol, RID, and its variation, RID-B, are explained. In Section IV, extensive performance evaluations of TDMA and TDMA-RID-B, in which the TDMA scheduling is based on RID-B's interference detection result, are analyzed. In addition, it is also explained how to use RID on contention-based MAC protocols. In Section VI, related work is analyzed, and finally in Section VII, conclusions are given and future work is pointed out.

II. EXPERIMENTS ON RADIO INTERFERENCE

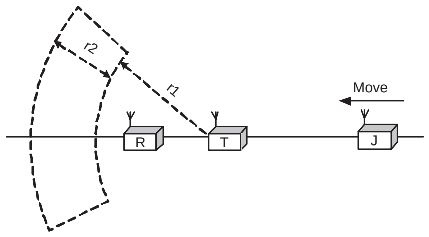
In this section, we present experimental results collected in an outdoor environment with MICA2 motes. From these experiments, it is confirmed that radio interferences are ubiquitous phenomena. It is also observed that for a weak link that has a low packet delivery ratio, the interference range is larger than the communication range, while for a strong link that has a high packet delivery ratio, the interference range is smaller than the communication range.

A. Experimental Setup

For each experiment, three MICA2 motes are used. One MICA2 mote is used as the transmitter, and another MICA2 mote is used as the receiver, and the third one is used as the jammer, whose transmission is synchronized with that of the transmitter to generate possible interference. The carrier sensing and backoff operations in the MAC layer are disabled to ensure packets are simultaneously sent out by the jammer and the transmitter. The radio interferences are reflected by the changing packet delivery ratios.



(a) A Weak Link From T to R



(b) A Strong Link From T to R

Fig. 1. Empirical Experiment Setting

All experiments are conducted in an open parking lot late at night, in order to separate possible influence from people and moving cars on the radio interference measurements. In the experiment, as Figure 1 illustrates, the transmitter T and receiver R are fixed in positions, and jammer J moves along the line determined by the positions of the transmitter and the receiver. The jammer is tried at different positions along the line to observe different degrees of interferences, and interference observations in different directions are measured as well.

In the experiments, two kinds of links are used, strong links and weak links. The setting of a weak link case is presented in Figure 1(a). The receiver R is put on the communication edge of transmitter T, i.e. the fan area [3], resulting in a weak link that only has 80% packet delivery ratio. The distance between the transmitter and the receiver is measured to be 16.2 feet. In Figure 1(b), the setting of the strong link case is shown. The receiver R is not put on the edge of transmitter T's communication range. Instead, it is put close to the transmitter, to get a strong link that has a stable 100% packet delivery ratio. The distance between the transmitter and the receiver is measured to be 8.5 feet. Similarly, the jammer is put on different positions in one direction to observe different interferences, and also interferences from different directions are measured.

All the experiments are repeated several times and consistent results are obtained.

B. Interference in one Direction

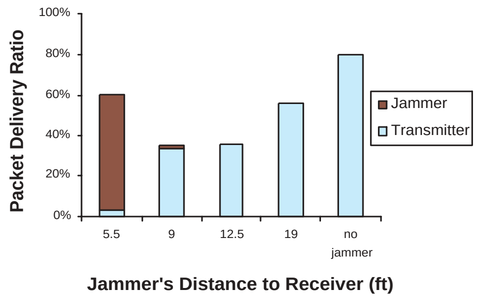
In this experiment, the jammer's interference is measured in one direction, on both a strong link and a weak link. In the weak link case, the packet delivery ratio is 80% when there is no interference, and the distance between transmitter T and receiver R (Figure 1(a)) is 16.2 feet. The experimental results of this observation are illustrated in Figure 2. As Figure 2(a) shows, when the distance between jammer J and receiver R increases, more packets from transmitter T are able to go through the channel and be correctly received by the receiver. On the other hand, less and less packets from the jammer are correctly received by the receiver. This is because when the jammer moves further away from the receiver, its own signal gets weaker when it arrives at the receiver, thus becoming less capable of interfering with the transmitter's signal. On the contrary, the transmitter's signal makes the signal of the jammer harder to receive.

In the case of a strong link that has 100% packet delivery ratio, similar phenomena are observed. The packet delivery ratio of the transmitter increases from 0% to 100%, with the increase of the distance between the jammer and the receiver from 2.92 feet to 19 feet. On the other hand, the packet delivery ratio of the jammer decreases from 99.2% to 0%.

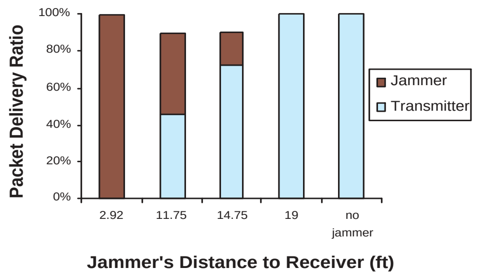
In addition, both Figure 2(a) and (b) illustrate that when the transmitter and the jammer have similar distances to the receiver, the total communication throughput of the link, including packets from both the jammer and the transmitter, goes down. This is because when the transmitter's distance to the receiver is similar to that of the jammer, their signals are at similar power levels, resulting in higher probability that both of them get corrupted.

C. Interference in Different Directions

Besides the interferences in one direction, interferences in different directions are also measured, and Figure 3 shows the experimental results. As Figure 3 shows, neither the radio interference pattern nor the radio communication pattern is spherical, which is consistent with the result in [2].



(a) Radio Interference for a Weak Link

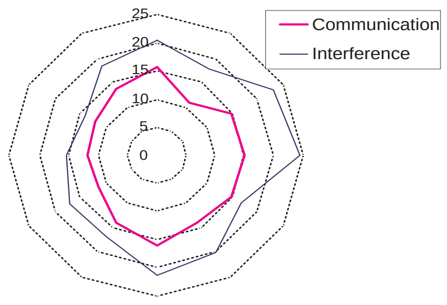


(b) Radio Interference for a Strong Link

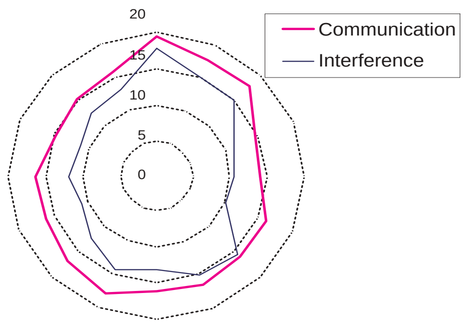
Fig. 2. Radio Interference in One Direction

In addition, our experiments also show that for different links, the relations between the communication range and the interference range are different. As Figure 3(a) shows, when the link is weak, the interference range is larger than the communication range. On the other hand, when the link is strong (Figure 3(b)), the interference range is smaller than the communication range. This is because whether the transmitter's packet is able to be correctly received by the receiver is determined by the relative strengths of the transmitter's signal, the receiver's signal, and the receiver's background noise. The transmitter's signal can only be correctly received if its power level is equal to or bigger than the product of the receiver's Signal-Noise-Ratio (SNR) threshold and the accumulative power level of the jammer's signal and the receiver's background noise.

Accordingly, when the link is weak, the transmitter's signal power level is very low and it can easily get interfered with by a distant jammer. On the contrary, in a strong link, the transmitter's signal is too strong to be disrupted by the signal of the jammer, no matter the jammer is outside the communication range or within the outer part of the communication range. So from the weak link case, we know that interference does not necessarily imply connectivity, while from the strong link case, we know that a connectivity does not necessarily result in interference.



(a) Interference Pattern Measured for a Weak Link



(b) Interference Pattern Measured for a Strong Link

Fig. 3. Radio Interference Pattern in Different Directions

Since neither interference-connectivity nor connectivity-interference assumptions is well maintained in real running systems, many existing concepts and protocols based on these assumptions are no longer logically correct. For example, the hidden terminal problem [10] is one of the most important and most frequent phenomena in wireless communication. Current research on MAC [8][7] assumes that if collision-free scheduling within two communication hops can be done, the whole network will be collision free. However, as Figure 3 presents, the communication range does not equal the interference range and the relation between them depends on how strong the link is. So it is not logically appropriate to use two hops of communication range as the basis to avoid interference.

Accordingly, the communication topology is not an accurate approximation of the interference topology, and it is challenging to design collision-free MAC protocols, without knowing the interference relations among nodes. Hence we are motivated to put forward a radio interference detection protocol, RID.

III. INTERFERENCE DETECTION PROTOCOLS

In this section, a radio interference detection protocol, RID, and its lightweight version, RID-B, are presented.

A. Radio Interference Detection Protocol: RID

The basic idea of RID is that a transmitter broadcasts a High Power Detection packet (HD packet), and immediately follows it with a Normal Power Detection packet (ND packet). This is called an HD-ND detection sequence. The receiver uses the HD-ND detection sequence to estimate the transmitter's interference strength. An HD packet includes the transmitter's ID, from which the receiver knows from which transmitter the following ND packet comes. The receiver estimates possible interference caused by the transmitter by sensing the power level of the transmitter's ND packet. In order to make sure every node within the transmitter's interference range is able to receive the HD packet, we assume that the communication range, when the high sending power is used, is at least as large as the interference range, when the normal sending power is used.

After the HD-ND detection, each node begins to exchange the detected interference information among its neighborhood, and then uses this information to figure out all collision cases within the system.

In what follows, the three stages of RID, (i) HD-ND detection, (ii) information sharing, and (iii) interference calculation, are discussed in detail.

1) HD-ND Detection: With a high sending power, the transmitter first sends out an HD packet, which only contains its own ID information (two bytes) and the packet type (one Byte) to minimize the packet length and to save transmission energy. Then the transmitter waits until the hardware is ready to send again. After the Minimal Hardware Wait Time (MHWT), the transmitter immediately sends out a fixed-length ND packet, with the normal sending power. The ND packet's length is fixed in order that the receiver is able to estimate when the ND packet's transmission will end once it starts to be sensed. At the receiver side, the HD-ND detection sequences are used to estimate the interference strength from corresponding transmitters.

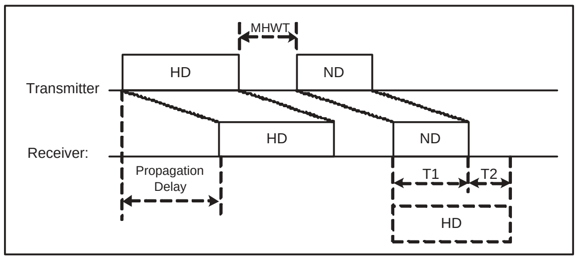


Fig. 4. Time Sequence of an HD-ND Detection

Figure 4 illustrates the time sequence of the whole HD-ND transmission, propagation and reception process. From the ID in the HD packet, the receiver gets to know which node is transmitting. The receiver also gets to know that an ND packet from the same transmitter will arrive later after the MHWT time. So it senses the signal strength of the ND packet during that time period, that is, the $T1$ time period in Figure 4. In the following, we present the detection estimation rules the receiver uses:

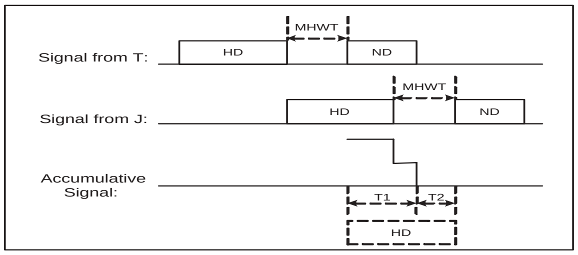
1) If the power level sensed in time period $T1$ is as low as that of the background noise, the receiver knows that the corresponding transmitter's interference strength is extremely weak, and does not record any information.
2) If the power level sensed in time period $T1$ is clearly above that of the background noise, the receiver thinks this data is useful and records the (transmitter ID, power level) pair for later use.

We also note that multiple HD-ND detection sequences from different transmitters may overlap and disturbance among these detection sequences may happen. Even though each transmitter can choose a random backoff before sending its HD-ND detection sequence, trying to avoid their HD-ND detection sequences from overlapping, the overlapping and disturbance among different detection sequences can not be completely prevented. So we provide an add-on rule for receivers to detect disturbance and avoid recording the disturbed detection results. This add-on rule can be presented as follows: if either of the following two conditions is violated, the receiver gets to know that this HD-ND detection sequence is disturbed by another HD-ND detection sequence, and the result is not useful and marked invalid.

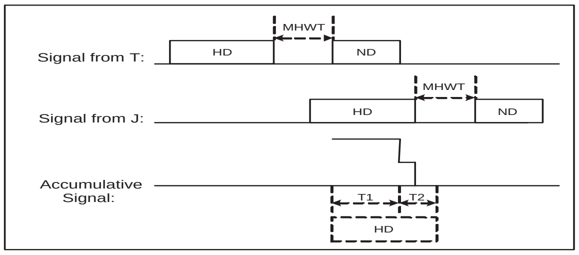
1) The power level sensed during time period $T1$, which is determined by the fixed length of ND packets, is stable.
2) The power level sensed during time period $T2$, which is determined by the fixed size of both ND and HD packets, is always as low as that of the background noise.

We illustrate the importance of this add-on rule with examples (Figure 5). Figure 5(a) presents a disturbance case the receiver is not able to be aware of, without the first requirement of the add-on rule. In Figure 5(a), the HD packet from jammer J overlaps with the ND packet of transmitter T at the receiver side, which results in the unstable power level sensed at the receiver side during time period $T1$. So it violates the first requirement of the add-on rule, and the receiver gets to know that this HD-ND detection is disturbed, and it marks the detection result invalid. In Figure 5(b), the overlapping detection sequences from jammer J and transmitter T can be detected, because the sensed power level in time period $T2$ is not always as low as that of the background noise, and the second requirement of the add-on rule is violated. These two requirements in the add-on rule can be used to detect most disturbances and reduce their adverse effects.

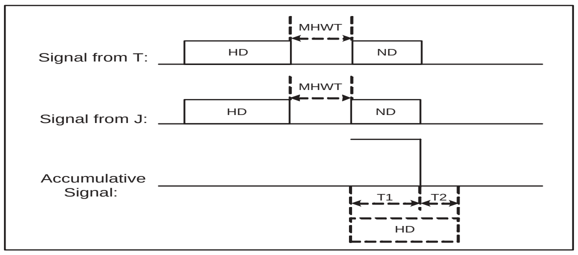
However, as Figure 5(c) illustrates, there are some cases, in which neither of these two conditions in the add-on rule is violated, but there is disturbance. However, the probability such a case happens is low. In addition, each transmitter can send out the HD-ND detection sequences multiple times at different time, and the average sensed power level at the receiver side can be used in the (transmitter ID, power level) pair. This method is also helpful to deal with engineering issues brought by a dynamic environment, as well as to give more opportunities to transmitters whose HD-ND detections are marked invalid at the receiver side to avoid dirty detection results. All these (transmitter ID, power level) pairs are put in a local table of the receiver, called the *Interference_In* table.



(a) Variable Sensed Power Level During $T1$



(b) Variable Sensed Power Level During $T1$



(c) Stable Sensed Power Level During $T1$ and $T2$

Fig. 5. Overlapping of Multiple Transmitters' HD-ND Detection Sequences

2) Information Sharing: During the HD-ND detection, each node puts in its *Interference_In* table the information about which nodes may cause potential interference when the node itself is the receiver, as well as how much interference the node may get. For each node, the other important side of interference detection is to get to know on which nodes the node itself has potential interference and how much the interference will be. Also, the interference topology among two interference hops is necessary to deal with the hidden terminal problem [10]. For these reasons, information sharing is designed.

There are two options for performing information sharing. Each node can choose to use a high sending power to broadcast its interference information among its neighborhood. The other way is to use the normal sending power to relay interference information among multiple hops. In our implementation, the high power broadcast packet is used for each node to broadcast the interference information in its own *Interference_In* table. When the broadcast packet is received, the receiver node R builds two other tables, besides the *Interference_In* table:

- *Interference_Out Table*: This table contains information of nodes on which node R has potential interfere.
- *Interference_HTP Table*: This table contains information of nodes that are hidden from node R, when one of R's neighbors is the receiver.

These two tables can be built according to the following two rules:

1) If receiver R's ID is in the broadcast packet, receiver R gets to know that it has potential interference on the transmitter of the broadcast packet. So node R puts the transmitter's ID in its *Interference_Out* table.
   To reduce redundancy, nodes already in R's *Interference_In* table are not inserted into the *Interference_Out* table again.
2) For any ID in the broadcast packet, if it is not receiver R's ID, it is put in R's *Interference_HTP* table.
   Also, to reduce redundancy, nodes already in either R's *Interference_In* table or *Interference_Out* table are not inserted into the *Interference_HTP* table again.

3) Interference Calculation: In the three tables, *Interference_In*, *Interference_Out* and *Interference_HTP*, enough information about potential interferences is collected. All this information is processed, in the interference calculation phase, to figure out all the scenarios in which collisions are sure to happen.

$$\begin{aligned} N_2(D) &= \{(i_1, i_2) | (P_{i_1D} < (P_{i_2D} + P_{\text{idle}}) * SNR_T) \\ &\quad \wedge (P_{i_1D} > \text{receiver\_sensitivity})\} \end{aligned} \tag{1}$$

Formula 1 defines the set of possible interference cases at receiver D, when there are only two simultaneous transmitters. Parameter P_{i_1D} represents the power level node D senses when the normal power packet from node i_1 arrives, while P_{i_2D} represents the power level node D senses when the normal power packet from node i_2 arrives. Parameter P_{idle} denotes the power level of the background noise around node D, when there are no radio signals. Parameter SNR_T is the receiver's SNR threshold for correct packet reception.

If $P_{i_1D} >$ receiver_sensitivity, node i_1's signal is strong enough to be received by node D, provided that there is no other radio signals. If $P_{i_1D} < (P_{i_2D} + P_{\text{idle}}) * SNR_T$, node i_1's signal is not strong enough and will be disturbed by node i_2's signal [11]. Accordingly, $(i_1, i_2) \in N_2(D)$ carries two meanings: first, node i_1's signal can be disturbed by node i_2's signal, and second, if there is no interference, node i_1's signal is able to be received by node D.

According to the membership conditions of Formula 1, RID is able to calculate all members of $N_2(D)$ for each receiver D, and thus obtain all collision cases when only two simultaneous transmitters get involved. Many TDMA protocols [7][8] only consider these collision cases.

However, when neither jammer A nor jammer B individually is able to interfere with the communication from transmitter T to receiver R, it does not mean that transmitter T's signal will not be disturbed, when A, B and T transmit data packets at the same time. That is, the composite of multiple negligible jammers is not necessarily negligible. In order to deal with this case, and also to make RID's detection complete, RID uses Formula 2 to calculate the remaining collision cases.

$$\begin{aligned} N_k(D) \quad = \quad & \{(i_1, i_2, \ldots, i_k)| \\ & (P_{i_1D} < (P_{i_2D} + \ldots + P_{i_kD} + P_{\text{idle}}) * SNR_T) \\ & \wedge (P_{i_1D} > \text{receiver\_sensitivity}) \\ & \wedge (\forall t(2 \leq t \leq k-1 \Rightarrow \\ & (\forall j_1, \ldots, j_{t-1}(i_2 \leq j_1, \ldots, j_{t-1} \leq i_k) \Rightarrow \\ & (i_1, j_1, \ldots, j_{t-1}) \notin N_t(D))))\} \qquad (2) \end{aligned}$$

The parameters in $N_k(D)$ are defined similarly as those in $N_2(D)$. According to the definition in Formula 1 and Formula 2, $N_k(D)$ satisfies the following two properties:

$$\forall i, j(i \neq j \Rightarrow (N_i(D) \cap N_j(D) = \phi)) \qquad (3)$$

$$\text{All Collision Scenarios in System} = \bigcup_{k=2}^{N} \bigcup_{i=1}^{N} N_k(D_i) \qquad (4)$$

Here N is the number of sensor devices actually deployed, and $\{D_i\}$ is the set consisting of all nodes in the system. From Formula 2, RID can calculate $N_k(D)$ for any k value, which means that all possible interference cases at the receiver D, no matter how many simultaneous transmitters get involved, are able to be obtained by calculation. However, all the $N_k(D)$ don't have to be calculated at the same time. Their members can be calculated separately, in an on-demand way.

B. Lightweight Radio Interference Detection Protocol: RID-B

In this section, motivations for a lightweight RID are presented, and corresponding design differences are given.

1) Motivations of RID-B: The full version of RID as described above is able to detect all collision scenarios that could happen some time in the system. But, to take full use of the detected information from RID to achieve collision-free scheduling as well as to maximize the network bandwidth, information about which nodes have packets to send, which nodes have no transmission requirements, and which nodes will be the desired transmission destinations are needed. In TDMA like TRAMA [7], packet delivery information in the future from higher layer applications is assumed known. In MAC layer, nodes exchange this information among neighborhoods to perform the TDMA scheduling.

However, in wireless sensor networks, most applications are designed for unattended environments [12][13][14][15][16]. They are developed to monitor objects in environments, detect possible events, and report important events back to the base station. That is, they are event based applications, so to obtain the data delivery requests from higher layer applications in the future is extremely hard. In addition, to exchange the application layer's future traffic requirements [7] is very expensive, and hence is not desired in wireless sensor networks, because the limited power supply and communication bandwidth are already big problems.

Accordingly, in the rest of the paper, a lightweight RID, called RID-Basic (RID-B), is given and its corresponding applications are presented.

2) Design Differences of RID-B from RID: The main body of RID-B is similar with that of RID. Each node also sends out HD-ND detection sequence for receivers to estimate the interference. The detection estimation rules and the add-on rule for receivers are the same as those of RID presented in Section III-A.1.

However, in RID-B, after a receiver puts all (transmitter ID, power level) pairs in its $Interference\_In$ table, the table gets reorganized again, according to the following condition:

$$\begin{aligned} P_{minR} < \quad & (P_{JR} + P_{\text{idle}}) * SNR_T \\ \text{where} \quad P_{minR} = \quad & min\{P_{iR} | i \neq J \\ & \wedge P_{iR} > \text{receiver\_sensitivity}\} \qquad (5) \end{aligned}$$

In Formula 5, P_{JR} represents the sensed power level when jammer J's signal arrives at receiver R. Similarly, P_{iR} represents the sensed power level when node i's signal arrives at receiver R. Parameter P_{idle} represents R's background noise level when there is no radio signals. When $P_{iR} >$ receiver_sensitivity, node i's packets are able to be correctly received by node R, and node R is within node i's communication range. So P_{minR} represents the power level node R senses from R's most distant neighbor it can hear packets from, and $P_{minR} < (P_{JR} + P_{\text{idle}} * SNR_T)$ carries the information that node J is able to interfere with the weakest communication from R's neighbors to R.

Accordingly, if the sensed signal power from node J at receiver R (P_{JR}) satisfies the condition in Formula 5, node J is able to interfere with R's packet reception. In this case, the corresponding (transmitter ID, power level) pair in the $Interference\_In$ table is replaced by just the transmitter ID. On the contrary, if the power level in the (transmitter ID, power level) pair does not satisfy the condition in Formula 5, this pair is removed from the $Interference\_In$ table. After this reorganization, the $Interference\_In$ table no longer consists of rows of (transmitter ID, power level) pairs, but rows of transmitter IDs.

In addition, RID-B does not take into consideration the interference cases when multiple transmitters get involved. If neither jammer J_1 nor jammer J_2 can individually interfere with R's communication, RID-B does not put J_1 or J_2 in its *Interference_In* table. However, the accumulative signal power of J_1 and J_2 may be able to interfere with R's reception from its most distant neighbor. So RID-B is optimistic in some degree. But the probability that multiple transmitters' packets overlap is very low, because in wireless sensor networks, the payload of the MAC layer is short, usually 32 Bytes. So the transmission time of each packet is short, and the probability for multiple packets to overlap simultaneously is low.

In information sharing, content in the *Interference_In* table is exchanged among nodes, in the same way as RID does. The *Interference_Out* and *Interference_HTP* tables are also built in the same way. There is no interference calculation phase in RID-B. Instead, RID-B uses the *Interference_In* and *Interference_Out* tables to avoid direct interferences, and uses the *Interference_HTP* table to avoid hidden terminal problems. Accordingly, compared with RID, RID-B is simple and lightweight.

IV. USING RADIO INTERFERENCE DETECTION

Radio interference detection provides interference relations at a very low layer, which can then be widely used in upper layer applications, such as media access control (MAC), topology control, and localization. There are two kinds of MAC protocols: contention-based MAC protocols [10][17][18][19][20][21][22] and scheduling-based TDMA protocols [8][7][23][24]. A contention-based MAC protocol like CSMA allows collisions and retransmits lost packets, which reduces transmission time in light load, but suffers severe collisions in heavy load, resulting in frequent backoffs and long transmission time. A TDMA protocol schedules nodes to use the shared channels at different time to avoid collisions, which results in unnecessary transmission delay in light load, but is efficient in heavy load, maximizing network bandwidth usage. Due to space limitation, in this paper we focus on the evaluation of RID-B's application in TDMA protocols, and we also analyze RID-B's application on backoff algorithms in collision-based MAC protocols. We leave the rest as future work.

A. Using RID-B in TDMA

TDMA protocols can be classified into two groups: centralized TDMAs and distributed TDMAs. Centralized TDMA protocols like UxDMA [24] is not preferred in wireless sensor networks, because centralized scheduling is not scalable. NAMA [8] and TRAMA [7] are distributed TDMA protocols that try to schedule collision-free transmissions by using the knowledge of the communication range. Without considering radio interference, these TDMA algorithms can operate poorly, because a TDMA slot may be assigned to a node whose transmission may suffer from interference by a distant node, even though logically this should not occur. However, these scheduling algorithms can make use of the explicit interference knowledge from the protocols we put forth, RID and RID-B, to assign slots to avoid this problem and therefore greatly improve the channel utilization.

In this section, we choose NAMA as the typical MAC protocol to conduct performance evaluation. In NAMA, nodes within two communication hops are scheduled to avoid transmitting at the same time, to avoid collisions, while in NAMA that uses RID-B (called NAMA-RID-B), the *Interference_In* table, the *Interference_Out* table and the *Interference_HTP* tables are used to achieve collision free scheduling.

Separate performance evaluation for TRAMA is not presented here, because TRAMA uses the same principles as NAMA does to achieve collision avoidance.

1) Simulation Design: Since NAMA does not achieve real collision free operations, the MAC layer may drop packets, and the upper layer applications will have to retransmit the lost packets many times until the maximal retransmission limit is reached. On the other hand, with the help of NAMA-RID-B, interference relations are detected to make collision-free scheduling. So the upper layers do not retransmit packets due to collisions, and the control overhead in the upper layer is much less. In order to set a fair context for comparison, we move the retransmission function in the upper layer to the MAC layer, so that the MAC will try to minimize the possible packet loss, which is the original goal of TDMA designs. In addition, ACK packets are sent back from the receiver to the transmitter to acknowledge the reception of data packets, to provide a reliable hop-by-hop communication.

Since radio interference is related to many factors, we conducted three groups of separate experiments to explore system performance, when different factors are considered. In each group of experiments, the performance is evaluated with five metrics: average single hop loss ratio, average single hop transmission time, #retransmission, #control packets and energy consumption.

The first experiment is designed to explore the system performance when different system loads are used. The many-to-one pattern of CBR streams is used to simulate the environment monitoring application scenarios in wireless sensor networks, and the increasing system load is simulated by the increasing number of CBR streams.

The second experiment is designed to explore the sensitivity to different ICR ratios, which is defined as $ICR = R_I/R_C$. Here R_I is the interference range and R_C is the communication range. This experiment is important because different hardware have different communication abilities, and ICR values may be different from device to device. Since the interference range is different between a long link and a short link, as explained in Figure 3, here we use the interference range of the longest link for a node, in which the receiver is put at the exact edge of the transmitter's communication range.

The third experiment is designed to explore the result sensitivity to different Signal-Noise-Ratio (SNR) thresholds. In current applications, different types of low power wireless hardware are used, which have different receiver sensitivities and different SNR thresholds. In addition, devices produced in different years or by different companies also differ in hardware abilities and hence the SNR thresholds.

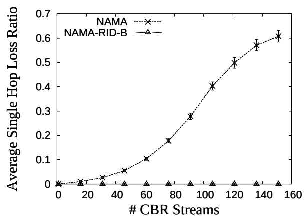
The event-driven simulation tool, GlomoSim [25], developed by ULCA, is used in our simulation and the general setting in GlomoSim is shown in Table I. Also, 90% confidence intervals are shown in each figure.

2) Performance Evaluation with Different System Loads: Figure 6 shows the performance difference between NAMA and NAMA-RID-B, when the system load increases.

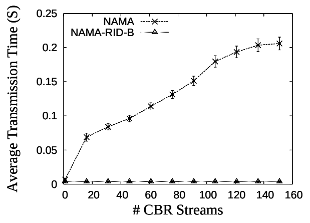
TABLE I

SIMULATION CONFIGURATION

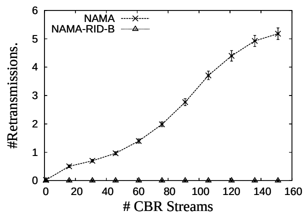
TERRAIN	(144m X 144m) Square
Node Number	144
Node Placement	Uniform
Application	Many-to-one CBR Streams
Payload Size	32 Bytes
Routing Layer	GF
MAC Layer	NAMA/NAMA-RID-B
Radio Layer	RADIO-ACCNOISE
Radio Bandwidth	250Kb/s
Radio Range	25m



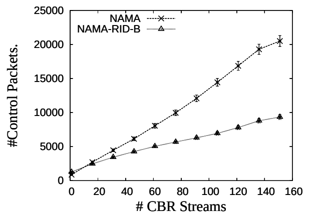
(a) Loss Ratio



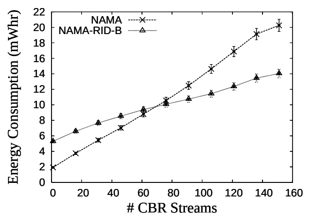
(b) Transmission Time



(c) #Retransmission



(d) #Control Packets



(e) Energy Consumption

Fig. 6. Performance Evaluation with Different System Loads

From Figure 6(a), we observe that NAMA's packet loss ratio increases from 0% (#CBR=1) to 60% (#CBR=151) when the system load increases, because more nodes beyond two communication hops begin to compete for shared channels, while NAMA only considers collision avoidance within two communication hops. The increasing #retransmission (from 0.04 to 5.38 in Figure 6(c)) for each successfully delivered data packet also makes it clear that NAMA performs worse when the system load increases. When a packet gets lost, NAMA tries to retransmit it. On the other hand, the retransmission scheme is useful, which reduces the packet loss ratio, as can be seen from Figure 6(c). In Figure 6(c), the #retransmission for each successfully delivered packet is less than 8, the maximal retransmission limit, which means that a lot of lost packets are successfully delivered after several retransmissions. However, when #retransmission increases, the transmission time increases as well, from 8ms (#CBR=1) to 215ms (#CBR=151) as shown in Figure 6(b).

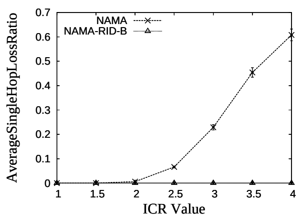
In both Figure 6(a) and (c), we observe that NAMA-RID-B has no packet loss. This is because RID-B detects all the nodes that can cause potential interferences, and NAM-RID-B schedules those nodes to transmit in different time slots, and hence avoids collisions. So in spite of the increase of system load, NAMA-RID-B always maintains 100% packet delivery ratio (Figure 6(a)) and has no retransmission due to collisions (Figure 6(c)). For the same reason, the transmission time of NAMA-RID-B in Figure 6(b) is low, less than 4ms.

Figure 6(d) shows that NAMA's control overhead increases rapidly with the increase of system load. This is because of two reasons. First, when #CBR streams increases, more packets are transmitted, even though the delivery ratio decreases. So more ACK packets are needed to acknowledge the successful transmission. Second, more data packets get lost and more overhead is paid to retransmit these packets. On the other hand, the control overhead of NAMA-RID-B increases slowly. As Figure 6(d) shows, RID-RID-B has less than 50% control overhead compared to NAMA when the load is very heavy (#CBR=151). This is because NAMA-RID-B does not have transmission failure due to collisions, and does not retransmit corrupted packets. So the only source of the increasing overhead is the increasing ACK packets.

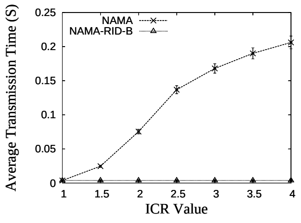
Similarly, the energy consumption of NAMA increases rapidly with the increase of system load, while the energy consumption of NAMA-RID-B increases slowly, as illustrated in Figure 6(e). The reason is that both NAMA and NAMA-RID-B spend more energy in acknowledging delivered packets, while NAMA also spends more energy to retransmit the lost packets. However, when there is only one CBR stream, the energy consumption of NAMA-RID-B is larger than that of NAMA (Figure 6(e)), because RID-B uses HD-ND detection sequences, rather than individual packets in NAMA, to detect the interference relations. Besides, the HD packets consume more energy than normal packets.

3) Performance Evaluation with Different ICR: In this experiment, different ICR (defined in Section IV-A.1) values are used, and the simulation results are presented in Figure 7. When ICR is 1, the interference range equals the communication range. That is why both NAMA and NAMA-RID-B perform well, achieving 100% data delivery ratio(Figure 7(a)) and no retransmission (Figure 7(c)) when ICR is 1. Also the transmission time of NAMA and NAMA-RID-B are the same (Figure 7(b)). But, from Figure 7(d) and (e), we observe that NAMA-RID-B pays slightly higher control overhead and energy consumption than NAMA, when ICR is 1. This is because RID-B uses HD-ND detection sequences, rather than individual packets as NAMA uses, and also because HD packets consume more energy than normal packets.

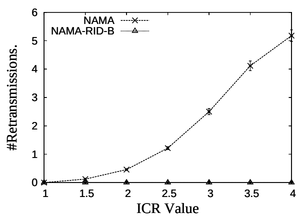
With the increase of the ICR value, NAMA loses its control of collision avoidance, and transmission begins to fail due to collisions, as Figure 7(a) shows. The #retransmission in Figure 7(c) increases from 0 to 5.38, and the transmission time in Figure 7(b) increases from less than 4ms to 215ms.



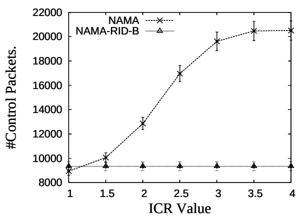
(a) Loss Ratio



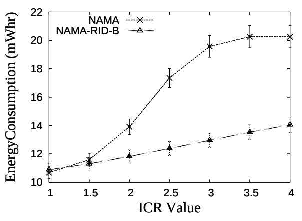
(b) Transmission Time



(c) #Retransmission



(d) #Control Packets



(e) Energy Consumption

Fig. 7. Performance Evaluation with Different ICR

However, in spite of the increase of the ICR values, NAMA-RID-B is always able to maintain 100% packet delivery ratio (Figure 7(a)), and there are no retransmissions (Figure 7(c)). Besides, it keeps constantly low transmission time, less than 4ms as Figure 7(b) shows. This is because RID-B is able to detect the increasing interference range, and the exact interference information detected is used to make collision-free time slot scheduling.

Since NAMA has more collisions, with the increase of ICR values, it pays more overhead (Figure 7(d)) to retransmit the lost packets, and the energy consumption (Figure 7(e)) increases rapidly as well. NAMA-RID-B also pays a little more control overhead, but much lower than that of NAMA, to detect the increasing interference range, which is hard to observe in Figure 7(d) because the increase is small compared with the scale of the Y coordinate, but it's clear in Figure 7(e) as it shows up as energy consumption.

4) Performance Evaluation with Different SNR Thresholds: Figure 8 shows the simulation results when different Signal-Noise-Ratio (SNR) thresholds are used. From Figure 8(a), (b) and (c), it's clear that NAMA suffers more interferences and more packets get corrupted, with the increase of the SNR threshold. The packet loss ratio (Figure 8(a)) increases, the #retransmission (Figure 8(c)) increases, and the transmission time (Figure 8(b)) also increases. The reason is that when SNR threshold increases, the receiver becomes more and more sensitive to interference. So a transmission gets easier to be interfered with by nodes from longer distances.

However, since NAMA-RID-B can detect possible interferences, it does not get affected by the increasing SNR threshold. Figure 8(a) and (c) show that the collision-free packet delivery is always maintained in NAMA-RID-B, and Figure 8(b) shows that the transmission time is short.

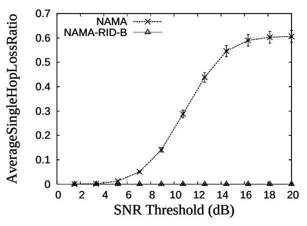
As Figure 8(a) illustrates, the packet loss ratio of NAMA stops increasing when it arrives at 60%. The existence of this upper bound reflects that NAMA has certain degree of collision avoidance ability, since NAMA is designed to avoid collisions within two communication hops. Performance result from Figure 8(b), (c), (d) and (e) also confirm the existence of the upper bound.

The control overhead of NAMA increases (Figure 8(d)) with the increase of SNR thresholds, because more packets get corrupted and retransmitted. And the energy consumption of NAMA also increases, as shown in Figure 8(e). Since NAMA-RID-B does not spend more overhead to detect the interference relation, nor does it take effort to retransmit lost packets when SNR threshold increases, NAMA-RID-B shows constantly low control overhead and constant energy consumption. The initial energy consumption of NAMA-RID-B, when the SNR threshold is low, is bigger than that of NAMA, because the HD-ND sequences consume more energy than the individual packets in NAMA. However, when the SNR threshold increases, NAMA-RID-B saves as much as 30% energy compared to NAMA.

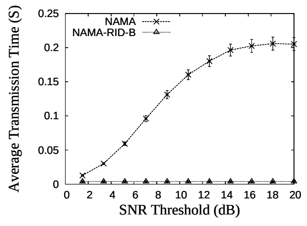
V. Using RID-B in Backoff Algorithms

Contention-base MAC protocols such as CSMA [17] and 802.11 DCF [10] use backoff to avoid further collisions after a collision happens.

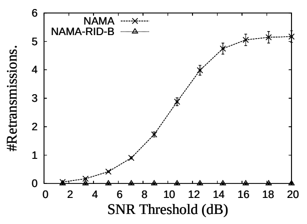
In a typical backoff algorithm, each node adopts the same initial window size, and each time when collision happens, the backoff window size doubles. After the channel is sensed clear and data has been retransmitted successfully, the backoff window size is reset to the initial value. Usually, nodes choose the same parameter settings to achieve fairness in channel access. This traditional mechanism works well in many situations where nodes are treated as logical independent entities. Obviously, without customizing parameters such as initial windows size, according to network configuration (e.g. interference density) surrounding individual nodes, it is hard to achieve the optimal aggregate throughput. We note here



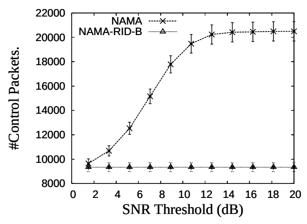
(a) Loss Ratio



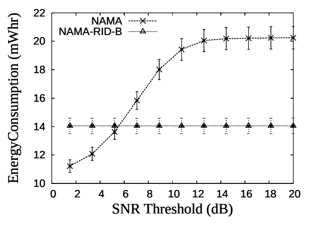
(b) Transmission Time



(c) #Retransmission



(d) #Control Packets



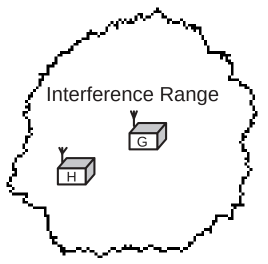
(e) Energy Consumption

Fig. 8. Performance Evaluation with Different SNR Thresholds

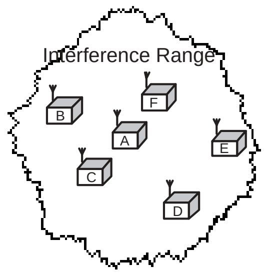
that due to the breakdown of connectivity-interference and interference-connectivity assumptions, the neighborhood of connectivity can no longer precisely reflect real contention situations. With RID, we are able to identify the neighborhood of potential interference, thus adaptively adjust the backoff strategy of individual nodes.

We illustrate this point through an example (Figure 9). The area near node G has low interference density and the area near node A has high interference density. Since node G has only one potential node to compete with, for the shared channel, while node A has much more potential nodes to compete with, it's desirable to assign different initial window sizes to node A and G. Otherwise, either node G suffers unnecessary communication delays, or node A suffers excessive number of backoffs due to contention.

With RID-B, each node gets to know the set of nodes that are able to interfere with its communication. So different nodes can set different initial window sizes according to the number of nodes that show up in their *Interference_In* tables, or their *Interference_Out* tables, or their *Interference_HTP* tables to improve aggregated throughput of the network. Due to space



(a) Interference range of G



(b) Interference range of A

Fig. 9. Implications on Backoff Algorithms

limitation, we leave the evaluation as future work.

VI. Related Work

To the best of our knowledge, there is no previous work on radio interference detection in run-time systems. Interference-connectivity and connectivity-interference assumptions are still widely used to design collision-free MAC designs. NAMA [8] and TRAMA [7] are such protocols that make collision-avoidance scheduling with node information two communication hops away, which is shown to perform poorly in heavy load, because interference range is not the same as communication range. Our work differs by detecting the real radio interference relations among nodes in run-time systems, and then uses this information to achieve collision-free communication.

The Shadowing Phenomenon work [9] points out that a connectivity does not necessarily lead to corruptions of all involved packets, and it designs algorithms to recover the stronger packet involved in the collision and drop the weaker one. In all our experiments, this recovery scheme is considered. Besides, we also point out that interference does not necessarily come from connectivity. We also put forth RID and RID-B to detect interference relations among nodes, and use this information to assist TDMA design.

Many recent works [2][3][4][5][6] conduct extensive experiments to study radio irregularity and asymmetry links. Their work indirectly reflects the existence and complexity of radio interference. However, they don't try to address radio interference. Neither interference detection nor collision avoidance is addressed in their work.

VII. Conclusions and Future Work

In this paper, we focus on a very important issue in wireless sensor networks, the radio interference. Our contributions are as follows:

- To the best of our knowledge, our work is the first to detect radio interference relations among nodes in run-time systems. We present the design of the first radio interference detection protocol, RID, as well as its variation, RID-B.
- We implement RID-B in GlomoSim, and conduct extensive simulation experiments to study the application of

RID-B in TDMA design. We observe that the traditional TDMA protocol, NAMA, can have up to 60% packet loss in heavy load, while the RID-B supported TDMA, NAMA-RID-B, can maintain 100% packet delivery.

- We also analyze the application of radio interference detection, on how to design adaptive backoff algorithms.

In future work, we will concentrate on the following aspects. First, we plan to design schemes to predict the future traffic information of higher layer applications, and then combine this information with RID to achieve more bandwidth efficient TDMA designs. Second, we plan to analyze the combination of RID with topology control protocols. Third, we plan to further evaluate the radio interference detection in a large-scale sensor network system, and also do research on the interaction between radio interference and radio irregularity.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[2] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Impact of radio irregularity on wireless sensor networks," in *ACM MobiSys 2003*, June 2004, pp. 125–138.

[3] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *ACM SenSys 2003*, November 2003.

[4] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *ACM SenSys 2003*, November 2003.

[5] A. Cerpa, N. Busek, and D. Estrin, "SCALE: A tool for simple connectivity assessment in lossy environments," Tech. Rep., 2003.

[6] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker, "Complex behavior at scale: an experimental study of low-power wireless sensor networks," Tech. Rep., 2002.

[7] V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves, "Energy-efficient, collision-free medium access control for wireless sensor networks," in *ACM SenSys 2003*, November 2003.

[8] L. Bao and J. J. Garcia-Luna-aceves, "A new approach to channel access scheduling for ad hoc networks," in *ACM MobiCom 2001*, July 2001, pp. 210–221.

[9] A. Woo, K. Whitehouse, F. Jiang, J. Polastre, and D. Culler, "The shadowing phenomenon: implications of receiving during a collision," Tech. Rep., 2004.

[10] "IEEE 802.11, part II: wireless LAN medium access control (MAC) and physical layer (PHY) specification," ANSI/IEEE Std. 802.11, 1999.

[11] T. S. Rappaport, *Wireless systems principles and practice*, Prentice Hall PTR, 1999.

[12] G. Simon, M. Maróti, and Á. Lédeczi, "Sensor Network-Based Countersniper System," in *ACM SenSys 2004*, November 2004.

[13] T. He, S. Krishnamurthy, J. A. Stankovic, T. F. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh, "An energy-efficient surveillance system using wireless sensor networks," in *ACM MobiSys 2004*, June 2004, pp. 270–283.

[14] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson., "Wireless sensor networks for habitat monitoring," in *ACM WSNA 2002*, September 2002.

[15] B. Thorstensen, T. Syversen, T. A. Bjornvold, and T. Walseth, "Electronic shepherd - a low-cost, low-bandwidth, wireless network system," in *ACM MobiSys 2004*, June 2004, pp. 245–255.

[16] T. Liu, C. M. Sadler, P. Zhang, and M. R. Martonosi, "Implementing software on resource-constrained mobile sensors: experimences with impala and ZebraNet," in *ACM MobiSys 2004*, June 2004, pp. 256–269.

[17] L. Kleinrock and F. A. Tobagi, "Packet switching in radio channels: part I - carrier sense multiple-access modes and their throughput-delay characteristics," *IEEE Transactions on Communications*, vol. COM-23, pp. 1400–1416, December 1975.

[18] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *IEEE InfoCom*, June 2002, pp. 1567–1576.

[19] S. Singh and C. S. Raghavendra, "PAMAS: Power aware multi-access protocol with signaling for ad hoc networks," 1999.

[20] C. L. Fullmer and J. J. Garcia-Luna-Aceves, "Floor acquisition multiple access (FAMA) for packet-radio networks," in *ACM SigComm 1995*, 1995, pp. 262–273.

[21] P. Karn, "MACA - A new channel access method for packet radio," in *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, 1990, pp. 134–140.

[22] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang, "MACAW: A media access protocol for wireless LANs," in *ACM SigComm 1994*, 1994, pp. 212–225.

[23] V. Rajendran, J. J. Garcia-Luna-Aceves, and K. Obraczka, "An energy-efficient channel access scheduling for sensor networks," in *The Fifth International Symposium on Wireless Personal Multimedia Communication*, October 2002.

[24] S. Ramanathan, "A Unified framework and algorithm for channel assignment in wireless networks," *Wireless Networks*, vol. 5, no. 2, pp. 81–94, 1999.

[25] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: a library for parallel simulation of large-scale wireless networks," in *The 12th Workshop on Parallel and Distributed Simulations*, May 1998.

A High-Accuracy, Low-Cost Localization System for Wireless Sensor Networks

Radu Stoleru, Tian He, John A. Stankovic, David Luebke
Department of Computer Science
University of Virginia, Charlottesville, VA 22903
{stoleru, tianhe, stankovic, luebke}@cs.virginia.edu

ABSTRACT

The problem of localization of wireless sensor nodes has long been regarded as very difficult to solve, when considering the realities of real world environments. In this paper, we formally describe, design, implement and evaluate a novel localization system, called Spotlight. Our system uses the spatio-temporal properties of well controlled events in the network (e.g., light), to obtain the locations of sensor nodes. We demonstrate that a high accuracy in localization can be achieved without the aid of expensive hardware on the sensor nodes, as required by other localization systems. We evaluate the performance of our system in deployments of Mica2 and XSM motes. Through performance evaluations of a real system deployed outdoors, we obtain a 20cm localization error. A sensor network, with any number of nodes, deployed in a 2500m^2 area, can be localized in under 10 minutes, using a device that costs less than $1000. To the best of our knowledge, this is the first report of a sub-meter localization error, obtained in an outdoor environment, without equipping the wireless sensor nodes with specialized ranging hardware.

Categories and Subject Descriptors

C.2.4 [**Computer-Communications Networks**]: Distributed Systems; C.3 [**Special-Purpose and Application-Based Systems**]: Real-Time and embedded systems.

General Terms

Algorithms, Measurement, Performance, Design, Experimentation

Keywords

Wireless Sensor Network, Localization, Event Distribution, Laser

1. INTRODUCTION

Recently, wireless sensor network systems have been used in many promising applications including military surveillance, habitat monitoring, wildlife tracking etc. [12] [22] [33] [36]. While many middleware services, to support these applications, have been designed and implemented successfully, localization - finding the position of sensor nodes - remains one of the most difficult research challenges to be solved practically. Since most emerging applications based on networked sensor nodes require location awareness to assist their operations, such as annotating sensed data with location context, it is an indispensable requirement for a sensor node to be able to find its own location.

Many approaches have been proposed in the literature [4] [6] [13] [14] [19] [20] [21] [23] [27] [28], however it is still not clear how these solutions can be practically and economically deployed. An on-board GPS [23] is a typical high-end solution, which requires sophisticated hardware to achieve high resolution time synchronization with satellites. The constraints on power and cost for tiny sensor nodes preclude this as a viable solution. Other solutions require per node devices that can perform ranging among neighboring nodes. The difficulties of these approaches are two-fold. First, under constraints of form factor and power supply, the effective ranges of such devices are very limited. For example the effective range of the ultrasonic transducers used in the Cricket system is less than 2 meters when the sender and receiver are not facing each other [26]. Second, since most sensor nodes are static, i.e. the location is not expected to change, it is not cost-effective to equip these sensors with special circuitry just for a one-time localization. To overcome these limitations, many range-free localization schemes have been proposed. Most of these schemes estimate the location of sensor nodes by exploiting the radio connectivity information among neighboring nodes. These approaches eliminate the need of high-cost specialized hardware, at the cost of a less accurate localization. In addition, the radio propagation characteristics vary over time and are environment dependent, thus imposing high calibration costs for the range-free localizations schemes. With such limitations in mind, this paper addresses the following research challenge: How to reconcile the need for high accuracy in location estimation with the cost to achieve it. Our answer to this challenge is a localization system called Spotlight. This system employs an asymmetric architecture, in which sensor nodes do not need any additional hardware, other than what they currently have. All the sophisticated hardware and computation reside on a single Spotlight device. The Spotlight device uses a steerable laser light source, illuminating the sensor nodes placed within a known terrain. We demonstrate that this localization is much more accurate (i.e., tens of centimeters) than the range-based localization schemes and that it has a much longer effective range (i.e., thousands of meters) than the solutions based on ultra-sound/acoustic ranging. At the same time, since only a single sophisticated device is needed to localize the whole network, the amortized cost is much smaller than the cost to add hardware components to the individual sensors.

2. RELATED WORK

In this section, we discuss prior work in localization in two major categories: the range-based localization schemes (which use either expensive, per node, ranging devices for high accuracy, or less accurate ranging solutions, as the Received Signal Strength Indicator (RSSI)), and the range-free schemes, which use only connectivity information (hop-by-hop) as an indication of proximity among the nodes.

The localization problem is a fundamental research problem in many domains. In the field of robotics, it has been studied extensively [9] [10]. The reported localization errors are on the order of tens of centimeters, when using specialized ranging hardware, i.e. laser range finder or ultrasound. Due to the high cost and non-negligible form factor of the ranging hardware, these solutions can not be simply applied to sensor networks.

The RSSI has been an attractive solution for estimating the distance between the sender and the receiver. The RADAR system [2] uses the RSSI to build a centralized repository of signal strengths at various positions with respect to a set of beacon nodes. The location of a mobile user is estimated within a few meters. In a similar approach, MoteTrack [17] distributes the reference RSSI values to the beacon nodes.

Solutions that use RSSI and do not require beacon nodes have also been proposed [5] [14] [24] [26] [29]. They all share the idea of using a mobile beacon. The sensor nodes that receive the beacons, apply different algorithms for inferring their location. In [29], Sichitiu proposes a solution in which the nodes that receive the beacon construct, based on the RSSI value, a constraint on their position estimate. In [26], Priyantha et al. propose MAL, a localization method in which a mobile node (moving strategically) assists in measuring distances between node pairs, until the constraints on distances generate a rigid graph. In [24], Pathirana et al. formulate the localization problem as an on-line estimation in a nonlinear dynamic system and proposes a Robust Extended Kalman Filter for solving it. Elnahrawy [8] provides strong evidence of inherent limitations of localization accuracy using RSSI, in indoor environments.

A more precise ranging technique uses the time difference between a radio signal and an acoustic wave, to obtain pair wise distances between sensor nodes. This approach produces smaller localization errors, at the cost of additional hardware. The Cricket location-support system [25] can achieve a location granularity of tens of centimeters with short range ultrasound transceivers. AHLoS, proposed by Savvides et al. [27], employs Time of Arrival (ToA) ranging techniques that require extensive hardware and solving relatively large nonlinear systems of equations. A similar ToA technique is employed in [3].

In [30], Simon et al. implement a distributed system (using acoustic ranging) which locates a sniper in an urban terrain. Acoustic ranging for localization is also used by Kwon et al. [15]. The reported errors in localization vary from 2.2m to 9.5m, depending on the type (centralized vs. distributed) of the Least Square Scaling algorithm used.

For wireless sensor networks ranging is a difficult option. The hardware cost, the energy expenditure, the form factor, the small range, all are difficult compromises, and it is hard to envision cheap, unreliable and resource-constraint devices make use of range-based localization solutions. However, the high localization accuracy, achievable by these schemes is very desirable.

To overcome the challenges posed by the range-based localization schemes, when applied to sensor networks, a different approach has been proposed and evaluated in the past. This approach is called range-free and it attempts to obtain location information from the proximity to a set of known beacon nodes.

Bulusu et al. propose in [4] a localization scheme, called Centroid, in which each node localizes itself to the centroid of its proximate beacon nodes. In [13], He et al. propose APIT, a scheme in which each node decides its position based on the possibility of being inside or outside of a triangle formed by any three beacon nodes heard by the node. The Global Coordinate System [20], developed at MIT, uses apriori knowledge of the node density in the network, to estimate the average hop distance. The DV-* family of localization schemes [21], uses the hop count from known beacon nodes to the nodes in the network to infer the distance. The majority of range-free localization schemes have been evaluated in simulations, or controlled environments. Several studies [11] [32] [34] have emphasized the challenges that real environments pose. Langendoen and Reijers present a detailed, comparative study of several localization schemes in [16].

To the best of our knowledge, Spotlight is the first range-free localization scheme that works very well in an outdoor environment. Our system requires a line of sight between a single device and the sensor nodes, and the map of the terrain where the sensor field is located. The Spotlight system has a long effective range (1000's meters) and does not require any infrastructure or additional hardware for sensor nodes. The Spotlight system combines the advantages and does not suffer from the disadvantages of the two localization classes.

3. SPOTLIGHT SYSTEM DESIGN

The main idea of the Spotlight localization system is to generate controlled events in the field where the sensor nodes were deployed. An event could be, for example, the presence of light in an area. Using the time when an event is perceived by a sensor node and the spatio-temporal properties of the generated events, spatial information (i.e. location) regarding the sensor node can be inferred.

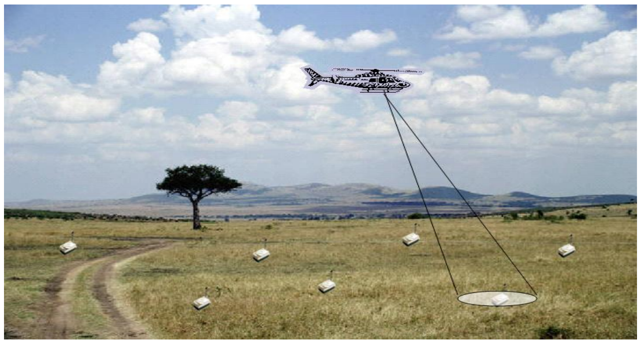


Figure 1. Localization of a sensor network using the Spotlight system

We envision, and depict in Figure 1, a sensor network deployment and localization scenario as follows: wireless sensor nodes are randomly deployed from an unmanned aerial vehicle. After deployment, the sensor nodes self-organize into a network and execute a time-synchronization protocol. An aerial vehicle (e.g. helicopter), equipped with a device, called Spotlight, flies over the network and generates light events. The sensor nodes detect the events and report back to the Spotlight device, through a base station, the timestamps when the events were detected. The Spotlight device computes the location of the sensor nodes.

During the design of our Spotlight system, we made the following assumptions:

- the sensor network to be localized is connected and a middleware, able to forward data from the sensor nodes to the Spotlight device, is present.
- the aerial vehicle has a very good knowledge about its position and orientation (6 parameters: 3 translation and 3 rigid-body rotation) and that is possesses the map of the field where the network was deployed.
- a powerful Spotlight device is available and it is able to generate

spatially large events that can be detected by the sensor nodes, even in the presence of background noise (daylight).
- a line of sight between the Spotlight device and sensor nodes exists.

Our assumptions are simplifying assumptions, meant to reduce the complexity of the presentation, for clarity. We propose solutions that do not rely on these simplifying assumptions, in Section 6.

In order to formally describe and generalize the Spotlight localization system, we introduce the following definitions.

3.1 Definitions and Problem Formulation

Let's assume that the space $A \subset R^3$ contains all sensor nodes N, and that each node N_i is positioned at $p_i(x, y, z)$. To obtain $p_i(x, y, z)$, a Spotlight localization system needs to support three main functions, namely an Event Distribution Function (EDF) $E(t)$, an Event Detection Function $D(e)$, and a Localization Function $L(T_i)$. They are formally defined as follows:

Definition 1: An event $e(t, p)$ is a detectable phenomenon that occurs at time t and at point $p \in A$. Examples of events are light, heat, smoke, sound, etc. Let $T_i=\{t_{i1}, t_{i2}, ..., t_{in}\}$ be a set of n timestamps of events detected by a node i. Let $T'=\{t_1', t_2', ..., t_m'\}$ be the set of m timestamps of events generated in the sensor field.

Definition 2: The Event Detection Function $D(e)$ defines a binary detection algorithm. For a given event e:

$$D(e) = \begin{cases} \text{true}, & \text{Event } e \text{ is detected} \\ \text{false}, & \text{Event } e \text{ is not detected} \end{cases} \tag{1}$$

Definition 3: The Event Distribution Function (EDF) $E(t)$ defines the point distribution of events within A at time t:

$$E(t) = \{p \mid p \in A \wedge D(e(t,p)) = true\} \tag{2}$$

Definition 4: The Localization Function $L(T_i)$ defines a localization algorithm with input T_i, a sequence of timestamps of events detected by the node i:

$$L(T_i) = \bigcap_{t \in T_i} E(t) \tag{3}$$

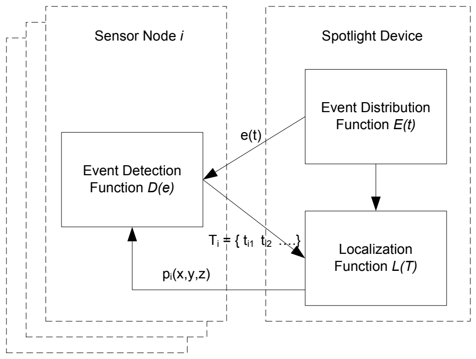

Sensor Node i
Spotlight Device
Event Distribution Function E(t)
e(t)
Event Detection Function D(e)
Ti = { ti1 ti2 ...}
Localization Function L(T)
pi(x,y,z)

Figure 2. Spotlight system architecture

As shown in Figure 2, the Event Detection Function $D(e)$ is supported by the sensor nodes. It is used to determine whether an external event happens or not. It can be implemented through either a simple threshold-based detection algorithm or other advanced digital signal processing techniques. The Event Distribution E(t) and Localization Functions $L(T_i)$ are implemented by a Spotlight device. The Localization function is an aggregation algorithm which calculates the intersection of multiple sets of points. The Event Distribution Function $E(t)$ describes the distribution of events over time. It is the core of the Spotlight system and it is much more sophisticated than the other two functions. Due to the fact that $E(t)$ is realized by the Spotlight device, the hardware requirements for the sensor nodes remain minimal.

With the support of these three functions, the localization process goes as follows:

1) A Spotlight device distributes events in the space A over a period of time.
2) During the event distribution, sensor nodes record the time sequence $T_i = \{t_{i1}, t_{i2}, ..., t_{in}\}$ at which they detect the events.
3) After the event distribution, each sensor node sends the detection time sequence back to the Spotlight device.
4) The Spotlight device estimates the location of a sensor node i, using the time sequence T_i and the known $E(t)$ function.

The Event Distribution Function $E(t)$ is the core technique used in the Spotlight system and we propose three designs for it. These designs have different tradeoffs and the cost comparison is presented in Section 3.5.

3.2 Point Scan Event Distribution Function

To illustrate the basic functionality of a Spotlight system, we start with a simple sensor system where a set of nodes are placed along a straight line ($A = [0, l] \subset R$). The Spotlight device generates point events (e.g. light spots) along this line with constant speed s. The set of timestamps of events detected by a node i is $T_i=\{t_{i1}\}$. The Event Distribution Function $E(t)$ is:

$$E(t) = \{p \mid p \in A \wedge p = t * s\} \tag{4}$$

where $t \in [0, l/s]$. The resulting localization function is:

$$L(T_i) = E(t_{i1}) = \{t_{i1} * s\} \tag{5}$$

where $D(e(t_{i1}, p_i)) = true$ for node i positioned at p_i.

The implementation of the Event Distribution Function $E(t)$ is straightforward. As shown in Figure 3(a), when a light source emits a beam of light with the angular speed given by $S_\alpha = \frac{d\alpha}{dt} = \frac{s \cos^2(\alpha)}{d}$, a light spot event with constant speed s is generated along the line situated at distance d.

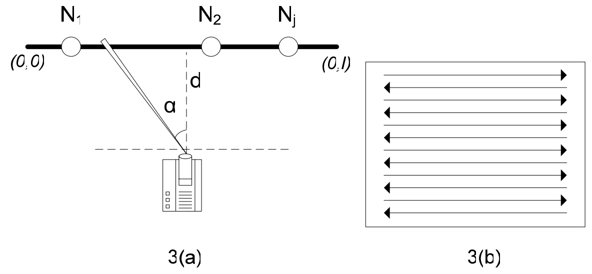


Figure 3. The implementation of the Point Scan EDF

The Point Scan EDF can be generalized to the case where nodes are placed in a two dimensional plane R^2. In this case, the Spotlight system progressively scans the plane to activate the sensor nodes. This scenario is depicted in Figure 3(b).

3.3 Line Scan Event Distribution Function

Some devices, e.g. diode lasers, can generate an entire line of events simultaneously. With these devices, we can support the Line Scan Event Distributed Function easily. We assume that the sensor nodes are placed in a two dimensional plane ($A=[l \times l] \subset R^2$) and that the scanning speed is s. The set of timestamps of events detected by a node i is $T_i=\{t_{i1}, t_{i2}\}$.

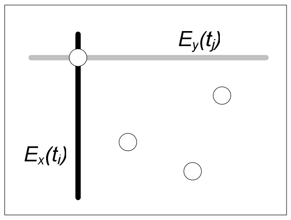


Figure 4. The implementation of the Line Scan EDF

The Line Scan EDF is defined as follows:

$$E_x(t) = \{p_k \mid k \in [0,l] \wedge p_k = (t * s, k)\}$$

for $t \in [0, l/s]$ and:

$$E_y(t) = \{p_k \mid k \in [0,l] \wedge p_k = (k, t * s - l)\} \tag{6}$$

for $t \in [l/s, 2l/s]$.

$$E(t) = E_x(t) \bigcup E_y(t)$$

We can localize a node by calculating the intersection of the two event lines, as shown in Figure 4. More formally:

$$L(T_i) = E(t_{i1}) \bigcap E(t_{i2}) \tag{7}$$

where $D(e(t_{i1}, p_i)) = true$, $D(e(t_{i2}, p_i)) = true$ for node i positioned at p_i.

3.4 Area Cover Event Distribution Function

Other devices, such as light projectors, can generate events that cover an area. This allows the implementation of the Area Cover EDF. The idea of Area Cover EDF is to partition the space A into multiple sections and assign a unique binary identifier, called code, to each section. Let's suppose that the localization is done within a plane ($A \subset R^2$). Each section S_k within A has a unique code k. The Area Cover EDF is then defined as follows:

$$BIT(k, j) = \begin{cases} \text{true}, & \text{if jth bit of k is 1} \\ \text{false}, & \text{if jth bit of k is 0} \end{cases} \tag{8}$$

$$E(t) = \{\, p \mid p \in S_k \wedge BIT(k,t) = true \,\}$$

and the corresponding localization algorithm is:

$$L(T_i) = \{p \mid p = COG(S_k) \wedge (BIT(k,t) = true \ if \ t \in T_i) \wedge (BIT(k,t) = false \ \ if \ t \in T\text{`}-T_i)\} \tag{9}$$

where $COG(S_k)$ denotes the center of gravity of S_k.

We illustrate the Area Cover EDF with a simple example. As shown in Figure 5, the plane A is divided in 16 sections. Each section S_k has a unique code k. The Spotlight device distributes the events according to these codes: at time j a section S_k is covered by an event (lit by light), if jth bit of k is 1. A node residing anywhere in the section S_k is localized at the center of gravity of that section. For example, nodes within section 1010 detect the events at time $T = \{1, 3\}$. At $t = 4$ the section where each node resides can be determined

A more accurate localization requires a finer partitioning of the plane, hence the number of bits in the code will increase. Considering the noise that is present in a real, outdoor environment, it is easy to observe that a relatively small error in detecting the correct bit pattern could result in a large localization error. Returning to the example shown in Figure 5, if a sensor node is located in the section with code 0000, and due to the noise, at time $t = 3$, it thinks it detected an event, it will incorrectly conclude that its code is 1000, and it positions itself two squares below its correct position. The localization accuracy can deteriorate even further, if multiple errors are present in the transmission of the code.

A natural solution to this problem is to use error-correcting codes, which greatly reduce the probability of an error, without paying the price of a re-transmission, or lengthening the transmission time too much. Several error correction schemes have been proposed in the past. Two of the most notable ones are the Hamming (7, 4) code and the Golay (23, 12) code. Both are perfect linear error correcting codes. The Hamming coding scheme can detect up to 2-bit errors and correct 1-bit errors. In the Hamming (7, 4) scheme, a message having 4 bits of data (e.g. *dddd*, where *d* is a data bit) is transmitted as a 7-bit word by adding 3 error control bits (e.g. *dddpdpp*, where *p* is a parity bit).

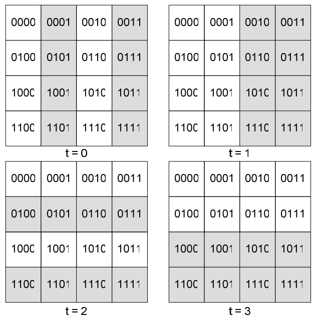


Figure 5. The steps of Area Cover EDF. The events cover the shaded areas.

The steps of the Area Cover technique, when using Hamming (7, 4) scheme are shown in Figure 6. Golay codes can detect up to 6-bit errors and correct up to 3-bit errors. Similar to Hamming (7, 4), Golay constructs a 23-bit codeword from 12-bit data. Golay codes have been used in satellite and spacecraft data transmission and are most suitable in cases where short codeword lengths are desirable.

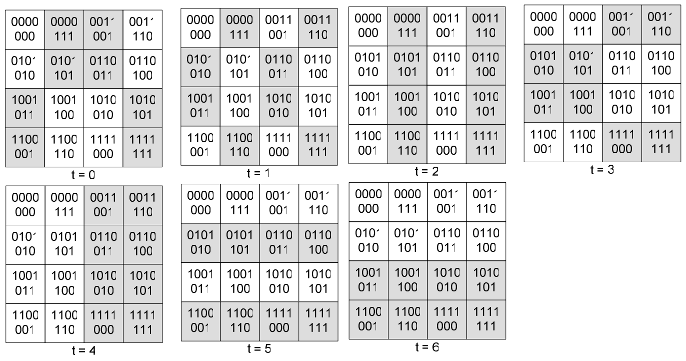


Figure 6. The steps of Area Cover EDF with Hamming (7, 4) ECC. The events cover the shaded areas.

Let's assume a 1-bit error probability of 0.01, and a 12-bit message that needs to be transmitted. The probability of a failed transmission is thus: 0.11, if no error detection and correction is used; 0.0061 for the Hamming scheme (i.e. more than 1-bit error); and 0.000076 for the Golay scheme (i.e. more than 3-bit errors). Golay is thus 80 times more robust that the Hamming scheme, which is 20 times more robust than the no error correction scheme.

Considering that a limited number of corrections is possible by any coding scheme, a natural question arises: can we minimize the localization error when there are errors that can not be corrected? This can be achieved by a clever placement of codes in the grid. As shown in Figure 7, the placement A, in the presence of a 1-bit error has a smaller average localization error when compared to the placement B. The objective of our code placement strategy is to reduce the total Euclidean distance between all pairs of codes with Hamming distances smaller than K, the largest number of expected 1-bit errors.

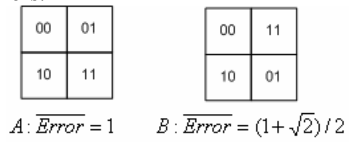

00
01
10
11
00
11
10
01
$A: \overline{Error} = 1$
$B: \overline{Error} = (1+\sqrt{2})/2$

Figure 7. Different code placement strategies

Formally, a placement is represented by a function P: $[0, l]^d \rightarrow C$, which assigns a code to every coordinate in the d-dimensional cube of size l (e.g., in the planar case, we place codes in a 2-dimensional grid). We denote by $d_E(i, j)$ the Euclidean distance and by $d_H(i, j)$ the Hamming distance between two codes i and j. In a noisy environment, $d_H(i,j)$ determines the crossover probability between the two codes. For the case of independent detections, the higher $d_H(i, j)$ is, the lower the crossover probability will be. The objective function is defined as follows:

$$\min\{ \sum_{d_H(i,j)\le K} d_E(i,j)\} \quad where\, i,j \in [0,l]^d \tag{10}$$

Equation 10 is a non-linear and non-convex programming problem. In general, it is analytically hard to obtain the global minimum. To overcome this, we propose a Greedy Placement method to obtain suboptimal results. In this method we initialize the 2-dimensional grid with codes. Then we swap the codes within the grid repeatedly, to minimize the objective function. For each swap, we greedily chose a pair of codes, which can reduce the objective function (Equation 10) the most. The proposed Greedy Placement method ends when no swap of codes can further minimize the objective function.

For evaluation, we compared the average localization error in the presence of K-bit error for two strategies: the proposed Greedy Placement and the Row-Major Placement (it places the codes consecutively in the array, in row-first order).

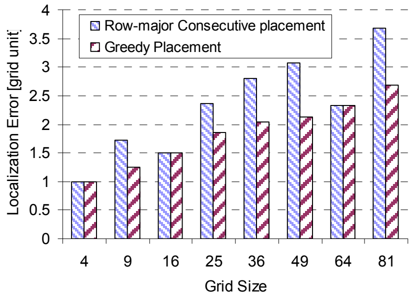


Figure 8. Localization error with code placement and no ECC

As Figure 8 shows, if no error detection/correction capability is present and 1-bit errors occur, then our Greedy Placement method can reduce the localization error by an average 23%, when compared to the Row-Major Placement. If error detection and correction schemes are used (e.g. Hamming (12, 8) and if 3-bit errors occur (K=3) then the Greedy Placement method reduces localization error by 12%, when compared to the Row-Major Placement, as shown in Figure 9. If K=1, then there is no benefit in using the Greedy Placement method, since the 1-bit error can be corrected by the Hamming scheme.

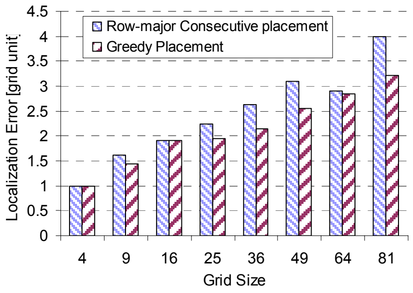


Figure 9. Localization error with code placement and Hamming ECC

3.5 Event Distribution Function Analysis

Although all three aforementioned techniques are able to localize the sensor nodes, they differ in the localization time, communication overhead and energy consumed by the Event Distribution Function (let's call it Event Overhead). Let's assume that all sensor nodes are located in a square with edge size D, and that the Spotlight device can generate N events (e.g. Point, Line and Area Cover events) every second and that the maximum tolerable localization error is r. Table 1 presents the execution cost comparison of the three different Spotlight techniques.

Table 1. Execution Cost Comparison

Criterion	Point Scan	Line Scan	Area Cover
Localization Time	$(D^2/r^2)/N$	$(2D/r)/N$	$log_r D/N$
# Detections	1	2	$log_r D$
# Time Stamps	1	2	$log_r D$
Event Overhead	D^2	$2D^2$	$D^2 log_r D/2$

Table 1 indicates that the Event Overhead for the Point Scan method is the smallest - it requires a one-time coverage of the area, hence the D^2. However the Point Scan takes a much longer time than the Area Cover technique, which finishes in $log_r D$ seconds. The Line Scan method trades the Event Overhead well with the localization time. By doubling the Event Overhead, the Line Scan method takes only $r/2D$ percentage of time to complete, when compared with the Point Scan method. From Table 1, it can be observed that the execution costs do not depend on the number of sensor nodes to be localized.

It is important to remark the ratio Event Overhead per unit time, which is indicative of the power requirement for the Spotlight device. This ratio is constant for the Point Scan (r^2*N) while it grows linearly with area, for the Area Cover ($D^2*N/2$). If the deployment area is very large, the use of the Area Cover EDF is prohibitively expensive, if not impossible. For practical purposes, the Area Cover is a viable solution for small to medium size networks, while the Line Scan works well for large networks. We discuss the implications of the power requirement for the Spotlight device, and offer a hybrid solution in Section 6.

3.6 Localization Error Analysis

The accuracy of localization with the Spotlight technique depends on many aspects. The major factors that were considered during the implementation of the system are discussed below:

- Time Synchronization: the Spotlight system exchanges time stamps between sensor nodes and the Spotlight device. It is necessary for the system to reach consensus on global time through synchronization. Due to the uncertainty in hardware processing and wireless communication, we can only confine such errors within certain bounds (e.g. one jiffy). An imprecise input to the Localization Function *L(T)* leads to an error in node localization.
- Uncertainty in Detection: the sampling rate of the sensor nodes is finite, consequently, there will be an unpredictable delay between the time when an event is truly present and when the sensor node detects it. Lower sampling rates will generate larger localizations errors.
- Size of the Event: the events distributed by the Spotlight device can not be infinitely small. If a node detects one event, it is hard for it to estimate the exact location of itself within the event.
- Realization of Event Distribution Function: EDF defines locations of events at time t. Due to the limited accuracy (e.g. mechanical imprecision), a Spotlight device might generate events which locate differently from where these events are supposed to be.

It is important to remark that the localization error is independent of the number of sensor nodes in the network. This independence, as well as the aforementioned independence of the execution cost, indicate the very good scalability properties (with the number of sensor nodes, but not with the area of deployment) that the Spotlight system possesses.

4. SYSTEM IMPLEMENTATION

For our performance evaluation we implemented two Spotlight systems. Using these two implementations we were able to investigate the full spectrum of Event Distribution techniques, proposed in Section 3, at a reduced “one time” cost (less than $1,000).

The first implementation, called μSpotlight, had a short range (10-20 meters), however its capability of generating the entire spectrum of EDFs made it very useful. We used this implementation mainly to investigate the capabilities of the Spotlight system and tune its performance. It was not intended to represent the full solution, but only a scaled down version of the system.

The second implementation, the Spotlight system, had a much longer range (as far as 6500m), but it was limited in the types of EDFs that it can generate. The goal of this implementation was to show how the Spotlight system works in a real, outdoor environment, and show correlations with the experimental results obtained from the μSpotlight system implementation.

In the remaining part of this section, we describe how we implemented the three components (Event Distribution, Event Detection and Localization functions) of the Spotlight architecture, and the time synchronization protocol, a key component of our system.

4.1 μSpotlight System

The first system we built, called μSpotlight, used as the Spotlight device, an Infocus LD530 projector connected to an IBM Thinkpad laptop. The system is shown in Figure 10.

The Event Distribution Function was implemented as a Java GUI. Due to the stringent timing requirements and the delay caused by the buffering in the windowing system of a PC, we used the Full-Screen Exclusive Mode API provided by Java2. This allowed us to bypass the windowing system and more precisely estimate the time when an event is displayed by the projector, hence a higher accuracy of timestamps of events. Because of the 50Hz refresh rate of our projector, there was still an uncertainty in the time stamping of the events of 20msec. We explored the possibility of using and modifying the Linux kernel to expose the vertical synch (VSYNCH) interrupt, generated by the displaying device after each screen refresh, out of the kernel mode. The performance evaluation results showed, however, that this level of accuracy was not needed.

The sensor nodes that we used were Berkeley Mica2 motes equipped with MTS310 multi-sensor boards from Crossbow. This sensor board contains a CdSe photo sensor which can detect the light from the projector.

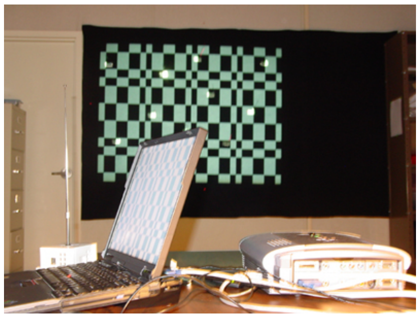


Figure 10. μSpotlight system implementation

With this implementation of the Spotlight system, we were able to generate Point, Line and Area Scan events.

4.2 Spotlight System

The second Spotlight system we built used, as the Spotlight device, diode lasers, a computerized telescope mount (Celestron CG-5GT, shown in Figure 11), and an IBM Thinkpad laptop. The laptop was connected, through RS232 interfaces, to the telescope mount and to one XSM600CA [7] mote, acting as a base station.

The diode lasers we used ranged in power from 7mW to 35mW. They emitted at 650nm, close to the point of highest sensitivity for CdSe photosensor. The diode lasers were equipped with lenses that allowed us to control the divergence of the beam.

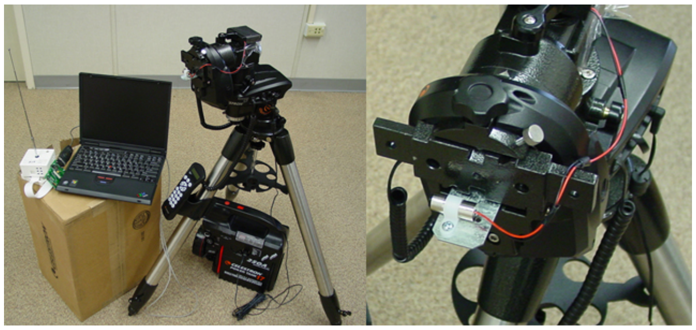


Figure 11. Spotlight system implementation

The telescope mount has worm gears for a smooth motion and high precision angular measurements. The two angular measures that we used were the, so called, Alt (from Altitude) and Az (from Azimuth). In astronomy, the Altitude of a celestial object is its angular distance above or below the celestial horizon, and the Azimuth is the angular distance of an object eastwards of the meridian, along the horizon.

The laptop computer, through a Java GUI, controls the motion of the telescope mount, orienting it such that a full Point Scan of an area is performed, similar to the one described in Figure 3(b). For each turning point i, the 3-tuple (Alt_i and Az_i angles and the timestamp t_i) is recorded. The Spotlight system uses the timestamp received from a sensor node j, to obtain the angular measures Alt_j and Az_j for its location.

For the sensor nodes, we used XSM motes, mainly because of their longer communication range. The XSM mote has the photo sensor embedded in its main board. We had to make minor adjustments to the plastic housing, in order to expose the photo sensor to the outside. The same mote code, written in nesC, for TinyOS, was used for both μSpotlight and Spotlight system implementations.

4.3 Event Detection Function *D(t)*

The Event Detection Function aims to detect the beginning of an event and record the time when the event was observed. We implemented a very simple detection function based on the observed maximum value. An event i will be time stamped with time t_i, if the reading from the photo sensor d_{ti}, fulfills the condition:

$$d_{\max} + \Delta < d_{t_i}$$

where d_{max} is the maximum value reported by the photo sensor before t_i and Δ is a constant which ensures that the first large detection gives the timestamp of the event (i.e. small variations around the first large signal are not considered). Hence Δ guarantees that only sharp changes in the detected value generate an observed event.

4.4 Localization Function *L(T)*

The Localization Function is implemented in the Java GUI. It matches the timestamps created by the Event Distribution Function with those reported by the sensor nodes.

The Localization Function for the Point Scan EDF has as input a time sequence $T_i = \{t_1\}$, as reported by node i. The function performs a simple search for the event with a timestamp closest to t_1. If t_1 is constrained by:

$$t_{e_n} < t_1 < t_{e_{n+1}}$$

where e_n and e_{n+1} are two consecutive events, then the obtained location for node i is:

$$x = x_{e_{n+1}}, y = y_{e_{n+1}}$$

The case for the Line Scan is treated similarly. The input to the Localization Function is the time sequence $T_i = \{t_1, t_2\}$ as reported by node i. If the reported timestamps are constrained by:

$$t_{e_n} < t_1 < t_{e_{n+1}} \text{, and } t_{e_m} < t_2 < t_{e_{m+1}}$$

where e_n and e_{n+1} are two consecutive events on the horizontal scan and e_m and e_{m+1} are two consecutive events on vertical scan, then the inferred location for node i is:

$$x = x_{e_{n+1}}, y = y_{e_{m+1}}$$

The Localization Function for the Area Cover EDF has as input a timestamp set $T_i=\{t_{i1}, t_{i2}, ..., t_{in}\}$ of the n events, detected by node i. We recall the notation for the set of m timestamps of events generated by the Spotlight device, $T'=\{t_1', t_2', ..., t_m'\}$. A code $d_i=d_{i1}d_{i2}...d_{im}$ is then constructed for each node i, such that $d_{ij}=1$ if $t_j' \in T_i$ and $d_{ij}=0$ if $t_j' \notin T_i$. The function performs a search for an event with an identical code. If the following condition is true:

$$d_i = d_{e_n}$$

where e_n is an event with code d_{en}, then the inferred location for node i is:

$$x = x_{e_n}, y = y_{e_n}$$

4.5 Time Synchronization

The time synchronization in the Spotlight system consists of two parts:

- Synchronization between sensor nodes: This is achieved through the Flooding Time Synchronization Protocol [18]. In this protocol, synchronized nodes (the root node is the only synchronized node at the beginning) send time synchronization message to unsynchronized nodes. The sender puts the time stamp into the synchronization message right before the bytes containing the time stamp are transmitted. Once a receiver gets the message, it follows the sender's time and performs the necessary calculations to compensate for the clock drift.

- Synchronization between the sensor nodes and the Spotlight device: We implemented this part through a two-way handshaking between the Spotlight device and one node, used as the base station. The sensor node is attached to the Spotlight device through a serial interface.

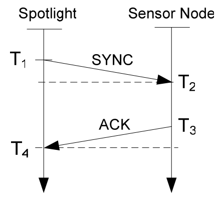


Figure 12. Two-way synchronization

As shown in Figure 12, let's assume that the Spotlight device sends a synchronization message (SYNC) at local time T_1, the sensor node receives it at its local time T_2 and acknowledges it at local time T_3 (both T_2 and T_3 are sent back through ACK). After the Spotlight device receives the ACK, at its local time T_4, the time synchronization can be achieved as follows:

$$Offset = \frac{(T_2 - T_1) + (T_3 - T_4)}{2} \qquad (11)$$

$$T_{global} = T_{node} = T_{spotlight} + Offset$$

We note that Equation 11 assumes that the one trip delays are the same in both directions. In practice this does not hold well enough. To improve the performance, we separate the handshaking process from the timestamp exchanges. The handshaking is done fast, through a 2 byte exchange between the Spotlight device and the sensor node (the timestamps are still recorded, but not sent). After this fast handshaking, the recorded time stamps are exchanged. The result indicates that this approach can significantly improve the accuracy of time synchronization.

5. PERFORMANCE EVALUATION

In this section we present the performance evaluation of the Spotlight systems when using the three event distribution functions, i.e. Point Scan, Line Scan and Area Cover, described in Section 3.

For the μSpotlight system we used 10 Mica2 motes. The sensor nodes were attached to a vertically positioned Veltex board. By projecting the light to the sensor nodes, we are able to generate well controlled Point, Line and Area events. The Spotlight device was able to generate events, i.e. project light patterns, covering an area of approximate size 180x140cm^2. The screen resolution for the projector was 1024x768, and the movement of the Point Scan and Line Scan techniques was done through increments (in the appropriate direction) of 10 pixels between events.

Each experimental point was obtained from 10 successive runs of the localization procedure. Each set of 10 runs was preceded by a calibration phase, aimed at estimating the total delays (between the Spotlight device and each sensor node) in detecting an event. During the calibration, we created an event covering the entire sensor field (illuminated the entire area). The timestamp reported by each sensor node, in conjunction with the timestamp created by the Spotlight device were used to obtain the time offset, for each sensor node. More sophisticated calibration procedures have been reported previously [35]. In addition to the time offset, we added a manually configurable parameter, called bias. It was used to best estimate the center of an event.

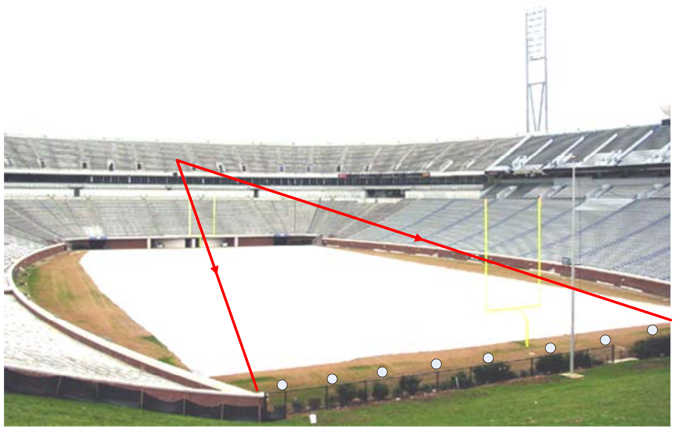


Figure 13. Deployment site for the Spotlight system

For the Spotlight system evaluation, we deployed 10 XSM motes in a football field. The site is shown in Figure 13 (laser beams are depicted with red arrows and sensor nodes with white dots). Two sets of experiments were run, with the Spotlight device positioned at 46m and at 170m from the sensor field. The sensor nodes were aligned and the Spotlight device executed a Point Scan. The localization system computed the coordinates of the sensor nodes, and the Spotlight device was oriented, through a GoTo command sent to the telescope mount, towards the computed location. In the initial stages of the experiments, we manually measured the localization error.

For our experimental evaluation, the metrics of interest were as follows:

- Localization error, defined as the distance, between the real location and the one obtained from the Spotlight system.

- Localization duration, defined as the time span between the first and last event.

- Localization range, defined as the maximum distance between the Spotlight device and the sensor nodes.

- A Localization Cost function *Cost*:{{localization accuracy}, {localization duration}} → [0,1] quantifies the trade-off between the accuracy in localization and the localization duration. The objective is to minimize the Localization Cost function. By denoting with e_i the localization error for the i^{th} scenario, with d_i the localization duration for the i^{th} scenario, with max(e) the maximum localization error, with max(d) the maximum localization duration, and with α the importance factor, the Localization Cost function is formally defined as:

$$Cost(e_i, d_i) = \alpha * \frac{e_i}{\max(e)} + (1-\alpha) * \frac{d_i}{\max(d)}$$

- Localization Bias. This metric is used to investigate the effectiveness of the calibration procedure. If, for example, all computed locations have a bias in the west direction, a calibration factor can be used to compensate for the difference.

The parameters that we varied during the performance evaluation of our system were: the type of scanning (Point, Line and Area), the size of the event, the duration of the event (for Area Cover), the scanning speed, the power of the laser and the distance between the Spotlight device and sensor field, to estimate the range of the system.

5.1 Point Scan - μSpotlight system

In this experiment, we investigated how the size of the event and the scanning speed affect the localization error. Figure 14 shows the mean localization errors with their standard deviations.

It can be observed, that while the scanning speed, varying between 35cm/sec and 87cm/sec has a minor influence on the localization accuracy, the size of the event has a dramatic effect.

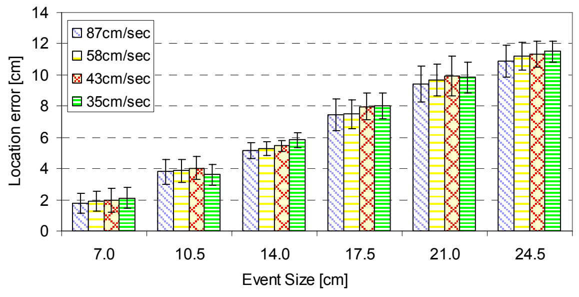


Figure 14. Localization Error vs. Event Size for the Point Scan EDF

The obtained localization error varied from as little as 2cm to over 11cm for the largest event. This dependence can be explained by our Event Detection algorithm: the first detection above a threshold gave the timestamp for the event.

The duration of the localization scheme is shown in Figure 15. The dependency of the localization duration on the size of the event and scanning speed is natural. A bigger event allows a reduction in the total duration of up to 70%. The localization duration is directly proportional to the scanning speed, as expected, and depicted in Figure 15.

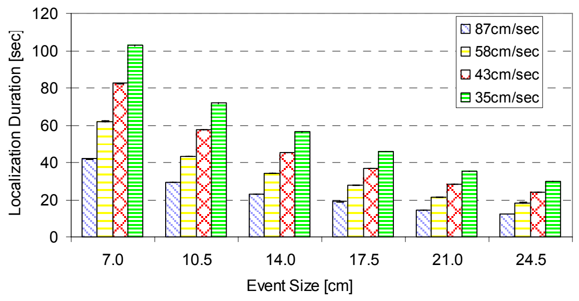


Figure 15. Localization Duration vs. Event Size for the Point Scan EDF

An interesting trade-off is between the localization accuracy (usually the most important factor), and the localization time (important in environments where stealthiness is paramount). Figure 16 shows the Localization Cost function, for α = 0.5 (accuracy and duration are equally important).

As shown in Figure 16, it can be observed that an event size of approximately 10-15cm (depending on the scanning speed) minimizes our Cost function. For α = 1, the same graph would be a monotonically increasing function, while for α = 0, it would be monotonically decreasing function.

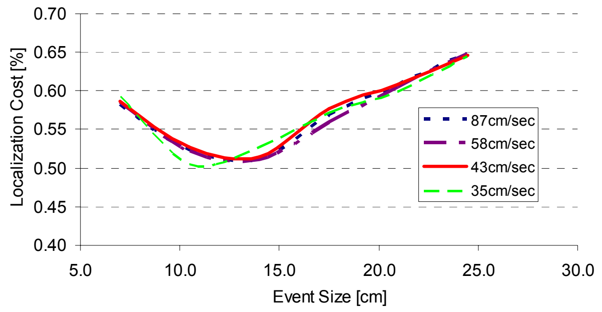


Figure 16. Localization Cost vs. Event Size for the Point Scan EDF

5.2 Line Scan - μSpotlight system

In a similar manner to the Point Scan EDF, for the Line Scan EDF we were interested in the dependency of the localization error and duration on the size of the event and scanning speed.

We represent in Figure 17 the localization error for different event sizes. It is interesting to observe the dependency (concave shape) of the localization error vs. the event size. Moreover, a question that should arise is why the same dependency was not observed in the case of Point Scan EDF.

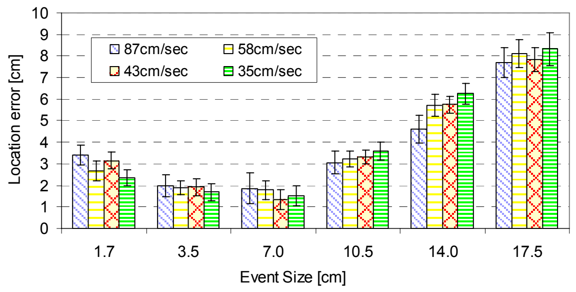


Figure 17. Localization Error vs. Event Size for the Line Scan EDF

The explanation for this concave dependency is the existence of a bias in location estimation. As a reminder, a bias factor was introduced in order to best estimate the central point of events that have a large size. What Figure 17 shows is the fact that the bias factor was optimal for an event size of approximately 7cm. For events smaller and larger than this, the bias factor was too large, and too small, respectively. Thus, it introduced biased errors in the position estimation.

The reason why we did not observe the same dependency in the case of the Point Scan EDF was that we did not experiment with event sizes below 7cm, due to the long time it would have taken to scan the entire field with events as small as 1.7cm.

The results for the localization duration as a function of the size of the event are shown in Figure 18. As shown, the localization duration is directly proportional to the scanning speed. The size of the event has a smaller influence on the localization duration. One can remark the average localization duration of about 10sec, much shorter then the duration obtained in the Point Scan experiment.

The Localization Cost function dependency on the event size and scanning speed, for α=0.5, is shown in Figure 19. The dependency on the scanning speed is very small (the Cost Function achieves a minimum in the same 4-6cm range). It is interesting to note that this 4-6cm optimal event size is smaller than the one observed in the case of Point Scan EDF. The explanation for this is that the smaller localization duration observed in the Line Scan EDF, allowed a shift (towards smaller event sizes) in the total Localization Cost Function.

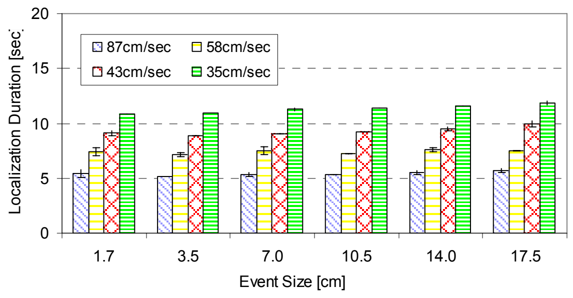


Figure 18. Localization Duration vs. Event Size for the Line Scan EDF

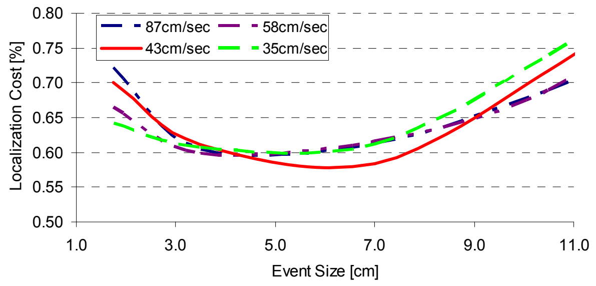


Figure 19. Cost Function vs. Event Size for the Line Scan EDF

During our experiments with the Line Scan EDF, we observed evidence of a bias in location estimation. The estimated locations for all sensor nodes exhibited different biases, for different event sizes. For example, for an event size of 17.5cm, the estimated location for sensor nodes was to the upper-left size of the actual location. This was equivalent to an "early" detection, since our scanning was done from left to right and from top to bottom. The scanning speed did not influence the bias.

In order to better understand the observed phenomena, we analyzed our data. Figure 20 shows the bias in the horizontal direction, for different event sizes (the vertical bias was almost identical, and we omit it, due to space constraints).

From Figure 20, one can observe that the smallest observed bias, and hence the most accurate positioning, was for an event of size 7cm. These results are consistent with the observed localization error, shown in Figure 17.

We also adjusted the measured localization error (shown in Figure 17) for the observed bias (shown in Figure 20). The results of an ideal case of Spotlight Localization system with Line Scan EDF are shown in Figure 21. The errors are remarkably small, varying between 0.1cm and 0.8cm, with a general trend of higher localization errors for larger event sizes.

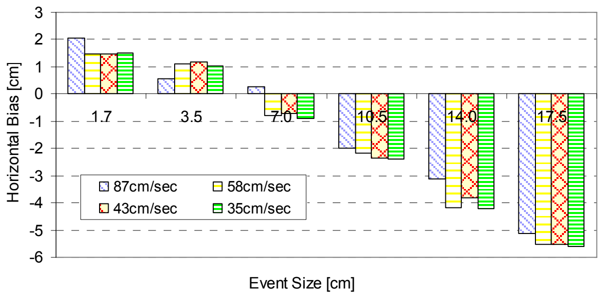


Figure 20. Position Estimation Bias for the Line Scan EDF

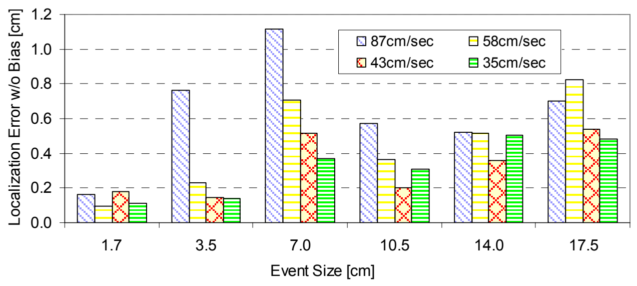


Figure 21. Position Estimation w/o Bias (ideal), for the Line Scan EDF

5.3 Area Cover - μSpotlight system

In this experiment, we investigated how the number of bits used to quantify the entire sensor field, affected the localization accuracy. In our first experiment we did not use error correcting codes. The results are shown in Figure 22.

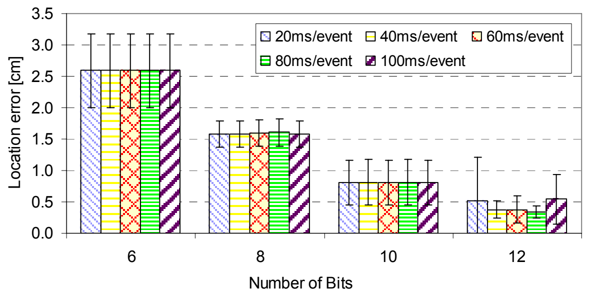


Figure 22. Localization Error vs. Event Size for the Area Cover EDF

One can observe a remarkable accuracy, with localization error on the order of 0.3-0.6cm. What is important to observe is the variance in the localization error. In the scenario where 12 bits were used, while the average error was very small, there were a couple of cases, where an incorrect event detection generated a larger than expected error. An example of how this error can occur was described in Section 3.4. The experimental results, presented in Figure 22, emphasize the need for error correction of the bit patterns observed and reported by the sensor nodes.

The localization duration results are shown in Figure 23. It can be observed that the duration is directly proportional with the number of bits used, with total durations ranging from 3sec, for the least accurate method, to 6-7sec for the most accurate. The duration of an event had a small influence on the total localization time, when considering the same scenario (same number of bits for the code).

The Cost Function dependency on the number of bits in the code, for α=0.5, is shown in Figure 24. Generally, since the localization duration for the Area Scan can be extremely small, a higher accuracy in the localization is desired. While the Cost function achieves a minimum when 10 bits are used, we attribute the slight increase observed when 12 bits were used to the two 12-bit scenarios where larger than the expected errors were observed, namely 6-7mm (as shown in Figure 22).

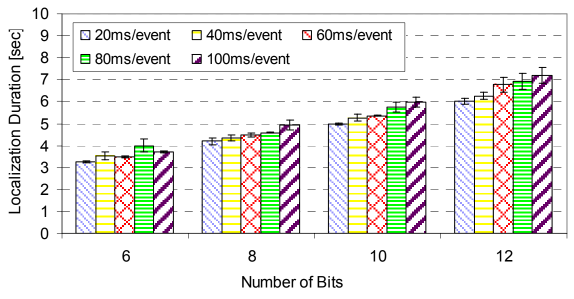


Figure 23. Localization Duration vs. Event Size for the Area Cover EDF

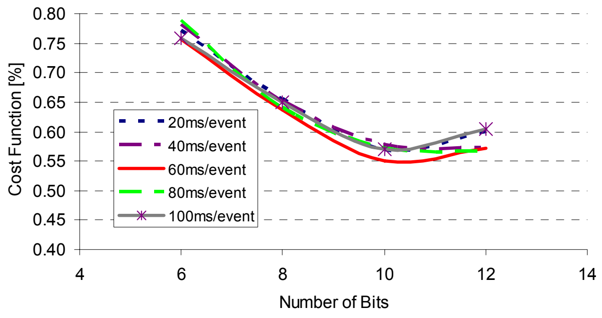


Figure 24. Cost Function vs. Event Size for the Area Cover EDF

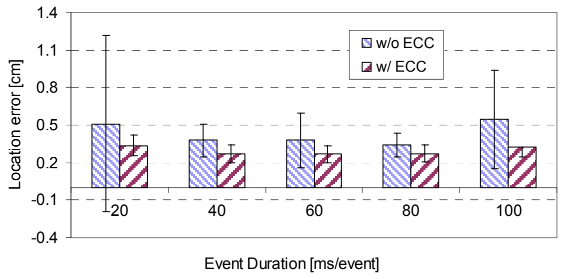


Figure 25. Localization Error w/ and w/o Error Correction

The two problematic scenarios (shown in Figure 22, where for 12-bit codes we observed errors larger than the event size, due to errors in detection) were further explored by using error correction codes. As described in Section 3.3, we implemented an extended Golay (24, 12) error correction mechanism in our location estimation algorithm.

The experimental results are depicted in Figure 25, and show a consistent accuracy. The scenario without error correction codes, is simply the same 12-bit code scenario, shown in Figure 22. We only investigated the 12-bit scenario, due to its match with the 12-bit data required by the Golay encoding scheme (extended Golay producing 24-bit codewords).

5.4 Point Scan - Spotlight system

In this section we describe the experiments performed at a football stadium, using our Spotlight system. The hardware that we had available allowed us to evaluate the Point Scan technique of the Spotlight system. In our evaluation, we were interested to see the performance of the system at different ranges. Figures 26 and 27 show the localization error versus the event size at two different ranges: 46m and 170m.

Figure 26 shows a remarkable accuracy in localization. The errors are in the centimeter range. Our initial, manual measurements of the localization error were most of the time difficult to make, since the spot of the laser was almost perfectly covering the XSM mote. We are able to achieve localization errors of a few centimeters, which only range-based localization schemes are able to achieve [25]. The observed dependency on the size of the event is similar to the one observed in the μSpotlight system evaluation, and shown in Figure 14. This proved that the μSpotlight system is a viable alternative for investigating complex EDFs, without incurring the costs for the necessary hardware.

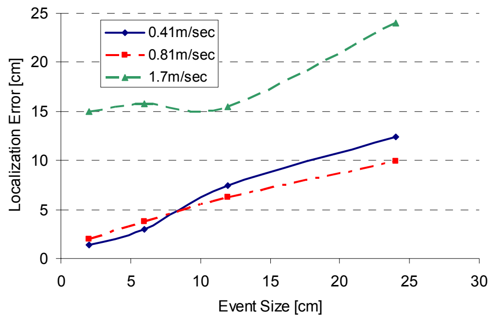


Figure 26. Localization Error vs. Event Size for Spotlight system at 46m

In the experiments performed over a much longer distance between the Spotlight device and sensor network, the average localization error remains very small. Localization errors of 5-10cm were measured, as Figure 27 shows. We were simply amazed by the accuracy that the system is capable of, when considering that the Spotlight system operated over the length of a football stadium. Throughout our experimentation with the Spotlight system, we have observed localization errors that were simply offsets of real locations. Since the same phenomenon was observed when experimenting with the μSpotlight system, we believe that with auto-calibration, the localization error can be further reduced.

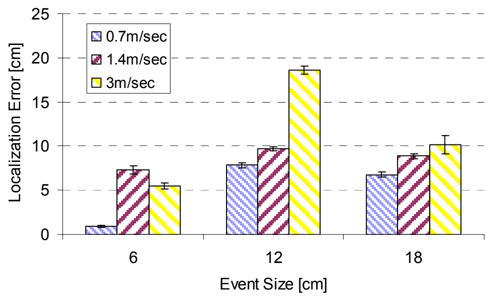


Figure 27. Localization Error vs. Event Size for Spotlight system at 170m

The time required for localization using the Spotlight system with a Point Scan EDF, is given by: $t=(L*l)/(s*E_s)$, where L and l are the dimensions of the sensor network field, s is the scanning speed, and E_s is the size of the event. Figure 28 shows the time for localizing a sensor network deployed in an area of size of a football field using the Spotlight system. Here we ignore the message propagation time, from the sensor nodes to the Spotlight device.

From Figure 28 it can be observed that the very small localization errors are prohibitively expensive in the case of the Point Scan. When localization errors of up to 1m are tolerable, localization duration can be as low as 4 minutes. Localization durations of 5-10 minutes, and localization errors of 1m are currently state of art in the realm of range-free localization schemes. And these results are achieved by using the Point Scan scheme, which required the highest Localization Time, as it was shown in Table 1.

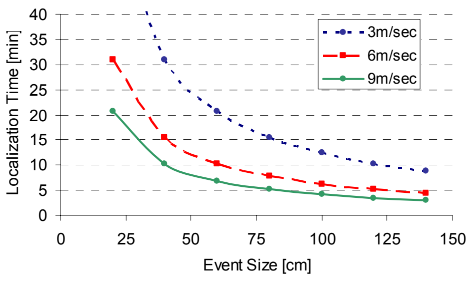


Figure 28. Localization Time vs. Event Size for Spotlight system

One important characteristic of the Spotlight system is its range. The two most important factors are the sensitivity of the photosensor and the power of the Spotlight source. We were interested in measuring the range of our Spotlight system, considering our capabilities (MTS310 sensor board and inexpensive, $12-$85, diode laser). As a result, we measured the intensity of the laser beam, having the same focus, at different distances. The results are shown in Figure 29.

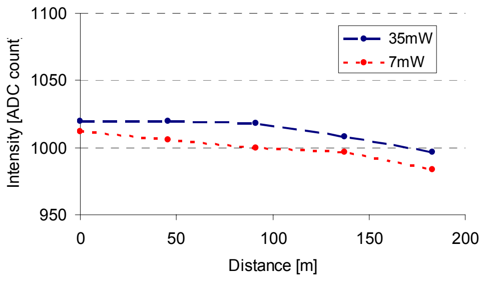


Figure 29. Localization Range for the Spotlight system

From Figure 29, it can be observed that only a minor decrease in the intensity occurs, due to absorption and possibly our imperfect focusing of the laser beam. A linear fit of the experimental data shows that distances of up to 6500m can be achieved. While we do not expect atmospheric conditions, over large distances, to be similar to our 200m evaluation, there is strong evidence that distances (i.e. altitude) of 1000-2000m can easily be achieved. The angle between the laser beam and the vertical should be minimized (less than 45°), as it reduces the difference between the beam cross-section (event size) and the actual projection of the beam on the ground.

In a similar manner, we were interested in finding out the maximum size of an event, that can be generated by a COTS laser and that is detectable by the existing photosensor. For this, we varied the divergence of the laser beam and measured the light intensity, as given by the ADC count. The results are shown in Figure 30. It can be observed that for the less powerful laser, an event size of 1.5m is the limit. For the more powerful laser, the event size can be as high as 4m.

Through our extensive performance evaluation results, we have shown that the Spotlight system is a feasible, highly accurate, low cost solution for localization of wireless sensor networks. From our experience with sources of laser radiation, we believe that for small and medium size sensor network deployments, in areas of less than 20,000m^2, the Area Cover scheme is a viable solution. For large size sensor network deployments, the Line Scan, or an incremental use of the Area Cover are very good options.

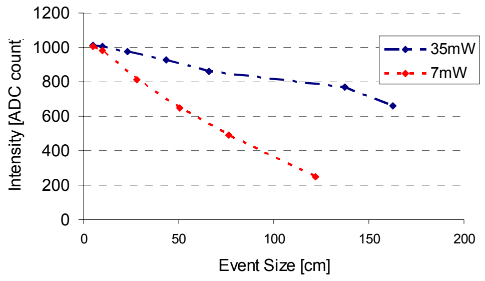


Figure 30. Detectable Event Sizes that can be generated by COTS lasers

6. OPTIMIZATIONS/LESSONS LEARNED

6.1 Distributed Spotlight System

The proposed design and the implementation of the Spotlight system can be considered centralized, due to the gathering of the sensor data and the execution of the Localization Function *L(t)* by the Spotlight device. We show that this design can easily be transformed into a distributed one, by offering two solutions.

One idea is to disseminate in the network, information about the path of events, generated by the EDF (similar to an equation, describing a path), and let the sensor nodes execute the Localization Function. For example, in the Line Scan scenario, if the starting and ending points for the horizontal and vertical scans, and the times they were reached, are propagated in the network, then any sensor in the network can obtain its location (assuming a constant scanning speed).

A second solution is to use anchor nodes which know their positions. In the case of Line Scan, if three anchors are present, after detecting the presence of the two events, the anchors flood the network with their locations and times of detection. Using the same simple formulas as in the previous scheme, all sensor nodes can infer their positions.

6.2 Localization Overhead Reduction

Another requirement imposed by the Spotlight system design, is the use of a time synchronization protocol between the Spotlight device and the sensor network. Relaxing this requirement and imposing only a time synchronization protocol among sensor nodes is a very desirable objective. The idea is to use the knowledge that the Spotlight device has about the speed with which the scanning of the sensor field takes place. If the scanning speed is constant (let's call it *s*), then the time difference (let's call it *Δt*) between the event detections of two sensor nodes is, in fact, an accurate measure of the range between them: $d=s*\Delta t$. Hence, the Spotlight system can be used for accurate ranging of the distance between any pair of sensor nodes. An important observation is that this ranging technique does not suffer from limitations of others: small range and directionality for ultrasound, or irregularity, fading and multipath for Received Signal Strength Indicator (RSSI). After the ranges between nodes have been determined (either in a centralized or distributed manner) graph embedding algorithms can be used for a realization of a rigid graph, describing the sensor network topology.

6.3 Dynamic Event Distribution Function *E(t)*

Another system optimization is for environments where the sensor node density is not uniform. One disadvantage of the Line Scan technique, when compared to the Area Cover, is the localization time.

An idea is to use two scans: one which uses a large event size (hence larger localization errors), followed by a second scan in which the event size changes dynamically. The first scan is used for identifying the areas with a higher density of sensor nodes. The second scan uses a larger event in areas where the sensor node density is low and a smaller event in areas with a higher sensor node density.

A dynamic EDF can also be used when it is very difficult to meet the power requirements for the Spotlight device (imposed by the use of the Area Cover scheme in a very large area). In this scenario, a hybrid scheme can be used: the first scan (Point Scan) is performed quickly, with a very large event size, and it is meant to identify, roughly, the location of the sensor network. Subsequent Area Cover scans will be executed on smaller portions of the network, until the entire deployment area is localized.

6.4 Stealthiness

Our implementation of the Spotlight system used visible light for creating events. Using the system during the daylight or in a room well lit, poses challenges due to the solar or fluorescent lamp radiation, which generate a strong background noise. The alternative, which we used in our performance evaluations, was to use the system in a dark room (μSpotlight system) or during the night (Spotlight system). While using the Spotlight system during the night is a good solution for environments where stealthiness is not important (e.g. environmental sciences) for others (e.g. military applications), divulging the presence and location of a sensor field, could seriously compromise the efficacy of the system.

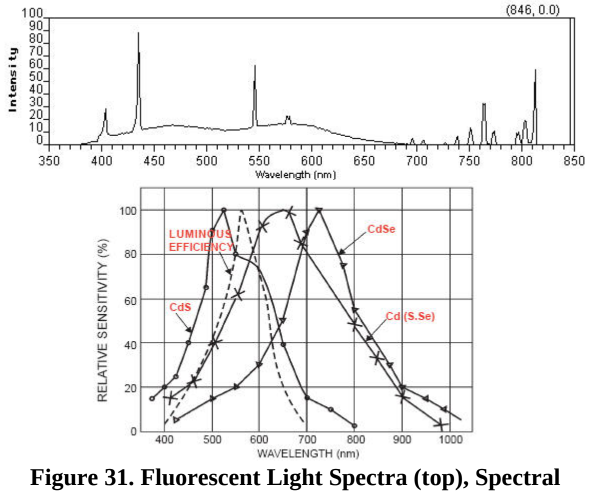


Figure 31. Fluorescent Light Spectra (top), Spectral Response for CdSe cells (bottom)

A solution to this problem, which we experimented with in the μSpotlight system, was to use an optical filter on top of the light sensor. The spectral response of a CdSe photo sensor spans almost the entire visible domain [37], with a peak at about 700nm (Figure 31-bottom). As shown in Figure 31-top, the fluorescent light has no significant components above 700nm. Hence, a simple red filter (Schott RG-630), which transmits all light with wavelength approximately above 630nm, coupled with an Event Distribution Function that generates events with wavelengths above the same threshold, would allow the use of the system when a fluorescent light is present.

A solution for the Spotlight system to be stealthy at night, is to use a source of infra-red radiation (i.e. laser) emitting in the range [750, 1000]nm. For a daylight use of the Spotlight system, the challenge is to overcome the strong background of the natural light. A solution we are considering is the use of a narrow-band optical filter, centered at the wavelength of the laser radiation. The feasibility and the cost-effectiveness of this solution remain to be proven.

6.5 Network Deployed in Unknown Terrain

A further generalization is when the map of the terrain where the sensor network was deployed is unknown. While this is highly unlikely for many civil applications of wireless sensor network technologies, it is not difficult to imagine military applications where the sensor network is deployed in a hostile and unknown terrain. A solution to this problem is a system that uses two Spotlight devices, or equivalently, the use of the same device from two distinct positions, executing, from each of them, a complete localization procedure. In this scheme, the position of the sensor node is uniquely determined by the intersection of the two location directions obtained by the system. The relative localization (for each pair of Spotlight devices) will require an accurate knowledge of the 3 translation and 3 rigid-body rotation parameters for Spotlight's position and orientation (as mentioned in Section 3).

This generalization is also applicable to scenarios where, due to terrain variations, there is no single aerial point with a direct line of sight to all sensor nodes, e.g. hilly terrain. By executing the localization procedure from different aerial points, the probability of establishing a line of sight with all the nodes, increases. For some military scenarios [1] [12], where open terrain is prevalent, the existence of a line of sight is not a limiting factor. In light of this, the Spotlight system can not be used in forests or indoor environments.

7. CONCLUSIONS AND FUTURE WORK

In this paper we presented the design, implementation and evaluation of a localization system for wireless sensor networks, called Spotlight. Our localization solution does not require any additional hardware for the sensor nodes, other than what already exists. All the complexity of the system is encapsulated into a single Spotlight device. Our localization system is reusable, i.e. the costs can be amortized through several deployments, and its performance is not affected by the number of sensor nodes in the network. Our experimental results, obtained from a real system deployed outdoors, show that the localization error is less than 20cm. This error is currently state of art, even for range-based localization systems and it is 75% smaller than the error obtained when using GPS devices or when the manual deployment of sensor nodes is a feasible option [31].

As future work, we would like to explore the self-calibration and self-tuning of the Spotlight system. The accuracy of the system can be further improved if the distribution of the event, instead of a single timestamp, is reported. A generalization could be obtained by reformulating the problem as an angular estimation problem that provides the building blocks for more general localization techniques.

8. ACKNOWLEDGEMENTS

This work was supported by the DARPA IXO office, under the NEST project (grant number F336616-01-C-1905) and by the NSF grant CCR-0098269. We would like to thank S. Cornwell for allowing us to run experiments in the stadium, M. Klopf for his assistance with optics, and anonymous reviewers and our shepherd, Koen Langendoen, for their valuable feedback.

9. REFERENCES

[1] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulharni, U. Arumugam, M. Nesterenko, A. Vora, M. Miyashita, "A Line in the Sand: A Wireless Sensor Network for Target Detection, Classification and Tracking", in Computer Networks 46(5), 2004.

[2] P. Bahl, V.N. Padmanabhan, "RADAR: An In-Building RF-based User Location and Tracking System", in Proceedings of Infocom, 2000

[3] M. Broxton, J. Lifton, J. Paradiso, "Localizing a Sensor Network via Collaborative Processing of Global Stimuli", in Proceedings of EWSN, 2005.

[4] N. Bulusu, J. Heidemann, D. Estrin, "GPS-less Low Cost Outdoor Localization for Very Small Devices", in IEEE Personal Communications Magazine, 2000.

[5] P. Corke, R. Peterson, D. Rus, "Networked Robots: Flying Robot Navigation Using a Sensor Net", in ISSR, 2003.

[6] L. Doherty, L. E. Ghaoui, K. Pister, "Convex Position Estimation in Wireless Sensor Networks", in Proceedings of Infocom, 2001

[7] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, D. Culler, "Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events", in Proceedings of IPSN, 2005.

[8] E. Elnahrawy, X. Li, R. Martin, "The Limits of Localization using RSSI", in Proceedings of SECON, 2004.

[9] D. Fox, W. Burgard, S. Thrun, "Markov Localization for Mobile Robots in Dynamic Environments", in Journal of Artificial Intelligence Research, 1999.

[10] D. Fox, W. Burgard, F. Dellaert, S. Thrun, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots", in Conference on Artificial Intelligence, 2000.

[11] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, S. Wicker, "Complex Behaviour at Scale: An Experimental Study of Low Power Wireless Sensor Networks", in Technical Report, UCLA-TR 01-0013, 2001.

[12] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, B. Krogh, "An Energy-Efficient Surveillance System Using Wireless Sensor Networks", in Proceedings of Mobisys, 2004.

[13] T. He, C. Huang, B. Blum, J. A. Stankovic, T. Abdelzaher, "Range-Free Localization Schemes for Large Scale Sensor Networks" in Proceedings of Mobicom, 2003.

[14] L. Hu, D. Evans, "Localization for Mobile Sensor Networks", in Proceedings of Mobicom, 2004.

[15] Y. Kwon, K. Mechitov, S. Sundresh, W. Kim, G. Agha, "Resilient Localization for Sensor Networks in Outdoor Environments", UIUC Technical Report, 2004.

[16] K. Langendoen, N. Reijers, “Distributed Localization in Wireless Sensor Networks, A Comparative Study”, in Computer Networks vol. 43, 2003.
[17] K. Lorincz, M. Welsh, “MoteTrack: A Robust, Decentralized Approach to RF-Based Location Tracking”, in Proceedings of Intl. Workshop on Location and Context-Awareness, 2005.
[18] M. Maroti, B. Kusy, G. Simon, A. Ledeczi, “The Flooding Time Synchronization Protocol”, in Proceedings of Sensys, 2004.
[19] D. Moore, J. Leonard, D. Rus, S. Teller, “Robust Distributed Network Localization with Noisy Range Measurements” in Proceedings of Sensys, 2004.
[20] R. Nagpal, H. Shrobe, J. Bachrach, “Organizing a Global Coordinate System for Local Information on an Adhoc Sensor Network”, in A.I Memo 1666. MIT A.I. Laboratory, 1999.
[21] D. Niculescu, B. Nath, “DV-based Positioning in Adhoc Networks” in Telecommunication Systems, vol. 22, 2003.
[22] E. Osterweil, T. Schoellhammer, M. Rahimi, M. Wimbrow, T. Stathopoulos, L.Girod, M. Mysore, A.Wu, D. Estrin, “The Extensible Sensing System“, CENS-UCLA poster, 2004.
[23] B.W. Parkinson, J. Spilker, “Global Positioning System: theory and applications”, in Progress in Aeronautics and Astronautics, vol. 163, 1996.
[24] P.N. Pathirana, N. Bulusu, A. Savkin, S. Jha, “Node Localization Using Mobile Robots in Delay-Tolerant Sensor Networks”, in Transactions on Mobile Computing, 2004.
[25] N. Priyantha, A. Chakaborty, H. Balakrishnan, “The Cricket Location-support System”, in Proceedings of MobiCom, 2000.
[26] N. Priyantha, H. Balakrishnan, E. Demaine, S. Teller, “Mobile-Assisted Topology Generation for Auto-Localization in Sensor Networks”, in Proceedings of Infocom, 2005.
[27] A. Savvides, C. Han, M. Srivastava, “Dynamic Fine-grained localization in Adhoc Networks of Sensors”, in Proceedings of MobiCom, 2001.
[28] Y. Shang, W. Ruml, “Improved MDS-Based Localization”, in Proceedings of Infocom, 2004.
[29] M. Sichitiu, V. Ramadurai,“Localization of Wireless Sensor Networks with a Mobile Beacon”, in Proceedings of MASS, 2004.
[30] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, “Sensor Network-Base Countersniper System”, in Proceedings of Sensys, 2004.
[31] R. Stoleru, T. He, J.A. Stankovic, “Walking GPS: A Practical Solution for Localization in Manually Deployed Wireless Sensor Networks”, in Proceedings of EmNetS, 2004.
[32] R. Stoleru, J.A. Stankovic, “Probability Grid: A Location Estimation Scheme for Wireless Sensor Networks”, in Proceedings of SECON, 2004.
[33] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, D. Culler, “An Analysis of a Large Scale Habitat Monitoring Application”, in Proceedings of Sensys, 2004.
[34] K. Whitehouse, A. Woo, C. Karlof, F. Jiang, D. Culler, “The Effects of Ranging Noise on Multi-hop Localization: An Empirical Study”, in Proceedings of IPSN, 2005.
[35] K. Whitehouse, D. Culler, “Calibration as Parameter Estimation in Sensor Networks”, in Proceedings of WSNA, 2002.
[36] P. Zhang, C. Sadler, S. A. Lyon, M. Martonosi, “Hardware Design Experiences in ZebraNet”, in Proceedings of Sensys, 2004.
[37] Selco Products Co. “Construction and Characteristics of CdS Cells”, product datasheet, 2004

Localization System for Outdoor Wireless Sensor Networks

Radu Stoleru, Tian He, John A. Stankovic
Department of Computer Science, University of Virginia
{stoleru, tianhe, stankovic}@cs.virginia.edu

1. INTRODUCTION

We demonstrate a localization system, called Spotlight [1], for Wireless Sensor Networks (WSN). The system, developed at the University of Virginia, has been tested in an outdoor, realistic, environment and has shown a very high accuracy (e.g., tens of centimeters). Our contribution to the area of localization in WSNs is two-fold: develop a novel scheme for localizing sensor nodes, and implement the system on real sensor nodes and common-off-the-shelf (COTS) hardware.

2. SYSTEM DESCRIPTION

The Spotlight localization system employs an asymmetric architecture, in that all the complexity associated with the localization is embedded in a single device, called the Spotlight device. In our system, no hardware changes are necessary to the existing line of Mica sensor boards.

The Spotlight device is a sophisticated, powerful device, capable of producing controlled events (e.g., light events) in the deployment area of a sensor network. A visual depiction of the Spotlight device, is a helicopter equipped with a search light.

The sensor network is time synchronized. Its sensor nodes execute an event detection algorithm, and report back to the Spotlight device, through a base station, the time stamps of the detected events. Using its knowledge of the distribution (in space and time) of events, and the time when a sensor node detected an event, the Spotlight device is able to compute the node's location.

3. DEMONSTRATION

For our indoor demonstration, we use the following hardware: a computerized telescope mount, three very low power (less than 1mW) diode lasers, able to produce circular and linear beams, an LCD projector, a laptop and a sensor network of 10 MicaZ motes. The setup is shown in Figure 1.

The sensor nodes are attached to a vertically positioned Veltex board (attached to an easel). The Spotlight device, comprised of the telescope mount and laptop is positioned 3-4 meters away from the sensor network. A Localization GUI, running on the laptop, controls the orientation of the computerized telescope mount and the communication with the sensor network, through a base station. The Localization GUI displays, through the LCD projector, an aerial image (map) of a real outdoor environment, called deployment area.

During the demonstration, the system goes through the following phases:

- Time synchronization phase, to synchronize the clocks of all sensor nodes and the PC. This phase executes only once, at the beginning of the demo. The following phases execute in a loop.

- Event Generation phase, during which the laser will scan the deployment area. We demonstrate two different Event Generation schemes: a Point Scan and a Line Scan. In the Point Scan scheme, the deployment area is scanned by a laser with a circular beam. In the Line Scan scheme, two lasers, with linear beams, scan the deployment area, on the horizontal and vertical axes. One run of the demo will exhibit one of the two Event Generation schemes.

- Reporting phase, initiated by the Localization GUI. In this phase all sensor nodes report back to the Localization GUI, the time stamps of the detected events.

- Localization phase in which the Localization GUI computes and displays on the map the position of sensor nodes.

- System reset phase, initiated from the Localization GUI, in which all sensor nodes clear the set (i.e. empty set) of timestamps for the observed events.

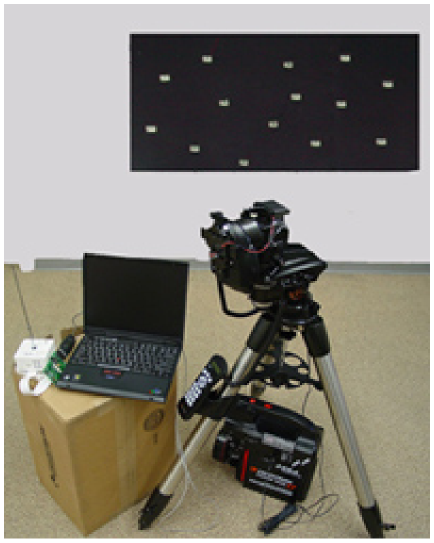


Figure 1. Indoor demonstration setup.

The error in localization of the system is represented by the distance between the position of the photo-sensor, on the MTS310 sensor board, and the position of a sensor node, shown as a red dot through the LCD projector.

4. REFERENCES

[1] R. Stoleru, T. He, J. A. Stankovic, "High-Accuracy, Low-Cost Localization System for Wireless Sensor Networks", in International Conference on Embedded Networked Sensor Systems (SenSys), 2005

Energy-Efficient Surveillance System Using Wireless Sensor Networks[*]

Tian He, Sudha Krishnamurthy, John A. Stankovic, Tarek Abdelzaher, Liqian Luo, Radu Stoleru, Ting Yan, Lin Gu

Department of Computer Science
University of Virginia, Charlottesville, VA 22903
{tianhe, skrish, stankvoic, zaher, ll4p, rs6bd, ty4k, lg6e}@cs.virginia.edu

Jonathan Hui, Bruce Krogh

Department of Electrical and Computer Engineering
Carnegie-Mellon University, Pittsburgh, PA 15213
jwhui@cs.berkeley.edu, krogh@ece.cmu.edu

ABSTRACT

The focus of surveillance missions is to acquire and verify information about enemy capabilities and positions of hostile targets. Such missions often involve a high element of risk for human personnel and require a high degree of stealthiness. Hence, the ability to deploy unmanned surveillance missions, by using wireless sensor networks, is of great practical importance for the military. Because of the energy constraints of sensor devices, such systems necessitate an energy-aware design to ensure the longevity of surveillance missions. Solutions proposed recently for this type of system show promising results through simulations. However, the simplified assumptions they make about the system in the simulator often do not hold well in practice and energy consumption is narrowly accounted for within a single protocol. In this paper, we describe the design and implementation of a running system for energy-efficient surveillance. The system allows a group of cooperating sensor devices to detect and track the positions of moving vehicles in an energy-efficient and stealthy manner. We can trade off energy-awareness and surveillance performance by adaptively adjusting the sensitivity of the system. We evaluate the performance on a network of 70 MICA2 motes equipped with dual-axis magnetometers. Our results show that our surveillance strategy is adaptable and achieves a significant extension of network lifetime. Finally, we share lessons learned in building such a complete running system.

[*]This work was supported by the DAPRPA IXO offices under the NEST project (grant number F336615-01-C-1905).

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design

General Terms

Design, Performance, Experimentation, Measurement

Keywords

Sensor networks, Energy conservation, Tracking, Wireless

1. MOTIVATION

One of the key advantages of wireless sensor networks (WSN) is their ability to bridge the gap between the physical and logical worlds, by gathering certain useful information from the physical world and communicating that information to more powerful logical devices that can process it. If the ability of the WSN is suitably harnessed, it is envisioned that WSNs can reduce or eliminate the need for human involvement in information gathering in certain civilian and military applications. In the near future, sensor devices will be produced in large quantities at a very low cost and densely deployed to improve robustness and reliability. They can be miniaturized into a cubic millimeter package (e.g., smart dust [16]) in order to be stealthy in a hostile environment. Cost and size considerations imply that the resources available to individual nodes are severely limited. We believe, however ,that limited processor bandwidth and memory are temporary constraints in sensor networks. They will disappear with fast developing fabrication techniques. The energy constraints on the other hand are more fundamental. According to R.A. Powers [20], battery capacity only doubles in 35 years. Energy constraints are unlikely to be solved in the near future with the slow progress in battery capacity and energy scavenging. Moreover, the untended nature of sensor nodes and the hazardous sensing environment preclude manual battery replacement. For these reasons, energy awareness becomes the key research challenge for sensor network protocol design. Several researchers have addressed energy conservation recently. Most of them

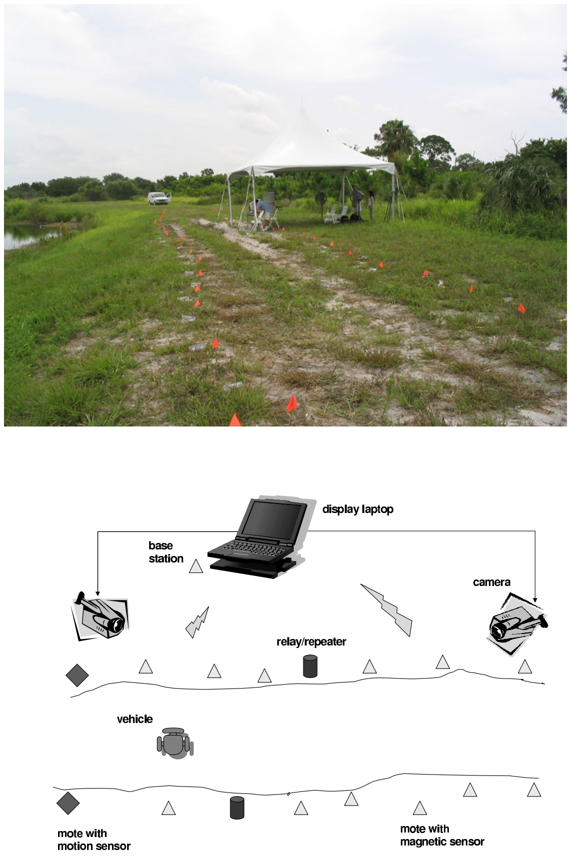


Figure 1: Sensor Network Deployment

focus on particular protocols and investigate whether their energy conservation goal can be achieved. To the best of our knowledge, none of them investigate energy-conservation for a running system as whole. Normally they evaluate their approach through simulations. Simulation approaches tend to make simplified assumptions that often do not hold well in practice and they are subject to incompleteness. For example, in [22][23][21], several sensing coverage schemes are proposed for energy conservation. None of them consider energy consumption in activities other than sensing.

In this paper, we describe our effort that involves system design and implementation on a MICA2 platform with 70 MICA2 motes. The primary goal of the system is to support the ability to track the position of moving targets in an energy-efficient and stealthy manner. Our experimental results show that the probability of false alarms observed reaches zero when aggregation is achieved among more than 3 member motes. The experimental results we obtained also show that with 5% of deployed motes serving as sentries and the non-sentries operating at a 4% duty cycle, our algorithm extends the lifetime of a sensor network by up to 900%.

The main contributions of this paper are: 1) the design and implementation of a running system with energy-awareness as the main design principle across multiple components, 2) Mechanisms for dynamic control, which allow tradeoffs between energy-efficiency and system performance by adjusting the sensitivity of the system, and 3) a physical implementation and field evaluation that reveal the practical issues that are hard to capture in simulation.

The remainder of this paper is organized as follows. Section 2 describes the requirements of a typical ground surveillance application. In Section 3, we describe the system setup and hardware components. In Section 4, we provide an overview of our system design. In Section 5, we elaborate on how the individual components of the system contribute to energy-efficient tracking. In Section 6, we discuss implementation issues concerning our system. We present experimental results in Section 7, and summarize the lessons learned from our experience in Section 8. Finally we conclude in Section 9 and discuss some future work in Section 10.

2. APPLICATION REQUIREMENTS

Our system design is motivated by the requirements of a typical ground surveillance application. The general objective of such an application is to alert the military command and control unit in advance to the occurrence of events of interest in hostile regions. The event of interest for our work is the presence of moving vehicles in the deployed region. The deployed sensor devices must have the ability to detect and track vehicles in the region of interest. Successful detection and tracking requires that the application obtain the current position of a vehicle with acceptable precision and confidence. When the information is obtained, it has to be reported to a remote base station within an acceptable latency. Several application requirements must be satisfied to make this system useful in practice:

- **Longevity:** The mission of a surveillance application typically lasts from a few days to several months. Due to the confidential nature of the mission and the inaccessibility of the hostile territory, it may not be possible to manually replenish the energy of the power-constrained sensor devices during the course of the mission. Hence, the application requires energy-aware schemes that can extend the lifetime of the sensor devices, so that they remain available for the duration of the mission.

- **Adjustable Sensitivity:** The system should have an adjustable sensitivity to accommodate different kinds of environments and security requirements. In critical missions, a high degree of sensitivity is desired to capture all potential targets even at expense of possible false alarms. In other case, we want to decrease the sensitivity of the system, maintaining a low probability of false alarms in order to avoid inappropriate actions and unnecessary power dissipation.

- **Stealthiness:** It is crucial for military surveillance systems to have a very low possibility of being detected and intercepted. Miniaturization makes sensor devices hard to detect physically; however, RF signals can be easily intercepted if sensor devices actively communicate during the surveillance stage. A zero communication exposure is desired in the absence of significant events.

- **Effectiveness:** The precision in the location estimate, and the latency in reporting an event are the metrics that determine the effectiveness of a surveillance system. Accuracy and latency are normally considered important metrics of tracking performance. However, the requirement of these two metrics can actually be slightly relaxed in many tracking applications. For example, it may be acceptable to obtain location estimation within a couple of feet and receive a detection report within a couple of seconds. We, therefore, focus primarily on the first three metrics mentioned above.

3. SYSTEM DESCRIPTION AND REQUIREMENTS

Figure 1 shows the deployment of our ground surveillance system. We deployed 70 tiny sensor devices, called MICA2 motes [14], along a 280 feet long perimeter in a grassy field that would typically represent a critical choke point or passageway to be monitored. Each of the motes is equipped with a 433 MHz Chipcon radio with 255 selectable transmission power settings. While this radio is sufficient to allow the motes deployed in the field to communicate with each other, it is not capable of long-range ($>$ 1000 ft) communication when put on the ground. Therefore, we assume that in a real system where the command and control units may be deployed several thousands of feet away from the sensor field, devices capable of long-range communication, such as repeaters, will be deployed as gateways to assist the sensors to relay back information from the motes in the field to the base station. In our prototypical deployment, we use a mote as the base station that is attached to a portable device, such as a laptop. The portable device is the destination of the surveillance information and is mainly used for visualization in our prototype system. The camera devices shown in Figure 1 are controlled by the laptop to provide the next level of surveillance information, when triggered by the sensor field.

Each mote is equipped with a sensor board that has magnetic, acoustic, and photo sensors on it. While the different sensors make it possible for a mote to detect different kinds of targets, only the magnetic sensors are relevant to the application described in this paper. We use the HMC1002 dual-axis magnetometers from Honeywell [13]. These magnetic sensors detect the magnetic field generated by the movement of vehicles and magnetic objects. They have an omni-directional field of view and are therefore less sensitive to orientation. They have a resolution of 27 μGauss and their sensing range varies with the size of the magnetic object they are sensing. From our experiments, we found that these sensors can sense a small magnet at a distance of approximately 1 ft and slowly moving passenger vehicles at a distance of approximately 8-10 ft.

4. SYSTEM OVERVIEW

The key contribution of this work is the design and implementation of a wireless sensor network prototype that enables energy-efficient tracking and detection of events. Such a system is useful for surveillance applications, such as the one outlined in Section 2. The system we have designed is organized into a layered architecture comprised of higher-level services and lower-level components, as shown in Figure 2. It is implemented on top of TinyOS [12]. We first provide an overview of the different software components we have designed and then follow that with a detailed discussion of the role played by those components in the context of our tracking and surveillance application.

Time synchronization, localization, and routing comprise

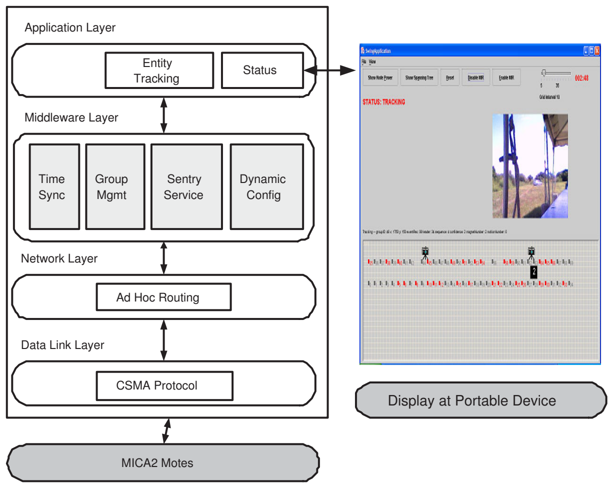


Figure 2: Energy-Efficient Tracking System

the lower-level components and form the basis for implementing the higher-level services, such as aggregation and power management. Time synchronization and localization are important for a surveillance application because the collaborative detection and tracking process relies on the spatio-temporal correlation between the tracking reports sent by multiple motes. The time synchronization module is responsible for synchronizing the local clocks of the motes with the clock of the base station. The localization module is responsible for ensuring that each mote is aware of its location. In our prototype system, we use a simple localization configuration, which statically assigns motes their location at the time they are programmed, assuming we know about where they will be placed. In actual deployment, such as a battlefield in which it is important to track the absolute geographical coordinates of the hostile tanks, the static configuration can be replaced with dynamic localization schemes such as in [10].

The routing component establishes routes through which the motes exchange information with each other and the base station.

Power management and collaborative detection are the two key higher-level services provided by our system. The sentry service component is responsible for power management, while the group management component is responsible for collaborative detection and tracking of events. The sentry service conserves energy of the sensor network by selecting a subset of motes, which we define as *sentries*, to monitor events. The remaining motes are allowed to remain in a low-power state until an event occurs. When an event occurs, the sentries awaken the other motes in the region and the group management component dynamically organizes the motes into groups in order to enable collaborative tracking. Together, these two components are responsible for energy-efficient event tracking.

All the deployed motes are programmed to run the distributed application. Our system supports the ability to reprogram the motes dynamically with new configuration parameters such as sensitivity . This eliminates the need to download the application code on all the motes each time the configuration is modified. We have a display module for portable devices (Figure 2)which is not part of the software that runs on each mote. We use it primarily for visualization and debugging purposes. Optionally, the display software also has the logic to filter out any residual false alarms that have not been filtered out in the network. We now elaborate how the individual components of the system shown in Figure 2 interact with each other in the context of a typical tracking application. In particular, we discuss the design decisions that make the target system energy-efficient and illustrate trade-offs between performance and energy-awareness.

5. TIME-DRIVEN SYSTEM DESIGN

In our system, the MICA2 motes prepare for tracking by going through an initialization process. This process is used to synchronize the motes, set up communication routes, and configure the system with the correct control parameters. The initialization process proceeds in a sequence of phases and the transition between phases is time-driven, as shown in Figure 3. Phases I through IV comprise the initialization process which normally takes about 2 minutes. At the end of phase IV, the motes begin the power management and tracking activity. After performing this activity for a certain duration of time (e.g., one day), they begin a new system cycle. The duration of each phase is a control parameter that can be dynamically configured by the base station. Our multi-phase cyclic process satisfies following design objectives:

- First, it eliminates interference between operations. The constrained bandwidth in MICA2 doesn't allow a high concurrency in communication. If all operations run simultaneously, the traffic will severely interfere with each other.
- Second, we can confine the exposure of sensor activity within a short period time during the initialization phase (phase I to IV). As a result, the system can achieve zero exposure (complete stealthiness) during surveillance when no significant event happens.
- Third, a new system cycle is a natural way to allow the rotation of sentry responsibility among motes in order to achieve uniform energy dissipation across the network.
- Last, the cycling introduces system-wide soft-states. It allows the motes to periodically synchronize their clocks to avoid significant clock drifts over time. In addition, since mote failures and new deployment may occur anytime during a cycle, a new system cycle gives the remaining motes an opportunity to repair routes and discover new neighbors.

We now discuss the activities occurring during each phase of the system cycle in more detail.

5.1 Phase I: Basic Initialization

We observe that three functions in our system need system-wide broadcast: time synchronization, network backbone creation and system-wide reconfiguration. These functions can be isolated into three different modules that perform separately. However, such a design would not be bandwidth and energy efficient due to the multiple flooding phases required. Instead, we use a unique application-specific design

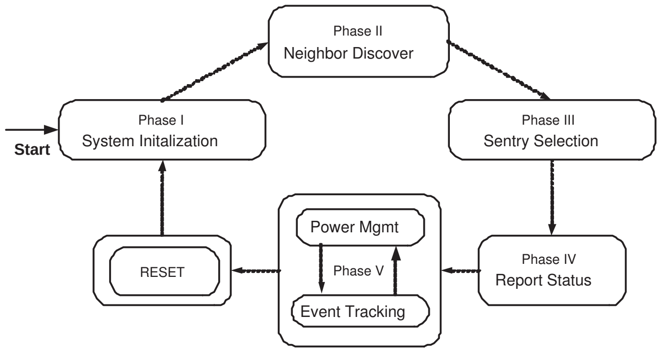


Figure 3: Time Driven System Transition

to perform these operations simultaneously in one flooding operation to reduce overhead as described in following sections.

5.1.1 *Time Synchronization*

System initialization begins with time synchronization. Several schemes proposed recently are able to achieve a high synchronization precision, however they do not match well with our system requirements. GPS-based schemes typically achieve persistent synchronization with a precision of about 200 ns. However, GPS devices are expensive and bulky. The reference broadcast scheme (RBS) proposed in [5] maintains information relating the phase and frequency of each pair of clocks in the neighborhood of a node. The relation is then used to perform time conversion when comparing the timestamps of two different nodes. While RBS achieves a precision of about 1 μs, the message overhead in maintaining the neighborhood information is high and may not be energy-efficient in large systems.

We argue that fine-grained clock synchronization [5] achieved by costly periodic beacon exchanges may not be suitable for the energy-constrained surveillance system. Moreover continuous adjustment through beaconing in these solutions defeats our purpose of stealthiness. In our system, we value energy-efficiency and stealthiness above high synchronization precision. Therefore, we use a lightweight scheme that synchronizes the motes only during initialization phase, using a synchronization beacon broadcast by the base station at the beginning of each initialization cycle. Since the underlying MAC layer provided by TinyOS does not guarantee reliable delivery, the base station retransmits the synchronization beacon multiple times. Receivers take the timestamp from the beacon plus a transmission delay as their own local time. The synchronization beacons are propagated across the network through limited flooding with timestamp values reassigned at intermediate motes immediately prior to transmission. To satisfy the stealthiness requirement, we confine time synchronization within the initialization phase. The timer drift accumulated overtime is rectified by a new system cycle.

5.1.2 *Diffusion Tree Creation*

While the primary purpose of the synchronization message is to coordinate the clocks of the motes, it also serves as an exploratory message for motes to set up reverse routes to the base station, like the technique used by directed diffusion [15]. The route that is set up during the propagation of the time synchronization message is essentially a diffusion tree rooted at the base station. The decision to use a diffusion tree is made based on several observations. 1) Sent along with the time synchronization operation, it is nearly free of cost in communication and code memory. 2) It allows any leaf motes to go to sleep without disrupting communication of other motes.

We encounter two practical issues when implementing the diffusion tree algorithm on the MICA2 platform.

- **Mote Failures:** The failure of a MICA2 mote can disable a subtree below it. Initially, we attempted to add failure detection to the MAC layer to quickly identify link failures and chose alterative routes. Soon, we discovered that link layer reliability in such a bandwidth constrained platform is too heavyweight and the effective data rate is reduced by nearly 50%. With such an observation, we introduce soft-state into the diffusion tree. The diffusion tree is refreshed per system cycle to prune failed links and discover new routes. After this modification, no bandwidth penalty is experienced during data communication.

- **Asymmetric Links:** If motes choose their parents without considering the distance separating them, it results in asymmetric links which leads to different reception rates along different directions between the same pair of motes. This asymmetry can be solved by link layer handshaking; however we discovered that it is very expensive. The practical strategy we adopt is that we use a lower transmission power when sending out synchronization messages to ensure that motes choose parents that are within a smaller radius. When transmitting application data, we use the maximum transmission power to ensure symmetric communication along the diffusion three in directions to and from the base station.

5.1.3 *Dynamic Reconfiguration*

The capability of dynamic reconfiguration facilitates retasking of sensor networks in future changes of mission requirements. Currently, this capability makes our work in system tuning and debugging much easier. When we deployed 70 motes on the field for the first time, it took us an hour to collect the motes and reprogram them manually, before the reconfiguration capability was added into the system. Now we can reconfigure the network within 1 minute. Our system supports reconfiguration with the help of the time synchronization message. The base station piggybacks the values of the control parameters in the synchronization message and motes adopt the new values when they accept the synchronization message. Such a strategy is energy-efficient, because it comes along with time synchronization beacons, obviating the need to send separate messages to reset parameters on the motes. Examples of control parameters that can be dynamically reconfigured include the duration of each phase shown in Figure 3, the duration for which a mote remains asleep and awake when power management is enabled, the sampling rate and the degree of in-network aggregation. This reconfiguration capability enables us to dynamically trade off between the energy-awareness and tracking performance as we show later in this paper.

5.2 Phase II: Neighbor Discovery

After the basic initialization phase, the motes make a transition to a neighbor discovery phase. Motes notify their neighbors by locally broadcasting `HELLO` messages. In the `HELLO` message, a sender sends its identifier, its status indicating whether it is a sentry or not, and the number of sentries that are currently covering it. The sender also identifies the sentry mote it reports to, if it is covered by at least one sentry. This local information is used to build a neighborhood table at each mote, and forms the basis for sentry selection in Phase III.

5.3 Phase III: Sentry Selection

In our sentry selection scheme, the decision to become a sentry is made locally by each mote, using the information gathered from its neighbors (The neighbor discovery goes through Phase II and III).

A mote decides to become a sentry if any one of the following conditions holds. 1) it is one of the internal nodes of the diffusion tree, or 2) it discovers that none of its neighbors either is a sentry or is covered by a sentry. When a mote decides to become a sentry, it advertises its intent. Three practical issues need to be solved to make this scheme work in a running system:

- **Race Conditions:** Contention occurs when multiple motes in the same neighborhood decide to become sentries at the same time. In order to reduce the collision probability, each mote uses a random backoff delay to transmit a `SENTRY DECLARE` message. If a mote receives a `SENTRY DECLARE` message from one of its neighbors during the backoff period, it updates its neighborhood table and cancels any pending outgoing `SENTRY DECLARE` messages. It then re-evaluates its decision to become a sentry based on the updated neighborhood information. If the mote finds that it is still necessary for it to become a sentry, it repeats the sentry declaration process described above.

- **Energy Balancing and Efficiency:** We set the backoff delay of a mote inversely proportional to its residual energy. Thus, a mote with higher residual energy has a greater likelihood of being selected as a sentry, thereby balancing the energy dissipation uniformly across the network. The backoff delay of a mote is also inversely proportional to the number of neighbors that are not covered by a sentry. Thus, motes in regions where there is insufficient sensing coverage are favored for being selected as sentries. The key feature of this sentry selection algorithm is that it provides an adaptive, self-configuring technique for choosing the sentries purely based on local information. However, the lack of global knowledge may result in a non-optimal number of sentries.

- **Sensing Coverage:** Surveillance addresses the sensing coverage problem of every physical point in the terrain, instead of communication coverage as in LEACH [11] and SPAN [3]. Since the sensing range of our Honeywell magnetometer [13] is much smaller than the Chipcon radio range, we need to use a smaller transmission power setting to send out `SENTRY DECLARE` messages in order to ensure sensing coverage. The power setting is chosen in such a way that there is

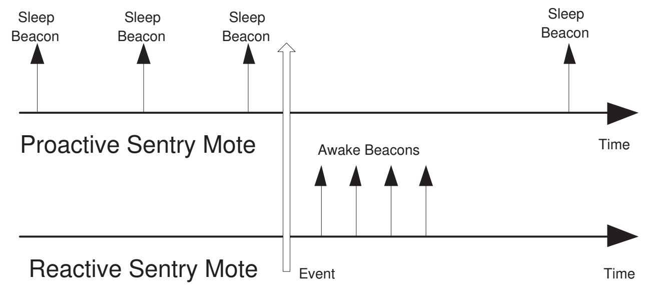


Figure 4: Two Power Management Schemes

at least one sentry within each sensing range. Unlike [22, 21], this unique design enables us to provide sensing coverage without the requirement of localization. More details can be found in the evaluation Section 7.1.

5.4 Phase IV: Status Report

After the routing backbone is finalized, all the motes use the backbone to report their status to the base station in Phase IV. The base station forwards those reports to the display module, which can then be used to visualize network topology, residual energy distribution and sentry distribution and detect any failed motes. Since the sole purpose of Phase IV is for visualization and debugging, it is optional.

5.5 Phase V-A: Power Management

The selection of sentries sets the stage for the power management phase. In this phase, the non-sentry motes alternate between sleep and wakeup states. A mote in the sleep state conserves power by disabling all processing, including those that are related to communication and sensing. We have proposed and implemented two different schemes to control the sleep-wakeup cycle. Now we discuss the pros and cons of these two schemes to clarify some practical issues

In the first implementation, which we call **proactive** control (Figure 4), the sentry mote sends out sleep beacons periodically. A non-sentry mote stays awake until it receives a beacon from its sentry mote, signaling the non-sentry mote to sleep for a certain duration of time. Upon receiving the sleep beacon, the non-sentry mote makes a transition to the sleep state and remains in that state for the specified amount of time. It wakes up when the timer expires and repeats the process by waiting for the next sleep beacon. Since neighboring non-sentry motes are likely to receive the same sleep beacon, their sleep-wakeup cycle proceeds in a lock-step fashion. The regular synchronization of the non-sentry motes with their respective sentries is beneficial in two ways. First, it allows multiple motes to receive the same beacon, and obviates the need to send out individual sleep beacons to put each non-sentry mote to sleep. This reduces the message overhead. Second, since motes in a neighborhood are all awake at the same time, the correlated sleep-wakeup cycle helps improve the tracking efficiency.

The second implementation to control the sleep-wakeup cycle is called the **reactive** control (Figure 4). In this scheme, the sentries are not required to send out explicit beacons to put the non-sentry motes to sleep. Instead, the transition between sleep and wakeup states is timer-driven. Each non-sentry mote remains awake for *awakeDuration* amount of time and then sleeps for *sleepDuration* amount of time. A non-sentry mote breaks out of its cycle and remains awake for a longer duration only when receiving an awake beacon from a sentry mote.

The reactive scheme is more stealthy compared to the proactive scheme, because no unnecessary beacons are sent unless an event occurs. Hence, the reactive approach is more appropriate for a surveillance application. However, one practical issue needs to be solved in the reactive scheme; since the non-sentries do not periodically synchronize their clocks with the clocks of their sentries, the clocks of the non-sentry motes may drift in course of time. Consequently, neighboring non-sentry motes may no longer have a sleep-wakeup cycle that is strictly in lock-step. As a result, a sentry no longer knows for certain which of its neighbors are awake. It has to retransmit the awake beacon multiple times in order to awaken non-sentries when an event occurs (Figure 4). We compare the message overhead between the proactive and reactive schemes in Section 7.3.1.

5.6 Phase V-B: Event Tracking and Reporting

After the sentry backbone has been created and power management is enabled, the motes are ready for tracking. Tracking and power management are toggle-states in phase V. When an event happens, motes wakeup and start tracking, when event disappears, motes toggle back to power management states.

A simple way to track events is by allowing each mote that has sensed an event to report its location and other relevant information about the event to the base station. The base station can then filter out the false alarms and infer the location of the event from the genuine reports. The advantage of this approach is that it allows all of the complex processing of the sensor readings to be deferred to the more powerful base station. However, the main drawback is that, if the motes are densely deployed, multiple motes may sense the event at the same time and send their individual reports to the base station. This results in higher traffic and wasteful expenditure of energy which can be reduced by aggregating multiple reports about the same event and sending a digest, instead of the individual reports to the base station. Previous in-network aggregation techniques fuse the data at the source through cluster headers [11] and/or along the route back to the sink [1][9][15][17]. In addition, Zhao [6] propose a optimal sensor selection approach to aggregate the fidelity of detections while eliminating redundant communication.

The system we have designed also performs in-network aggregation by organizing the motes into groups. However, distinguished from previous schemes, the groups in our work are more dynamic in the sense that they are formed in response to an external event and migrate when an event moves. A group represents an event uniquely and exists only as long as the event is in the scope of the sensor field. The design of our group management and tracking component is described in [2]. We review its key features here for completeness. It should be noted that the work reported in this paper is the first real implementation of the aforementioned design.

Each mote is programmed to detect an event by its sensory signature. This signature is a condition on the output of a filter that processes the raw sensory measurements (and removes noise). When the indicated condition is detected by a set of nearby motes, the group management component reacts by creating a group. All motes that detect the same event join the same group. The main contribution of the group management component, described in [2], is to establish a unique one-one mapping between a group and a physical event as well as to maintain the membership of the group as the event moves through the environment. It is assumed that different events are far enough apart that membership of motes to the corresponding groups can be decided without ambiguity based on spatial adjacency to one of the events.

Each group is represented by a *leader* to the external world. Group members (who by definition can sense the tracked event) periodically report to the group leader. The leader records each report keeping only the most recent one from each member. Reports that are older than a certain threshold are purged. We define the confidence level of event detection as the number of distinct motes that have reported the event in the last t_r units of time. When the confidence level of detecting an event is at least as high as the threshold required by the application, called the *degree of aggregation* (DOA), the leader sends a digest of the reports to the base station. The confidence threshold can be tuned to manipulate the sensitivity of the system. A low threshold increases sensitivity at the expense of possible false alarms. A high threshold could result in missing some smaller targets. The effect of manipulating the degree of aggregation is explored experimentally in Section 7.2.2.

This concludes the description of our system design. In the next section, we discuss the implementation issues on the TinyOS / MICA2 platform.

6. IMPLEMENTATION

The architecture described in Section 4 was built on top of TinyOS [12]. TinyOS is an event driven computation model, written in NesC [7] specifically for the motes platform. TinyOS provides a set of essential components such as hardware drivers, scheduler and basic communication protocols. These components provide low level support for application modules, which are also written in NesC. NesC is a C-like language that enables the programmers to define the function of components and the relations (dependencies) among them. Components from TinyOS and user applications are processed by the NesC compiler into a running executable, which runs (in our case) on the MICA2 mote platform. MICA2 is the third generation mote built for wireless sensor networks [4]. Besides normal computation and communication capabilities, MICA2 motes have (i) selectable transmission power settings (255 levels) which enable us to dynamically adjust the communication range, (ii) a snooze function with up to six sleep modes provided by the ATmega128 Microcontroller, and (iii) a wireless reprogramming capability that eliminates the need for manual code downloads. The first two functions are utilized extensively by our protocols. The last facilitates deployment. In particular, we use a lower communication power setting during neighbor discovery for diffusion tree creation. This ensures that when the diffusion tree is created and communication power is subsequently increased, all found edges along the tree are quite reliable. In contrast, running diffusion tree creation at the normal power setting could result in unreliable or asymmetric edges between some nodes. This choice would ultimately reduce performance. The snoozing function is used to put motes to sleep when in the power saving mode.

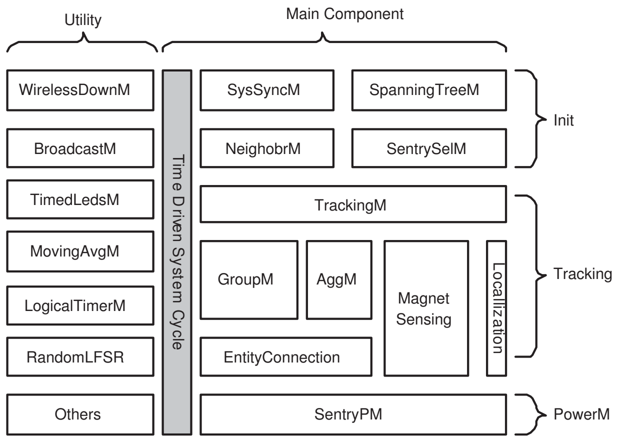


Figure 5: System Architecture

The implementation of our system on the MICA2 motes was driven by several requirements that arise from platform limitations. Namely:

- **Energy Efficiency:** MICA2 operates on a pair of batteries that approximately supply 2200 mAh at 3V. It consumes 20mA if running a magnetic sensing application continuously which leads to a lifetime of 5 days.
- **Bandwidth Efficiency:** The Chipcon radio on MICA2 provides an effective data rate of 12.4kbps, which equals a maximum packet rate of 43 pkts/sec. Our experiments show that a mote barely reaches 20 pkts/sec when it is exposed to channel contention.
- **Simplicity:** Our system requires many essential functions shown in Figure 5 to make target tracking efficient, while the whole system must fit in 4K data memory and 128K code memory. This necessitates a simple, yet effective, design for the MICA2 platform.
- **Flexibility:** Our prototype system spans 280 feet and comprises 70 motes. Once deployed, motes can not be easily collected. Dynamic configuration is desirable for fast performance tuning and debugging.

6.1 Software Architecture

The architecture of our system, written in NesC, is shown in Figure 5. The whole system occupies 39,496 bytes of code memory and 3,725 bytes of data memory. We divide system components into four major groups; initialization, tracking, power management, and general utilities. Initialization components are responsible for basic infrastructure establishment. Tracking components support the event tracking functions. The SentryPM module performs power management which puts motes to sleep as described earlier, when no significant events are detected. We also use some utilities to facilitate downloading, debugging, tuning and statistical logging. We provide a *backbone* module which is in charge of time-driven transitions between phases. We also use this module to pass state information among other modules to reduce the dependency among components.

In implementing the above architecture, several system challenges were met, primarily due to lack of common operating system support which TinyOS doesn't have. Some of the most important issues were the following:

Concurrency Control: TinyOS provides minimal support for concurrency control. The latest NesC compiler detects potential data races and give warnings at compile-time, however, it still requires the programmer to deal with it. Data race can be avoided by atomic sections or tasks. An atomic section is implemented through disabling and enabling interrupts. This requires the critical section to be very short. Otherwise, the system will become unresponsive. For example, if the soft timer cannot get updated by clock interrupts, time drift will happen. A better approach is to put all operations that access shared data into a task context. This guarantees sequential access to the data. However, the current task model doesn't allow parameter passing. The solution to this limitation is to put parameters into shared variables accessible by all tasks and use atomic sections to protect the read and write operation on these variables.

Packet Scheduling: For now, the TinyOS communication module doesn't provide a buffering mechanism. It is often the case that multiple components send out packets concurrently. All but one operation fails due to the mutual exclusion mechanism described above, used in the lower layer. The current solution we used is to provide application layer buffering. We reinitiate the transmission with linear backoff when contention happens.

Aggregation: The TinyOS communication module has a relatively high overhead. The packet header is 7 bytes (MAC header+ CRC) and the preamble overhead is 20 bytes in MICA2. For a default payload size of 29 bytes, the overhead to send a single packet is 48%! This limitation motivates us to use aggregation techniques. We use piggybacking whenever possible to increase the effective data rate. For instance, we piggyback system-wide parameters in time synchronization messages and piggyback sentry declaration information in neighbor beaconing. A more advanced aggregation technique such as in [9] is desired to efficiently use bandwidth.

Hardware Limitations: In general, the MICA2 platform is effective in supporting our system. However, in some cases, we have to modify our design to accommodate the limitations on hardware. First, the MICA2 mote has no circuit support for remote passsive wakeup. The current snooze implementation relies on a timer interrupt. This increases the chance of false negatives when the sleep duration of non-sentries is relatively long. Second, while the operating frequency of the Chipcon radio is selectable, external hardware attached to the chip can only support one frequency. This prevents us from designing a better collision avoidance algorithm to improve radio performance. Third, the current timer is supported through software which freezes when snooze is enabled. Though we can compensate for the lost time after a mote is awakened, more precise hardware timer support is desired.

Due to space limitations, here we only give a snapshot of the issues we encountered during the implementation. In general, we feel that platform-specific system designs are necessary to improve the performance.

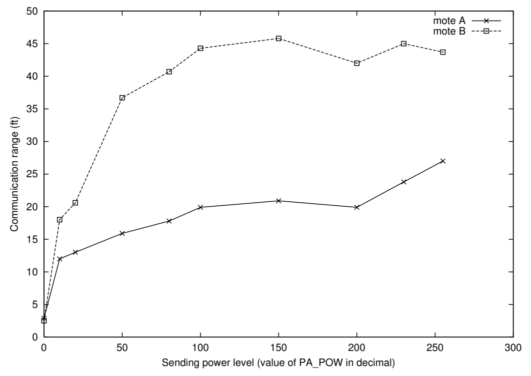


Figure 6: Impact of Sending Power on RF Range

7. PERFORMANCE EVALUATION

We now present experimental results that evaluate the performance of the physical system described in the previous section. We obtained most of the experimental results through an actual deployment of MICA2 motes in a grassy field, using the setup described in Section 3. However, for some experiments which require a long duration of time, we can not afford to deploy the system unattended due to security issues. Instead we conduct this type of experiments with a smaller number motes in controlled environments. In addition, simulations are also used to reveal the tradeoff between different design decisions.

We classify the experiments into three broad categories. The first set of experiments evaluate the MICA2 radio in different environments. The second set of experiments evaluate the performance of the tracking component. Finally, we evaluate the sentry service and the power management features of our system.

7.1 Evaluation of Capability of MICA2 Radio

The communication range of a MICA2 mote depends on several factors, such as the length of the antenna, the transmission power, the elevation above the ground, and the non-line-of-sight effects from objects in the surroundings (e.g., grass, trees, buildings, people, cars). Although the absolute values may vary in different environments, we can still draw some general observations about the MICA2 platform:

- We measure a set of MICA2 communication ranges under different sending power settings with two senders and one receiver. Results shown in Figure 6, indicate that 1) the communication range nonlinearly increases as the sending power increases. It increases more slowly when the power setting is large. 2) Asymmetry in communication range is more than what we expect, and it might primarily come from the differences in hardware calibration.

- We measure MICA2 communication ranges under different antenna lengths and different elevations above the ground. As expected, Table 1 indicates that longer antennas can significantly increase communication range in MICA2. Table 2 shows that the high elevation reduces floor attenuation, and hence increases RF range.

Table 1: Impact of Antenna Lengths on RF Range

Antenna	Power level = 50	Power level= 255
17.3 cm	37 ft	43 ft
34.6 cm	59 ft	> 84 ft

Table 2: Impact of Elevations on RF Range

Elevation	0 ft	0.5 ft	1 ft
Mote A	27 ft	30 ft	> 84 ft
Mote B	43 ft	> 84 ft	> 84 ft

7.2 Evaluation of In-Network Aggregation

In our experimental setup, we deployed 70 MICA2 motes along two sides of a road at a distance of 7-8 ft from each other. They were deployed densely in order to improve the data aggregation among motes.

Our goal is to track a car being driven along the stretch of road and study the impact of system parameters on the tracking performance. One key parameter is the degree of aggregation (DOA). This parameter decides the sensitivity of the surveillance system and is used to trade off between energy-awareness and surveillance performance. It is defined in our system as the minimum number of reports about an event that a leader of a group waits to receive from its group members, before reporting the event's location to the base station. In our implementation, the value of the DOA is dynamically configurable from the base station. We were interested in studying the impact of the degree of aggregation on the following metrics:

- the number of tracking reports (Figure 7),
- the number of false alarms generated (Figure 8), and
- the latency in reporting an event (Figure 9).

7.2.1 Impact of Aggregation on Transmission Overhead

In our tracking experiments we drove a car at a speed varying between 5-10 mph. We varied the degree of aggregation from 1 to 6 and repeated the tracking experiment for each value of DOA ten times. Figure 7 shows how the number of the tracking reports received by the base station varies with the DOA. From the figure, we see that when the value of DOA increases from 1 to 2, the number of tracking reports reduces by almost 50%. As the value of DOA increases even further, we observe that there is a steady drop in the number of tracking reports generated. These results verify the fact that the in-network aggregation, resulting from organizing the sensor motes into groups, significantly reduces the message overhead during tracking, and hence leads to much less energy consumption in data transmission.

7.2.2 Impact of Aggregation on False Alarms

Our next experimental result shows how the degree of in-network aggregation affects the false alarms generated when tracking an event. False alarms are normally caused by events such as burst distortions of readings due to power state transitions and incorrect readings from faulty sensors. Since a simulation-based approach normally assumes that sensors behave according to their specifications, such phenomena are usually not investigated in simulation. We classify false alarms into *false positives* and *false negatives*. A

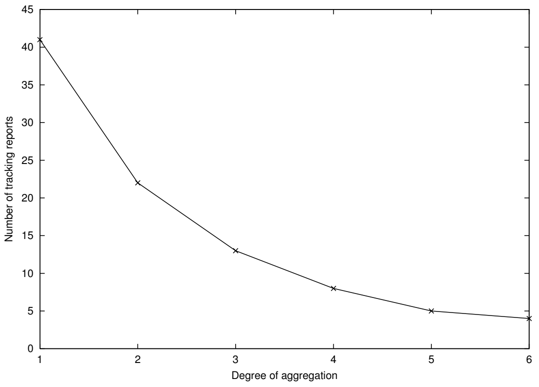


Figure 7: Impact of DOA on the Message Overhead

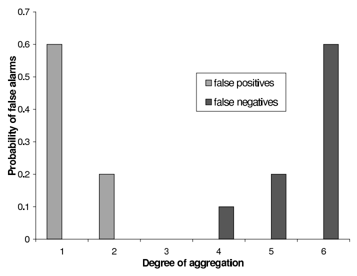


Figure 8: Impact of DOA on False Alarms

false positive occurs when a group of motes report the presence of the moving car in their neighborhood, when in reality, the car is not in their vicinity. A false negative occurs if the base station does not receive any reports of the car, although in reality, there is a car moving though the sensor field. In other words, if the car never appears on the display as it moves from one end of the sensor field to the other, we treat it as a false negative. It is important to emphasize that we do not consider a delayed report as a false negative.

We determined the probability of false alarms for each value of DOA by counting the number of false positives and false negatives we observed on the display during a set of 10 tracking rounds. Figure 8 shows how the probability of false positives and the probability of false negatives are each affected by the degree of aggregation. From Figure 8 we see that as the value of DOA increases from 1 to 6, the probability of false positives drops from 0.6 to 0, while the probability of false negatives increases from 0 to 0.6. These results can be explained as follows.

When the DOA = 1, the leader of a group reports the event to the base station, as soon as at least one member of the group detects the event. In an ideal scenario in which the sensing is perfect, even a single sensor reading should generate a high level of confidence. However, in practice, the sensor boards are sometimes inaccurate. This could result in an event being reported when it is not actually present. Hence, a single sensor reading may not be very reliable. One way to improve the reliability of event detection is to increase the redundancy, by either waiting for multiple reports from the same sensor mote (temporal redundancy), or by waiting for reports from multiple neighboring sensor motes (spatial

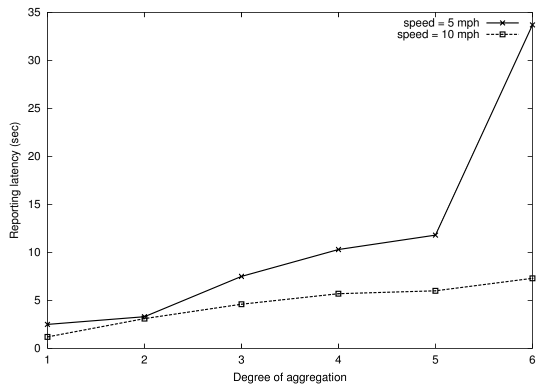


Figure 9: Impact of DOA on Reporting Latency

redundancy). We chose to experiment with the latter option because we assumed that the faults in the sensor boards are independently distributed. Therefore, the probability that multiple neighboring sensor motes are simultaneously in error is lower than the probability that a single sensor mote is in error. From Figure 8, we see that our assumption is validated. The figure shows that if the leader waits until at least 3 different sensor motes have detected the event, before reporting the event to the base station, the number of false positives drops to 0. However, if the sensing range and the density of deployment is not sufficiently high, it is harder to achieve a higher degree of aggregation. This results either in more false negatives, as shown in Figure 8, or in higher reporting latency as shown in the next section.

7.2.3 *Impact of Aggregation on Tracking Latency*

Figure 9 shows how the reporting latency increases with the degree of aggregation for a car moving at 5 mph through a sensor field where the motes are deployed 7-8 ft apart. We define the reporting latency as the time elapsed from the instant at which the car enters the sensor field until the instant at which the base station receives the first *genuine* report about the location of the car. In addition to the density, the increase in the latency and false negatives depends on the sleep cycle of the sensor motes and the speed of the moving vehicle. To our surprise, we found that we were able to reduce the latency and false negatives for higher degree of aggregation (DOA ≥ 4), by increasing the speed of the vehicle from about 5 mph to about 10 mph (Figure 9). However, increasing the speed beyond that value resulted in more false negatives. The reason is that when motes are some distance apart, a higher speed allows the vehicle to be in the sensing range of more motes during a period of time t_r. Hence, the vehicle can be detected even at a higher degree of aggregation. However, the sensors have a non-negligible reaction time, which further increases if the motes are sleeping. Hence, if the speed is increased beyond a certain threshold, the vehicle may move past the sensing range of the motes before they have a chance to react. That could result in more false negatives.

We must emphasize that the performance numbers we have presented above exhibit some degree of variance across different experimental runs and in different environments. Therefore, instead of using the above experimental results to deduce absolute performance numbers, we use them to draw some general conclusions about choosing the degree of in-network aggregation. First, a higher DOA certainly helps reduce the message overhead and the number of false positives. However, if the density with which the motes are deployed is not sufficiently high, a higher degree of aggregation may adversely affect the tracking performance. This effect is more pronounced in the case of slow-moving events. Even if the motes are densely packed and the events are fast-moving, it is harder to achieve a high degree of aggregation, if the motes sleep for a long duration and their sleep-wakeup cycles are not in lock-step. Thus, we see that the degree of aggregation represents a tradeoff between different parameters. The recommendation we follow based on our results is to choose a value of DOA that is large enough to maintain the probability of false negatives within a certain threshold. Our experiments show that a value of 2 or 3 for the degree of in-network aggregation is reasonable for MICA2 platform. If this value is not large enough to maintain the false positives within the desired threshold, then we recommend using a second tier of false alarm processing at the base station.

The above discussion motivates us to develop an analytical model in the future that captures the tradeoff between the key parameters, such as the degree of aggregation, density of node deployment, sleep duration, and the maximum probability of false alarms that a user can tolerate. Such a model can then be used to choose the appropriate degree of aggregation, when the values of the other parameters are known. Such a model is also valuable in estimating the probability of false alarms that a user can expect for a specific design and configuration.

7.3 Evaluation of Sentry Service

In this section, we analyze the key features of the sentry service component. We first analyze the stealthiness of the power management scheme, and then assess the extension in lifetime achieved for different sentry distributions and for different periods of the sleep-wakeup cycle of the non-sentries.

7.3.1 *Stealthiness of Power Management Component*

In Section 5.5, we compared and contrasted the proactive and reactive schemes for controlling the sleep-wakeup cycle of the non-sentry motes when power management is enabled. The proactive scheme provides better responsiveness when an event occurs, at the cost of transmitting more messages in the absence of an event. In contrast, the reactive scheme provides better stealthiness during the idle periods, at the cost of retransmitting multiple messages in order to awaken the non-sentries when an event occurs. A sentry chooses the interval between successive retransmissions in such a way that the beacon transmission coincides with the wakeup period of the neighboring non-sentry motes. We use the following equation to control the number of retransmissions of the awake beacon (n_r).

$$n_r = \frac{sleepDuration + awakeDuration}{awakeDuration + 1} \qquad (1)$$

A larger value of *awakeDuration* results in fewer retransmissions of the awake beacon when a sentry detects an event. However, if the motes are awake longer, more energy is consumed and therefore, the lifetime of the sensor network reduces.

Higher message overhead also translates to higher energy

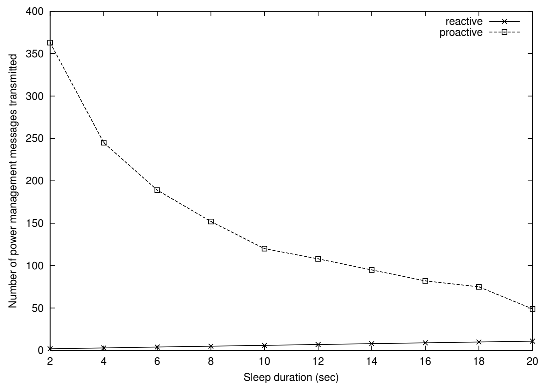


Figure 10: Power Management Message Overhead

consumption. In order to compare the message overhead between the reactive and proactive schemes, we implemented both the schemes and conducted experiments using the Nido simulator[19], a simulator that actually runs our system and TinyOS code. We simulated a simple scenario in which a tank moved across a sensor field in which 10 motes capable of magnetic sensing were deployed. The duration of each simulation run was 600 seconds. The *awakeDuration* of the motes was fixed at 2 seconds for each run. Figure 10 compares the number of messages sent out by the proactive and reactive schemes during the tracking phase when power management is enabled.

Figure 10 shows that the number of power management messages in the reactive scheme increases from 2 to 11 as the sleep duration increases from 2 seconds to 20 seconds. This is justified by Equation 1, which indicates that a longer sleep duration requires more retransmissions of the awake beacon, in order to ensure that one of the beacons is received by the non-sentry motes. In contrast, the message overhead in the case of the proactive scheme reduces as the sleep duration increases. This is because the periodicity with which a sentry sends out the sleep beacon is equal to $sleepDuration + awakeDuration$. As the sleep duration increases, the sleep beacons are sent out less frequently, thereby reducing the message overhead.

The results in Figure 10 also show that the message overhead due to power management is significantly lower in the reactive scheme compared to its proactive counterpart. This suggests that the reactive scheme is more stealthy compared to the proactive scheme. While this is true for the 2 second awake period we have chosen, it may not be true for smaller values of *awakeDuration*. In our experiment, we chose a relatively high value of 2 seconds for *awakeDuration*, in order to compensate for the high rate of drift in the software timers in the current TinyOS implementation. If the timer drift is smaller in future implementations of TinyOS, we would choose a smaller awake duration for the motes, so that the overall energy consumption of the network can be reduced. However, a smaller value of *awakeDuration* would increase the message overhead for the reactive scheme. We have currently adopted the reactive scheme for our surveillance application, because it provides better stealthiness for the duration of the sleep-wakeup cycle we have chosen. However, an investigation into a hybrid scheme that combines the advantages of both the proactive and reactive schemes would be worthwhile to pursue as future work. In addition, the

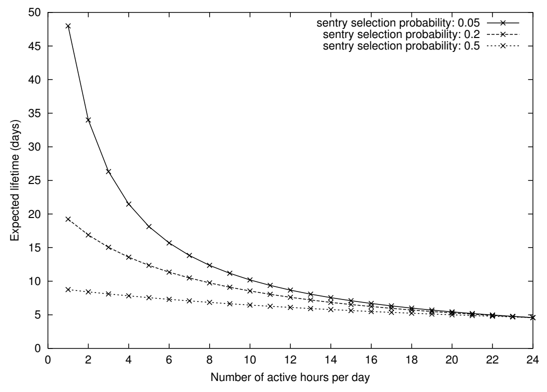


Figure 11: Expected Lifetime of a Sensor Network Using Sentry-based Power Management

hardware solution mentioned in [8] might also be an alternative strategy for aggressive energy conservation.

7.3.2 Power Savings

One of the main goals of the sentry service module is the extension of the lifetime of the sensor network. The sentry service extends the lifetime by conserving the energy consumption of the motes when the network is idle. Non-sentry motes alternate between sleep and wakeup states, and in Section 7.3.1, we justified our choice of a timer-driven, reactive approach to control the sleep-wakeup cycle. When a mote is in the sleep state, its radio is turned off, all of its I/O ports are configured appropriately to minimize the current consumption, the ADC module is turned off to disable any sampling, and the controller is placed in a power-save state. When the sleep timer expires, the controller is awakened by a timer interrupt, and all of the modules resume activity. The extent to which our power management approach increases the lifetime of a mote depends on the fraction of time the mote spends in the sleep state. We now use the current consumed in the sleep and wakeup states using the above power management scheme to predict how the expected lifetime of a sensor network varies with the fraction of sentries selected.

A MICA2 mote is powered by a pair of AA batteries, supplying a combined voltage of 3V. Assuming that a pair of batteries will supply 2200 mAh at 3V [18], we can estimate the lifetime of a mote, if we know the current consumed in the sleep and wakeup states and the duty cycle of the mote. The duty cycle of a mote is the number of hours per day it remains awake polling for events. Based on our measurements, we found that a MICA2 mote equipped with a magnetic sensor board and running our sentry-based power management software consumes 20 mA in the wakeup state. The wakeup current includes the current consumed by the magnetometer to sample at a rate of 10 samples per second. On the other hand, we measured the sleep current of the mote to vary between 50 μA to 130 μA, which results in a 99% reduction in the current consumption. We use a sleep current of 130 μA for the discussion in this section.

From the above data, we can determine the lifetime of a sensor network that uses our sentry-based power management scheme. The lifetime of a sensor network depends on the fraction of sentries selected and the fraction of time the non-sentry motes remain awake. Let $P(s)$ denote the probability that a mote is selected as a sentry, and $P(a)$ denote

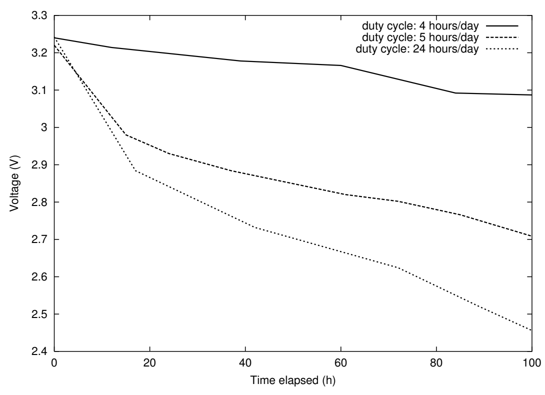


Figure 12: Impact of Sleep Duration on Power Consumption

the probability that a non-sentry mote is awake. The total current (C) consumed by a mote in the baseline case, when there are no events in the network, is given by Equation 2. The lifetime of the motes, L, is the ratio of the battery capacity to the total current consumed. Assuming a battery capacity of 2200 mAh, the lifetime of the motes in hours is simply $2200/C$.

$$C = P(s) * 20 + (1 - P(s)) * (P(a) * 20 + (1 - P(a)) * 0.13) \quad (2)$$

Figure 11 uses the above equation to predict the expected lifetime of the motes for different percentages of their duty cycle. The actual values of $P(s)$ and $P(a)$ are measured from the our prototype system. A mote that is always asleep is expected to survive for 2 years, whereas a mote that is always awake (i.e. always remains a sentry), can survive only up to 5 days. The exponential curves show that the lifetime greatly improves when the duty cycle is low. For example, when the probability that a mote is selected as a sentry is 0.5, and its duty cycle is reduced from 24 hours per day to one hour per day, its lifetime extends by nearly 100%. The graphs also show that the lifetime improves significantly as the number of sentries is reduced. For example, when the probability that a mote is selected as a sentry is reduced to 0.05, and its duty cycle is reduced to 4%, its lifetime extends by nearly 900%. The probability of selecting a mote as a sentry involves a tradeoff between the sensing coverage that can be achieved and the required network lifetime. A higher probability results in more sentries and provides better sensing coverage. However, it also reduces the lifetime of the network, as Figure 11 shows. In order to reduce the number of sentries without adversely affecting the sensing coverage, we can either choose magnetometers with a higher sensing range or increase the density with which the motes are deployed. For example, in our experiments we found that when the motes were placed at a distance of 8 ft from each other, the probability that a mote was selected as a sentry was nearly about 40%. However, in a more dense deployment in which the motes were placed within a few inches from each other, the probability of selecting a mote as a sentry dropped to about 20%. The reason is that a dense deployment results in a larger number of neighbors for each mote. Therefore, a single sentry is able to cover more neighbors, and that gives fewer motes a chance to elect themselves as a sentry.

In addition to predicting the lifetime of the network using a simple model, we also conducted experiments to compare the rate at which energy is dissipated for differe nt duty cycles in an actual deployment. In each of our experiments we deployed 6 motes, all equipped with magnetic sensor boards, inside an office building. Sentry rotation occurred once every 4 hours. Since there is no direct way to measure the energy consumed by the motes, we used the voltage drop across the batteries supplying power to the motes as an indirect way to measure the energy dissipation. We measured the voltage for each mote at regular intervals over a period of 100 hours and found that the voltage drop was reasonably uniform across the motes. Figure 12 shows the voltage drop during the observation period for one of the 6 motes for different values of duty cycles. From the figure, we see that the battery voltage for a mote does not drop uniformly with time. One of the reasons for the non-uniform energy dissipation is the periodic rotation of the sentry responsibility. The voltage drop of a mote is higher during an interval in which it is serving as a sentry than when it is serving as a non-sentry because the periodic sampling operation performed by a sentry consumes significant energy. The results also confirm that a higher duty cycle results in a higher energy dissipation. We see that when the mote is always awake, it loses most of its capacity within 100 hours (about 4 days). This reasonably matches with the results in Figure 11, which predicted that a mote operating 100% of the time will last only 5 days.

The experimental results we obtained are promising in that they show that the sentry-based power management algorithm is adaptive and that it is successful in extending the lifetime of the sensor network. While our current sentry selection algorithm does not choose the minimal number of sentries, by knowing the lifetime of the mission in advance, we can choose the density of deployment and the duty cycle in such a way that the lifetime requirement can be met.

8. LESSONS LEARNED

The work described in this paper is our experience in building a complete system for using wireless sensor networks for a practical application, and evaluating it through an actual deployment of motes. This practical experience has been valuable, because it has taught us that some of the simplified assumptions made about the hardware platform and operating system in much current research do not hold well in practice. The lessons we learned have greatly impacted some of the design choices we had to make in building our system.

1. **Application-specific Reliability :** We found that the packet loss in the MICA2 platform can be as large as 20%. A well-known approach to counter message loss is to retransmit the message multiple times, in order to improve the probability of delivery. Such retransmissions can be initiated either in the lower layers of the protocol stack or at the application layer. Since retransmitting a message consumes significant energy, it is important that the messages are retransmitted selectively, based on application-specific knowledge. For instance, applications that transmit ephemeral sensor readings, such as the instantaneous temperature, may not require reliability. Lower layers, such as the MAC layer, often lack domain-specific knowledge. So implementing reliability guarantees in the lower layers makes it harder to provide application-specific relia-

bility. Hence, for a system that strives to achieve energy efficiency, providing reliability guarantees at the application layer is a better option.

2. **False Alarm Reduction:** We found that our sensors generated false alarms at a non-negligible rate. This introduces unnecessary energy consumption and inappropriate actions. False alarms we experienced can be categorized into two major types: Transient and persistent false alarms. A simple exponential weighted moving average (EWMA) on the mote is sufficient to deal with transient false alarms such as the burst distortion of sensing readings. However, if the false alarms are persistent due to errors in the sensor device, more advance techniques are desired. In our system, we successfully eliminated individual persistent false alarms by utilizing in-network aggregation with a relatively high DOA value. In the worst case, when multiple persistent false alarms are generated simultaneously, we are able to filter out such false alarms by analyzing spatial-temporal correlations among the consecutive reports at the base station.

3. **Race Conditions Reduction:** Race conditions are another example of a phenomenon that is often ignored in simulation-based approaches, but must be addressed when building the running system. For example, contention occurs not only when different motes try to transmit simultaneously, but also when different software components on the same mote initiate transmissions simultaneously through split-phase operations. Due to the limited support from TinyOS, the latter can lead to race conditions. Race conditions can be avoided, if the OS can support synchronized processing, based on semaphores, in order to coordinate the shared resources among the contending modules. While TinyOS supports concurrency control through atomic sections and tasks, it is more flexible and efficient to use application level synchronization such as packet scheduling mentioned in Section 6.1 to coordinate the operations.

4. **Asymmetry Reduction:** Another issue we had to address was to account for the effect of asymmetric channels which is largely ignored in simulation approaches. Communication in low power devices, such as the motes, is asymmetric [24] due to differences in hardware, signal attenuation, and residual battery capacity. In practice, we were able to reduce the effect of asymmetric channels by restricting a mote to communicate with only those neighbors that are well within its communication range. This can be achieved by reducing transmission power during the network establishment as we mentioned in Section 5.1.2. Moreover, it also can be achieved by bounding relay distance , if the localization is available.

5. **Software Calibration:** In a simulation-based approach, it is common for sensor devices of the same type to generate the same readings under identical conditions. However, in practice, the same type of sensors are capable of generating quite different sensor readings under identical conditions. Such a phenomenon may occur because of differences in the way the devices are manufactured, and it is often hard to accurately capture those differences in a simulator. We found that the impact of such heterogeneity is significant in the MICA2 platform, such as shown in Figure 6. The variance in the sensor readings can be accounted for at the very outset through software calibration of the sensors.

6. **Other Lessons:** The drift in the software timers in TinyOS presents another practical issue, especially when motes transit into sleep state. In order to compensate for the drift in the soft timers, we need to increase the duration for which a mote remains awake, and design appropriate strategies to control the sleep-wakeup cycle, as described in Section 7.3.1. Another practical challenge we faced was the lack of appropriate tools for debugging a network of motes. We utilize the dynamic configuration method mentioned in 5.1.3 and overhearing tools to facilitate our work. However, more sophisticated debugging and configuration tools will greatly ease the burden on the programmer in the future. We acknowledge that our design choices sometimes are restricted by limited hardware and operation system support. It is desirable to have new features such as interruptible snoozing, sub-controllers for I/O, a more reliable RF module and process management, so that we can improve our design and implementation in the future.

9. CONCLUSIONS

Research in wireless sensor networks has been very active. Most of the published work studies an individual protocol and performs evaluations via simulations. In contrast, in this work we implement an entire integrated suite of protocols and application modules and evaluate the performance on a system composed of 70 MICA2 motes in a realistic outdoor setting. Empirical results identify the capability of the MICA2 radio, the value of in-network aggregation with respect to transmission overhead, false alarm processing and application layer tracking latency, and the value of power management. Design decisions and how those decisions were influenced by the empirical data were described. Key lessons learned were also itemized. From our experience in building and analyzing this system it is clear that key realistic hardware, software and environmental issues must not be ignored in developing usable solutions. This includes realism of sensor performance, asymmetries in communication, false alarms, and race conditions.

10. FUTURE WORK

System design and engineering is one of the keys to bring sensor network paradigm into reality. The system described in this paper is still an ongoing prototype. Many outstanding design issues are yet to be resolved. We are currently investigating 1) target classification under constraint resources through collaborative data fusion, 2) the possibility to design a more aggressive power management strategy with passive wake-up capabilities [8], 3) approaches to build extremely robust routing infrastructure, which can survive under hostile environments, 4) a practical localization scheme and 5) a scalable architecture up to thousands of nodes while maintaining operational performance requirement.

11. REFERENCES

[1] S. Bhattacharya, H. Kim, S. Prabh, and T. Abdelzaher. Energy-Conserving Data Placement and Asynchronous Multicast in Wireless Sensor Networks. In *The First International Conference on Mobile Systems, Applications, and Services (MobiSys)*, May 2003.

[2] B. M. Blum, P. Nagaraddi, A. Wood, T. F. Abdelzaher, S. Son, and J. A. Stankovic. An Entity Maintenance and Connection Service for Sensor Networks. In *The First Intl. Conference on Mobile Systems, Applications, and Services (MobiSys)*, May 2003.

[3] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *6th ACM MOBICOM Conference*, 2001.

[4] CrossBow. *Mica2 data sheet*. Available at `http://www.xbow.com/Products/Product_pdf_files/MICA%20data%20sheet.pdf`.

[5] J. Elson and K. Romer. Wireless Sensor Networks: A New Regime for Time Synchronization. In *Proc. of the Workshop on Hot Topics in Networks (HotNets)*, October 2002.

[6] Z. Feng, S. Jaewon, and R. James. Information-Driven Dynamic Sensor Collaboration for Target Tracking. *IEEE Signal Processing Magazine*, 19(2), Mar 2002.

[7] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC Language: A Holistic Approach to Networked Embedded Systems. In *Proceedings of Programming Language Design and Implementation (PLDI) 2003*, 2000.

[8] L. Gu and J. A. Stankovic. Radio-Triggered Wake-Up Capability for Sensor Networks. In *Proceedings of RTAS*, 2004.

[9] T. He, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher. AIDA: Adaptive Application Independent Data Aggregation in Wireless Sensor Networks. *ACM Transactions on Embedded Computing System, Special issue on Dynamically Adaptable Embedded Systems*, 2004.

[10] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-Free Localization Schemes in Large-Scale Sensor Networks. In *Proc. of the Intl. Conference on Mobile Computing and Networking (MOBICOM)*, September 2003.

[11] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In *Proc. of the Intl. Conference on System Sciences*, January 2000.

[12] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System Architecture Directions for Networked Sensors. In *Proc. of Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 93–104, 2000.

[13] Honeywell. *1- and 2-Axis Magnetic Sensors*. Available at `www.ssec.honeywell.com/magnetic/datasheets/hmc1001-2_1021-2.pdf`.

[14] M. Horton, D. E. Culler, K. Pister, J. Hill, R. Szewczyk, and A. Woo. MICA: The Commercialization of Microsensor Motes. *Sensors Online*, April 2002. www.sensorsmag.com/articles/0402/40.

[15] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *the Sixth Annual International Conference on Mobile Computing and Networks*, 2000.

[16] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next Century Challenges: Mobile Networking for Smart Dust. In *Proc. of Intl. Conference on Mobile Computing and Networking (MOBICOM)*, August 1999.

[17] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks. In *Operating Systems Design and Implementation*, December 2002.

[18] A. Mainwaring, J. Polastre, R. Szewczyk, D. E. Culler, and J. Anderson. Wireless Sensor Networks for Habitat Monitoring. In *Proc. of the ACM Workshop on Sensor Networks and Application (WSNA)*, September 2002.

[19] TOSSIM: A Simulator for TinyOS Networks. Available at `webs.cs.berkeley.edu/tos/tinyos-1.x/doc/nido.pdf`.

[20] R. Powers. Batteries for Low Power Electronics. *In Proceedings of the IEEE*, pages 687–693, April 1995.

[21] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks. In *First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, November 2003.

[22] T. Yan, T. He, and J. Stankovic. Differentiated Surveillance Service for Sensor Networks. In *First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, November 2003.

[23] F. Ye, G. Zhong, S. Lu, and L. Zhang. PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks. In *Proc. of International Conference on Distributed Computing Systems (ICDCS)*, May 2003.

[24] G. Zhou, T. He, and J. A. Stankovic. Impact of Radio Irregularity on Wireless Sensor Networks. In *The Second International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2004.

Impact of Radio Irregularity on Wireless Sensor Networks

Gang Zhou, Tian He, Sudha Krishnamurthy, John A. Stankovic

Department of Computer Science
University of Virginia, Charlottesville, VA 22903
{gz5d, tianhe, skrish, stankovic}@cs.virginia.edu

ABSTRACT

In this paper, we investigate the impact of radio irregularity on the communication performance in wireless sensor networks. Radio irregularity is a common phenomenon which arises from multiple factors, such as variance in RF sending power and different path losses depending on the direction of propagation. From our experiments, we discover that the variance in received signal strength is largely random; however, it exhibits a continuous change with incremental changes in direction. With empirical data obtained from the MICA2 platform, we establish a radio model for simulation, called the Radio Irregularity Model (RIM). This model is the first to bridge the discrepancy between spherical radio models used by simulators and the physical reality of radio signals. With this model, we are able to analyze the impact of radio irregularity on some of the well-known MAC and routing protocols. Our results show that radio irregularity has a significant impact on routing protocols, but a relatively small impact on MAC protocols. Finally, we propose six solutions to deal with radio irregularity. We evaluate two of them in detail. The results obtained from both the simulation and a running testbed demonstrate that our solutions greatly improve communication performance in the presence of radio irregularity.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Network**]: Network Architecture and Design; I.6 [**Computing Methodologies**]: Simulation and Modeling

General Terms

Design, algorithms, measurement, performance, experimentation

Keywords

Sensor networks, wireless communication, radio irregularity, sending power, path loss, link asymmetry, packet loss

1. INTRODUCTION

Radio irregularity is a common and non-negligible phenomenon in wireless sensor networks. It results in irregularity in radio range and variations in packet loss in different directions, and is considered as an essential reason for asymmetric links as viewed by upper layers in the protocol stack. Several empirical studies [4][9][23][28] on the Berkeley mote platform have shown that the radio range varies significantly in different directions and the percentage of asymmetric links in a system varies depending on the average distance between nodes.

The impact of radio irregularity on protocol performance can be investigated through a running system. However, few researchers have actually pursued this direction, because of two reasons: First, the complexity and cost of performance evaluations on a running system escalate, when sensor networks scale up to thousands or more nodes. Second, repeatable results of radio performance are extremely hard to obtain from uncontrolled environments, hence leading to difficulties in system tuning and performance evaluation. As a result, simulation techniques are used as an efficient alterative to evaluate protocol performance. Unfortunately, most existing simulations don't take radio irregularity, a common phenomenon in wireless communication, into account. The spherical radio patterns assumed by simulators such as [27] may not approximate real radio properties well enough and hence may lead to an inaccurate estimation of application performance.

Several researchers [4][9][23][28] have already shown extensive evidence of radio irregularity in wireless communication. Their main focus is to observe and quantify such phenomena. This paper is distinguished from the previous ones for the initiative in bridging the gap between spherical radio models used by simulators and the physical reality of radio signals. We first verify the presence of radio irregularity using empirical data obtained from the MICA2 platform. The results demonstrate that the radio pattern is largely random; however, it exhibits a continuous change with incremental changes in direction. Based on experimental data, a radio model for simulations, called the Radio Irregularity Model (RIM), is formulated. RIM takes into account both the non-isotropic[1] properties of the propagation media and the heterogeneous properties of devices.

[1] Exhibiting properties with different values when measured along axes in different directions.

With the help of the RIM model, we explore the impact of radio irregularity on MAC and routing performance. Among the protocols we evaluate, we find that radio irregularity has a significant impact on the routing protocols, but a relatively small impact on the MAC protocols. We also find that location-based routing protocols, such as Geographic Forwarding (GF) [17] perform worse in the presence of radio irregularity than on-demand protocols, such as AODV [21] and DSR [14]. We propose several potential solutions to deal with radio irregularity in wireless sensor networks. We evaluate two of them through a simulation and a running system, respectively. Our results illustrate that our solutions do succeed in alleviating the performance penalties due to radio irregularity.

The rest of this paper is organized as follows: we briefly analyze the causes and impact of radio irregularity in Section 2. In Section 3, we describe experimental data collected from the MICA2 platform and make some general conclusions about radio irregularity. Based on these conclusions, we propose the RIM radio model in Section 4. We then use the RIM model in simulations to analyze the impact of radio irregularity on MAC and network layer protocols in Section 5 and Section 6, respectively. Solutions to deal with radio irregularity are proposed and evaluated in Section 7. Finally, we conclude the paper in Section 8.

2. ANALYSIS OF RADIO IRREGULARITY

In this section, we first identify the causes of radio irregularity, and then briefly discuss the impact of irregularity on the different protocol layers.

2.1 Causes of Radio Irregularity

Radio irregularity is caused by two categories of factors: devices and the propagation media. Device properties include the antenna type (directional or omni-directional), the sending power, antenna gains (at both the transmitter and receiver), receiver sensitivity, receiver threshold and the Signal-Noise Ratio (SNR). Media properties include the media type, the background noise and some other environmental factors, such as the temperature and obstacles within the propagation media.

In general, the radio irregularity is caused by the non-isotropic properties of the propagation media and the heterogeneous properties of devices. Among all these factors, we focus on the non-isotropic path losses and the differences in signal sending power, which are commonly regarded as the key causes of radio irregularity.

- **Non-isotropic Path Losses:** The variance in the signal path loss is one of the major causes of radio irregularity. When a signal propagates within a medium, it may be reflected, diffracted, and scattered [22]. Reflection occurs when an electromagnetic signal encounters an object, such as a building, that is larger than the signal's wavelength. Diffraction occurs when the signal encounters an irregular surface, such as a stone with sharp edges. Scattering occurs when the medium through which the electromagnetic wave propagates contains a large number of objects smaller than the signal wavelength. The medium is normally different in different directions. Consequently, radio propagation exhibits non-isotropic patterns in most environments.

  Another significant reason for non-isotropic path loss is hardware calibration. A node may not have the same antenna gain along all propagation directions, possibly due to hardware manufacturing. Hence, the non-isotropic antenna gain of each node also contributes to the non-isotropic path loss.

- **Heterogeneous Sending Powers:** Sensor devices may transmit RF signal at different sending powers, even though they are the same kind of devices. This difference may arise from some random factors during the manufacture of sensor devices. In addition, after the sensor devices are deployed, the batteries of different sensor devices deplete at different rates, due to different workloads and different environments in which they are deployed. Heterogeneous sending powers result in variable communication ranges, and cause non-isotropic connectivity.

2.2 Impact of Radio Irregularity

Radio irregularity is a non-negligible phenomenon in wireless communication. It's an essential reason for asymmetric radio interference and asymmetric links in upper layers. It can directly or indirectly affect many aspects of upper layer performance.

Asymmetric radio interference between neighboring nodes affects the correctness of MAC layer functions. For example, in the presence of radio irregularity, a node might not be able to successfully reserve the wireless channel through RTS and CTS handshaking, because those neighboring nodes of the receiver, which cannot hear the CTS control packet, might disrupt the receiving node. This impacts the delivery ratios of data frames at the MAC layer.

Radio irregularity can also affect the performance and even correctness of networking protocols such as [12] [14] [16] [17] [21]. For example, link asymmetry is one of the ways in which radio irregularity manifests itself at the higher layer. Link asymmetry has an adverse impact on protocols that use path-reversal techniques to establish an end-to-end connection.

Actually, the impact of radio irregularity is not only confined to the MAC and routing layers, radio irregularity also influences other protocols, such as the localization, sensing converge and topology control protocols.

Localization protocols such as DV-Hop [20] and Centroid [2] assume a spherical radio range. The study in [10] shows that the performance of such protocols degrades when the radio range becomes irregular. The sensing coverage scheme in [26] assumes that sensing and communication ranges are spherical. In the presence of radio irregularity, they might not be able to guarantee full coverage and blind points would occur. The topology-control scheme in GAF [25] builds a communication mesh based on the assumption of a spherical range. This might lead to the network partition in the presence of a non-spherical range. We note that some other topology-control protocols, such as ASCENT [3] and Span [5] don't depend on such an assumption, however, performance evaluations of those protocols considering radio irregularity are desired.

Due to space limitation, in the rest of paper, we only focus on the impact of radio irregularity on MAC and routing performance and leave the rest as future work.

3. RADIO IRREGULARITY IN REALITY

We conduct several experiments to study the irregularity of the radio using MICA2 motes, and in this section we discuss some of the experimental results we obtain in an outdoor environment. Our results confirm that radio propagation is largely non-isotropic and exhibits a continuous variation with incremental changes in direction.

3.1 Experimental Setup

We use a pair of MICA2 motes for our experiments. One of the motes periodically transmits probing beacons and the other mote samples its ADC port while receiving these beacons. The ADC reads the signal on the analog pin of the Chipcon transceiver [6] and converts it into a 10-bit voltage value. The voltage reading is mapped into the received signal strength in dBm according to the specification in [6]. All experiments are conducted in an open parking lot near a building.

3.2 Experimental Results

In this section, we demonstrate the presence of radio irregularity using three different metrics: 1) the received signal strength, 2) the packet reception ratio and 3) the communication range.

1) Non-isotropic Signal Strength: In the first experiment, the receiver is placed 10 feet away from the sender (both on the ground) and the received signal strength is measured in four different geographical directions by sampling 100 beacons received in each direction.

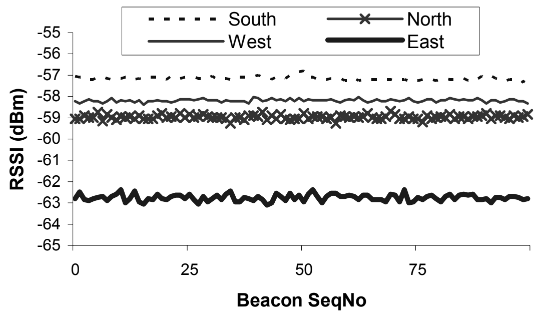


Figure 1: Signal Strength over Time in Four Directions

Figure 1 shows that the received signal strength in each direction is relatively stable over time (The small variance comes from the fading effect [22]). However, the signal strength received in the south is much higher than that received in the east, although nodes have the same distance from the sender. We also measure the variation of signal strength with the changes in the angular direction of the receiver with respect to the sender. Figure 2 shows the variation of the received signal strength as a function of the angular direction with respect to the sender, when the distance between the sender and receiver is 10 feet and 20 feet, respectively. These results show that the received signal strength varies continuously with the direction. In other words, incremental changes in direction result in incremental variation in the received signal strength.

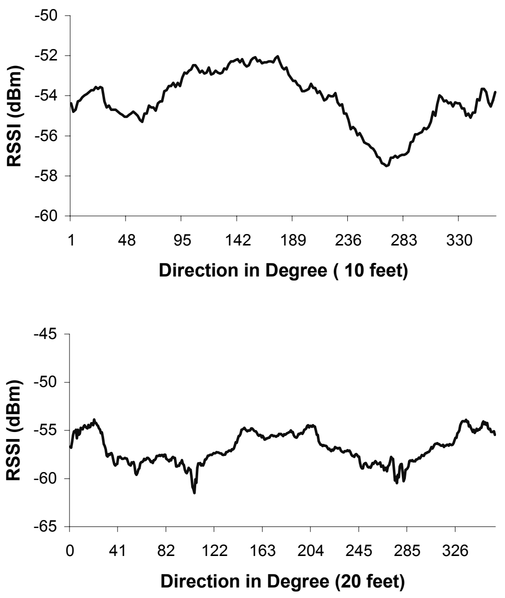


Figure 2: Signal Strength Values in Different Directions

2) Non-isotropic Packet Loss Ratio: Figure 3 shows how the packet reception ratio varies in different directions. When the sender and receiver are placed 10 feet apart, the packet reception ratio is nearly 100% in all the directions, because the signal is still strong in all the directions. However, when they are placed 20 feet apart, there is a 90% packet loss in the east direction. This result is consistent with the results shown in Figure 1, which demonstrates that the received signal strength measured in the east is lower than that in the other three directions.

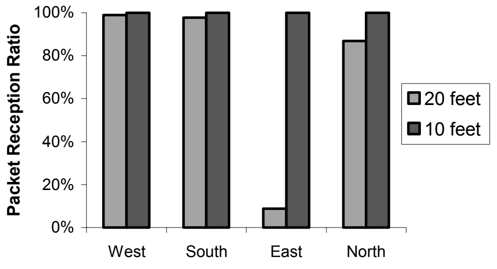


Figure 3: Non-isotropic Packet Reception

3) Non-isotropic Radio Range: Another aspect in which we demonstrate the irregularity is to show that the communication range of a mote is not uniform in all directions. In the experiment, we fix the received signal strength threshold at -55.5 dBm and -59 dBm, respectively. Then with such thresholds, we measure the communication ranges in different directions. Figure 4 shows the communication range of a mote as the receiver direction varies from degree 0 to degree 359. The range map shown in Figure 4 is another confirmation of radio irregularity in a wireless medium.

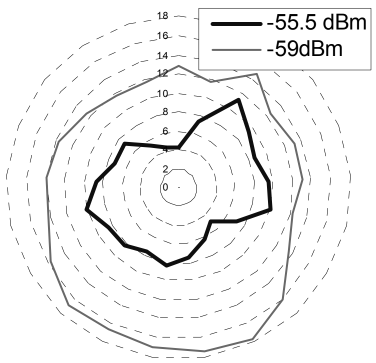
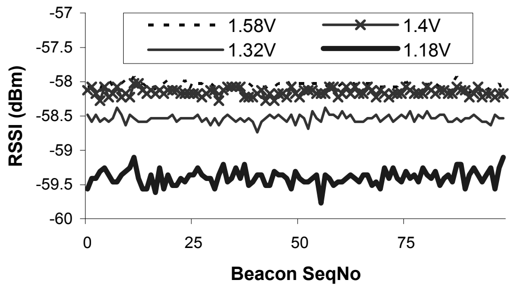
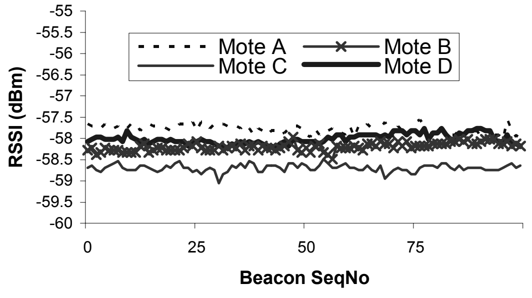


Figure 4: Non-isotropic Range

4) Range Irregularity with Varying Sending Power: We also investigate the received signal strength when the sending power varies due to different battery status and different hardware calibration. In Figure 5 (a), we use the same sender and receiver, placed 10 feet apart. We change the batteries at the sender side each time. The result indicates that different battery status at the same sender can affect the received signal strength. In Figure 5 (b), we use the same batteries but in different senders each time. The same receiver is used, placed 10 feet apart from the sender. The result shows that different senders with the same batteries can also affect the received signal strength.



(a) One mote with different battery status



(b) Different motes with the same battery status

Figure 5: Radio Irregularity with Sending Powers

3.3 Summary of Experimental Results

From the experimental results, we infer that the radio of sensor devices has the following main properties:

1. **Non-isotropic**: The radio signal from a transmitter has different path loss in different directions (Figure 1 and Figure 2).
2. **Continuous variation**: The signal path loss varies continuously with incremental changes of the propagation direction from a transmitter (Figure 2 and Figure 4).
3. **Heterogeneity**: Differences in hardware calibration and battery status lead to different signal sending powers, hence different received signal strengths (Figure 5).

4. MODEL THE RADIO IRREGULARITY

As we have shown in our experiments as well as in other research results [4] [9] [23] [28], radio irregularity is a common phenomenon in wireless sensor networks. Therefore, it is essential for wireless simulations to capture such effects. This section describes our effort to model such a phenomenon in simulation environments.

4.1 Isotropic Radio Models

In isotropic radio models, the received signal strength is usually represented with the following formula:

Received Signal Strength = *Sending Power*–*Path Loss*+*Fading* (1)

The *Sending Power* of a node is determined by the battery status and the type of transmitter, amplifier and antenna. *Path Loss* describes the signal's energy loss as it travels to the receiver. Many models are used to estimate the *Path Loss*, such as the free-space propagation model, the two-ray model and the Hata model [22]. All these models are isotropic, meaning that the signal attenuates exactly the same in all directions. However, our experience as well as results obtained by others [4] [9] [23] [28] all indicate that the isotropic models don't hold well in practice.

4.2 Radio Irregularity Model (RIM)

The RIM model proposed here is an extension to isotropic radio models. It enhances isotropic radio models by approximating three main properties of radio signals: non-isotropic, continuous variation and heterogeneity, as we summarized in section 3.3. These properties are normally ignored by previous isotropic radio models.

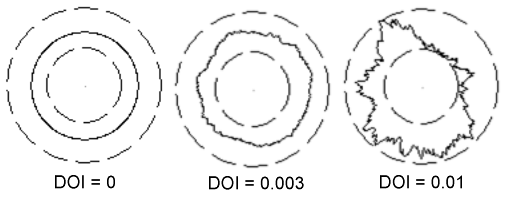


Figure 6: Degree of Irregularity

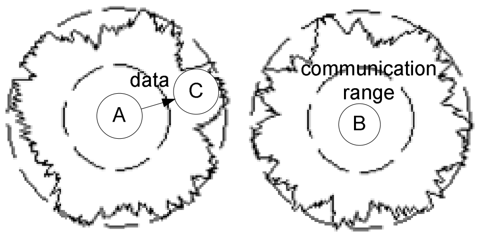
The RIM model is motivated by a simple DOI (Degree of Irregularity) model briefly mentioned in the localization work [10]. In the DOI model, the DOI is used to denote the irregularity of the radio pattern. It was originally defined as the maximum range variation per unit degree change in the direction of radio propagation. As seen in Figure 6, when the DOI is set to zero, there is no range variation, and the communication range is a perfect sphere. However, when we increase the DOI value, the communication range becomes more and more irregular.

The DOI model assumes an upper and lower bound on signal propagation (Figure 6). Beyond the upper bound, all nodes are out of communication range; and within the lower bound, every node is guaranteed to be within the communication range. If the distance between a pair of nodes is between these two boundaries, three scenarios are possible: 1) symmetric communication, 2) asymmetric communication, and 3) no communication.

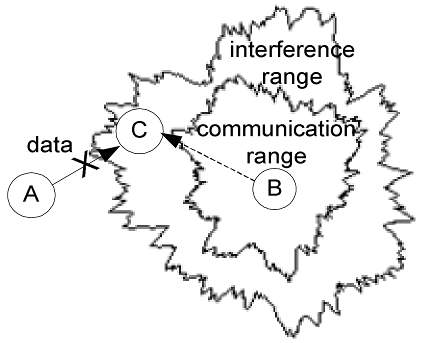
The DOI model is a good start to model signal irregularity. However, it doesn't model interference in real devices well. Since the DOI model is based on an absolute communication range, it assumes that within the inner range, the signal is very strong and can always be received correctly, while beyond the outer range there is no signal at all. This binary pattern is not true in reality. For example, in Figure 7 (a), the DOI model assumes that there is no interference between nodes B and C.

However in reality, there are no such clear boundaries and the communications of nodes do interfere with each other. Different from the DOI model, the RIM model proposed here takes the radio sending energy, the energy loss, the background noise, and the interference among different communication signals into account.

The difference can be further explained with an example. In Figure 7 (b), the RIM model allows node B's signal to propagate beyond its communication range to reach node C, even though it is not strong enough for node C to receive it as a valid packet. This weak signal from node B acts as one source of background noise around node C. In this case, node C may not be able to receive packets from node A, if the received signal is not stronger than the product of the Signal-Noise Ratio (SNR) value and background noise level of node C.



(a) No interference in the DOI model



(b) Interference in the RAM energy model

Figure 7: Communication Interference

The DOI model only models an absolute range based on the distance and determines whether one node can hear another node only by comparing the distances between these two nodes with the sender's communication range. With such a binary decision, it can't deal with interference as we mentioned earlier.

To enhance the isotropic radio model and the DOI radio model, we propose the RIM model that combines the energy models and the DOI factor together. We redefine DOI in the RIM model as *the maximum received signal strength percentage variation per unit degree change in the direction of radio propagation.* We note that RIM is a general radio model which can default to the isotropic model when the DOI value is zero. Also, it can default to the DOI model when there is no interference among nodes.

We note that our RIM model is established based on data from real sensor devices. It is a hybrid approach, which introduces real data (DOI value) into simulations, so that the radio irregularity pattern in reality can be approximated well. DOI values can be calculated according to its definition. We repeat the experiments shown in Figure 2 six times and the results are in Figure 8. It shows that the variances of the received signal strength with incremental changes in direction are small, which validates our conclusion about continuous variation.

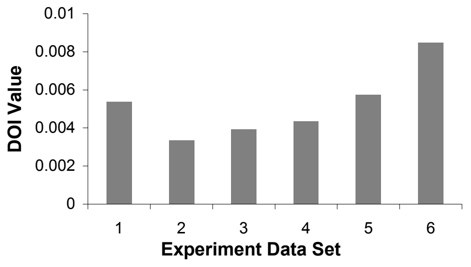


Figure 8: DOI Values from MICA2 Experiments

4.2.1 Non-isotropic Properties in RIM Model

Many models are used to estimate path loss, such as the free-space propagation model, the two-ray model and the Hata model [22]. These models are isotropic in the sense that the path losses in different directions are the same. To reflect the two main properties of radio irregularity, namely non-isotropic and continuous variation, we adjust the value of path loss models in Equation 1 based on DOI values, resulting in the following formula:

Received Signal Strength = Sending Power – DOI Adjusted Path Loss + Fading (2)

DOI Adjusted Path Loss = Path Loss × K_i (3)

Here K_i is a coefficient to represent the difference in path loss in different directions. Specifically, K_i is the ith degree coefficient, which is calculated in the following way:

$$K_i = \begin{cases} 1 \ , & i = 0 \\ K_{i-1} \pm Rand \times DOI \ , & 0 < i < 360 \ \wedge \ i \in N \end{cases}$$
$$\text{where } |K_0 - K_{359}| \le DOI \quad (4)$$

We can generate 360 K_i values for the 360 different directions, based on Equation 4, by randomly fixing a direction as the starting direction represented by *i*=0. For the direction which doesn't have an integer value of angle from the start direction, we interpolate the K_i value based on the values of the two adjacent directions which have integer angles from the starting direction.

$$K_i = K_s + (i-s) \times (K_t - K_s)$$
$$where \quad s = \lfloor i \rfloor \wedge t = \lceil i \rceil \bmod 360 \wedge 0 < i < 360 \wedge i \notin N \qquad (5)$$

The statistical analysis of our experimental data indicates that the variance of received signal strength in different directions fits the Weibull [7] distribution. The Weibull distribution can be used to model natural phenomena such as variation of wind speed, scattering of radiation, etc. The Rayleigh distribution, which is commonly used for modeling multi-path fading in wireless communication, is a special case of the Weibull distribution. Analysis details are provided in Appendix A. In Equation 4, we generate a random number according to the Weibull distribution.

4.2.2 *Heterogeneity Property in RIM Model*

Due to the difference in hardware calibration and battery status, received signal strength can be different from two sending nodes of the same type in the same experimental setting. In RIM, we use the variance of signal sending power to account for such a difference. We introduce a second parameter named VSP (Variance of Sending Power), which is defined as the maximum percentage variance of the signal sending power among different devices. The new signal sending power is modeled by the following equation:

VSP Adjusted Sending Power = Sending Power × (1 + Rand × VSP) (6)

In Equation 6, we assume that the variance of sending power follows a Normal distribution, which is broadly used to measure the variance caused by the hardware.

With the two parameters: DOI and VSP, the RIM model can be formulated as follows:

Received Signal Strength = VSP Adjusted Sending Power – DOI Adjusted Path Loss + Fading (7)

With the help of the RIM model, we explore the impact of radio irregularity on MAC and routing protocols in the next two sections, respectively.

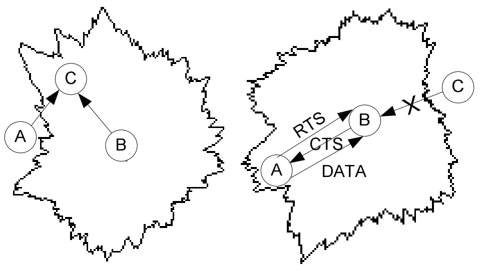
5. IMPACT ON MAC LAYER

In this section, we first analyze how operations in the MAC layer are affected by radio irregularity. We then quantify the degree of MAC performance degradation in the presence of radio irregularity.

5.1 Logical Analysis of the Impact

Most contention-based MAC protocols are based on carrier sensing or handshaking techniques. In this section, we analyze the impact of radio irregularity from the technical point of view.

- **Impact on Carrier Sensing:** Radio irregularity increases the chance for MAC protocols that use the carrier sensing technique to get involved in the hidden terminal problem. For example, in Figure 9 (a), while node B is transmitting packets to node C, due to the irregularity, node A cannot detect the signal from node B, so node A senses a clear channel and starts to transmit packets. As a result, a collision happens at receiver C. This scenario doesn't occur in isotropic radio situations, where node A detects node B's signal and refrains from transmitting the packets. Typical protocols using the carrier sensing technique are CSMA [18], MACA [15], MACAW [1] and 802.11 DCF [8].
- **Impact on handshaking:** The handshaking technique is specially designed to resolve hidden and exposed terminal problems. However, they cannot resolve the hidden and exposed terminal problems due to asymmetry, which can be produced by radio irregularity. This can be demonstrated in an example (Figure 9 (b)). We assume that node A sends a RTS message to node B, and then node B responds with a CTS message to node A. Any node overhearing the CTS message is supposed to wait long enough for node A to send out the data packet. If node C can't hear the CTS message from node B while node B can hear node C, there will be a collision if node C sends data. Similar examples can be found for the exposed terminal case.



(a) Carrier Sensing (b) Handshaking

Figure 9: Impact on MAC Protocols

5.2 Quantitative Analysis of the Impact

We implemented the RIM model in the radio layer of GloMoSim [27], a scalable discrete-event simulator developed by UCLA. We first describe our simulation configuration, and then evaluate the performance impact under different DOI and different VSP values, respectively.

5.2.1 *Simulation Configuration*

Table 1: **Simulation Configuration**

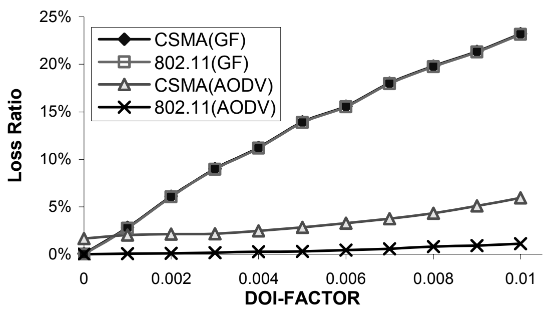
TERRAIN	(150m,150m)
Node Number	100
Node Placement	Uniform
Payload Size	32 Bytes
Routing Protocol	AODV, DSR, GF
MAC Protocol	CSMA, 802.11 (DCF)
Radio Model	RIM
Radio Bandwidth	200Kb/s

In the experiments, we use six CBR streams as the workload and set the CBR rate at a low rate, in order to isolate the effect of congestion and radio irregularity. We choose two typical MAC protocols: CSMA and 802.11 DCF. Two metrics are used: 1) the loss ratio (number of frames lost / number of frames sent) and 2) the average single hop delay of received packets. We vary the

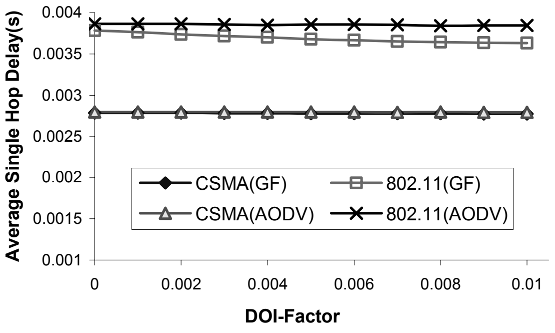
DOI and VSP values[2] separately in order to isolate and identify the impact individually. Each result shown in the graphs is an average of 140 runs. The 95% confidence intervals are within 0~25% of the mean.

In order to make our evaluation close to existing hardware proposed for use in wireless sensor network environments [24], we use the simulation configuration shown in Table 1. In all experiments, we investigate the range of DOI values according to the experimental data obtained from MICA2 motes as shown in Figure 8.

5.2.2 MAC Performance with Different DOI



(a)Loss Ratio vs. DOI



(b)Average Single Hop Delay vs. DOI

Figure 10: MAC Performances with Different DOI Values

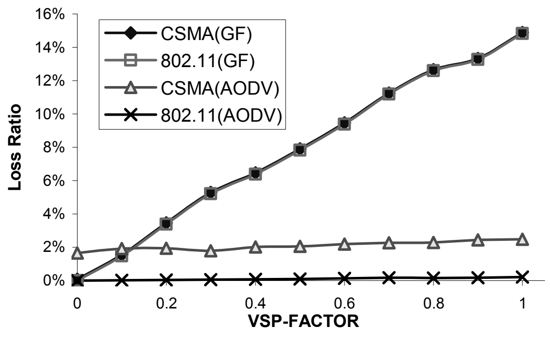
In the initial setup, we use Geographic Forwarding (GF) for the routing layer and compare the MAC performance between 802.11 and CSMA. We found that the MAC loss ratio increases rapidly with the increase in DOI values (Figure 10 (a)). However, 802.11 and CSMA yield roughly the same results. We realize that MAC performance can be strongly affected by routing, because an incorrect routing decision might lead to the failure at MAC layer. For instance, the routing layer designates that the MAC layer send a packet to a node that is out of reach. So we repeat the experiments with the AODV protocol as the routing layer. We find that MAC loss ratio increases slightly with the increase of DOI. Such a discrepancy is a strong indication that the radio irregularity has a much larger impact on routing protocols than MAC protocols. We explain this in more detail in Section 6.

[2] Spherical radio model is configured by setting DOI to zero.

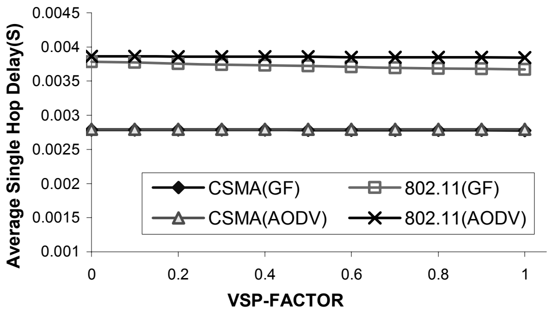
From Figure 10 (b), we can see that with the increase of DOI values, the average single hop delay remains almost the same. The reason is that increasing the DOI value only increases the communication asymmetry, but not the congestion. This is also a confirmation that packet loss in Figure 10 (a) is not due to congestion.

5.2.3 MAC Performance with Different VSP

In this experiment, we set the DOI value to zero, which means that the radio range is isotropic. However, different VSP values make radio ranges different among nodes.



(a) Loss Ratio vs. VSP



(b) Average Single Hop Delay vs. VSP

Figure 11: MAC Performances with Different VSP Values

The results shown in Figure 11 are similar to the results shown in Figure 10, which we obtain by varying the DOI values. The average single hop delay remains almost the same, because the different sending power only increases the degree of communication asymmetry, but does not increase the congestion.

The loss ratio increases with the increase of VSP values because the irregularity results in more asymmetric links. The loss ratio when AODV is used is much lower than when GF is used because asymmetric links have a larger impact on GF than on AODV. This result indicates that varying the VSP values has a much larger impact on routing protocols than on MAC protocols, which is similar to the behavior we observed by varying the DOI values.

6. IMPACT ON ROUTING LAYER

In this section, we analyze and quantify the impact of radio irregularity on routing protocols. We first discuss three techniques that are widely used in most routing protocols: path-reversal, multi-round discovery, and neighbor discovery. Our analysis shows that both path-reversal and neighbor-discovery are greatly influenced by radio irregularity. However, the multi-round discovery technique is able to deal with radio irregularity, but with relatively high overhead. Our simulation results also show that radio irregularity has a great impact on Geographic Forwarding (GF), but a small impact on AODV and DSR.

6.1 Logical Analysis of the Impact

We analyze the influence of radio irregularity on path-reversal, multi-round discovery, and neighbor-discovery techniques in this section.

6.1.1 Impact on Path-Reversal Technique

Protocols that use path-reversal technique are built based on the assumption that if there is a path from node A to node B, there is also a reverse path from node B to node A. The path may consist of a single link or multiple links. Most on-demand routing protocols used in ad hoc networks such as AODV [21], DSR [14], Direct Diffusion [13] and LAR [19] depend on this technique.

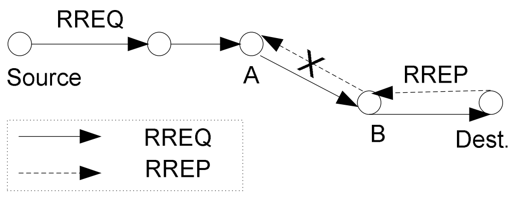


Figure 12: Impact on Path-Reversal Technique

Radio irregularity may result in asymmetric links and hence, it may have an adverse impact on protocols that use path-reversal techniques. For example, in Figure 12, node B can hear node A, but node A cannot hear node B. So even though there is a path from source S to destination D, we cannot assume that the reverse path from D to S exists. So during route discovery, if source S broadcasts a route request (RREQ) to discover the path to destination D, it may not be possible to deliver the reply (RREP) message to source S along the reverse path, even though node D replies to the request. In such a case, the route discovery fails.

The above analysis leads one to believe that it would be inappropriate to use any routing protocol that uses path-reversal in route discovery, such as AODV, DSR, DD and LAR, in an asymmetric environment, because they would have a very high loss ratio. However, the simulation results we present later show that AODV and DSR work reasonably well despite the asymmetric nature of communication. The reason is that in addition to path-reversal technique, these routing protocols also use the multi-round discovery, which is capable of dealing with asymmetry, but with a high overhead.

6.1.2 Multi-Round Discovery Technique

In AODV and DSR, the RREQ is broadcast towards the destination D. So node D receives RREQ messages from multiple paths, as shown in Figure 13. It chooses one of the many available paths to send the RREP message back to source S, according to some runtime configurable parameter, such as the RREQ arrival time, path load, or end-to-end delay of the path. If the reverse path doesn't exist, the RREP fails to arrive at sender S and the route discovery is repeated due to timeout. In the next attempt, thanks to the random nature of flooding, node D might receive a RREQ message from another path, which happens to be a symmetric connection.

The chance to establish a symmetric connection increases after retries. If there is no limitation on the number of retries, a symmetric path will sooner or later be discovered on the condition that such a path exists. We note that the rediscover technique provides a viable way to work around the effects of asymmetry, but with significant overhead.

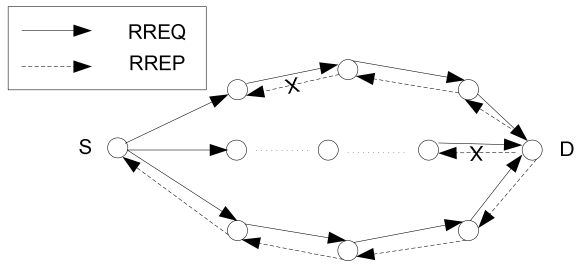


Figure 13: Route Discovery Using Rediscovery Technique

6.1.3 Impact on Neighbor Discovery Technique

Many location-based routing protocols [12] [16] [17] use the neighbor discovery technique in order to maintain the neighborhood information. However, the neighbor discovery technique works well only if the links are symmetric. For example, in Figure 14, node A discovers its neighbors by receiving beacons. Node A might choose one of its neighbors, node B, C, or D for forwarding packets. However, if node A picks node B which is unable to hear node A, node B will never receive the packet forwarded by node A. If node A does not retry its transmission with the other neighbors, the transmission of the packet will fail. So the routing protocol based on the neighbor discovery technique is subject to failures when communication is asymmetric.

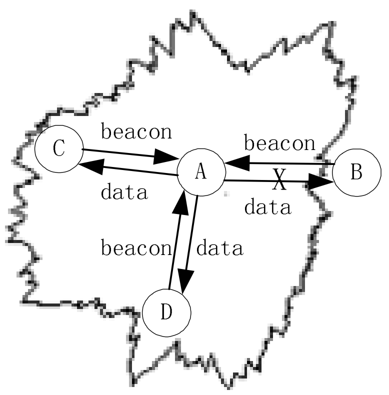
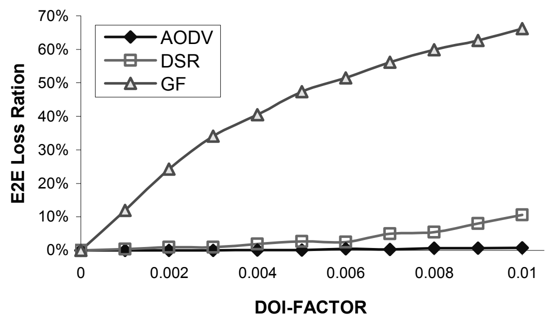
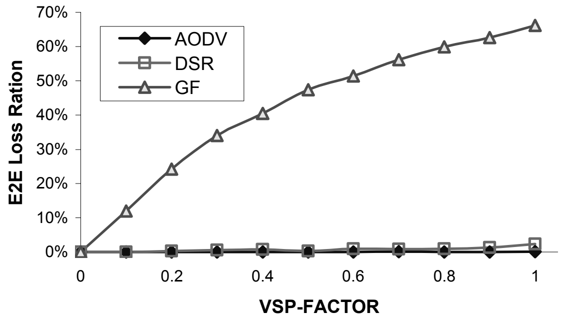


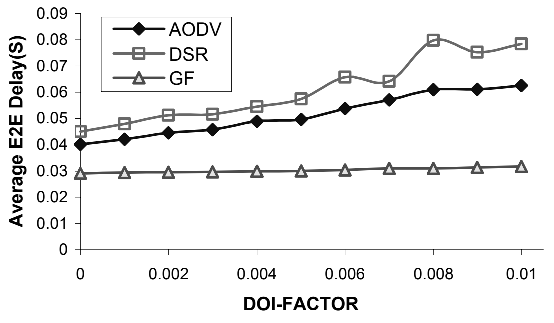
Figure 14: Impact on Neighbor Table Technique



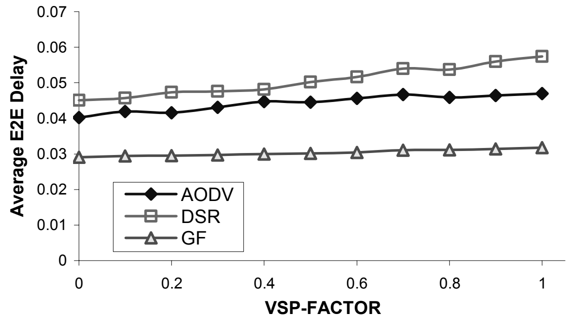
(a) E2E Loss Ratio vs. DOI



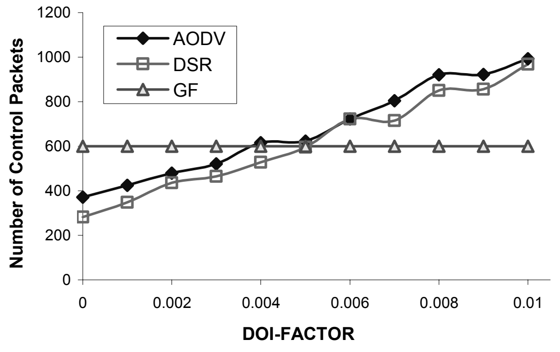
(a) E2E Loss Ratio vs. VSP



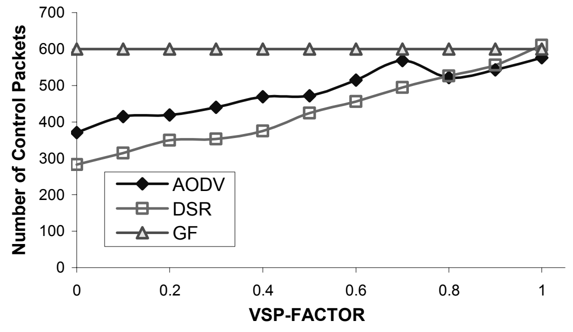
(b)Average E2E Delay vs. DOI



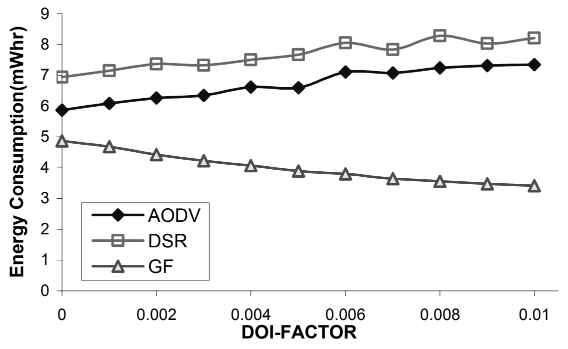
(b) Average E2E Delay vs. VSP



(c) Number of Control Packets vs. DOI

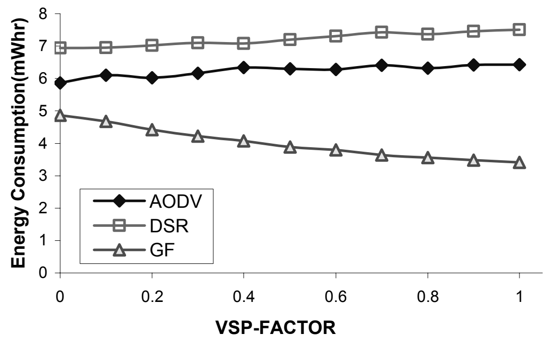


(c) Number of Control Packets vs. VSP



(d) Energy Consumption vs. DOI

Figure 15: Routing Performance with Different DOI



(d) Energy Consumption vs. VSP

Figure 16: Routing Performance with Different VSP

6.2 Quantitative Analysis of the Impact

In this section, we quantify the performance penalty of radio irregularity. We conducted two sets of experiments by varying DOI and VSP. In each set, we measure four metrics: end-to-end (E2E) loss ratio, average E2E delay, number of control packets, and energy consumption.

6.2.1 Routing Performance with Different DOI

Figure 15 (a) shows that GF is greatly influenced by radio irregularity. It loses 70 percent of packets when the DOI is 0.01. The reason is that according to the greedy forwarding rule, GF tends to choose a node near the border, which is more likely to have an asymmetric link with the sender. AODV and DSR perform well because they use multi-round discovery, exploring alterative paths to find a symmetric connection. However, they achieve a low loss ratio at the cost of increasing overhead in control packets shown in Figure 15 (c).

In Figure 15 (b), the average E2E delay of DSR and AODV increases with the increase in the DOI value. That is because more rounds of route discovery are needed as the radio irregularity increases. In Figure 15 (b) DSR has a higher delay than AODV, because the source routing technique in DSR adds the whole path in the header of data packets, which increases the transmission time. However, the E2E delay of GF remains the same because packets in GF either go through successfully or get lost.

Figure 15 (c) shows that while AODV and DSR need more control packets to do multi-round discovery, when the DOI value increases GF needs only a constant number of control packets for neighbor exchange.

From Figure 15 (d), we see that AODV and DSR consume more energy when DOI increases, because more control packets are used for rediscovery. However, GF transmits fewer data packets when the DOI value increases (Figure 15 (a)). Hence, the energy consumption reduces.

6.2.2 Routing Performance with Different VSP

In Figure 15, the impact of radio irregularity on the routing layer is measured for different DOI values. In this section, we measure the impact of radio irregularity on the routing layer by varying the VSP values. From our results, we find that an increasing value of VSP has a similar impact on AODV, DSR and GF, as an increasing value of DOI, because both lead to a higher degree of irregularity and therefore, a higher degree of link asymmetry.

From Figure 16 (a), we see that all routing protocols have higher loss ratio when the VSP value is increased, because there are more asymmetric links. GF has much higher loss ratio than AODV and DSR, because GF uses neighbor discovery and tends to choose the same node near the border of the radio range as the candidate, while AODV and DSR use multi-round discovery to try different paths.

As in the case of larger DOI values, larger VSP values result in more asymmetric links, which lead to larger average E2E delays (Figure 16 (b)) and higher energy consumption (Figure 16 (d)). However, GF does not require more beacons, so there is no increase in the control packets (Figure 16 (c)) and the delay remains the same (Figure 16 (a)). The energy consumption of GF reduces because GF drops more packets in the presence of asymmetry.

To summarize, as DOI and VSP increase, radio irregularity has a greater adverse impact on the GF protocol compared to on-demand routing protocols that use multi-round discovery such as AODV and DSR.

7. SOLUTIONS FOR RADIO IRREGULARITY

Having analyzed the causes and impact of radio irregularity, the key results can be summarized as follows:

- Radio irregularity is a common and non-negligible phenomenon in wireless communication.
- Radio irregularity has a greater impact on the routing layer than on the MAC layer.
- Routing protocols, such as AODV and DSR, that use multi-round discovery technique, can deal with radio irregularity, but with a high overhead.
- Routing protocols, such as geographic forwarding, which are based on neighbor discovery technique, are severely affected by radio irregularity.

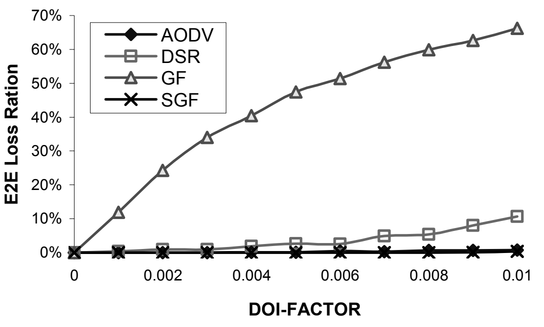
Based on both analytical and experimental results, we propose 6 potential solutions to improve the protocol performance in the presence of radio irregularity. We first describe the Symmetric Geographic Forwarding solution and the Bounded Distance Forwarding solution in detail and discuss their performance evaluation. We then follow that by briefly describing four other solutions.

7.1 Symmetric Geographic Forwarding

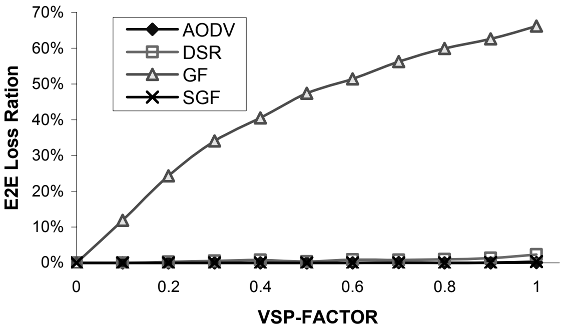
In location-based protocols, such as GF and GPSR, the beacon message only contains the node's ID and position. In the Symmetric Geographic Forwarding (SGF) solution, we allow a node to add the IDs of all its neighbors it has discovered into the beacon message. When a node receives a beacon message, it registers the sender as its neighbor in its local neighbor table, and then checks whether its own ID is in the beacon message. If the receiver finds its own ID in the neighbor list in the beacon message, then it marks the communication link connecting it to the sender as "SYMMETRIC". Otherwise, it marks the communication link between them as "ASYMMETRIC". Whenever a node needs to forward a packet, it selects only those neighboring nodes with which it is connected through "SYMMETRIC" links. Here we must emphasize that when a node broadcasts a beacon message, it should add the IDs of the nodes with which it has "SYMMETRIC" connectivity as well as those nodes with which it has "ASYMMETRIC" connectivity.

The SGF provides a basic prototype of incorporating symmetric detection into routing protocols. More sophisticated algorithms, such as measuring link quality with multiple rounds, can be introduced to deal with engineering issues in running systems.

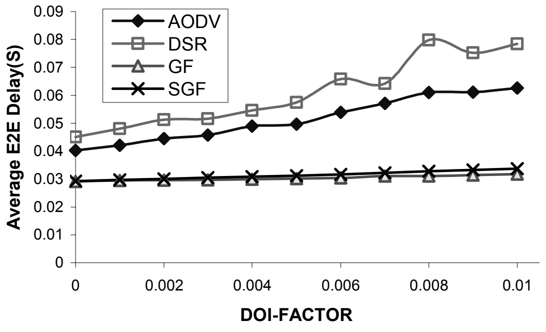
We simulate SGF in GloMoSim. We find that SGF maintains most of the advantages of GF, such as scalability, and the absence of flooding. Furthermore, SGF is able to deal with asymmetry as effectively as the multi-path route discovery protocols, such as AODV and DSR, but at lower cost. The simulation setup use the same configuration as mentioned in Table 1.



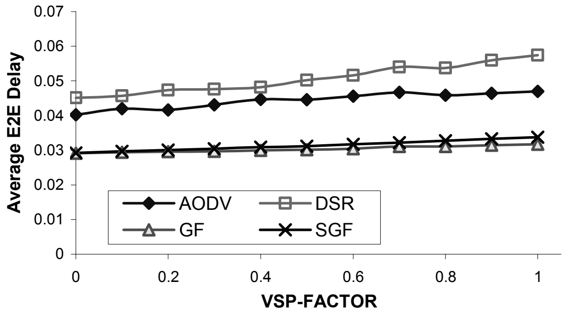
(a) E2E Loss Ratio vs. DOI



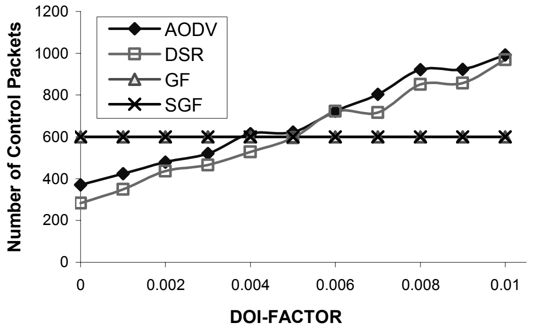
(a) E2E Loss Ratio vs. VSP



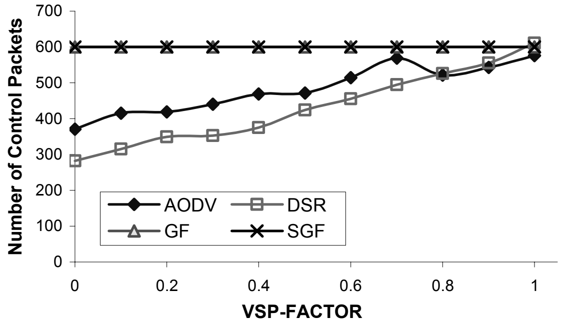
(b)Average E2E Delay vs. DOI



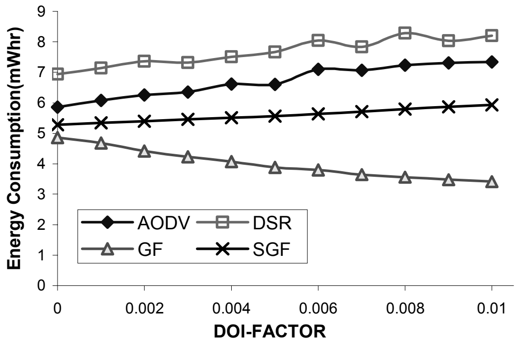
(b) Average E2E Delay vs. VSP



(c) Number of Control Packets vs. DOI

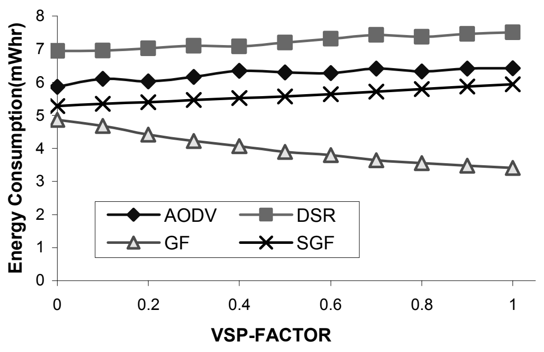


(c) Number of Control Packets vs. VSP



(d) Energy Consumption vs. DOI

Figure 17: SGF Performance with Different DOI



(d) Energy Consumption vs. VSP

Figure 18: SGF Performance with Different VSP

7.1.1 SGF Performance with Different DOI

In this experiment, we incrementally increase the degree of irregularity (DOI) to measure the SGF performance.

From Figure 17 (a), we observe that SGF has significantly lower loss ratio than GF, and performs as well as AODV. This is because it avoids forwarding data along asymmetric links. From Figure 17 (b), we observe that SGF has almost the same average E2E delay as GF. The delay is much lower than that of ADOV and DSR. An interesting point from Figure 17 (c) is that SGF consumes the same number of control packets as GF, and the number of control packets remains the same with the increase of DOI value. From Figure 17 (d), it is clear that SGF consumes almost the same amount of energy with the increase of DOI. The reason is that as DOI increases, AODV and DSR require more control packets, while SGF maintains the same number of control packets. As a result, the energy consumption of AODV and DSR increases with DOI, while that of SGF almost remains the same. The energy consumption of SGF is larger than that of GF on account of two reasons. First, SGF needs a larger packet to piggyback the neighbors' IDs, even though both GF and SGF transmit the same number of control packets. Second, unlike GF, SGF does not drop packets with the increasing DOI. Hence, while the energy consumption of GF decreases, because GF drops useful data packets, the energy consumption of SGF almost remains the same.

7.1.2 SGF Performance with Different VSP

Similar conclusions can be drawn from the results in Figure 18. Compared with GF, SGF has much lower loss ratio, almost the same average E2E delay, and the same number of control packets, with an increase of the VSP value. The loss ratio of SGF is comparable to that of AODV and DSR. However, SGF has a much lower average E2E delay, constant number of control packets, and lower energy consumption. SGF consumes almost constant energy with an increase of the VSP value. In contrast, GF consumes less energy because it drops data packets, while AODV and DSR consume more energy because of a higher number of control packets.

To summarize, the SGF protocol not only maintains GF's scalability, but also successfully deals with radio irregularity. Compared with AODV and DSR, it achieves similar delivery ratio in the presence of radio irregularity with a lower E2E delay, a lower number of control packets and lower energy consumption.

7.2 Bounded Distance Forwarding

Bounded Distance Forwarding restricts the distance over which a node can forward a message in a single hop. It can act as an add-on rule to many routing protocols. The distance bound is decided based on the degree of radio irregularity of the real devices in a physical system.

We add the Bounded Distance Forwarding rule on the spanning tree module in a vehicle tracking system [11] in which we deploy 60 MICA2 motes. In the experiments, we incrementally increase the single hop forwarding bound from 8 feet to 100 feet and count the number of nodes that report their status and Figure 19 shows this data as a percentage of the total number of nodes deployed. Data points here are average values over five runs. Figure 19 indicates two interesting phenomena. First, when we use a very low forwarding bound (8 feet) to eliminate the asymmetric links, the performance, however, is not good. This is because relative node density decreases when the enforced communication range is small. Hence, the chance of a network partition increases. Moreover, a smaller forward bound per hop leads to a longer route, thus a higher chance of loss. Second, when the forwarding bound reaches larger values (16~100 feet), link asymmetry becomes the dominating factor. Figure 19 shows that when the forwarding bound is 16 feet, we receive almost every report. This bound is about half of the MICA2 radio range on the ground. Above 16 feet, performance reduces monotonically because of increase in link asymmetry.

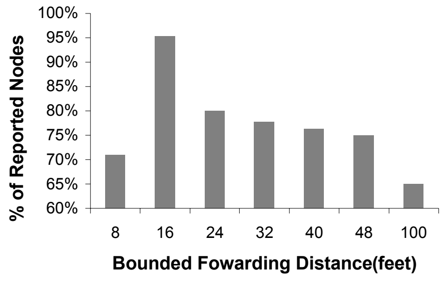


Figure 19: Percentage of Reporting Nodes

7.3 Other Solutions

In this section, we propose four additional potential solutions to deal with radio irregularity.

- **Bidirectional Flooding**: The multi-round discovery technique can deal with radio irregularity. However, it needs multiple rounds of flooding to explore different paths, which can be very expensive. In Bidirectional Flooding, the source propagates the RREQ towards the destination through flooding. After the destination receives the RREQ, it propagates the RREP to the source through flooding, instead of using the reverse path along which it received the RREQ from the source. Multi-round discovery cannot guarantee finding symmetry connections within a bounded number of flooding stages. In contrast, bidirectional flooding completes the discovery by flooding twice.
- **Learning Function**: In an earlier section we mentioned that GF has a higher loss ratio than AODV and DSR, because GF tends to choose the same candidate near the border to forward packets to a destination, while AODV and DSR attempt different paths due to the nature of flooding. To address this shortcoming of GF, we can enhance GF with a learning function, which allows a node to make better decisions based on previous routing failures. In the learning function, we distinguish the routing failures arising due to congestion from those that arise due to asymmetric links. This can be done with the help of the 802.11 (DCF) in the MAC layer. If a node receives the CTS, but not the ACK, then the link should be symmetric and the routing failure might be a result of congestion. Such a failure can be solved by retransmissions. However, if a node fails to receive the CTS despite several retransmissions, then the chances are that the link is asymmetric. This learning function allows a node to remember such an asymmetric link and to avoid trying it again.

- **RTS Broadcast:** Another solution we propose is called the RTS Broadcast, which involves both the MAC and routing layers. We first broadcast a special RTS message, which sets the destination as ANY_NODE. Any node hearing it backs off for a random amount of time and replies with a CTS message. Among all the nodes that send the CTS message, the one that is closest to the destination is chosen as the forwarding candidate. Since the RTS and CTS detect connectivity along the forward and reverse directions of a channel, forwarding packets along asymmetric channels can be avoided.
- **High Energy Asymmetry Detection:** IEEE 802.11 (DCF) uses a collision-avoidance strategy in which any node upon hearing an RTS, CTS, or DATA message defers its transmission until the data is sent out. However, a node can still interfere with the message transmission even though it is not able to hear any of the RTS, CTS and DATA messages in the presence of asymmetry. The sixth solution we propose is to send out a High Energy Asymmetry Detection (HEAD) control message which has a higher sending power than the other control messages. So more nodes will hear the high-powered signal, and prevent themselves from sending messages. The HEAD message is sent out before the RTS message. Any node other than the destination, upon hearing the HEAD message, sets its NAV to a value large enough so that data can be sent out without contention. The wait time and destination ID are included in the HEAD message. Conflicts may arise if two nodes send out the HEAD messages simultaneously. That is resolved in a manner similar to the way to resolve conflicts arising from the simultaneous transmission of two RTS messages. Hence, the transmission sequence is modified from RTS-CTS-DATA-ACK to HEAD-RTS-CTS-DATA-ACK. While the higher sending power of the HEAD message lowers the collision rate, it also reduces the channel utilization. This tradeoff between collision rate and desired channel utilization can be balanced by choosing an appropriate value for the sending power.

We note that the solutions we mentioned above are still open topics and require further refinements. Extensive analysis and evaluation in the future are required to demonstrate their applicability and effectiveness.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we confirm the existence of radio irregularity which is the main focus of several recent research papers [4][9][23][28]. Our contributions are as follows:

1. To the best of our knowledge, our work is the first to bridge the gap between isotropic radio models assumed by most simulators and the non-isotropic radio properties in reality.
2. We propose a novel RIM model that approximates three essential properties exhibited in radio irregularity: non-isotropy, continuous variation and difference in sending power.
3. We implement the RIM model in GloMoSim, and run a set of simulation experiments to investigate radio irregularity's impact on MAC and routing layer performance. We discover that, among the protocols we evaluate, the radio irregularity has a greater impact on the routing layer than MAC layer. We also discover that radio irregularity has a greater impact on location-based routing protocols than on-demand protocols that use multi-round discovery technique.
4. Finally, we propose six potential solutions. We implement SGF in GloMoSim, and implement the Bounded Distance Forwarding rule in a real running tracking system consisting of 60 MICA2 motes. From the data we collect from the simulator and the running system, we find that SGF and Bounded Distance Forwarding can successfully deal with radio irregularity.

In future work, we will concentrate on the following aspects. First, we plan to further refine the RIM model and incorporate our work into the standard release of GloMoSim and NS-2. Second, we plan to analyze the impact of radio irregularity on other protocols, such as localization and topology control. Third, we plan to analyze and evaluate the remaining four approaches mentioned in Section 7.3.

Acknowledgements

The authors would like to thank the MobiSys shepherd, Hari Balakrishnan, and the anonymous reviewers for their helpful suggestions that improved our work. This research was supported by the DARPA IXO offices under the NEST project (grant number F336615-01-C-1905).

9. REFERENCES

[1] V. Bharghavan, A. Demers, S. Shenker and L. Zhang, MACAW: A Media Access Protocol for Wireless LANs, In *Proc. of ACM SIGCOMM*, pp. 212-225, 1994.

[2] N. Bulusu, J. Heidemann and D. Estrin, GPS-less Low Cost Outdoor Localization for Very Small Devices, In *IEEE Personal Communications Magazine*, pp. 28-34, October 2000.

[3] A. Cerpa and D. Estrin, ASCENT: Adaptive Self-Configuring Sensor Networks Topologies, In *Proc. of the IEEE Infocom*, 2002.

[4] A. Cerpa, N. Busek and D. Estrin, SCALE: A Tool for Simple Connectivity Assessment in Lossy Environments, In *CENS Technical Report 0021*, September 2003.

[5] B. Chen, K. Jamieson, H. Balakrishnan and R. Morris, Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad-hoc Wireless Networks, In *ACM MobiCom*, July 2001.

[6] Chipcon CC1000 Low Power Radio Transceiver, http:// www.chipcon.com/files/cc1000_data_sheet_2_1.pdf

[7] J. L. Devore, Probability and Statistics for Engineering and the Sciences*, Brooks/Cole Publishing*, 1982.

[8] IEEE 802.11, Part II: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, *ANSI/IEEE Std. 802.11,* 1999.

[9] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin and S. Wicker, Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks, In *Technical Report UCLA/CSD-TR 02-0013*, 2002.

[10] T. He, C. Huang, B. M. Blum, J. A. Stankovic and T. F. Abdelzaher, Range-Free Localization Schemes in Large Scale Sensor Networks, In *Proc. MOBICOM*, 2003.

[11] T. He, S. Krishnamurthy, J. A. Stankovic, T. F. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, B. Krogh, Energy-Efficient Surveillance System Using Wireless Sensor Networks, In *ACM MobiSys 2004,* Boston, MA, June 2004.

[12] T. He, J. A. Stankovic, C. Lu and T. F. Abdelzaher, SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks, In *IEEE ICDCS 2003*, Providence, RI, May 2003.

[13] C. Intanagonwiwat, R. Govindan and D. Estrin, Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks, In *Proc. MOBICOM*, pp. 56-67, March 2000.

[14] D. B. Johnson and D. A. Maltz, Dynamic Source Routing in Ad Hoc Wireless Networks, Mobile Computing, edited by Tomas Imielinski and Hank Korth, *Kluwer Academic Publishers*, ISBN: 0792396979, 1996.

[15] P. Karn, MACA - A New Channel Access Method for Packet Radio, In *ARRL/CRRL Amateur Radio 9th Computer Networking Conference*, pp. 134-140, 1990.

[16] B. Karp and H. T. Kung, GPSR: Greedy Perimeter Stateless Routing for Wireless Networks, In *Proc. MOBICOM*, pp. 243-254, August 2000.

[17] B. Karp, Geographic Routing for Wireless Networks, *Ph.D. Dissertation*, Harvard University, Cambridge, MA, October 2000.

[18] L. Kleinrock and F. A. Tobagi, Packet Switching in Radio Channels: Part I - Carrier Sense Multiple-access Modes and Their Throughput-delay Characteristics, In *IEEE Trans. Commun.*, vol. COM-23, pp. 1400-1416, Dec. 1975.

[19] Y. B. Ko and N. H. Vaidya, Location-Aided Routing (LAR) in Mobile Ad Hoc Networks, In *Proc. MOBICOM*, pp. 66-75, 1998.

[20] D. Niculescu and B. Nath, DV Based Positioning in Ad hoc Networks, In *Journal of Telecommunication* Systems, 2003.

[21] C. E. Perkins, and E. M. Royer, Ad-hoc On-Demand Distance Vector Routing, In *Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications,* pp. 90-100, February 1999.

[22] P. M. Shankar, Introduction to Wireless Systems, *John Wiley & Sons, Inc,* ISBN 0-471-32167-2, 2001.

[23] A. Woo, T. Tong and D. Culler, Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks, In *ACM SenSys 2003*, Los Angeles, CA, November 2003.

[24] XBOW MICA2 Mote Specifications: http://www.xbow.com/

[25] Y. Xu, J. Heidemann and D. Estrin, Geography-informed Energy Conservation for Ad Hoc Routing, In *ACM MOBICOM 2001*, Rome, Italy, July 2001

[26] T. Yan, T. He, J. A. Stankovic, Differentiated Surveillance for Sensor Networks, In *ACM SenSys 2003,* Los Angeles, CA, November 2003.

[27] X. Zeng, R. Bagrodia and M. Gerla, GloMoSim: a Library for Parallel Simulation of Large-scale Wireless Networks, In *Proc. of the 12th Workshop on Parallel and Distributed Simulations*, May 1998.

[28] Y. J. Zhao and R. Govindan, Understanding Packet Delivery Performance in Dense Wireless Sensor Network, In *ACM SenSys 2003*, Los Angeles, CA, November 2003.

Appendix A

We use the goodness-of-fit statistical testing to determine the statistical distribution of the variance of the received signal strength (in dBm) per degree in the direction that is obtained in our experiments. We find that among different continuous distributions, the Weibull distribution [3] has the maximum likelihood of matching our experimental data. A random variable X that has a Weibull distribution with parameters has a probability density function defined by the following equation, where a is the shape parameter and b is the scale parameter.

$$f(x) = \begin{cases} (a/b^a) \times x^{(a-1)} \times e^{-(\frac{x}{b})^a}, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{8}$$

Table 2 shows the likelihood values and the parameters of the Weibull distribution that fits our experimental data. These values are computed at a 95% confidence level.

Table 2: Data fitting to the Weibull distribution

	Likelihood	a	b
Dataset 1	48.55	1.13	0.28
Dataset 2	154.43	1.01	0.17
Dataset 3	145.25	0.86	0.18
Dataset 4	277.44	0.67	0.16
Dataset 5	204.51	0.58	0.17
Dataset 6	111.15	0.53	0.22

Lightweight Detection and Classification for Wireless Sensor Networks in Realistic Environments

Lin Gu [1] Dong Jia [2] Pascal Vicaire [1] Ting Yan [1] Liqian Luo [1]
Ajay Tirumala [3] Qing Cao [1] Tian He [1]
John A. Stankovic [1] Tarek Abdelzaher [1] Bruce H. Krogh [2]

{lingu, pv9f, ty4k, ll4p, qc3h, th7c, stankovic, zaher}@cs.virginia.edu
{djia, krogh}@ece.cmu.edu tirulama@uiuc.edu

ABSTRACT

A wide variety of sensors have been incorporated into a spectrum of wireless sensor network (WSN) platforms, providing flexible sensing capability over a large number of low-power and inexpensive nodes. Traditional signal processing algorithms, however, often prove too complex for energy-and-cost-effective WSN nodes. This study explores how to design efficient sensing and classification algorithms that achieve reliable sensing performance on energy-and-cost-effective hardware without special powerful nodes in a continuously changing physical environment. We present the detection and classification system in a cutting-edge surveillance sensor network, which classifies vehicles, persons, and persons carrying ferrous objects, and tracks these targets with a maximum error in velocity of 15%. Considering the demanding requirements and strict resource constraints, we design a hierarchical classification architecture that naturally distributes sensing and computation tasks at different levels of the system. Such a distribution allows multiple sensors to collaborate on a sensor node, and the detection and classification results to be continuously refined at different levels of the WSN. This design enables reliable detection and classification without involving high-complexity computation, reduces network traffic, and emphasizes resilience and adaptation to the realistic environment. We evaluate the system with performance data collected from outdoor experiments and field assessments. Based on the experience acquired and lessons learned when developing this system, we abstract common issues and introduce several guidelines which can direct future development of detection and classification solutions based on WSNs.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; C.3 [**Computer System Organization**]: Special Purpose And Application-Based Systems—*Real-Time and embedded systems*; C.4 [**Performance of Systems**]: Design Studies

General Terms

Design, Experimentation, Measurement, Performance

Keywords

Classification, Wireless Sensor Networks, VigilNet

1. INTRODUCTION

Sensing is a fundamental function in wireless sensor networks. Researchers have built WSN platforms with a wide spectrum of sensors, ranging from simple thermistors to micropower impulse radars [4, 13, 14]. They provide flexible sensing capability with a large number of low-power and inexpensive sensor nodes. Nontrivial as it is, the selection and integration of sensors on a WSN platform is often a manageable task given a certain amount of engineering effort. The situation is, however, completely different above the physical sensor and computing hardware layer – the acquisition and processing of sensor data impose great challenges on WSN design because of strict resource constraints.

Cost-effectiveness being an important objective, WSN designers often choose mass produced commercial off the shelf (COTS) sensors when designing a sensor network system. Moreover, a sensor node must be energy efficient. As a result, the raw sensor data is often of low-quality – they are not always reliable, not always repeatable, usually not self-calibrated, and often not shielded to environment and circuit board noise. Obviously, it is necessary to use signal processing algorithms to filter, process, and abstract sensor data with software to provide precise, reliable, and easy-to-use information to applications.

Traditional signal processing algorithms, however, often prove too complex to implement on inexpensive sensor network hardware without digital signal processing co-processors. For example, the popular Berkeley Mica series has an 8-bit micro-processor running at 7.3827MHz, no hardware floating-point support, and only 4KB data memory. Though recent versions of MicaZ and Telos motes

[1]Department of Computer Science, University of Virginia.
[2]Department of Electrical and Computer Engineering, Carnegie Mellon University.
[3]Department of Computer Science, University of Illinois at Urbana-Champaign
[4]This work is supported in part by NSF grant CCR-0098269 and CCR-0325197, the MURI award N00014-01-1-0576 from ONR, and the DAPRPA IXO offices under the NEST project (grant number F336615-01-C-1905).

employ a bigger data memory, we expect the growth of computational resources on WSN platforms to be rather slow because of the emphasis on low power consumption, low cost, and small form factor. Generally, resource constraints will continue to represent the reality of energy-and-cost-effective embedded systems. This strict resource limitation makes it very difficult to execute Fast Fourier Transformation (FFT) and other signal processing algorithms with moderate or high time/space complexity. Also, the stringent energy budget favors simple and quick algorithms over complex algorithms that require prolonged execution time.

While the computation/energy resources are limited, the application requirement is not. Specifically, the development of recent surveillance WSNs requires the network to provide functionalities well beyond sensing and routing. Such surveillance WSNs are designed to detect and report certain classes of events of interest. When such an event happens, the WSN needs to detect it quickly, classify it into one category (e.g., person, vehicle), and compute its attributes (e.g., location, velocity).

Designing such surveillance WSNs is a research challenge. Besides the obviously severe resource constraints, the following factors also contribute to the difficulty of the task.

- To provide sensing coverage for a relatively large area, the network is usually comprised of a large number of densely deployed nodes. This imposes a challenge on efficient data propagation and reliable operation.
- The detection, classification, and reporting must be performed in a timely manner. It is usually required that the network complete the detection and classification before the target travels out of the field so that the system can respond to the event. As a result, offline-style processing performed by base stations with global and relatively "complete" data is often not feasible in this context.
- To perform quality signal processing, the sensors often need to sample at a high sampling rate, stressing resource utilization. The sensing data is bursty and in large quantity.
- Surveillance networks are often deployed on rough terrains for a long period of time. Hence, it must be adaptive to the realistic, ever-changing environment.

Given the numerous technical challenges, important research questions are: Can we construct a reliable surveillance WSN that meets the requirements within the strict resource constraints? What performance will such a system achieve? This study attempts to answer these questions by presenting the detection and classification system in VigilNet [10], which is a recently deployed surveillance WSN detecting and classifying vehicles, persons and persons with ferrous objects. Specifically, this paper explores the design choices involved in constructing an efficient detection and classification system that achieves reliable performance on a network of energy-and-cost-effective sensor nodes, analyzes the performance, and proposes a set of guidelines for future designs of WSNs in a similar design context.

It is worth clarifying that advanced signal processing mathematics and algorithms are not the emphasis of this paper. Instead, this paper focuses on the system design issues involved in creating a reliable and realistic classification system for a surveillance WSN using homogeneously low-end sensor nodes, as well as evaluation of the effectiveness of these designs. To the authors' best knowledge, there has not been a large-scale deployment of such a sophisticated surveillance network without using special powerful nodes. Hence, our study is focused on answering this challenge: Without enhancing any individual nodes' capability and cost, can a network of distributed sensor nodes provide advanced functions and work reliably in realistic environments? We believe that the experience acquired and lessons learned in constructing such a system, and the analysis of the trade-offs and design decisions in it, will benefit the research in this area, and help transform the research potential of WSNs into real-world technology and market success.

The paper is organized as follows. Section 2 presents background information and surveys related work. Section 3 gives an overview of the VigilNet surveillance system. Section 4 presents the design of the hierarchical classification architecture. System level evaluation is shown in Section 5. Section 6 discusses several guidelines for designing a large-scale WSN for detection and classification tasks. Finally, Section 7 concludes the paper.

2. BACKGROUND

Focusing on VigilNet's hardware platform, we present a brief overview of the sensing subsystem – sensors and their supporting circuitry – on a sensor node. The sensing subsystem is the hardware foundation on which classification systems are constructed. We also survey the related work in the area of detection and classification WSN systems.

2.1 Overview of the sensing subsystem

VigilNet uses the ExScal motes as sensor nodes. Based on the Mica2 [3] mote design, the ExScal mote, shown in Fig. 1, is designed by CrossBow Inc. and Ohio State University for large-scale surveillance WSNs [8]. The major difference between the ExScal mote and the Berkeley Mica2 mote is that the former integrates a magnetometer (Honeywell HMC1052[2]), a microphone, and 4 PIR sensors on the same circuit board as the processor's. After the first prototype ExScal motes were delivered in March 2004, CrossBow released several versions with various improvements throughout the year of 2004.

Several correlated factors contribute to the complexity of the sensing circuitry. First, applications require a long sensing distance, which implies a finer granularity for the sensor readings. Second, as a general purpose platform, designers hope to choose sensors with a wide measuring range. Third, the wider measuring range combined with a finer granularity maps to more numeric values which, however, have to all fall into the representation capability of the A/D converter (ADC), I/O bus, and CPU I/O port width. Finally, as the sensors on the sensor board grow in both number and sophistication, the support circuitry may need to support better filtering, handle more advanced signaling protocols, employ a faster or wider bus, provide wider functionality (such as waking up the sensor node), or build better shielding to avoid cross-talk among various components. These factors make the design of the sensing subsystem a significant engineering effort involving numerous design choices which often depend on the application domain.

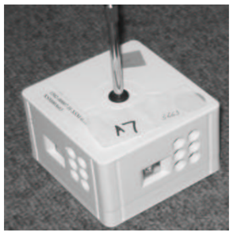


Figure 1: ExScal mote

To solve the aforementioned range and granularity problems for the magnetometer, the ExScal mote includes circuitry that allows the application program to adjust the input signal to be amplified. To provide a quality signal for acoustic processing, the microphone circuitry incorporates a high-pass filter and a low-pass filter. Both the input adjustment and filtering are controllable by the processor, with an I^2C bus connecting the processor and sensor components.

2.2 Related work

With the development of WSN systems, sensing, detection, and tracking have been a prosperous research area. Specifically, Wang et. al. studied acoustic tracking using Mica motes [22]. Simon et. al. designed a sniper localization system with acoustic signal processing [19] and accomplished good performance. Different from VigilNet's homogeneous approach, these systems employ special powerful nodes or DSP co-processors to process acoustic data. Zhao et. al. described collaborative signal processing [25] to retrieve more accurate information from sensor data and achieve better target tracking performance. Pattem et. al. build a framework to evaluate the tracking strategies in an energy aware context [18]. Most of the performance analysis in [25] and [18] are conducted by simulations, concentrating on exploring the design space and trade-offs under specific constraints and assumptions.

Along the direction of real-world application and deployments, researchers have also constructed a number of successful systems. Szewczyk et. al. [21] developed a habitat monitoring WSN on the Great Duck Island and the system operated for months. Zhang et. al. developed a WSN for wild life tracking [24]. These systems demonstrate the flexibility and capability of the WSN technology in various applications. However, VigilNet faces more demanding application requirements. As a result, many design choices are different in these systems than in VigilNet. For example, many current systems typically employ centralized processing which is not feasible in many surveillance networks [7].

In [11], the authors describe a surveillance network that can detect moving targets. The system uses Mica2 motes [3] equipped with a magnetometer (Honeywell HMC1002 [2]), an acoustic sensor and, on some nodes, a motion sensor. The motion sensor is an Advantaca MIR (micropower impulse radar) sensor which transmits microwave signals and detects motion by capturing distortion of the reflected signal. The network reports a target as a walking person or a vehicle. Therefore, it has a preliminary classification capability. However, there is very limited signal processing in it. As a result, the classification is limited in both functionality and performance. Also, the MIR sensors, worth four thousand dollars each, are not a typical choice for energy-and-cost-effective systems.

Brooks et. al. [7] introduced a collaborative signal processing framework for sensor networks using location-aware routing and collaborative signal processing. Their study provides many insights into the distributed collaborative classification in WSNs. Nevertheless, the CSP framework involves non-trivial training and computation overhead, which our system cannot afford. Also, the system implementation and evaluation of the CSP framework employ nodes with higher power than the energy-and-cost-effective WSN nodes our system is targeting. In fact, VigilNet must satisfy three conflicting requirements simultaneously – low-end hardware, long lifetime, and sophisticated function. This challenging design context is different than what past solutions assume.

Among recently deployed WSNs, the Extreme Scaling project is the most similar to VigilNet in functionality and hardware platform [1, 8]. However, a major difference is that the Extreme Scaling WSN employs a heterogeneous network topology and uses a more powerful Stargate node for some computation and communication intensive tasks.

3. OVERVIEW OF THE DETECTION AND CLASSIFICATION IN VIGILNET

The VigilNet surveillance system [5] is a WSN with 200 sensor nodes (ExScal motes). The WSN is required to perform timely detection, tracking and classification of vehicles, persons, and persons with ferrous objects. When a target is detected, the WSN reports the detection to an external device. The external device can be a more powerful sensor, a communication device connecting to a control center, or any device that handles the information delivered by the WSN. A base mote connects to the external device through a UART interface, and serves as a router between the WSN and the external device. As the target travels in the network, the WSN garners enough information to classify the target and compute its attributes, such as location and velocity, and the results are delivered to the external device as periodic updates. Fig. 2 shows a screen snapshot of VigilNet deployed along two roads forming a "T" shape. It illustrates the detection and classification of a "person with ferrous objects" target. Moreover, the WSN is to be deployed in a rough terrain and operate for months. Hence, the detection and classification algorithms must be adaptive to environmental variety and weather changes.

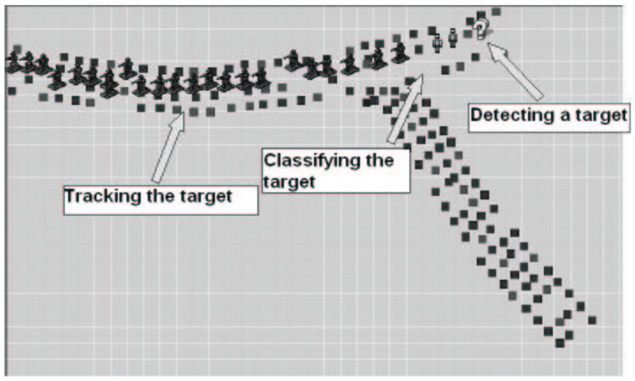


Figure 2: Screen of tracking a person with ferrous objects

As in many surveillance systems, VigilNet emphasizes that the false negative rate (the possibility of a target not being detected) must be very low. Meanwhile, it also requires a low false positive rate (the possibility of an event being reported without a real target in the field) since false positives waste energy and reduce the overall system lifetime. This implies that the wake-up (most of the network nodes are in sleep mode when there are no events of interest), sensing and classification must complete within a time constraint. These two factors – energy efficiency and low latency – make it undesirable to have a centralized semi-offline algorithm that collects all data from the network, transports them to a base station, and lets a powerful node analyze data and perform classification. Instead, the network, including the base mote (also an energy-and-cost-effective device), must perform reliable detection and classification functions independently in a timely manner without powerful nodes involved.

To build a complete VigilNet for realistic outdoor environments, other middleware services are also integrated. In brief, the localization is done through the walking GPS solution [20], which assigns nodes their location at the time they are deployed. The time synchronization used in VigilNet is a variation of the FTSP protocol [17] without periodic adjustments for the sake of stealthiness. Routing infrastructure is a set of multi-parent diffusion trees (forest) rooted at the base nodes. To achieve long-term surveillance, a multi-dimensional power management scheme is proposed in [12]. In this paper, we focus on the design of the detection and classification system in VigilNet, which is not addressed in other papers, but is a major part of the system and directly determines the system's functionality and performance.

4. CLASSIFICATION SYSTEM DESIGN

In this section, we present the design of the classification architecture, including the sensing algorithms for the magnetometer, motion sensor, and microphone (acoustic sensor).

We call the sensor reading at a specific time on a specific sensor on a specific node a *sample point*. When a sensor network starts operation, each sensor on each node in the network produces a sequence of sample points. All the sample points produced by the network form a set and we call it the *global sample set*.

The global sample set is the complete information about what happens in the network. If all the nodes report their sample points to a base station, the base station can collect the global sample set and perform computation with it. This solution has been successfully used in a number of WSNs. For surveillance WSNs, however, this is often not feasible because it is too expensive to collect the global sample set in a sensor network. As an example, a 150-node habitat monitoring WSN, presented in [21], collected temperature, humidity, and barometric pressure sensor readings and routed them back to base stations for analysis. During its 115 days of operation, the network collected and routed 650,000 observations. In VigilNet, the data for a one-minute target detection and classification event, with 200 nodes and acoustic processing, well exceeds 1,000,000 observations. If targets enter the network once a day and we routed all the data (the global sample set) back to the base mote, the system could hardly last a week. Hence, the "sense-store-send" style processing is not suitable for latency-sensitive surveillance systems that require a high sampling rate.

On the other hand, the sequence of sample points on a single node does not have enough information to support reliable detection and classification. As an example, a transient disturbance (such as a curious bird landing on the sensor) may shake the node and trigger PIR and magnetic detections. Individual sensor nodes cannot distinguish such an unexpected event from a moving person with ferrous objects. Generally, observations on an individual node are not a reliable indication of events in a network.

Hence, we must design the detection and classification system so that the sensing and classification functions are reasonably distributed in the network and the sensor nodes can cooperate to detect target signatures, reduce false positives, and achieve reliable and timely classification at reasonable energy cost. This motivates us to choose a hierarchical architecture for the classification system. In fact, the concept of hierarchical processing is not new in WSNs. The unique characteristics of our hierarchical design are in the organization of various components and the distribution of the detection and classification tasks in such a hierarchy so that the system accomplishes the required performance with minimal overhead. Illustrated in Fig. 3, the hierarchical classification architecture is comprised of four tiers – sensor-level, node-level, group-level, and base-level. The classification result is represented by a data structure called the confidence vector. The confidence vector comprises the confidence levels for the corresponding targets, and is used as a common data structure to transport information between different levels of the classification hierarchy.

The lowest level deals with individual sensors and comprises the sensing algorithms for the corresponding sensors. With communication being a costly operation, the sensing algorithms need to perform local detection and classification as much as possible. After processing the sensor data, each sensing algorithm delivers the confidence vector to the higher level module – the node-level detection and classification module.

The node-level classification deals with output from multiple sensors on the node. The fusion of the data from various sensors exposes more useful information than can be obtained from any individual sensor. Hence, the node-level sensing algorithm must correlate the sensor data from individual sensors and form node-level classification results. Such a correlation can enhance the detection and classification accuracy on individual nodes – different sensors may strengthen the confidence of each other's classification results and invalidate false positives. Furthermore, the node-level classification module monitors the sensors' status and performs sanity control over sensors. For example, it detects and shuts down faulty sensors. Though these functions are all important aspects in the hierarchical classification architecture, this paper does not detail the design of the node-level classification because, compared to other components, it is not the challenging part in the system.

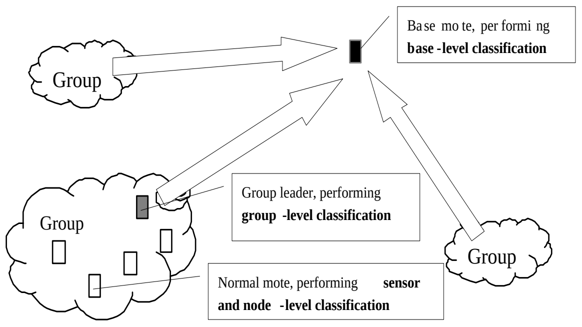

Base mote, performing base-level classification
Group
Group leader, performing group-level classification
Group
Normal mote, performing sensor and node-level classification
Group

Figure 3: Hierarchical Classification Architecture

The sensor-level and node-level classification functions both reside on a single node. The level above, the group-level classification, is performed by groups of nodes. Such groups are managed by a middleware called EnviroSuite [15], which provides a set of distributed group management protocols to dynamically organize nodes in the vicinity of targets into groups and elect leaders among them. These leaders are designated to collect the node-level classification results from individual members and, based on them, perform the group-level classification. Thus, the input to the group-level classification is the node-level confidence vectors rather than a bulk of sample points. This greatly reduces the volume of information transmitted between group leaders and members. Group leaders have much better views of targets compared with individual nodes. Therefore, besides group-level classification they are able to execute more complicated tasks which are extremely hard or even impossible for the node-level. Examples include suspicious report/node detection (based on spacial and temporal correlations among members) and aggregate attribute computation (e.g., computing average member locations as estimates of target positions).

The highest level in the hierarchical classification architecture is the base-level classification. The group-level classification results are transported via multiple hops to the base mote, serving as the input to the base-level classification algorithm. The base-level classification algorithm finalizes the sensing and classification result, as well as computing attributes (e.g., velocity) of the event.

In the following subsections, we present sensing algorithms for the magnetometer, the motion sensor, and the acoustic sensor, focusing on their unique characteristics. Some techniques are used in more than one sensing algorithm. To avoid redundancy, we present them in the sensing algorithm where their purpose and effects can be most clearly explained. In Section 4.4 and 4.5, we describe the group-level and base-level classification, respectively.

4.1 Sensing Algorithm for Magnetometer

In VigilNet, the requirement for magnetic sensing is to detect vehicles and persons with ferrous objects. Since the magnetometer circuitry in the ExScal mote senses a wide range of signals with a fine granularity, we can use it to measure deflection of the magnetic field caused by motion of ferrous objects (e.g., vehicles or weapons). Straightforward as it looks, challenges abound in designing a reliable magnetic sensing algorithm for the low-power sensor network platform.

First, raw ADC readings easily saturate due to the aforementioned granularity/range problem. The ADC on the ExScal platform is 10 bits wide, representing 1024 values. But the wide range of signal intensity combined with a fine granularity requires a much larger value set than the available 0 to 1023. Second, the response latency is too long for accurate signal waveform extraction. The magnetometer circuitry needs about 40 milliseconds to stabilize, and each tuning of the potentiometer needs about 50 milliseconds to stabilize. Third, electromagnetic noise from the circuit board lowers the S/N ratio and imposes serious problems on the magnetic sensing algorithm to distinguish signals from noise. Fourth, thermal drift is a severe issue – When the ambient temperature changes, the sensor readings change accordingly. Finally, radio transmission interferes with the magnetometer sensing circuit.

Among the five issues, the response time is a hardware characteristic. We cannot eliminate such delays. Instead, we measure such delays and reduce them to as low as safety allows. The radio/magnetometer interference can be solved by scheduling the radio and magnetometer to work in separate time slots. The other three issues are more interesting research questions with practical importance to a number of amplitude based sensors. Hence, Section 4.1.1 discusses the sensor reading and signal/noise ratio, and Section 4.1.2 discusses how to deal with the thermal drift. We also use the magnetic sensing algorithm as an example to discuss the trade-off between sensitivity and resilience in Section 4.1.3.

4.1.1 Mag-Points

As mentioned above, raw ADC readings are not suitable to represent the magnetic field intensity, and the magnetometer suffers from a low signal/noise ratio. Both issues relate to a basic question: how to provide credible sensor readings with semantics that higher-level signal processing algorithms can easily use? Hence, we handle these two issues together, by transforming the the raw sensor reading into a 32-bit uniform measure, the Mag-Point.

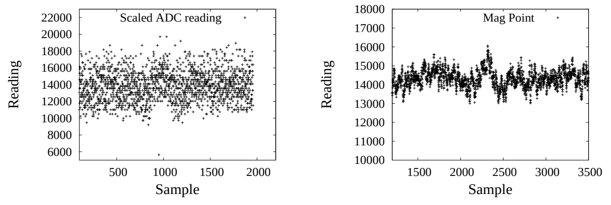
First, the sensing algorithm transforms the raw ADC reading into a scaled ADC reading. The numeric value of the raw ADC reading (r) is determined by the voltage across the magnetic signal line and a reference line. The voltage on the reference line is determined by a digital potentiometer setting. By studying the relation of the changes of potentiometer value (p) with the changes of ADC reading, we map the potentiometer value into certain ADC units. On ExScal motes, experiments reveal that 1 unit of potentiometer change equals 210 ADC units. At run time, as the magnetic signal varies, the sensing algorithm dynamically searches and sets the potentiometer to adjust the reference voltage to a suitable level, and combines r and p to acquire scaled ADC readings (s) using the linear formula: $s = 210 \cdot p + r$

Then, the sensing algorithm averages scaled ADC readings to acquire Mag-Points, using the following moving average

$$\mathfrak{m}_0 = s_0$$
$$\mathfrak{m}_n = \alpha_{mp} \cdot s_n + (1 - \alpha_{mp})\mathfrak{m}_{n-1}$$

Here $\mathfrak{m}_n$ is the n^{th} Mag-Point, and s_n is the n^{th} scaled ADC reading. The process of generating Mag-Points from raw magnetometer signals filters out high frequency noise and the results are relatively reliable measures representing the current magnetic field intensity. As a comparison, Fig. 4(a) shows the waveform of raw magnetic signals (scaled ADC readings) sampled at 32Hz when an iron bar moved at 5 feet away. As we can see, the signals of the moving iron bar are hidden in high noise.

In contrast, Fig. 4(b) shows the waveform of the Mag-Points, with $\alpha_{mp} = 1/18$, for the same target. The signal is more evident with Mag-Points, which filters out a large part of the noise. Therefore, the Mag-Point is not only a uniform numeric value that is easy to use, but also a loyal indication of the magnetic field intensity that higher-level algorithms can rely on. Such a low-complexity technique is applicable to many amplitude based signals.



(a) Scaled ADC readings (b) Mag-Point readings

Figure 4: Scaled ADC readings and Mag-Points collected from the same sensor in two consecutive runs

4.1.2 Thermal Drift

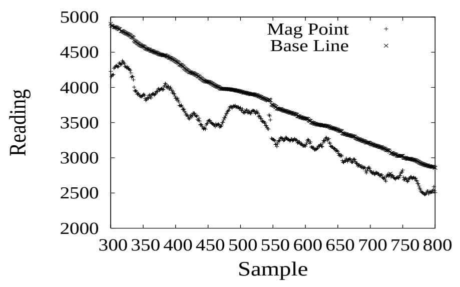
Thermal drift is the most difficult noise the sensing algorithm needs to filter out. Fig. 5(a) shows the magnetometer observations on the X-axis when a sensor node was moved from an air-conditioned room to outdoors on a sunny day. The Mag-Point readings, sampled at 32Hz, fluctuated and dropped quickly in about 15 seconds. Sometimes, the thermal drift is identical to a ferrous target. Fig. 5(b) shows readings collected at noon on a cloudy and windy day. The sensor node was an ExScal mote version 1, which has no enclosure. The frequent alternations of sunshine and shadow caused the temperature of the exposed magnetometer to change quickly. Note that the readings from 300 to 500 (about 6 seconds) is similar to a car moving slowly. Such an intrinsically ambiguous thermal drift cannot be filtered out algorithmically. In such situations, other measures must be employed to avoid such ambiguity. Packaging is the most important supplementary factor that ensures that thermal drift does not produce ambiguous signal waveforms.

Assuming intrinsically ambiguous thermal drifts are eliminated by methods other than software, frequency based analysis can be used to filter out other thermal drifts. The thermal drift is a relatively slow change, i.e., low frequency noise. To eliminate this, the sensing algorithm uses another moving average, which assigns more weight on history, to compute the current base signal line. The formula for B_n (the n_{th} point in the base signal line) is

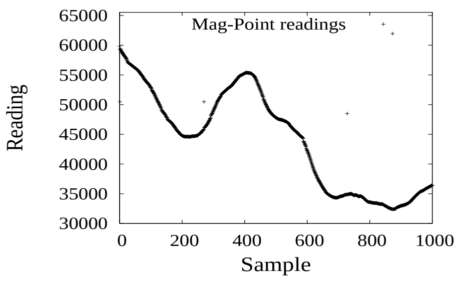
$$B_0 = s_0$$
$$B_n = \alpha_B \cdot s_n + (1 - \alpha_B) \cdot B_{n-1}$$

Fig. 5(a) also shows the base signal line. As we may notice, when the sensor readings change, the base signal line readings change at a slower speed than the Mag-Points. When the Mag-Points deviate from the base signal line for an amount larger than a threshold, detection occurs.

By using two moving averages with very low computational complexity, the magnetic sensing algorithm filters out both high frequency and low frequency noise, solves the problems of non-uniform



(a) Readings as temperature increases



(b) Readings on a cloudy and windy day

Figure 5: The impact of temperature on the magnetometer

sensor reading, low signal/noise ratio, and thermal drift, and accomplishes a resilient detection algorithm.

4.1.3 Trade-off between sensitivity and resilience

The parameters α_{mp} and α_B affect the performance of the magnetic sensing and must be carefully chosen so that the magnetic sensing algorithm is not only sensitive, but also resilient to noise and environmental changes.

The parameter α_{mp} affects how effectively the algorithm averages out high frequency noise. If $\alpha_{mp} = 1$, there is no noise filtering. As we decrease α_{mp}, high frequency noise is filtered out by the averaging process, and small signals are able to emerge from the background noise. When α_{mp} approaches 0, however, the history readings overwhelm the new reading so much that signals lasting for a short period of time cannot distinguishably change the Mag-Point readings. Hence, the algorithm becomes unable to detect a target unless it is moving very slowly. This means that the sensitivity decreases, and the false negative rate increases, when α_{mp} is too large, or too small.

The parameter α_B aims to establish a baseline to characterize the ambient magnetic field strength without targets. If $\alpha_B = \alpha_{mp}$, the baseline reading B_n is the same as the Mag-Point $\mathfrak{m}_n$, and there can be no detection since their difference is always 0. With α_{mp} fixed and α_B decreasing, the baseline becomes more stable. When a target approaches, the Mag-Points change faster than the baseline. Generally, the smaller the α_B, the larger the difference between the baseline and Mag-Points, and the more sensitive the magnetic algorithm is. However, when α_B increases, the baseline also becomes less adaptive to environmental changes, such as the temperature change, and becomes more likely to report false positives. When $\alpha_B = 1$, the algorithm has the maximum sensitivity, but shows a very weak resilience because it does not adapt to the environment at all and thus any environmental change can trigger a false positive.

As we can see from the analysis above, choosing a suitable α_{mp} and α_B is a design decision that affects the magnetic sensing algorithm's performance. Their ranges of suitable values are dependent on the application requirements, the sensor properties, and the expected environmental variability. In VigilNet, we choose $\alpha_{mp} = 1/4$ and $\alpha_B = 1/64$, after weighing the above factors and experimenting with a number of settings.

4.2 Sensing Algorithm for Motion Sensors

The task for motion sensors is to detect movement of an object in the region where the sensor network is deployed. The motion sensors on sensor boards are peroelectric infra-red (PIR) sensors. They sense changes in the thermal field over the region. During the time when an object is moving through, the variations of the thermal field result in unbalanced infra-red signals detected by the lens pairs in the PIR sensor, leading to positive detections. Unlike the magnetometer, the PIR signals are AC signals, not amplitude based. A distinctive challenge to designing a reliable motion sensing algorithm is the weather. Hence, we introduce a motion sensing algorithm, focusing on its low-complexity frequency based processing and environmental resilience.

4.2.1 Increasing S/N ratio by filters

In outdoor environments, the performance of PIR sensors depends heavily on the weather conditions, including wind, temperature and humidity. Wind makes the air move and grass and trees swing, causing the thermal field to change since the air temperature is not uniform and grass and trees have different temperatures. Fig. 6(a) shows PIR data collected by a sensor in grass on a hot, humid and windy day and Fig. 6(b) is the spectrum of the signal. There is a moving target in the area between 60s and 70s. On hot, humid and windy days, when the sensors are placed in grass, a simple energy detector either generates false positives, if using a low threshold, or misses targets, if using a high threshold. We observe that the low frequency component, less than 1 Hz, dominates the noise. When a target moves through, the frequency components larger than 2Hz become significant. This motivates us to explore frequency based signal processing on PIR data.

Because of the limited computation resource and the time constraints of the application, we design a high pass filter as follows.

$$\begin{aligned} &\mathfrak{m}_0 = 0 \\ &\mathfrak{m}_n = s_n - s_{n-1} + 0.9\mathfrak{m}_{n-1} \end{aligned}$$

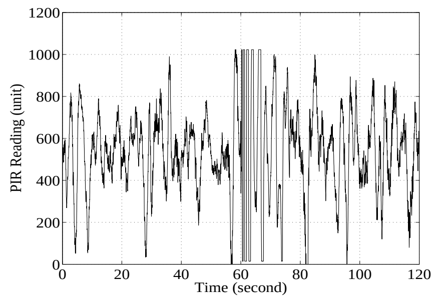
Fig. 7(a) shows the frequency response of the filter. Fig. 7(b) shows the spectrum of filtered PIR data on a hot and windy day collected by a sensor in grass. The coefficient 0.9 is decided empirically with different filters on PIR data collected outdoors. Although a higher order filter could achieve lower gain for components less than 1 Hz, it does not significantly improve the performance.

Fig. 8(a) is the filtered signal of the one in Fig. 6(a). When the moving object passes, there is considerable energy variation in the signal. A simple energy detector can then be applied to the filtered signal to detect movements with a low false positive rate.

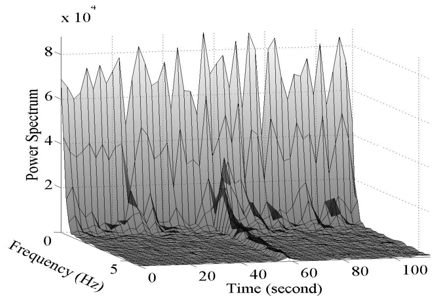
4.2.2 Unsupervised adaptation to environment

In the motion sensing algorithm, the energy based target detection threshold must be set based on the noise level. However, in realistic environments, the noise level is not fixed. The low frequency noise is very weak on cold and arid days, but can be strong on hot and windy days. Fig. 8(a) and Fig. 8(b) compare the PIR data for two different scenarios. Obviously, we cannot achieve good performance with a fixed threshold in all types of weather conditions.

To solve this problem, we use an unsupervised adaptation tech-

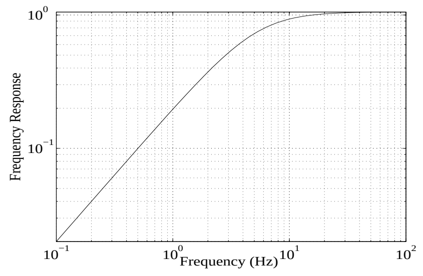


(a) PIR data

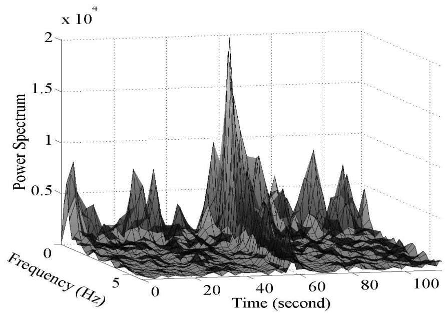


(b) Spectrum of PIR data

Figure 6: PIR readings from sensor in grass



(a) Frequency response



(b) Spectrum(filtered data)

Figure 7: PIR data filter

nique to adjust the threshold. The sensors continuously compute the noise level based on local measurements and adapt the threshold proportional to the noise level. To compute the noise level, the motion sensing algorithm monitors the maximum power p_n of the filtered signal within a time window. The noise level ϵ_n is updated by the following computation.

$$\epsilon_n = \begin{cases} p_0 & n = 0 \\ 0.98\epsilon_{n-1} + 0.02p_n & \epsilon_{n-1} < p_n \\ 0.75\epsilon_{n-1} + 0.25p_n & \epsilon_{n-1} \geq p_n \end{cases}$$

The motivation of this formula is to let ϵ_n increase and decrease at different speeds. This is because the weather changes slowly therefore we don't need to increase the noise level quickly to adapt to the weather change. A small weight on p_n for $p_n > \epsilon_{n-1}$ avoids the identified noise level increasing too fast when there are moving targets. Once there is no target, we decrease the noise level quickly with large weight on p_n for $p_n \leq \epsilon_{n-1}$. Fig. 9(a) shows the signal power of filtered PIR data in Fig. 8(a). The dashed curve is the identified noise level and the dashed-dotted curve is the updated threshold that is $1.5\epsilon_n$. An exceptionally large noise after 80 seconds causes a false detection.

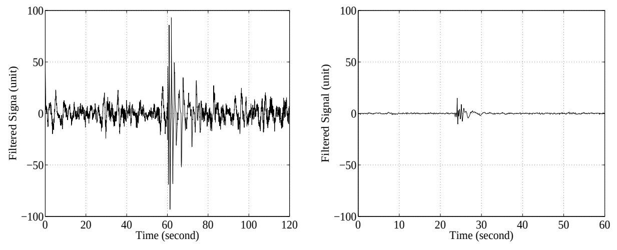
The motion sensing algorithm monitors the number of detections within a time window and defines the percentage of the detection within the time window to be the confidence of a target in the field. Fig. 9(b) shows the sensor confidence for the signal in Fig. 6(a).

4.3 Sensing Algorithm for Microphone

VigilNet uses acoustic sensing to differentiate between vehicles and humans. Acoustic sensing is unique in its relatively high frequency in sampling and processing. The resource constraints make it challenging to design a reliable acoustic sensing algorithm. First, the simultaneous use of magnetic and motion sensing limits the rate at which we can collect acoustic samples. Second, the CPU must remain available at all times to process incoming messages. Third, the system must continuously process acoustic data in order to detect and identify targets in time. Fourth, our whole surveillance system only has 4KB RAM for its functioning: our acoustic algorithm should occupy as little memory as possible.

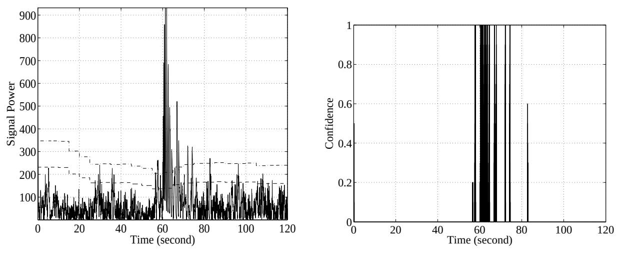
Frequency analysis could be an effective method to conduct acoustic detection and classification. Unfortunately, computing the frequency spectrum by FFT and analyzing the spectrum are expensive operations in our design context. The number of multiplications it takes to get the frequency domain results is $\Theta(N \log_2 N)$. The microcontroller ATmega128L used in the Mica2 and ExScal motes does not support native floating-point multiplication and the clock rate is between 4 MHz to 8 MHz. Xu shows that in [23], it takes a Mica mote with a 4MHz processor 30 seconds to finish a 512-point FFT. Hence, an ExScal mote with a similar processor runs 15 seconds for a 512-point FFT even if it is running at its maximum 8 MHz clock rate. Such a long latency is not acceptable in our application. The space complexity is another issue. Although there are in-place fixed-point FFT solutions, even when we consider a 1024-point FFT and each data point is 16 bit, an in-place solution still uses at least 2 KB space just for the data points. In order to save the 16-bit trigonometric value table, which is necessary for FFT calculation, another 2 KB is needed. In Mica/Mica2 series motes, the RAM size is 4KB and a large proportion of the RAM needs to be assigned to other modules. Of course, the off-chip Flash can be used as secondary storage, but frequent writes to the Flash makes the FFT computation even slower and quickly damage the Flash.

We, therefore, choose a less costly power-based scheme. Each time we obtain a new acoustic sample, we update an exponentially


(a) Data from a sensor in grass(hot and windy)

(b) Data from a sensor on ground(cold day, no wind)

Figure 8: Filtered PIR readings


(a) Signal power and noise level

(b) Sensor confidence

Figure 9: Signal power and detection confidence

weighted moving average of acoustic sample values, noted m_1:

$$\begin{aligned} m_{1,0} &= s_0 \\ m_{1,t} &= \alpha_1 \cdot s_t + (1-\alpha_1) \cdot m_{1,t-1} \end{aligned} \tag{1}$$

Where $m_{1,t}$ is the current value of m_1, $m_{1,t-1}$ is the previous value of m_1, s_t is the current microphone reading and α_1 is a constant determining the relative importance of recent readings. In our current system, α_1 is empirically determined to be 0.001. Fig. 10 graphs the raw acoustic data provided by an ExScal mote when three vehicles pass. The corresponding evolution of $m_{1,t}$ is also presented. We use this moving average to serve as a reference in the computation of E_t, a variable related to the instantaneous acoustic energy:

$$E_t = |s_t - m_{1,t}| \tag{2}$$

Then we compute an auto-adapting acoustic threshold that detects acoustic events. We chose this threshold to be the sum of an exponentially weighted moving average of E_t, noted m_2, plus what we name an exponentially weighted moving standard deviation, noted d_2. equations:

$$\begin{aligned} m_{2,t} &= \alpha_2 \cdot E_t + (1-\alpha_2) \cdot m_{2,t-1} \\ v_{2,t} &= \alpha_2 \cdot (E_t - v_{2,t})^2 + (1-\alpha_2) \cdot v_{2,t-1} \\ d_{2,t} &= \sqrt{v_{2,t}} \end{aligned} \tag{3}$$

Here v_2 represents an exponentially weighted moving variance. In our current implementation, α_2 is empirically determined to be 0.02. When $E_t > m_{2,t} + d_{2,t}$, the algorithm considers that the acoustic threshold has been crossed.

A circular table maintains the number N_t of acoustic threshold crossings that occurred during the last 1280 milliseconds. The value of N_t determines the nature of an acoustic event. When N_t is greater than a certain predetermined value T ($T = 8$ in our experiments), the system signals the detection of a vehicle. Fig. 11 graphs the values of E_t, $m_{2,t}$ and $d_{2,t}$ for the previously mentioned data

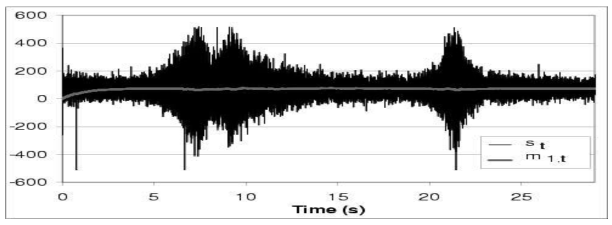


Figure 10: Raw acoustic data from three vehicles

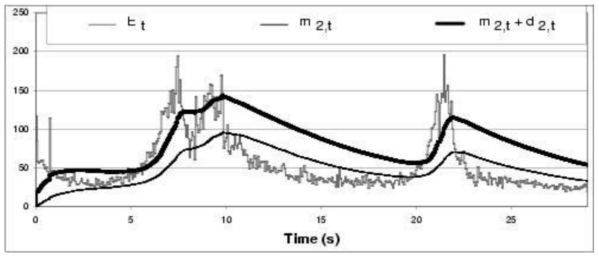


Figure 11: Acoustic Energy and threshold for three vehicles

set for three passing vehicles. Fig. 12 represents the corresponding values of N_t. We remark that the mote triggers a car detection event during the first three seconds of the algorithm execution. This erroneous detection is due to the fact that, at the beginning of the algorithm execution, the moving averages are arbitrarily set to zero. To resolve this problem, the system disregards acoustic detection results during the first five seconds of its execution.

4.4 Group-Level Classification

Distinguished from previous in-network data processing schemes [6, 9, 16], the groups in the VigilNet are more dynamic in the sense that they are formed in response to an external "event", which corresponds to a target in VigilNet, and migrate with the movement of the event. The details of the group forming, migration, and deletion can be found in [15]. In this section, we introduce groups' functions in the classification system – collecting, filtering, and aggregating node-level classification results, as well as triangulating the estimated target locations.

Each group has a statically assigned group leader. When events occur, group members periodically report to the group leader. The reports usually consist of node information (e.g., node ID and location), group information (e.g., leader ID and group ID) and event information (e.g., confidence vectors). The group leader aggregates the confidence vectors from group members, computes group-level confidence vectors and reports them to the base-level classification module via multi-hop communication. This scheme greatly reduces the amount of network traffic and, consequently, the energy consumption of the WSN.

Data aggregation contains several tunable parameters that affect different aspects of its performance. One parameter, minimum degree of aggregation (MDOA), defines the minimum number of distinct reports required to form a valid group-level confidence vector. An adequately high MDOA value enhances the credibility of group-level classification results. Hence, it is an important system parameter and has an impact on the performance of the detection and classification, which is to be discussed in Section 5.2.

In the classification system, the groups have a one-to-one mapping to physical events. An implicit assumption is that events always keep a far enough distance among them, so that membership

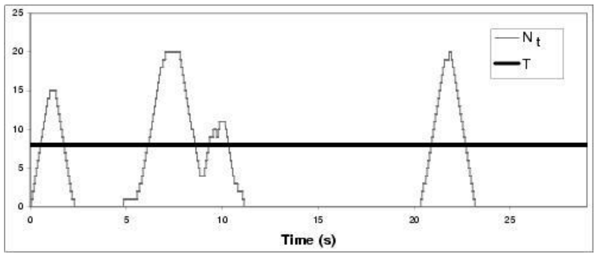


Figure 12: Number of threshold crossings for three vehicles

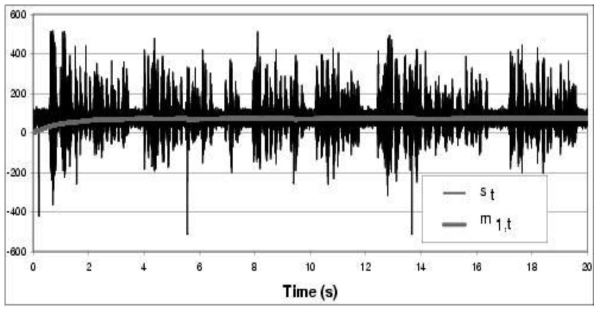


Figure 13: Raw acoustic data from human speaking

of nodes to the corresponding groups can be determined without ambiguity based on spatial adjacency to one of the events. This results in a limitation on detecting multiple simultaneous targets – for events that become close enough or cross each other, if they share the same sensory signature (e.g., two persons walking together), the current classification system cannot separate them.

If events are with different sensory signatures, different classes of events can be resolved based on history data after events deviate from each other. However, before such events deviate, there is still a temporary ambiguity. For instance, when a group for a vehicle and a group for a person cross, the person triggers detections on the motion sensors, and the vehicle triggers detections on the motion, acoustic, and magnetic sensors. Hence, the two groups merge to be a group for a vehicle, sensing an event with motion, magnetic, and acoustic features. Later, when the person and vehicle deviate, the ambiguity will be resolved and two groups will be formed. This is another limitation of the current classification system – events of different signatures may still have "temporary ambiguity" because the groups are formed by detecting nodes, not by nodes detecting a specific type of signature.

Potentially, temporary ambiguity can be resolved by group management with a finer granularity – a group is formed for a specific type of signature, hence multiple groups co-exist in the same vicinity. Another solution is to have the base mote disambiguate the events, based on track history and assumptions on trajectory. However, our current system does not pursue either of the approaches in order to keep time, space, and communication complexity low. Instead, we design the system so that the effect of the ambiguity is minimized. Specifically, a group still reports an event even when there is temporary ambiguity, allowing the system to still reacts to the event. Furthermore, when there is ambiguity, the classification tends to categorize it as a class of a higher alert level (e.g., a person and a vehicle are identified as a vehicle). On the other hand, for applications that require a better disambiguity capability, the hierarchical classification architecture allows more sophisticated group-level and base-level algorithms to be incorporated.

4.5 Base-Level Classification

The highest level detection and classification are conducted on the base mote. It takes the group-level classification results as input and computes the final classification results. Since the base mote has a global view of the classification process, it conducts the tasks requiring global knowledge, which is not available to individual nodes or groups. In order to further reduce false positives, spatial and temporal correlations among the tracking reports must be leveraged. Intuitively, the base mote deems that two reports in a certain time frame are from the same target if their locations are close. The base mote keeps a history of recently received reports. With the history of reports, the classification and velocity calculation of each target can be accomplished with high accuracy.

In the RAM of the base mote, a small data structure for each target is maintained. The data structure includes the recent location of the target, the latest timestamp, accumulated sensor values and a pointer to the information of the last report for the target. The base mote chooses the target whose recent location is the closest to the location of the incoming report and decides that the report belongs to the target. If there is no target or the closest distance from the recent location of any target to the location of the reports is greater than a predefined threshold, the report is considered to be from a new target. This threshold needs to be tuned in real-system testing. If it is too large, reports from multiple targets may be categorized into one group. If it is too small, two consecutive reports from a single target may be categorized into two groups. Currently in our system we use a threshold of 60 meters, which shows good performance results in experiments. In order to minimize the number of false positives, a target is reported to the front end interface only if the number of reports for it exceeds a predefined threshold. With this approach, most sporadic false positives can be filtered out.

Once a target accumulates enough reports, the base mote reads its history and applies a linear regression to calculate the velocity of the target, because velocity is one of the most important aspects for moving target tracking, and it is of great interest to the end users. The least square regression approach has been used in many scientific and engineering fields for a long time and is believed to be highly robust against small numbers of outliers. For each direction, the timestamps and the coordinates of the locations of the the most recent reports are used in the regression. The least square algorithm gives the average changing rate of the coordinates over time. This rate serves as the component of velocity along the direction. With the information of both velocity components, we can get the velocity of the target including the knowledge of its moving direction.

5. SYSTEM PERFORMANCE

In this section, we evaluate the the performance of VigilNet with a focus on the detection and classification performance. First, we evaluate the performance of sensing algorithms in Section 5.1. Then, Section 5.2 studies the group-level classification by analyzing the impact of MDOA on the classification performance. Finally, Section 5.3 assesses the overall system performance.

5.1 Evaluation of sensing algorithms

Among the three sensing algorithms, the acoustic sensing algorithm has the highest sampling rate and CPU utilization. Since a detailed analysis of all three algorithms has to be lengthy, we choose to study the acoustic sensing algorithm as a representative and evaluate its detection rate. To evaluate the performance of the acoustic sensing algorithm, we deploy 7 sensor nodes in a line with 3-meter spacing. This line is perpendicular to the trajectory of a

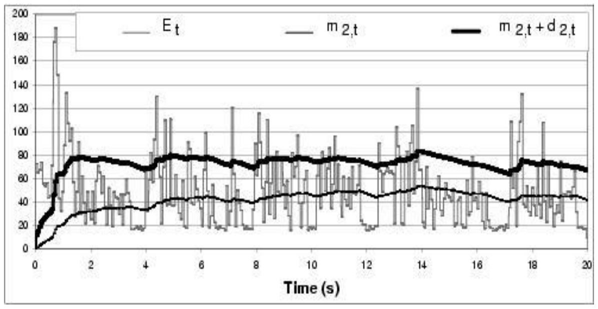


Figure 14: Energy and auto-adaptive threshold for human speaking

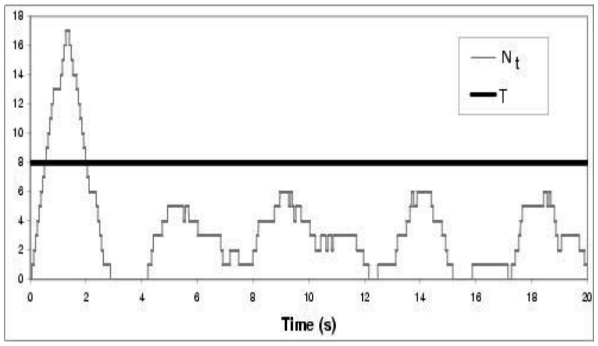


Figure 15: Number of threshold crossings for human speaking

passing car, with the first node located 3 meters from the trajectory. We drive the car at three different speeds: 10, 15 and 20 miles per hour. We realize ten trials for each speed in a parking lot and compute the success rate of our algorithm at various distances and speeds. Fig. 16 presents the results of this experiment. We observe that the success rate of our algorithm decreases as the distance to the car increases. Also, the algorithm is more successful when the car moves at higher speeds: this is not surprising as a rapidly moving car generates more acoustic power. In VigilNet, sensor nodes are approximately 33 feet (10 meters) away from each other. A sensing range of 16.5 feet (5 meters) guarantees the detection of a target traversing the field. Considering the resource constraints, our design does not emphasize very high detection rate on individual nodes. Hence, the performance of the acoustic sensing, with a detection rate of 90% at 30 feet (9 meters), is sufficiently good.

To demonstrate how our acoustic algorithm reacts to other sound sources, we experiment with a human speaking loudly at a distance of 1.8 meters (6 feet) from an ExScal mote. Note that, without sophisticated frequency analysis, the acoustic sensing algorithm is not designed to distinguish human voice and vehicle sound. However, according to experimentation, a human, even if speaking loudly, does not generate as much acoustic energy as a car passing close to the sensor node. This is why the algorithm, evaluating acoustic energy, can differentiate between these two types of targets. On the other hand, if the acoustic sensing algorithm shows a good resilience to human voice, which is strong at such a short distance, it indicates a good performance against background noise. The results are presented in Fig. 13, Fig. 14, and Fig. 15. Clearly, the acoustic algorithm does not trigger any vehicle detection event except during the first three seconds. As we said earlier, acoustic detections during this initial phase are ignored. Hence, from the

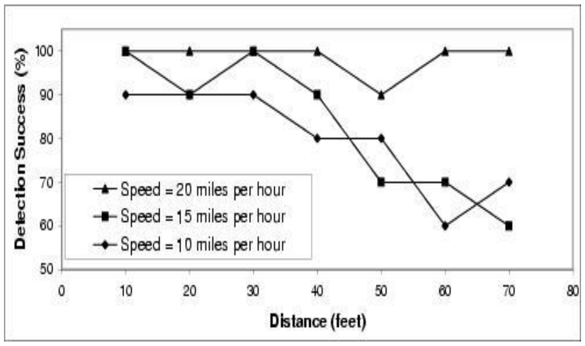


Figure 16: Acoustic detection performance.

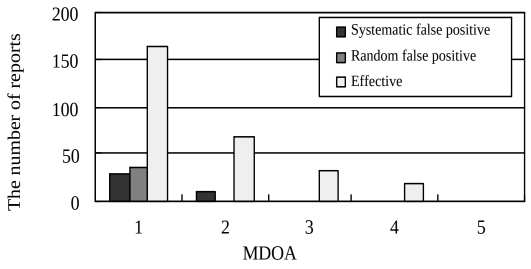


Figure 17: The impact of the MDOA.

system's view, no human speaking events are reported as vehicles in this test.

5.2 Evaluation of the group level classification

Among the functions of group level classification, the MDOA (the minimum degree of aggregation) is of special importance to the detection and classification performance. It not only controls the aggregation of member reports and reduces the network traffic, but also reduces the false positive rate of the system.

From a system designer's point of view, false positives roughly fall into two categories. One category of false positives are due to unexpected disturbance imposed on a small number of nodes. For example, when a wild animal touches a sensor node, that node may sense motion and magnetic signals and report false positives. A loose connector, a piece of shortcut wire, or a malfunctioning sensor may trigger continuous wrong readings from the sensor. Such false positives are mostly independently and randomly distributed and only occur at a reasonably low rate. We call this category of false positives "random false positives". In some other situations, a large percentage of the sensors in a WSN become much more probable to report false positives, because of flawed design or unexpected disturbances imposed on a large percentage of the network. False positives in this category are usually correlated, and sometimes bursty. For example, when the sensor driver has a bug, or there is a storm, the entire network may report false positives in large quantity. We call this category "systematic false positives".

By using a suitable MDOA, the group level classification can significantly reduce random false positives, and noticeably mitigate the effect of systematic false positives. Fig. 17 shows the result of a test studying the relationship between the MDOA and the number of false positive reports. Performed at an outdoor parking lot with a test VigilNet system of 37 nodes, the test involves the magnetometer and motion (PIR) sensor detecting a moving vehicle. The thresholds for the motion sensing algorithms are set to an extremely low value so that there are frequent false positives. Also, on the current hardware platform, the magnetometer is sensitive to LEDs. Hence, we let the LEDs blink at the beginning of the classification stage so that the magnetometers acquire some wrong data and generate false positives.

The testing procedure is as follows. First, we start the system, with MDOA set to 1 (all reports are delivered to the base mote). At this point, the LED blinks and the initial low threshold triggers false positives on the magnetometers and motion sensors. We record the number of reports in the first 32 seconds, and use them, as a reasonable approximation, as the number of systematic false positives. We record the number of reports in the following 3 minutes as the number of random false positives. Then, we send a middle-sized car to the WSN field, record the number of reports delivered during the tracking process (35 seconds) as the number of effective reports. We also examine whether the classification result is correct. Tests are repeated and statistics are collected for various MDOA settings.

As we can see from Fig. 17, when the MDOA is 1, there are 29 systematic false positives and 36 random positives. Such a high false positive rate confuses the base level classification algorithm and the system reports false targets. On the other hand, the system is still able to track the real target – the 165 effective reports make the system successfully detect and classify the vehicle.

When the MDOA increases past 1, all random false positives are eliminated. And the number of systematic false positives reduces by 85% – from 65 to 10. Meanwhile, the number of effective reports also reduces from 165 to 69. But the system is still able to detect and classify the target vehicle correctly.

When MDOA increases to 3 or 4, all the systematic and random false positives are removed. And we verified that, though the number of effective reports is further reduced, the system detects and classifies the target correctly. When MDOA is 5, no reports are delivered in the system.

In conclusion, adjusting the MDOA is an important method to reduce the number of false positives in a WSN, and significantly enhance it's performance. Meanwhile, a too-high MDOA lowers the system's sensitivity.

5.3 System level performance

In this subsection, we evaluate the VigilNet's performance as a holistic system. Especially, we measure how fast the network classifies targets and how accurately it computes the target's attributes. Among a number of attributes, velocity is our major interest and a good representative of high-level target attributes that cannot be accurately computed on individual nodes. Hence, in this section, the discussion of attribute computation focuses on velocity. We deployed and tested the VigilNet in an airfield in the July and December of 2004. Unless otherwise specified, the performance data in this section is collected from tests on these two deployments.

The test scenario involves moving targets traveling through the network following a straight trajectory. A moving target may be a vehicle, a person, or a person carrying a ferrous object. The network comprises 200 ExScal motes.

Table 1 shows statistics collected in an outdoor test site. Ten targets are tracked in two runs. In this test, the required minimum degree of aggregation is set to one in the group level classification in order to inspect the base mote's ability to filter out false positives. All the 10 targets are detected and correctly classified. In total, the network delivers 441 reports to the base mote, which, after processing these reports, delivers 71 reports to the external device. Interestingly, the network delivers more reports in run 1 than in run 2, even though there are more targets in run 2. The reason is that the number of reports from the network depends not only the number of targets, but also the amount of time the targets stay in the network, the types of the targets, and the number of group leaders in the network. In these two runs, the network generated totally 29 false positive reports at the node level and the group level, but all of them are filtered out by the base mote. Hence, from the system's view, there are no false positives and, since all targets are detected, there are no false negatives. Not surprisingly, the network's performance is better than individual nodes. The reasons is that the natural redundancy in a densely deployed network help reduce the false negative rate at the system level.

Run No.	Run 1	Run 2
Duration(s)	271	758
Group-level reports	261	180
Reports after filtering	29	42
Actual targets	4	6
Correctly classified targets	4	6
False negatives	0	0
Filtered false positives	5	24

Table 1: Statistics of classifying 10 targets in two runs

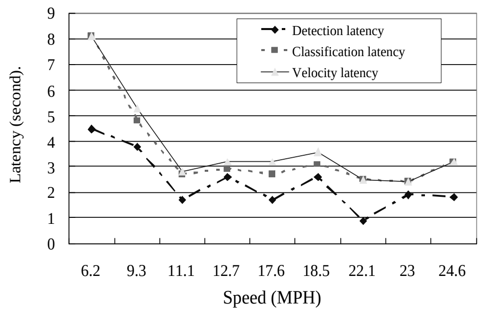


Figure 18: Latencies for vehicles at different speeds

Fig. 18 plots the detection latencies, classification latencies and velocity calculation latencies for vehicles at various speeds. Generally, the detection latency is lower than the classification latency, and the classification latency is lower than the velocity calculation latency. This difference reflects different amounts of information required for the detection, classification, and attribute calculation. The classification of a target employs a longer history of reports than the detection. The velocity calculation needs a even longer history, in order to accomplish a precise linearly-fit inference of the velocity. Also, the latencies reduce when the speed increases. The reason is that, when the target is traveling at a low speed, the time for the target to travel past multiple nodes dominates the total latency. When the speeds increase, the latency remains at a certain level. This is because, when the speed is high, the processing, queuing, and group-level aggregation latency dominate the total latency.

In the runs shown in Fig. 18, all targets are classified correctly, indicating a satisfactory classification capability. Meanwhile, it is interesting to compare the calculated velocity with the velocity shown on the vehicle speedometer. Our record shows that the range of error between the calculated velocity and the real velocity ranges from -7.5% to $+15\%$.

The motion of persons and persons with ferrous objects have are similar characteristics because the moving carriers are of the same type – human beings. Hence, they show similar latencies in the tests. However, their latencies are longer than those for vehicles. Fig. 19 shows the average detection, classification and velocity

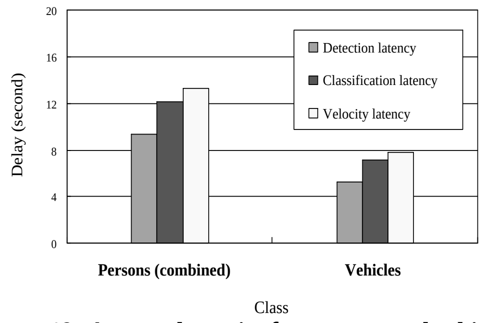


Figure 19: Average latencies for persons and vehicles

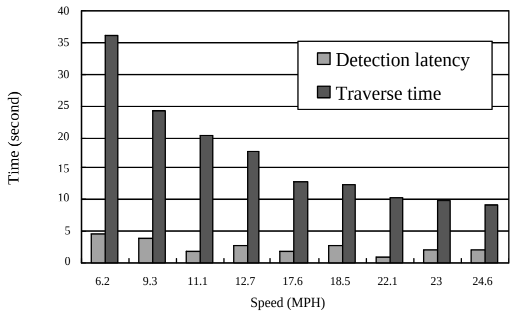


Figure 20: Detection latencies and traverse time for vehicles

calculation latencies for the two classes of persons combined, and vehicles. We notice detection latency for persons combined is 77% longer than that for vehicles. This is because the persons travels much slower in the network than vehicles, hence it takes longer time for persons to hit enough sensor nodes and trigger enough detection and classification reports to establish a sufficient confidence level for detection on the base mote.

As mentioned in Section 1, a surveillance WSN must operate in a timely manner. Given that the detection latencies for persons and some slow vehicles can approach 10 seconds, it is necessary to verify that the latencies are all within an acceptable range. One design detail in VigilNet is that the deployment ensures that a target has to travel about 330 feet (100 meters) to traverse the network. Based on this we can calculate the minimum amount of time that it takes a target to traverse the network, henceforth called the "traverse time". Suppose a person travels at 2–10MPH, it takes 22–112 seconds to traverse the network. As shown in Fig. 19, the detection latencies for persons are much shorter than the traverse time. As for vehicles, the traverse times are shorter for faster vehicles, but the detection latencies are usually shorter, too. To examine the timeliness of the detections, we plot the detection latencies and the traverse times at different speeds in Fig. 20. As we can see, the detection latencies are much shorter than the traverse times at various speeds.

From the analysis of performance at the sensor level, the group level, and the base level, we can clearly see the refining process of the detection and classification results. The detection rate at the sensor level is often not perfect. For instance, the acoustic sensing algorithm's detection rate is about 90% at a distance of 9 meters (30 feet). However, the redundancy of the network nodes ensures that the holistic system has a high detection rate (low false negative rate). The group-level classification significantly reduces the false positive rate and minimizes the network traffic. Finally, the base-level classification refines the detection and classification results by analyzing tracking reports from multiple groups. In summary, the evaluation shows that VigilNet accomplishes an excellent performance in reliable sensing and classification, accurate attribute (velocity) computation, resilient operation in realistic environments, and timely information delivery.

6. METHODOLOGICAL DISCUSSIONS

We believe that the design, implementation and evaluation of the sensing subsystem and classification algorithms should evolve from an ad hoc "art" to established methodologies. Though our experience with VigilNet and a study of several recent surveillance WSNs are not sufficient to abstract such a methodology, the challenges we faced do represent a series of common issues and the design choices we made reflect the diligent thoughts, careful trade-offs, and realistic concerns involved in constructing a realistic, sophisticated, and evolving system. Hence, we conceive that it is valuable to share our view and conception on how to design a detection and classification system, and believe this can help current and future WSN research to establish a systematic methodology for designing such WSNs. For this purpose, we abstract some general guidelines for the development of sensing and classification systems. Most of them are not new concepts, but are of critical importance to the success of realistic systems.

- *Mechanical design.* Though programmers traditionally do not care about the mechanical details of a computer system, designers of a sensor network must pay careful attention to it. For example, without a suitable enclosure, the magnetometer would suffer degraded performance at sudden temperature changes. Generally, the enclosure design for WSN nodes should provide a suitable operating environment to the hardware components [8], besides protecting the node hardware from harsh environmental conditions. Specifically, the positioning and wiring of various components should avoid interference from each other and maximize sensors' capability.
- *Autonomous operation.* It is infeasible to individually manage the network nodes in a large-scale WSN. Hence, each must operate in an autonomous manner. Specifically, a node must identify, calibrate, and operate its sensors automatically.
- *Fault tolerance.* For a large system to operate for a long period of time on a rough terrain, it must expect the unexpected. For example, strong wind or wild animals may disturb the sensor nodes, displace them, or even destroy them. Also, the large size of the network makes faults a common phenomenon – if each node has a 0.001 possibility to have a hardware fault, a network of 200 such nodes has a 0.18 possibility to contain a faulty node. Though it is infeasible to analyze and intelligently handle all possible situations, the design of such WSNs must, at minimum, deal with failures at various levels.
- *Adaptivity.* WSNs, especially when they are deployed outdoors, show a high level of dynamics. The quality of communication links, the electric characteristics of sensors, and the topology of the network, may continuously change due to internal and external conditions. Hence, many system parameters need to continuously adapt to changes inside and outside the network.
- *Redundancy and collaboration.* The performance of a network of energy-and-cost-effective nodes largely relies on how the nodes collaborate with each other. Enhancing the performance and capability of individual nodes is important. Many developers, including the authors, feel it intellectually exciting to answer the challenge of high-quality signal processing with low-end hardware by novel and prudent architectural and algorithmic designs of individual sensor nodes. Meaningful and important as it is, such an effort, if overemphasized, may prove inefficient or, in some situations, even hazardous, in realistic settings.

For example, an intrinsically difficult trade-off is that the more sensitive the sensing algorithms are, the more vulnerable to the changing environments they become. As another approach, we may choose to make the sensing algorithms less sensitive, but more resilient to environmental changes, and take advantage of the density of the network nodes to enhance the overall sensitivity of the system. Also, as we discussed in Section 5.2, a group of sensor nodes can collaborate with each other to reduce the false positive rates. Such a redundant and collaboration based approach proves to be highly effective.

7. CONCLUSION

We discussed the sensing subsystem in the VigilNet surveillance system and described how the hierarchical classification architecture enables the system to conduct efficient information processing, including detection and classification, in a large-scale WSN. The hierarchical architecture naturally distributes sensing and computation tasks at different levels of the system so that the sensor network can support high-quality sensing and reliable classification without involving special high-power nodes. With evaluation data collected from field tests in physical environments, the evaluation of VigilNet demonstrates excellent performance on the detection rate, classification result, attribute (velocity) computation accuracy, and timely information delivery.

8. ACKNOWLEDGMENTS

Thanks to Anthony Wood, Shashi Prabh, Prabal Dutta, Cory Sharp, Michael Grimmer, Qiuhua Cao, William Bain, Cole Timothy, Mark Satterfield, and Ed Roach, and many other researchers and staff, whose help makes this work possible.

9. REFERENCES

[1] Exscal web site. *http://www.cast.cse.ohio-state.edu/exscal/index.php?page=main*.

[2] Honeywell magnetometers. *http://www.ssec.honeywell.com/magnetic/*.

[3] Mica2 mote. *http://www.xbow.com/Products/productsdetails.aspx?sid=72*.

[4] Micropower inpulse radar by advantaca. *http://www.advantaca.com/radar.htm*.

[5] VigilNet web site. *http://www.cs.virginia.edu/ control/SOWN/index.html*.

[6] S. Bhattacharya, H. Kim, S. Prabh, and T. Abdelzaher. Energy-conserving data placement and asynchronous multicast in wireless sensor networks. In *Intl. Conf. on Mobile Systems, Applications, and Services (MobiSys)*, May 2003.

[7] R. Brooks, P. Ramanathan, and A. Sayeed. Distributed target classification and tracking in sensor networks. *Proceedings of the IEEE*, 91(8):1163–1171, 2003.

[8] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler. Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In *Proc. of Fourth Intl. Conf. on Information Processing in Sensor Networks (IPSN'05)*, 2005.

[9] Z. Feng, S. Jaewon, and R. James. Information-driven dynamic sensor collaboration for target tracking. *IEEE Signal Processing Magazine*, 19(2), March 2002.

[10] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, G. Zhou, J. Hui, and B. Krogh. Vigilnet:an integrated sensor network system for energy-efficient surveillance. *In submission to ACM Transaction on Sensor Networks*, 2004.

[11] T. He, S. Krishnamurthy, J. A. Stankovic, T. F. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh. An energy-efficient surveillance system using wireless sensor networks. In *Proc. of Intl. Conf. on Mobile Systems, Applications, and Services (MobiSys)*, June 2004.

[12] T. He, P. Vicaire, T. Yan, Q. Cao, G. Zhou, L. Gu, L. Luo, R. Stoleru, J. A. Stankovic, and T. Abdelzaher. Achieving Long-Term Surveillance in VigilNet. In *submission*.

[13] J. Hill and D. Culler. Mica: A wireless platform for deeply embedded networks. In *IEEE Micro*, volume 22, pages 12–24, Nov./Dec. 2002.

[14] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. *Proc. of ASPLOS 2000*, Nov. 2000.

[15] L. Luo, T. F. Abdelzaher, T. He, and J. A. Stankovic. Design and comparison of lightweight group management strategies in envirosuite. In *Distributed Computing in Sensor Systems (DCOSS '05)*, June 2005.

[16] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: A tiny aggregation service for ad-hoc sensor networks. In *Proc. of Operating Systems Design and Implementation*, Dec. 2002.

[17] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *Proc. of the 2nd ACM Intl. Conf. on Embedded Networked Sensor Systems (SenSys'04)*, pages 39–49, New York, NY, USA, 2004.

[18] S. Pattem, S. Poduri, and B. Krishnamachari. Energy-quality tradeoffs for target tracking in wireless sensor networks. In *Proc. of 2nd Intl. Conf. on Information Processing in Sensor Networks (IPSN'03)*, 2003.

[19] G. Simon, M. Maroti, A. Ledeczi, G. Balogh, B. Kusy, A. Nadas, G. Pap, J. Sallai, and K. Frampton. Sensor network-based countersniper system. In *Proc. of the 2nd ACM Intl. Conf. on Embedded Networked Sensor Systems (SenSys'04)*, Nov. 2004.

[20] R. Stoleru, T. He, and J. A. Stankovic. Walking GPS: A Practical Solution for Localization in Manually Deployed Wireless Sensor Networks. In *1st IEEE Workshop on Embedded Networked Sensors EmNetS-I*, October 2004.

[21] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler. An analysis of a large scale habitat monitoring application. In *Proc. of the 2nd ACM Intl. Conf. on Embedded Networked Sensor Systems (SenSys'04)*.

[22] Q. Wang, W. Chen, R. Zheng, K. Lee, and L. Sha. Acoustic target tracking using tiny wireless sensor devices. In *Proc. of 2nd Intl. Conf. on Information Processing in Sensor Networks (IPSN'03)*, 2003.

[23] N. Xu. Implementation of data compression and FFT in TinyOS. *http://enl.usc.edu/ ningxu/papers/lzfft.pdf*.

[24] P. Zhang, C. Sadler, S. Lyon, and M. Martonosi. Hardware design experiences in zebranet. In *Proc. of the 2nd ACM Intl. Conf. on Embedded Networked Sensor Systems (SenSys'04)*, Nov. 2004.

[25] F. Zhao, J. Liu, L. Guibas, and J. Reich. Collaborative signal and information processing: An information directed approach. *Proceedings of the IEEE*, 91(8):1199–1209, 2003.

Range-Free Localization Schemes for Large Scale Sensor Networks[1]

Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, Tarek Abdelzaher

Department of Computer Science, University of Virginia
151 Engineer's Way, P.O. Box 400740, Charlottesville, Virginia 22904-4740, USA
{TianHe, ch4pp, bmb5v, Stankovic, Zaher}@cs.virginia.edu

ABSTRACT

Wireless Sensor Networks have been proposed for a multitude of location-dependent applications. For such systems, the cost and limitations of the hardware on sensing nodes prevent the use of range-based localization schemes that depend on absolute point-to-point distance estimates. Because coarse accuracy is sufficient for most sensor network applications, solutions in range-free localization are being pursued as a cost-effective alternative to more expensive range-based approaches. In this paper, we present APIT, a novel localization algorithm that is range-free. We show that our APIT scheme performs best when an irregular radio pattern and random node placement are considered, and low communication overhead is desired. We compare our work via extensive simulation, with three state-of-the-art range-free localization schemes to identify the preferable system configurations of each. In addition, we study the effect of location error on routing and tracking performance. We show that routing performance and tracking accuracy are not significantly affected by localization error when the error is less than 0.4 times the communication radio radius.

Categories and Subject Descriptors

C.2. [**Computer Communication Networks**]: Network Protocols

General Terms

Algorithms, Performance, Design

Keywords

Localization, Positioning, Location discovery, Sensor Networks[1]

[1] This work was supported in part by NSF grant CCR-0098269, the MURI award N00014-01-1-0576 from ONR, and the DAPRPA IXO office under the NEST project (grant number F336615-01-C-1905).

1. INTRODUCTION

Wireless Sensor Networks (WSNs) distinguish themselves from other traditional wireless or wired networks through sensor and actuator based interaction with the environment. Such networks have been proposed for various applications including search and rescue, disaster relief, target tracking, and smart environments. The inherent characteristics of these sensor networks make a node's location an important part of their state. For such networks, location is being used (i) to identify the location at which sensor readings originate, (for example, identifying a target's position during tracking, providing the location of an earthquake survivor buried underneath rubble during disaster relief operations) (ii) in novel communication protocols that route to geographical areas instead of IDs ([13], [16], [17], [19], [35]), and (iii) when providing other location based services (such as sensing coverage [36] and location directory service [20] to provide medical information about a nearby patient in a smart hospital). In addition to the applications and protocols mentioned, continued research in WSNs will serve to invent and identify many additional protocols and applications, many of which will likely depend on location aware sensing devices.

Many localization algorithms for sensor networks have been proposed to provide per-node location information. With regard to the mechanisms used for estimating location, we divide these localization protocols into two categories: *range-based* and *range-free*. The former is defined by protocols that use absolute point-to-point distance estimates (range) or angle estimates for calculating location. The latter makes no assumption about the availability or validity of such information. Because of the hardware limitations of WSN devices, solutions in range-free localization are being pursued as a cost-effective alternative to more expensive range-based approaches.

This paper makes three major contributions to the localization problem in WSNs. First, we propose a novel range-free algorithm, called APIT, with enhanced performance under realistic system configurations. Second, though many different protocols [3][22][26] have been proposed to solve the localization problem in a range-free context, no prior work has been done to compare them in realistic settings. This paper is the first to provide a realistic and detailed quantitative comparison of existing range-free algorithms to determine the system configurations under which each is optimized. We perform such a study to serve as a guide for future research. Third, no attempt has previously been made to broadly study the impact of location error on various location-dependent applications and protocols. This paper provides insight into the effect of localization accuracy on application performance degradation and identifies bounds on the estimation error tolerated by applications.

The remainder of the paper is organized as follows: Section 2 discusses previous work in localization for sensor networks. Section 3 describes APIT. Section 4 gives brief descriptions of three other state-of-the-art range-free protocols to which we compare our work. Section 5 describes our simulation. Section 6 follows with a detailed performance comparison of the four range-free localization algorithms described. Section 7 further investigates the impact of localization error on various location-dependent applications and protocols such as routing and target tracking. Finally, we conclude in Section 8.

2. STATE OF THE ART

Many existing systems and protocols attempt to solve the problem of determining a node's location within its environment. The approaches taken to solve this localization problem differ in the assumptions that they make about their respective network and device capabilities. These include assumptions about device hardware, signal propagation models, timing and energy requirements, network makeup (homogeneous vs. heterogeneous), the nature of the environment (indoor vs. outdoor), node or beacon density, time synchronization of devices, communication costs, error requirements, and device mobility. In this section, we discuss prior work in localization with regard to these network characteristics, device restrictions, and application requirements. We divide our discussion into two subsections where we present both range-based and range-free solutions..

2.1 Range-Based Localization Schemes

Time of Arrival (TOA) technology is commonly used as a means of obtaining range information via signal propagation time. The most basic localization system to use TOA techniques is GPS [33]. GPS systems require expensive and energy-consuming electronics to precisely synchronize with a satellite's clock. With hardware limitations and the inherent energy constraints of sensor network devices, GPS and other TOA technology present a costly solution for localization in wireless sensor networks.

The Time Difference of Arrival (TDOA) technique for **ranging** (estimating the distance between two communicating nodes) has been widely proposed as a necessary ingredient in localization solutions for wireless sensor networks. While many infrastructure-based systems have been proposed that use TDOA [1][12][28], additional work such as AHLos ([30][31]) has employed such technology in infrastructure-free sensor networks. Like TOA technology, TDOA also relies on extensive hardware that is expensive and energy consuming, making it less suitable for low-power sensor network devices. In addition, TDOA techniques using ultrasound require dense deployment (numerous anchors distributed uniformly) as ultrasound signals usually only propagate 20-30 feet.

To augment and complement TDOA and TOA technologies, an Angle of Arrival (AOA) technique has been proposed that allows nodes to estimate and map relative angles between neighbors [27]. Similar to TOA and TDOA, AOA estimates require additional hardware too expensive to be used in large scale sensor networks.

Received Signal Strength Indicator (RSSI) technology such as RADAR [1] and SpotOn [15] has been proposed for hardware-constrained systems. In RSSI techniques, either theoretical or empirical models are used to translate signal strength into distance estimates. For RF systems [1][15], problems such as multi-path fading, background interference, and irregular signal propagation characteristics (shown in an empirical study of this technology [10]) make range estimates inaccurate. Work to mitigate such errors such as robust range estimation ([11]), two-phase refinement positioning ([29], [31]), and parameter calibration ([34]) have been proposed to take advantage of averaging, smoothing, and alternate hybrid techniques to reduce error to within some acceptable limit. While solutions based on RSSI have demonstrated efficacy in simulation and in a controlled laboratory environment, the premise that distance can be determined based on signal strength, propagation patterns, and fading models remains questionable, creating a demand for alternate localization solutions that work independent of this assumption.

2.2 Range-Free Localization Schemes

In sensor networks and other distributed systems, errors can often be masked through fault tolerance, redundancy, aggregation, or by other means. Depending on the behavior and requirements of protocols using location information, varying granularities of error may be appropriate from system to system. Acknowledging that the cost of hardware required by range-based solutions may be inappropriate in relation to the required location precision, researchers have sought alternate range-free solutions to the localization problem in sensor networks.

In [3], a heterogeneous network containing powerful nodes with established location information is considered. In this work, anchors beacon their position to neighbors that keep an account of all received beacons. Using this proximity information, a simple centroid model is applied to estimate the listening nodes' location. We refer to this protocol as the *Centroid algorithm.*

An alternate solution, *DV-HOP* [26] assumes a heterogeneous network consisting of sensing nodes and anchors. Instead of single hop broadcasts, anchors flood their location throughout the network maintaining a running hop-count at each node along the way. Nodes calculate their position based on the received anchor locations, the hop-count from the corresponding anchor, and the average-distance per hop; a value obtained through anchor communication. Like DV-Hop, an *Amorphous Positioning* algorithm proposed in [22] uses offline hop-distance estimations, improving location estimates through neighbor information exchange.

These range-free techniques are described in more depth in section 4, and are used in our analysis for comparison with our work.

3. APIT LOCALIZATION SCHEME

In this section, we describe our novel area-based range-free localization scheme, which we call APIT. APIT requires a heterogeneous network of sensing devices where a small percentage of these devices (percentages vary depending on network and node density) are equipped with high-powered transmitters and location information obtained via GPS or some other mechanism. We refer to these location-equipped devices as **anchors**. Using beacons from these anchors, APIT employs a novel *area-based* approach to perform location estimation by isolating the environment into triangular regions between beaconing nodes (Figure 1). A node's presence inside or outside of these triangular regions allows a node to narrow down the area in which it can potentially reside. By utilizing combinations of anchor positions, the diameter of the estimated area in which a node resides can be reduced, to provide a good location estimate.

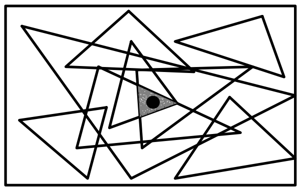


Figure 1: Area-based APIT Algorithm Overview

3.1 Main Algorithm

The theoretical method used to narrow down the possible area in which a target node resides is called the Point-In-Triangulation Test (PIT). In this test, a node chooses three anchors from all audible anchors (anchors from which a beacon was received) and tests whether it is inside the triangle formed by connecting these three anchors. APIT repeats this PIT test with different audible anchor combinations until all combinations are exhausted or the required accuracy is achieved. At this point, APIT calculates the center of gravity (COG) of the intersection of all of the triangles in which a node resides to determine its estimated position.

The APIT algorithm can be broken down into four steps: 1) Beacon exchange, 2) PIT Testing, 3) APIT aggregation and 4) COG calculation. These steps are performed at individual nodes in a purely distributed fashion. Before providing a detailed description of each of these steps, we first present the basic pseudo code for our algorithm:

```
 Receive location beacons $(X_i,Y_i)$ from N anchors.
InsideSet = $\Phi$ // the set of triangles in which I reside
For (each triangle $T_i \in \binom{N}{3}$ triangles) {
   If (Point-In-Triangle-Test ($T_i$) == TRUE)
         InsideSet = InsideSet $\cup$ { $T_i$ }
   If( accuracy(InsideSet) > enough ) break;
}
 /* Center of gravity (COG ) calculation */
Estimated Position = COG ( $\cap T_i \in$ InsideSet);
```

We note that the size of *InsideSet* grows cubically with the number of anchor beacons heard. For example, with 30 audible beacons in a sensor network of 1,500 nodes, the radio region will be divided by 4,060 triangles into small pieces. If the PIT tests render correct inside/outside decisions, each decision will narrow down the area in which a target node can possibly reside, making the final error small. In the next two sections, we describe the perfect PIT test and discuss the infeasibility of performing this test in a WSN. We then introduce a practical approximation to this perfect PIT test, applicable to our work.

3.2 Perfect PIT Test

In this section, we provide a perfect, albeit theoretical, solution to the following problem: For three given anchors: $A(a_x,a_y)$, $B(b_x,b_y)$, $C(c_x,c_y)$, determine whether a point *M* with an unknown position is inside triangle ΔABC or not.

Propositions I: *If M is inside triangle ΔABC, when M is shifted in any direction, the new position must be nearer to (further from) at least one anchor A, B or C.* (Figure 2A)

Proposition II: *If M is outside triangle ΔABC, when M is shifted, there must exist a direction in which the position of M is further from or closer to all three anchors A, B and C.* (Figure 2B).

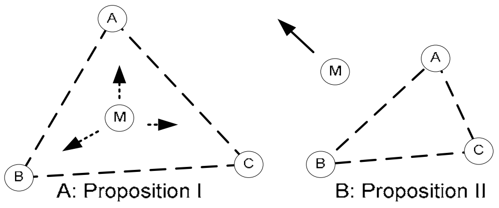


Figure 2: Propositions I and II

Propositions I and II are intuitively correct (the formal proofs are in appendix A). Accordingly, the Perfect PIT test methodology derived from propositions I and II is as follows:

Perfect P.I.T Test Theory: *If there exists a direction such that a point adjacent to M is further/closer to points A, B, and C simultaneously, then M is outside of ΔABC. Otherwise, M is inside ΔABC.*

The Perfect P.I.T test is guaranteed to be correct in deciding whether a point M is inside triangle ΔABC. However, there are two major issues when performing this in a WSN:

- How does a node recognize directions of departure from an anchor without moving?
- How to exhaustively test all possible directions in which node M might depart/approach vertexes A, B, C simultaneously?

We address these issues in the next section.

3.3 Approximation of the Perfect PIT Test

The Perfect P.I.T. test is infeasible in practice; however, we can still obtain a very high level of accuracy by an approximation method introduced in this section.

3.3.1 Departure Test

In previous work [1][15], researchers have assumed a circular, or otherwise well-defined, mathematical or empirical model such as a log-normal attenuation model for radio propagation characteristics that describes the relationship between the signal strength degradation and the distance a radio signal travels.

However, according to a recent empirical study by D. Ganesan at UCLA [10], this assumption does not hold well in practice. In our work, we make a much weaker assumption about radio propagation characteristics. We assume that in a certain propagation direction, defined to be within a narrow angle from the sending anchor (Figure 3), the received signal strength is monotonically decreasing in an environment without obstacles. This simply says that in a given direction, the further away a node is from the anchor, the weaker the received signal strength will be. Through signal strength comparisons between neighboring nodes, this assumption allows a node to determine whether a neighboring node is closer to a given anchor.

Departure Test Definition: *Test whether M is further away from anchor A than N.*

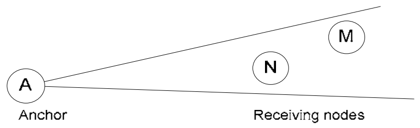


Figure 3: Departure Test

In addition to gathering evidence drawn from prior empirical studies of WSNs [10], we checked the validity of our assumption on Berkeley's MICA mote testbed in an obstruction free laboratory environment. In this experiment, we incrementally increased the distance between sending (anchor) and receiving motes. Figure 4 shows the measured signal strength of 40 beacons from a single anchor at varying distances.

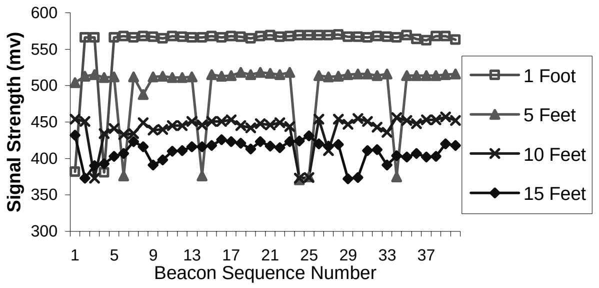


Figure 4: Signal Strength at Different Distances

We conclude from Figure 4 that our assumption of monotonically decreasing signal strength in a given direction is usually valid. For example, the signal strength readings shown in Figure 4 are usually about 560 mv at one-foot, and about 510 mv at five-feet. However, we note that there are various points on the graph where this signal strength property is violated due to burst disturbance effects. Two approaches to minimize the effect of such disturbances include taking a running average of the signal strength over time and using our robust aggregation, a technique discussed further in section 3.4.

It should be noted that our scheme does not make any assumptions about the correlation between *absolute* distance and signal strength; hence, we consider our scheme a range-free solution. More importantly, though we use radio signal comparisons throughout the paper, our scheme can actually work with any system, so long as it can support a form of the departure test.

3.3.2 Approximate PIT Test

To perform PIT testing in sensor networks without requiring that nodes move, we define an Approximate PIT Test (APIT) that takes advantage of the relatively high node density of these networks (usually with connectivity above 6). The basic idea behind the APIT test is to use neighbor information, exchanged via beaconing, to emulate the node movement in the Perfect PIT test. The APIT test is formally described below.

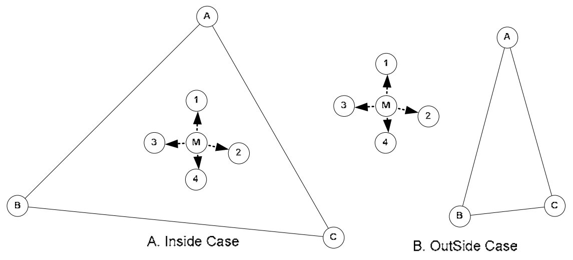


Figure 5: Approximate P.I.T Test

Approximate P.I.T Test: *If no neighbor of M is further from/closer to all three anchors A, B and C simultaneously, M assumes that it is inside triangle ΔABC. Otherwise, M assumes it resides outside this triangle.*

We further explain the APIT test through an example. Figure 5A presents a scenario where none of M's neighbors, 1, 2, 3 or 4, is further from/closer to all three anchors A, B and C simultaneously. In this scenario, M will assume that it is inside the triangle ΔABC according to the definition. The other scenario is shown in Figure 5B, where neighbor 3 will report to node M that it is further away from A, B, and C than M. This allows M to assume it resides outside of triangle ΔABC.

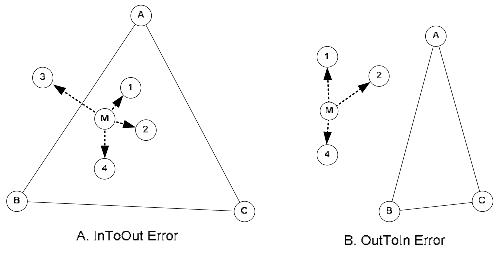


Figure 6: Error Scenarios for the APIT Test.

Because APIT can only evaluate a finite number of directions (the number of neighbors), APIT can make an incorrect decision. The two scenarios where incorrect decisions are made are depicted in Figure 6. In Figure 6A, we show what we deem *InToOut* error, where the node is inside the triangle, but concludes based on the APIT test that it is outside the triangle. This can happen when M is near the edge of the triangle, while some of M's neighbors (3 in this case) are outside the triangle and further from all points ABC, in relation to node M. As a result, M mistakenly thinks it is outside of triangle ABC due to this **edge effect**. On the other hand, the **irregular placement** of neighbors can result in *OutToIn* error. Figure 6B depicts a scenario where M is outside of triangle ABC and none of its neighbors is further from/closer to all three anchors, A, B and C, simultaneously. This makes M mistakenly assume it is inside triangle ABC.

Fortunately, from experimentation, we find that the percentage of APIT tests exhibiting such an error is relatively small (14% in the worst case). Figure 7 demonstrates this error percentage as a function of node density. When node density increases, APIT can evaluate more directions, considerably reducing *OutToInError* (Figure 6B). On the other hand, *InToOutError* will slightly increase due to the increased chance of **edge effects**.

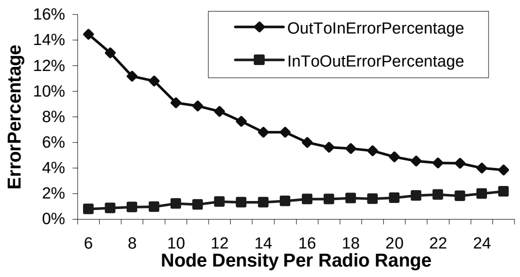


Figure 7: APIT Error under Varying Node Densities

3.4 APIT Aggregation

Once the individual APIT tests finish, APIT aggregates the results (inside/outside decisions among which some may be incorrect) through a grid SCAN algorithm (Figure 8). In this algorithm, a grid array is used to represent the maximum area in which a node will likely reside. In our experiments, the length of a grid side is set to 0.1R, to guarantee that estimation accuracy is not noticeably compromised.

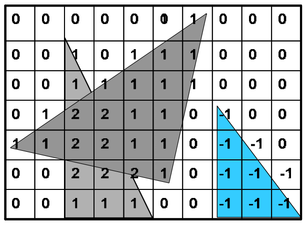


Figure 8: SCAN Approach

For each APIT **inside decision** (a decision where the APIT test determines the node is inside a particular region) the values of the grid regions over which the corresponding triangle resides are incremented. For an **outside decision,** the grid area is similarly decremented. Once all triangular regions are computed, the resulting information is used to find the maximum overlapping area (e.g. the grid area with value 2 in Figure 8), which is then used to calculate the center of gravity for position estimation.

The pseudo code for APIT aggregation is as follows:

```
For (each triangle $T_i \in \binom{N}{3}$ triangles) {
    If (APIT($T_i$) == Out ) AddNegtiveTriangle($T_i$);
    If (APIT($T_i$) == In ) AddPositiveTriangle($T_i$);
};
Find the area with Max values;
```

APIT aggregation is a robust approach that can mask errors in individual APIT tests. As we know from Figure 7, the majority (more than 85% in the worst case) of APIT tests are correct. With limited error, the correct decisions build up on the grid and the small number of errors only serves as a slight disturbance to the final estimation.

3.5 A Walk through the APIT Algorithm

In this section, we present an example to further explain our APIT algorithm.

1. Having received beacons from anchors A, B, and C, each node maintains a table (Anchor ID, Location, Signal Strength) for each anchor heard (Figure 9).

	(X,Y)		SS
A	20	20	1mv
B	45	31	2mv
C	23	56	3mv

Node M

	(X,Y)		SS
A	20	20	2mv
B	45	31	3mv
C	23	56	1mv

Node 1

Figure 9: Table of heard Anchors

2. Each node beacons once to exchange anchor tables with its neighbors. These tables are merged at every node to maintain neighborhood state (Figure 10).

	(X,Y)		MySS	SS1	…..	SSn
A	20	20	1mv	2mv		6mv
B	45	31	2mv	3mv		7mv
C	23	56	3mv	1mv		7mv

Node M

Figure 10: Combined Table

3. APIT runs on every column of the node's table to determine whether a neighboring node exists that has consistently larger/smaller signal strengths from the three anchors A, B and C[2]. If such a neighbor is found, M assumes that it is outside triangle ABC. If no such neighbor is found, M assumes it is inside this region.

4. Each node repeats step 3 for varying combinations of three anchors. (Note: we only demonstrate 1 combination of three anchors in this example).

5. The algorithm described in Section 3.4 is then used to determine the area with maximum overlap.

6. Finally, the center of gravity of this area is used as the final location estimation.

[2] No P.I.T. test is performed when neighboring nodes do not share three common anchor points.

3.6 APIT Performance Analysis

We consider a static senor network with *N* anchors and *M* nodes. Since APIT requires each anchor and node to broadcast once, the communication overhead of our APIT algorithm is *N+M* under collision-free situation. We have proven (see authors for proof) that if a target node can receive beacons from *K* anchors, the maximum number of polygons partitioned by these anchors can be achieved by placing all anchors on a convex curve. This anchor placement creates $(K-1)(K-2)/2 + K(K-1)(K-2)(K-3)/24$ partitions. Assuming the nominal anchor radio range is *R,* the average size of each partition is then:

$$\frac{\pi R^2}{(K-1)(K-2)/2 + K(K-1)(K-2)(K-3)/24}$$

It should be noted that the above formula only indirectly reflects the upper bound performance of the Perfect PIT test. APIT has less accuracy due to approximation as we will show in our evaluations.

By using our SCAN algorithm during APIT aggregation, we bound the computational complexity of the APIT algorithm by $O(L)$ (L is the number of APIT tests and each test only requires several comparisons). If we use a geometric algorithm to perform APIT aggregation precisely, the computational complexity will be $O(L^2)$.

In a mobile sensor network, periodic beaconing is a straightforward solution to maintain the current anchor and node positions. A more sophisticated method to minimize localization cost under such a network is left as future work.

3.7 Key Observations

We note several key observations here to justify the use of our APIT algorithm in sensor networks.

- Redundancy and high node density are the key positive characteristics of sensor networks over traditional ad hoc networks. By exploiting this redundancy, aggregated decisions can provide good accuracy during location estimation, regardless of the fact that information obtained by an individual test is coarse and error prone.
- In order to obtain high redundancy without increasing deployment costs, we can use a single moving anchor that sends out beacons at different locations to localize all nodes inside a sensor network.

4. RANGE-FREE SCHEMES

In this section, we briefly describe the key features of three state-of-the-art range-free localization algorithms studied in our simulation. These algorithms are implemented in accordance with the published design; with the exception of a few enhancements, made to ensure that our comparison is as fair as possible. The protocols discussed include:

- **Centroid Scheme** [3] by N.Bulusu and J. Heidemann
- **DV-Hop Scheme** [26] by D.Niculescu and B. Nath
- **Amorphous Scheme** [22] [23] by R. Nagpal

In addition to the aforementioned range-free algorithms, we implement an oracle version of APIT that uses the Perfect PIT Test defined in Section 3.2. For completeness, we provide brief descriptions of these algorithms. More details can be found in [3], [22], and [26].

4.1 Centroid Localization

N. Bulusu and J. Heidemann [3] proposed a range-free, proximity-based, coarse grained localization algorithm, that uses anchor beacons, containing location information (X_i,Y_i), to estimate node position. After receiving these beacons, a node estimates its location using the following centroid formula:

$$(X_{est}, Y_{est}) = \left(\frac{X_1 + \cdots + X_N}{N}, \frac{Y_1 + \cdots + Y_N}{N} \right)$$

The distinguished advantage of this Centroid localization scheme is its simplicity and ease of implementation. In a later publication [4], N. Bulusu augments his work by suggesting a novel density adaptive algorithm (HEAP) for placing additional anchors to reduce estimation error. Because HEAP requires additional data dissemination and incremental beacon deployment, while other schemes under consideration only use ad hoc deployment, we do not include this later work in our simulations.

4.2 DV-Hop localization

DV-Hop localization is proposed by D. Niculescu and B. Nath in the Navigate project [25]. DV-Hop localization uses a mechanism that is similar to classical distance vector routing. In this work, one anchor broadcasts a beacon to be flooded throughout the network containing the anchors location with a hop-count parameter initialized to one. Each receiving node maintains the minimum counter value per anchor of all beacons it receives and ignores those beacons with higher hop-count values. Beacons are flooded outward with hop-count values incremented at every intermediate hop. Through this mechanism, all nodes in the network (including other anchors) get the shortest distance, in hops, to every anchor. The hop count for a single anchor A, generated by simulation, is shown in Figure 11.

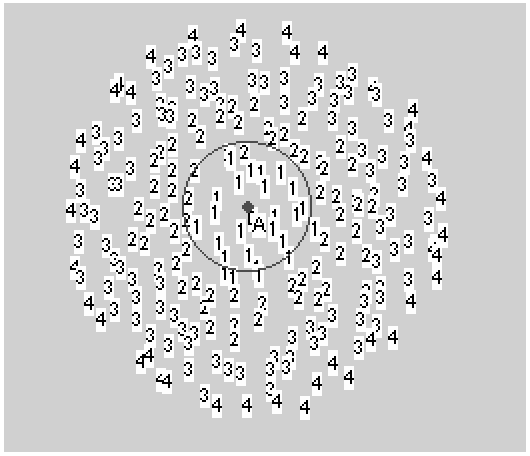


Figure 11: Anchor Beacon Propagation Phase

In order to convert hop count into physical distance, the system estimates the average distance per hop without range-based techniques. Anchors perform this task by obtaining location and hop count information for all other anchors inside the network.

The average single hop distance is then estimated by anchor *i* using the following formula:

$$HopSize_i = \frac{\sum \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum h_j}$$

In this formula, (x_j,y_j) is the location of anchor *j*, and h_j is the distance, in hops, from anchor *j* to anchor *i*. Once calculated, anchors propagate the estimated HopSize information out to the nearby nodes.

Once a node can calculate the distance estimation to more than 3 anchors in the plane, it uses triangulation (multilateration) to estimate its location. Theoretically, if errors exist in the distance estimation, the more anchors a node can hear the more precise localization will be.

4.3 Amorphous localization

The Amorphous Localization algorithm [22], proposed independently from DV-Hop, uses a similar algorithm for estimating position. First, like DV-Hop, each node obtains the hop distance to distributed anchors through beacon propagation.

Once anchor estimates are collected, the hop distance estimation is obtained through local averaging. Each node collects neighboring nodes' hop distance estimates and computes an average of all its neighbors' values. Half of the radio range is then deducted from this average to compensate for error caused by low resolution.

The Amorphous Localization algorithm takes a different approach from the DV-Hop algorithm to estimate the average distance of a single hop. This work assumes that the density of the network, n_{local}, is known *a priori*, so that it can calculate HopSize offline in accordance with the Kleinrock and Slivester formula [18]:

$$HopSize = r\left(1 + e^{-n_{local}} - \int_{-1}^{1} e^{-\frac{n_{local}}{\pi}\left(\arccos t - t\sqrt{1-t^2}\right)} dt\right)$$

Finally, after obtaining the estimated distances to three anchors, triangulation is used to estimate a node's location.

4.3.1 *Amorphous Localization Enhancement*[3]

By using only three anchors, Nagpal suggests in [22] a critical minimum average neighborhood size of 15, imposed to obtain good accuracy. As shown in the APIT algorithm, increasing estimation redundancy reduces estimation error. We, therefore, argue that the same design philosophy can be applied to [22]. By increasing the number of anchors used in their estimation, we can effectively reduce the critical minimum average neighborhood requirement from 15 nodes per communication area, to 6, under uniform node placement (Figure 12) without reducing estimation accuracy (this number would be 8 for random node placement).

This enhancement uses work done by Jan Beutel [2] in the Picoradio Project at UC Berkeley. A minimum mean square error (MMSE) algorithm triangulates node positions based on the locations of multiple anchors (in this case more than 3), and associates distances between each anchor and the target node.

[3] A recent publication [23] in ISPN'03 by Nagpal etc. makes a similar enhancement to the one we propose here.

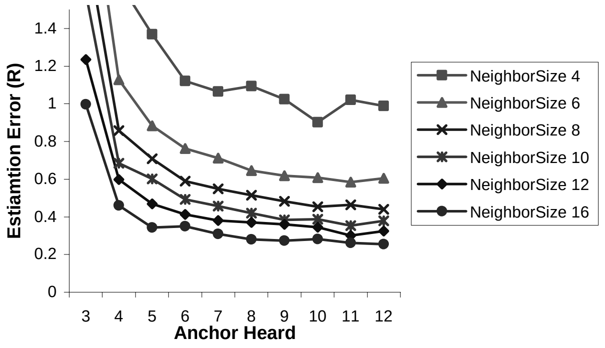


Figure 12: Phase Transition in the DV-Based Algorithm

Using this enhancement, we show that the Amorphous algorithm can actually work in a sparsely connected network. Increasing the number of anchors participating in multilateration can dramatically reduce the required level of network connectivity. In Figure 12, we see that when 3 anchors are used, the estimation error (normalized to units of node radio range R) is large, regardless of the level of connectivity. By increasing the number of anchors to 5, we obtain better precision than that with 3 anchors, when the levels of connectivity as low as 6.

More importantly, Figure 12 shows two kinds of phase transitions that occur. First, when the neighbor size exceeds 8, increasing the number of anchors participating in multilateration brings down the estimation error below half of the radio range, a bound tolerated by the applications we studied in section 7. Second, the estimation accuracy increases dramatically as the number of anchors heard increases to 6. However, after that, continuing to increase the number of anchors heard only slightly increases precision. In accordance with Figure 12, for DV-based algorithms, in order to confine the average estimation error to reside within half of the radio range, we suggest that both the neighborhood size, and the number of anchors used in multilateration, remain about 8~10. We argue that it is not quite cost-effective to further increase node density or the number of anchors used in multilateration for better accuracy after these phase transition points.

4.4 Perfect PIT algorithm

As previously mentioned, the precision of our APIT algorithm is highly dependent on the correctness of the APIT Test. To obtain boundary conditions for a best estimate in our localization scheme, we simulate a perfect PIT algorithm that utilizes an oracle. This oracle can guarantee correctness when determining whether a node resides within the triangular region created by the three anchors. We use this as a precise bound on our APIT algorithm

5. SIMULATION SETTINGS

This section describes the simulation settings we use in our evaluation.

5.1 Radio Model

Some previous work in localization assumes that a perfect circular radio model exists. As stated before, empirical studies [10] on real testbeds have shown that this assumption is invalid for WSNs. To ensure that our evaluation is as true to reality as possible, we use a more general radio model in our evaluation.

Specifically, we assume a model with an upper and lower bound on signal propagation (Figure 13). Beyond the upper bound, all nodes are out of communication range; and within the lower bound, every node is guaranteed to be within communication range. If the distance between a pair of nodes is between these two boundaries, three scenarios are possible: 1) symmetric communication. 2) uni-directional asymmetric communication, and 3) no communication.

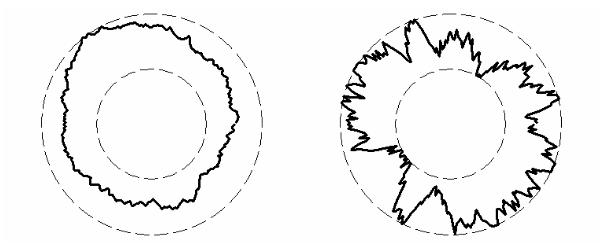


Figure 13: Irregular Radio Pattern

The parameter DOI is used to denote the irregularity of the radio pattern. It is defined as the maximum radio range variation per unit degree change in the direction of radio propagation. When the DOI is set to zero, there is no range variation, resulting in a perfectly circular radio model. To get a better idea of how this DOI parameter affects signal propagation characteristics, Figure 13 shows the radio patterns generated in simulation with DOI values set to 0.05 and 0.2 respectively.

5.2 Placement Model

In our simulations, nodes and anchors are distributed in a rectangular terrain in accordance with predefined densities. Two common placement strategies are investigated, namely random and uniform.

- Random placement: it distributes all nodes and anchors randomly throughout the terrain.
- Uniform placement: the terrain is partitioned into grids and nodes and anchors are evenly divided amongst these grids (random distribution inside each grid).

5.3 System Parameters

In our experiments, we study several system-wide parameters that we feel directly affect estimation error in range-free localization algorithms. A description of these parameters follows:

- Node Density (ND): Average number of nodes per node radio area.
- Anchors Heard (AH): Average number of Anchors heard by a node and used during estimation.
- Anchor to Node Range Ratio (ANR): The average distance an anchor beacon travels divided by the average distance a regular node signal travels. When this value equals one, the anchor and nodes have the same average radio range. The larger this value, the fewer anchors required to maintain a desired AH value.
- Anchor Percentage (AP): The number of anchors divided by the total number of nodes. This value can be derived from the three parameters described above using the formula: $AP=AH/(AH+ND*ANR^2)$.
- Degree of Irregularity (DOI): DOI is defined in section 5.1 as an indicator of radio pattern irregularity.
- GPS Error: In reality, GPS equipped anchors will render imprecise readings. In our evaluation, this parameter is defined as the maximum possible distance from the real anchor position to the GPS estimated anchor position in units of node radio range (R).
- Placement: Random and Uniform node/anchor placements are investigated in the evaluation.

In the evaluation, all distances including error estimation are normalized to units of node radio range (R) to ensure generally applicable results.

5.4 A Note about Comparisons

The range-free localization algorithms studied in this paper share a common set of system parameters, and most of them are defined in a consistent way across the algorithms we analyze. However, due to different anchor beacon propagation methods utilized in different algorithms, the Anchor to Node Range Ratio (ANR) parameter varies between algorithms. In the Centroid and APIT algorithms, direct communication between anchors and **target nodes** (nodes attempting to determine their location) is used. In this case, ANR is set to the physical radio range ratio between anchor and target nodes. In the Amorphous and DV-Hop algorithms studied, the physical radio range of anchors is the same as that of target nodes, and the ANR is set to the distance an anchor beacon can propagate in units of node radio range (R). In our evaluation, we indicate any performance implications that result from this implementation difference.

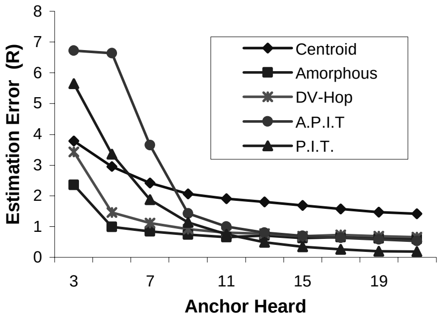
6. EVALUATION

This section provides a detailed quantitative analysis comparing the performance of the range-free localization algorithms described in Sections 3 and 4. The obvious metric for comparison when evaluating localization schemes is location estimation error. We have conducted a variety of experiments to cover a wide range of system configurations including varying 1) anchor density, 2) target node density, 3) radio range ratio (ANR), 4) radio propagation patterns, and 5) GPS error. Because communication can have a significant impact on sensor network systems with low bandwidth, we also use communication overhead, in terms of number of beacons exchanged, as a telling secondary metric to evaluate the cost and performance of the localization schemes studied.

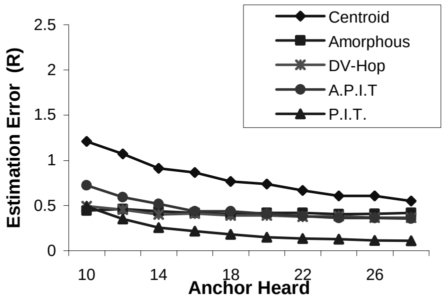
Outside of studying the effect of certain parameters on localization error, we use default values of AH=16, ND=8, and ANR=10 (Anchor Percentage = 2%) in most of our experiments. These settings are in line with our expectation of future sensor network technology and facilitate comparisons between figures. In all of our graphs, each data point represents the average value of 600 trials with different random seeds and the 90% confidence intervals for the data are within 5~10% of the mean shown. We note that for legibility reasons, we do not plot these confidence intervals in this paper. Full experimental data can be obtained from the authors upon request.

6.1 Localization Error when Varying AH

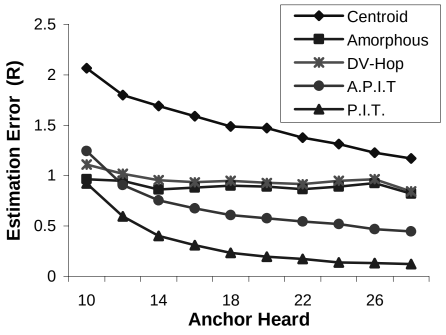
In this experiment, we analyze the effect of varying the number of anchors heard (AH) at a node to determine its effect on localization error.



A. AH=3~21, DOI=0, ANR = 10, ND = 8, Random



B. AH=10~28, DOI=0, ANR = 10, ND = 8, Uniform



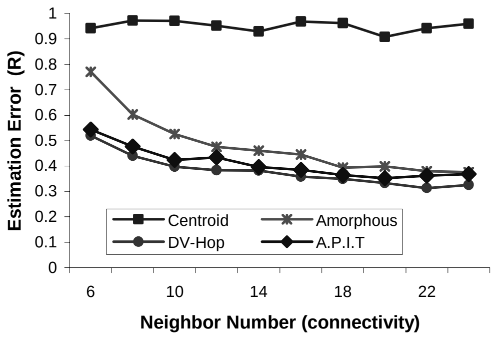
C. AH=10~28, DOI=0, ANR = 10, ND = 8, Random

Figure 14: Error Varying AH

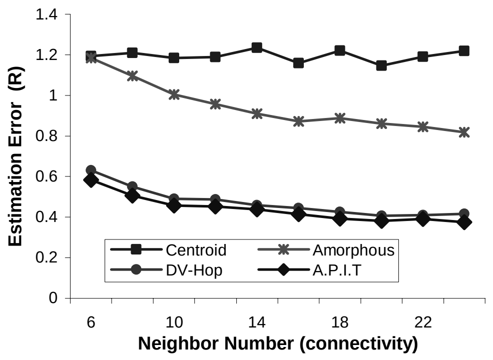
Figure 14A shows that the overall estimation error decreases as the number of anchors heard increases. However, it is important to note that different algorithms transition at different points in the graph. For example, the Amorphous and DV-Hop schemes improve rapidly when AH is below 7, and are nearly insensitive to the addition of anchors above 7. In contrast, the precision of APIT and the Centroid localization scheme constantly improve as AH is increased (Figure 14B and Figure 14C). Our APIT algorithm performs worse than the Centroid algorithm when AH is below 8 due to the fact that the diameter of the divided area is not small enough. This effect is significantly reduced by increasing AH values. For larger AH values, APIT consistently outperforms the Centroid scheme. Figure 14B extends AH to higher values in order to show estimation error below 0.6 R. We note that our APIT algorithm requires only 12 anchors to reach the 0.6R level while the Centroid scheme requires 24. Finally, Figure 14C presents the same experimental results for random node placement. By comparing graphs B (uniform placement) and C (random placement), we show that the DV-Based algorithm is more sensitive to irregular node placement than both APIT and the Centroid scheme. This is mainly due to the fact that *HopSize* estimation in the DV-Hop and Amorphous schemes, is less precise in non-isotropic deployment.

6.2 Localization Error when Varying ND

Figure 15 explores the effect of node density (ND) on the localization estimation accuracy. For all but the Centroid algorithm, localization error decreases as the number of neighbors increases. Since there is no interaction between nodes in the Centroid algorithm, we see nearly constant results while varying ND. However, due to its relatively simple design, the Centroid localization scheme does not perform as well as the others.



A. DOI=0.1, ANR = 10, AH=16, Uniform



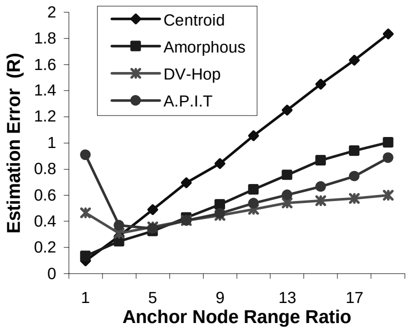
B.DOI=0.2, ANR = 10, AH=16, Uniform

Figure 15: Error Varying ND

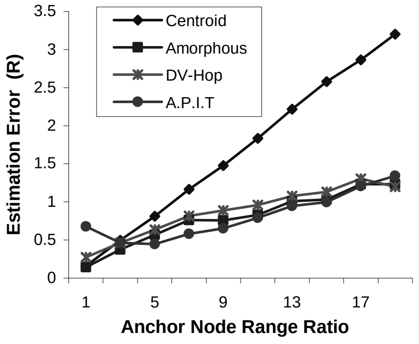
Because the offline estimation of *HopSize* in the Amorphous algorithm has large error when the node density is small, the estimation error is large when the node density is below 10. APIT and DV-Hop however, are robust to varying ND, and produce good results as long as the neighbor density remains above 6. By comparing Figure 15A (DOI=0.1) and Figure 15B (DOI=0.2), we show that the DV-Based algorithms, especially the Amorphous algorithm, are more sensitive to irregular radio patterns than the APIT scheme. This is mainly due to the fact that *HopSize* estimation in the previous schemes is less precise in the presence of irregular radio patterns. However, it should be noted that DV-Hop abates this error by online estimation.

6.3 Localization Error when Varying ANR

Section 6.1 demonstrated that a large number of anchors are desired for good estimation results. The cost of having such a large percentage of anchors can be ameliorated by increasing the anchor radio range to which beacons travel. This happens because larger beacon propagation distances mean less anchors required to achieve the same AH value. For example, if an algorithm requires AH equal to the neighborhood node density (ND), we need 50% of the nodes to be anchors when the ANR equals one. By increasing the ANR by a factor of 10, we can reduce the required anchor percentage to only 1%.



A. ND = 8, AD=16, DOI = 0.1, Uniform



B. ND = 8, AD=16, DOI = 0.1, Random

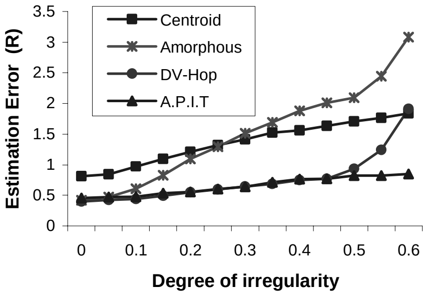
Figure 16: Error under Different ANR

The implication of this solution, as shown in Figure 16, is that estimation error increases as ANR increases. This occurs because larger beacon propagation distances result in larger accumulated error. We note from Figure 16 that while all algorithms possess this relationship, the estimation error of the Centroid algorithm increases more significantly with increased ANR, in comparison to the other three algorithms. However, we also note that when the ANR is smaller than 3, APIT has a large InToOutErrorRatio due to the edge effect (described in Section 3.3.2). In this system configuration, a Centroid algorithm has its advantages.

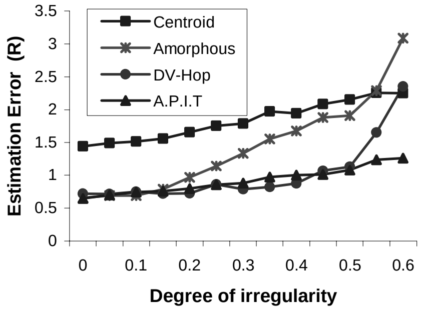
From an alternate perspective, we show that we can increase accuracy by using a smaller ANR. For example, the estimation error, shown in previous sections, can be reduced by about 30~50% when we use an ANR value of 5 instead of 10. However, this will increase the anchor percentage (AP) from 2% to 8%, requiring that more anchors be deployed.

6.4 Localization Error when Varying DOI

In this experiment, we investigate the impact of irregular radio patterns on the precision of localization estimation. It is intuitive that irregular radio patterns can affect the network topologies resulting in irregular hop count distributions in the Amorphous and DV-Hop algorithms. The HopSize formula, used in the Amorphous algorithm, assumes that radio patterns are perfectly circular. We can see, in Figure 17, how this inaccurate estimate directly contributes to localization error as the DOI increases. In contrast, the DV-Hop scheme estimates HopSize using online information exchanged between anchors. This results in much better performance than the Amorphous algorithm, even though they are both DV-Based algorithms. Because the Centroid and APIT algorithms do not depend on hop-count and HopSize estimations, and because the effect of DOI is abated by the aggregation of beaconed information, these algorithms are more robust than the Amorphous algorithm.



A. ANR = 10, ND = 8, AH=16, Uniform

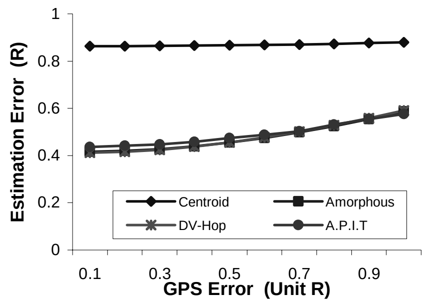


B. ANR = 10, ND = 8, AH=16, Random

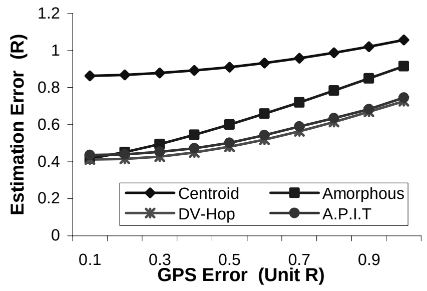
Figure 17: Error under Varying DOI

6.5 Localization Error when Varying GPS Error

In other experiments, we consider the distinct possibility that the GPS or an alternative system, which provides anchor nodes with location information, is error prone. Figure 18A and B demonstrate how initial location error at anchors directly affects the error of the range-free localization protocols studied. In general, in all four schemes GPS error is abated considerably by utilizing location information from multiple anchors. In the random error case (Figure 18A), we assume GPS error is isotropic; that is, the estimation error can occur in any direction. In this situation, the error impact of GPS is very small. We also see (Figure 18B) that when GPS error is biased (skewed in a particular direction) due to non-random factors, the estimation error of all schemes increases at a much slower rate than GPS error due to aggregation.



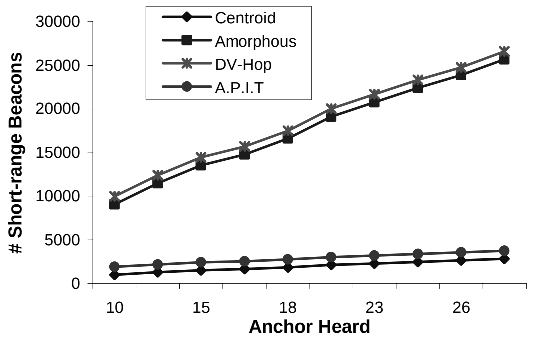
A.ANR = 10, ND = 8, AH=16, Uniform, Random Error



B. ANR = 10, ND = 8, AH=16, Uniform, Bias Error

Figure 18: Error under Different GPS Error

6.6 Communication Overhead for Varied AH



ANR=10, ND = 8, DOI = 0.1, Uniform

Figure 19: Communication Overhead for Varied AH

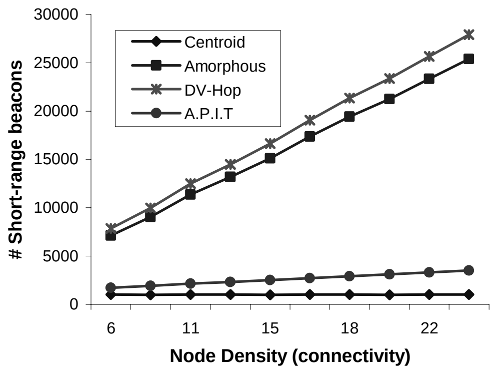
Figure 19 shows the results of experiments that test the communication overhead with regard to AH. It is important to note that the Centroid and APIT schemes use *long-range* anchor beacons, while the Amorphous and DV-hop algorithms use *short-range* beacons. Considering that energy consumption quadratically increases with increased beacon range, in Figure 19 we equate one long-range beacon to ANR^2 short-range beacons. This means that one long-range beacon sent out by APIT is counted as 100 short-range beacons when ANR = 10. Figure 19 shows that without flood-based beacon propagation, the Centroid and APIT algorithms use much fewer beacons than DV-based algorithms. For example, the APIT algorithm uses only about 10% of the beacons that the DV-Hop scheme uses when AH is set to 16.

Figure 19 also shows that APIT requires more beacons than the Centroid algorithm because of the neighborhood information exchange. In addition, DV-Hop requires more beacons than the Amorphous algorithm because of additional online HopSize estimation requirements.

It should be noted that the evaluation of communication overhead here assumes a collision-free environment. If taking the collision into account, we expect that Amorphous and DV-hop algorithms introduce even more control overhead because of the flooding required by those two schemes.

6.7 Communication Overhead for Varied ND

Figure 20 demonstrates the effect of neighborhood density on required communication for localization. We can see from this graph that because there is no interaction between nodes in the Centroid scheme, the overhead stays constant. Communication overhead in our APIT scheme does increase with increased node density; however, it does so at a much lower rate than the DV-based schemes.



ANR=10, AH = 16, DOI = 0.1, Uniform

Figure 20: Overhead for Varied Node Density

Drawing conclusions from Figure 19 and Figure 20, we argue that as far as the communication overhead is concerned, the DV-Hop and Amorphous schemes are less suitable solutions for sensor networks with limited bandwidth when compared to the APIT and Centroid schemes. This is due to the large number of beacons required in these schemes.

6.8 Evaluation Summary

In addition to the experiments previously discussed, we have conducted a variety of experiments to cover a varying range of system configurations. These experiments help us better understand the situations where the different localization schemes considered are more or less appropriate than one another.

Table 1 provides an overview of our results, and it can be used as a design guide for applying range-free schemes in WSN systems. This table shows that no single algorithm works best under all scenarios, and that each localization algorithm has preferable system configurations. Though the Centroid scheme has the largest estimation error, its performance remains independent of node density and it boasts the smallest communication overhead and simplicity of implementation. Although DV-Hop requires more communication beacons to perform online estimation, it is notably more robust than the Amorphous algorithm in HopSize estimation. Finally, our APIT algorithm trumps the other algorithms when an irregular radio pattern and random node placement are considered, and low communication overhead is desired. However, we acknowledge that APIT has more demanding requirements for both ANR values and the number of anchors used in localization.

	Centroid	DVHop	Amorp.	APIT
Accuracy	Fair	Good	Good	Good
NodeDensity	>0	>8	>8	>6
AnchorHeard	>10	>8	>8	>10
ANR	>0	>0	>0	>3
DOI	Good	Good	Fair	Good
GPSError	Good	Good	Fair	Good
Overhead	Smallest	Largest	Large	Small

Table 1 Performance and requirements summary

7. LOCALIZATION ERROR IMPACT

In localization for WSNs, achieving better results (usually with regard to location accuracy) requires increasing the relative cost of the localization scheme via additional hardware, communication overhead, or the imposition of constraints and system requirements. Although more accurate location information is preferable, the desired level of granularity should depend on a cost/benefit analysis of the protocols that utilize this information. In this section, we investigate the impact of localization error on other communication protocols and proposed sensor network applications. Designers of sensor network systems with certain performance requirements can use this analysis to aid in their architectural design and in setting system parameters. Although requirements are expected to vary between deployments, we found that in the general case for the protocols studied, performance degradation is moderate and tolerable when the average localization error is less than 0.4R.

7.1 Routing Performance

A localization service is critical for location-based routing protocols such as GF [24], GPSR [17], LAR [19] and GAF [35]. In these protocols, individual nodes make routing decisions based on knowledge of their geographic location. While most work in location-based routing assumes perfect location information, the fact is that erroneous location estimates are virtually impossible to avoid. Problems arise as error in the location service can influence location-based routing to choose the best next hop (the neighbor closest to the destination), or can make a node inadvertently think that the packet could not be routed because no neighbors are closer to the final destination.

To investigate the impact of localization error on routing, we studied the GF [24] routing protocol under the low traffic network conditions so that network congestion does not influence our results. Our baseline is "perfect localization", the protocol where every sensor node knows its correct physical location.

Figure 21 shows the delivery ratio (the percentages of packets that reach destination over all packets sent) with regard to node density for various levels of location error. From this graph, we can see that for average localization errors of 0.2 and 0.4 times the node radio range, the delivery ratios of GF are very close to the baseline (no error). Beyond these numbers, the results diminish with increased error; a trend that could be problematic and costly depending on the implemented architecture, reliability semantics, tolerance of message loss, and application requirements. For example, when localization error is the same as the node radio range, even with high node density (20 nodes per radio range), the delivery ratio still falls below 60%.

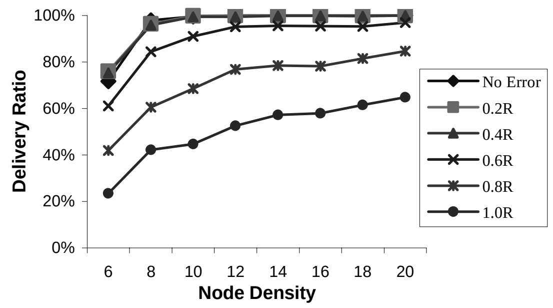


Figure 21: Delivery ratio with different localization errors, changing node density

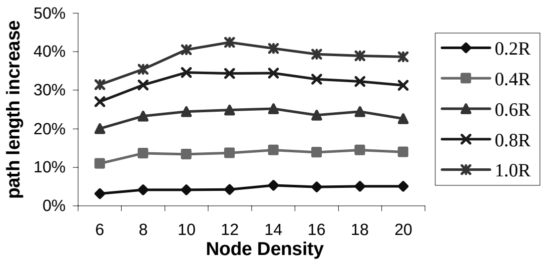


Figure 22: Path length overhead with different localization errors under varying node density

Another metric affected by localization error is the route path length. In Figure 22 we measure the hop count increase (in percentage) due to location error to assess the cost in communication overhead of this error. We see from this graph that for low localization error (less than 0.4R), this routing overhead remains moderate (<15%). However, as was the case for the delivery ratio metric, when localization error grows beyond 0.4R, the routing overhead increases to as high as 45%. We also note that this trend occurs regardless of the network node density, a fact that was not true for our previous metric.

We acknowledge that here we only chose GF as the representative protocol, and an in depth study about localization's impact on various routing protocols and its implications on design of location-dependent system is left as future work.

7.2 Target Estimation Performance

Many of the most frequently proposed applications for WSNs utilize target position estimations for tracking, search and rescue, or other means. In these proposed applications, when a target is identified, some combination of the nodes that sensed that target report their location to a centralized node (leader or base station). This node then performs aggregation on the received data to estimate the actual location of the target. Because target information could be used for locating survivors during a disaster, or identifying an enemy's position for strategic planning, the accuracy of this estimation is crucial to the application that uses it.

Intuitively an increase in localization error will directly lead to target estimation error. To better understand the degree to which this error will propagate to other protocols, we investigate average estimation error under different node densities for varying degrees of location error. For these experiments, we use a simple and widely used target estimation algorithm: the average *x* and *y* coordinates of all reporting nodes[4] are taken as the target location estimation. We set the sensing range equal to the node radio range so that the node density is equivalent to the average number of sensors involved in target estimation. The results of various experiments are depicted in Figure 23. This graph shows that target estimation error due to location error is dampened during the aggregation process. As before, our baseline occurs when no localization error exists. Aside from showing varying degrees of estimation error with respect to node location error, Figure 23 also shows that the absolute target estimation error decreases with increased node density. For example, with localization error is equal to 1.0R, and node density reaches 12 nodes per radio range, the estimation error is only about 67% as large as when the node density is 6. From this chart we see that more nodes participating in estimation results in more random estimation error being ameliorated through aggregation.

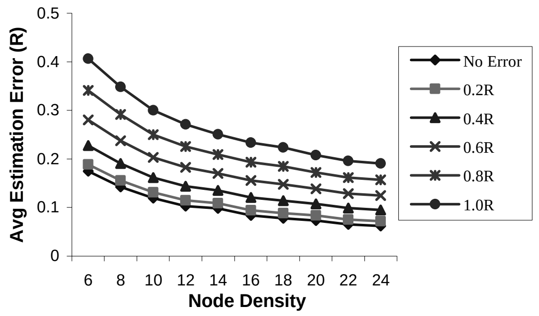


Figure 23: Target estimation error with different localization errors under varying node density

7.3 Object Tracking Performance

We further evaluate the performance of target estimation by simulating a tracking application that uses estimation in context. In this experiment, a mobile evader randomly walks around the specified terrain while a pursuer attempts to catch it. In this simple experiment, the pursuer is informed of the current location of the evader periodically via sensing nodes in the terrain that detect the evader, coordinate to estimate the targets position with regard to their own positions, and periodically report this result to the mobile pursuer. When receiving a report, the pursuer readjusts its direction in an attempt to intercept the evader. When the pursuer comes within the node communication radius of the evader, the evader is considered caught and the simulation ends. For this experiment, we compare the average tracking time (the time from pursuer take-off to when the evader is caught) under different localization errors, to the tracking time in the case of no localization error. Figure 24 shows normalized tracking time in relation to the pursuer speed for various degrees of localization error.

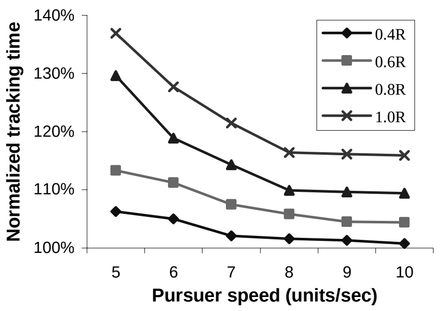


Figure 24: Normalized tracking time with different localization errors varying pursuer speed. Terrain size 1000x1000 units, Radio range = 40units, density = 8 nodes per radio circle. Evader speed = 5 units /second

From Figure 24 we can see that the tracking time overhead decreases with increased pursuer speeds. More importantly, Figure 24 shows that tracking time increases as localization error increases. This result implies that it is important for tracking applications with real-time requirements to take localization error into consideration. For example, when the average localization error is known to be 0.8R, and the Pursuer speed is 5 units per second, the Pursuer requires 30% more time in comparison to the ideal situation in which no localization error exists. To reduce this overhead to 10%, either the pursuer's speed must be increased to 10 units per second, or we must reduce the estimation error to 0.4R. Again, Figure 24 shows that 0.4R is a tolerable bound for estimation error since tracking time only increases by 7% in the worst case.

8. CONCLUSION

Given the inherent constrains of the sensor devices envisioned and the estimation accuracy desired by location-dependent applications, range-free localization schemes are regarded as a cost-effective and sufficient solution for localization in sensor networks. From our extensive comparison study, we identify preferable system configurations of four different recently proposed range-free localization schemes as a design guideline for further research. In particular, an APIT scheme, proposed in this paper, performs best when irregular radio patterns and random node placement are considered, and low communication overhead is desired. Moreover, we provide insight on how localization error affects a variety of location-dependent applications. These results show that the accuracy provided by the range-free schemes considered is sufficient to support various applications in sensor networks with only slight performance degradation.

[4] Nodes report when they sense the event of interest in the environment.

REFERENCES

[1] P. Bahl and V. N. Padmanabhan, RADAR: An In-Building RF-Based User Location and Tracking System, In *Proceedings of the IEEE INFOCOM '00*, March 2000.

[2] J. Beutel, Geolocation in a PicoRadio Environment, M.S. Thesis, ETH Zurich, *Electronics Laboratory*, Dec. 1999.

[3] N. Bulusu, J. Heidemann and D. Estrin, GPS-less Low Cost Outdoor Localization for Very Small Devices, *IEEE Personal Communications Magazine*, 7(5):28-34, October 2000.

[4] N. Bulusu, J. Heidemann and D. Estrin, Density Adaptive Algorithms for Beacon Placement in Wireless Sensor Networks, In *IEEE ICDCS '01*, Phoenix, AZ, April 2001.

[5] N. Bulusu, J. Heidemann, D. Estrin and T. Tran, Self-configuring Localization Systems: Design and Experimental Evaluation , *ACM Transactions on Embedded Computing Systems* (TECS), Special Issue on Networked Embedded Computing, 2003.

[6] J. Caffery, Jr. A New Approach to the Geometry of TOA Location, In *IEEE Vehicular Technology Conference (VTC)*, Boston, Mass, September 2000.

[7] S. Capkun, M. Hamdi and J.P. Hubaux, GPS-Free Positioning in Mobile Ad-Hoc Networks, In *Proceedings of HICCSS '01*, Maui, Hawaii, January 2001.

[8] L. Doherty, L. E. Ghaoui and K. S. J. Pister, Convex Position Estimation in Wireless Sensor Networks, In *Proceedings of the IEEE INFOCOM '01*, Anchorage, AK, April 2001.

[9] D. Estrin, R. Govindan, J. Heidemann and S. Kumar, Next Century Challenges: Scalable Coordination in Sensor Networks,In *Proceedings of MOBICOM '99*, Seattle, Washington, 1999.

[10] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin and S. Wicker, Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks, *Technical Report UCLA/CSD-TR 02-0013*, 2002.

[11] L. Girod and D. Estrin, Robust Range Estimation using Acoustic and Multimodal Sensing, In *Proceedings of IROS '01*, Maui, Hawaii, October 2001.

[12] A. Harter, A. Hopper and P. Steggles, A. Ward and P. Webster, The anatomy of a context-aware application, In *Proceedings of MOBICOM '99*, Seattle, Washington, 1999.

[13] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher, SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks, In *Proceedings of IEEE ICDCS '03*, Providence, RI, May 2003.

[14] J. Hightower and G. Boriello, Location Systems for Ubiquitous Computing, *IEEE Computer*, 34(8):57-66, August 2001.

[15] J. Hightower, G. Boriello and R. Want, SpotON: An indoor 3D Location Sensing Technology Based on RF Signal Strength, *University of Washington CSE Report #2000-02-02*, February 2000.

[16] X. Hong, K. Xu, and M. Gerla, Scalable routing protocols for mobile ad hoc networks, *IEEE Network magazine*, vol 16, No. 4, 2002.

[17] B. Karp and H. T. Kung, GPSR: Greedy Perimeter Stateless Routing for Wireless Networks, In *Proceedings of MOBICOM '00*, New York, August 2000.

[18] L.Kleinrock and J.Slivester, Optimum transmission radii for packet radio networks or why six is a magic number, In *proceedings of national Telecomm conf*erence, Pages 4.3.1-4.3.5, 1978

[19] Y. B. Ko and N. H. Vaidya, Location-Aided Routing (LAR) in Mobile Ad Hoc Networks, In *Proceedings of MOBICOM '98*, Dallas, TX, 1998.

[20] J. Li, J. Jannotti, D. S. J. De Couto, D. Karger and R. Morris, A Scalable Location Service for Geographic Ad-Hoc Routing, In *Proceedings of MOBICOM '00*, New York, August 2000.

[21] MICA Sensor Board Information, http://www.xbow.com

[22] R. Nagpal, Organizing a Global Coordinate System from Local Information on an Amorphous Computer, *A.I. Memo 1666*, MIT A.I. Laboratory, August 1999.

[23] R. Nagpal, H. Shrobe, J. Bachrach, Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network, In *the 2nd International Workshop on Information Processing in Sensor Networks* (IPSN '03), Palo Alto, April, 2003.

[24] J. C. Navas and T. Imielinski, Geographic Addressing and Routing, In *Proceedings of MOBICOM '97*, Budapest, Hungary, September 26, 1997.

[25] D. Nicolescu and B. Nath, Ad-Hoc Positioning Systems (APS), In *Proceedings of IEEE GLOBECOM '01*, November 2001.

[26] D. Niculescu and B. Nath, DV Based Positioning in Ad hoc Networks, In *Journal of Telecommunication* Systems, 2003.

[27] D. Niculescu and B. Nath, Ad Hoc Positioning System (APS) using AoA, *INFOCOM' 03*, San Francisco, CA,2003

[28] N. B. Priyantha, A. Chakraborty and H. Balakrishnan, The Cricket Location-Support System, In *Proceedings of MOBICOM '00*, New York, August 2000.

[29] C. Savarese, J. Rabay and K. Langendoen, Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks, *USENIX Technical Annual Conference*, Monterey, CA, June 2002.

[30] A. Savvides, C. C. Han and M. B. Srivastava, Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors, In *Proceedings of MOBICOM '01*, 2001, Rome, Italy, July 2001.

[31] A. Savvides, H. Park and M. Srivastava, The Bits and Flops of the N-Hop Multilateration Primitive for Node Localization Problems, In *First ACM International Workshop on Wireless Sensor Networks and Application*, Atlanta, GA, September 2002.

[32] R. Want, A. Hopper, V. Falcao and J. Gibbons, The Active Badge Location System, *ACM Transactions on Information Systems*, January 1992.

[33] B. H. Wellenhoff, H. Lichtenegger and J. Collins, *Global Positions System: Theory and Practice*, Fourth Edition. Springer Verlag, 1997.

[34] K. Whitehouse and D. Culler, Calibration as Parameter Estimation in Sensor Networks, In *First ACM International Workshop on Wireless Sensor Networks and Application*, Atlanta GA, September 2002.

[35] Y. Xu, J. Heidemann and D. Estrin, Geography-informed Energy Conservation for Ad Hoc Routing , In *Proceedings of MOBICOM '01*, Rome, Italy, July 2001.

[36] T. Yan, T. He, J. A. Stankovic, Differentiated Surveillance Service for Sensor Networks, In *Proceeding of First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, Los Angeles, CA 2003.

APPENDIX A:

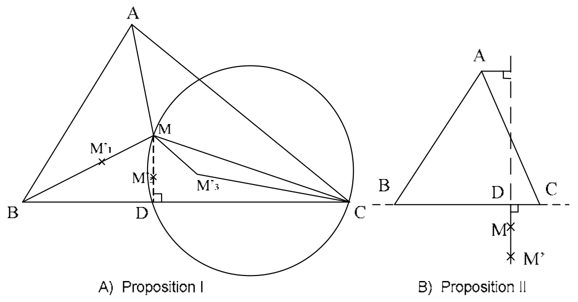


Figure 25: Proofs for Propositions

Proof of Proposition I:

We first prove that M' is closer to at least one vertex than M. As shown in Figure 25A, let M move a short distance to another point M' which is also within the ΔABC. Consider the different cases:

Case 1: M' is on one of the three line segments AM, BM or CM. Without loss of generality (WLOG), we can say that M' is on BM, call it M'1. Clearly, BM'1 < BM, i.e., the point moves towards B – hence proved.

Case 2: M' is in one of ΔABM , ΔBCM or ΔACM . WLOG let M' be in ΔBCM . From M, draw a line perpendicular to BC that meets BC at D.

Case 2.1: M' is on MD. Let us call this point M'2. Since DM'$_2$ < DM, by Pythagorean Theorem, BM'2 < BM, i.e., the point moves towards B – hence proved.

Case 2.2: M' is in ΔBDM or ΔCDM . Again, WLOG, let M' be in ΔCDM . Call it M'3. Now, draw the circum-circle of ΔCDM (A circle that passes through three vertices C,D and M). Note that CM is the diameter of this circle and M'3 is an interior point. Obviously, CM'3 < CM, i.e., the point moves towards C – hence proved

Second, by drawing three line segments AM', BM' and CM', we prove symmetrically that M is closer to at least one vertex than M', hence M' is further from at least one vertex than M.

Proof of Proposition II:

As shown in Figure 25B, we prove this proposition by construction. For any point M exterior to ΔABC, there is always an edge connecting two vertices of the triangle such that the third vertex lies on one side of the edge while M is on the other. WLOG, we can assume that BC to be such an edge. From M draw a perpendicular line to BC meeting it at D. Choose M' to be a point on line DM below M. By Pythagorean Theorem, AM<AM', BM<BM' and CM<CM', hence proved.

Demo Abstract: Electronic Tripwires for Power-Efficient Surveillance and Target Classification

Tian He, Qiuhua Cao, Liqian Luo, Ting Yan, Lin Gu, John Stankovic, Tarek Abdelzaher
Department of Computer Science, University of Virginia
{tianhe,qc9b,ll4p,ty4k,lg6e,,stankovic,zaher}@cs.virginia.edu

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design

General Terms

Design, Performance, Experimentation, Measurement

Keywords

Sensor networks, Energy conservation, Tracking, Classification

1. DEMO DESCRIPTION[1]

This demonstration presents a mini-scale surveillance system developed at the University of Virginia that is currently undergoing a technology transition. The ultimate goal of this project is to develop a clandestinely deployable, environmentally rugged, robust, self-organizing, wireless sensor network for long-term, persistent surveillance, involving detection, tracking, classification, and identification of vehicle and personnel targets over various types of terrain.

One key research challenge for this type of system is to reconcile the need for network longevity (which requires most sensors to remain inactive) with the need for fast and accurate target detection and classification (which requires as many sensors to be awake as possible). In order to reconcile the two conflicting design goals, we developed a suite of services that includes (i) a tripwire-based power management with sentry service, (ii) a radio-based wakeup service, (iii) an entity-based tracking service, and (iv) a three-tier target classification scheme. These services work in conjunction with a localization and time synchronization protocol needed to support their operation.

The conceptual operation of the system is as follows: 1) The sentry service allows most motes to go to sleep, while a small fraction of them (the sentries) remain awake to watch for potential targets. It is ensured that sentries form a communication backbone to deliver important information quickly to long-distance relays in the network when needed. Sentries rotate periodically to balance power consumption.

[1] This work was supported in part by NSF grant CCR-0098269, the MURI award N00014-01-1-0576 from ONR, and the DAPRPA IXO office under the NEST project (grant number F336615-01-C-1905).

2) When a target enters the field, the wakeup service allows a portion of the network near the target to be quickly awakened. Once awake, motes perform target tracking and classification. With the help of localization and time synchronization, the tracking component computes target location and velocity. Classification is done through a three-tier architecture. In essence, at the first tier, sensors differentiate real targets from false alarms and provide different sensing modalities. At the second tier, sensor groups perform data fusion among adjacent nodes that detect the target concurrently. At the third tier, the base node takes current group-level reports as well as history into account to make final classification decisions. This information is sent to a command center through the nearest relay (a device capable of long-distance communication).

3) To scale the system and allow further power savings, a tripwire service can be activated which partitions the field into regions (called tripwires). Individual regions can be de-activated in which case they go to sleep (including their sentries). Active tripwire regions operate as described above, using sentries to look for targets and waking up the remaining nodes only when targets are detected. A target detected by an active tripwire is tracked and classified. Tracking continues across tripwire boundaries waking up nodes on the target's trajectory.

The complete system is designed to scale to at least 1000 XSM motes and cover at least 100,1000 square meters. In this demonstration, a field of 20 - 40 XSM motes is used to form a mini-size sensor network. Two laptops are used as relays. The network is configured into two tripwire regions, each reporting to a different laptop. Active tripwires are shown to perform sentry selection. Sentry nodes identify themselves by showing different colors on the laptop screen (perhaps using a projection device). The remaining nodes go to sleep. Tripwire configuration commands can also be sent if needed to turn off a tripwire for further energy savings. One or a set of (small) targets then enters the field. As the target crosses an active tripwire, the wake-up service is automatically invoked and non-sentry nodes are awakened in the vicinity of the target. These nodes flash their LEDs to indicate active tracking state. Information about the location, speed, and type of the target is promptly sent to the relay(s) and displayed on the screen. The experiment may be repeated with different targets, different configurations of tripwire regions, and different target paths through the network. When the target departs the network and no further activity is detected, nodes go back to sleep leaving only sentries (in active tripwires) to keep watch. After a period of continued inactivity, sentries are shown to rotate to balance power. Re-introduction of targets at any time repeats the wakeup and tracking scenario described above.

Differentiated Surveillance for Sensor Networks

Ting Yan Tian He John A. Stankovic

Department of Computer Science,

School of Engineering, University of Virginia

151 Engineer's Way, P.O.Box 400740, Charlottesville, Virginia 22904-4740, USA

{ty4k, tianhe, stankovic}@cs.virginia.edu

ABSTRACT

For many sensor network applications such as military surveillance, it is necessary to provide full sensing coverage to a security-sensitive area while at the same time minimizing energy consumption and extending system lifetime by leveraging the redundant deployment of sensor nodes. It is also preferable for the sensor network to provide differentiated surveillance service for various target areas with different degrees of security requirements. In this paper, we propose a differentiated surveillance service for sensor networks based on an adaptable energy-efficient sensing coverage protocol. In the protocol, each node is able to dynamically decide a schedule for itself to guarantee a certain degree of coverage (DOC) with average energy consumption inversely proportional to the node density. Several optimizations and extensions are proposed to provide even better performance. Simulation shows that our protocol accomplishes differentiated surveillance with low energy consumption. It outperforms other state-of-the-art schemes by as much as 50% reduction in energy consumption and as much as 130% increase in the half-life of the network.

Categories and Subject Descriptors

C.2. [**Computer Communication Networks**]: Network Protocols

General Terms

Algorithms, Performance, Design

Keywords

Sensor Networks, Sensing Coverage, Energy Conservation, Differentiated Service

1. INTRODUCTION

Wireless Sensor Networks have emerged as a new information-gathering paradigm based on the collaborative effort of a large number of sensing nodes. In such networks, nodes deployed in a remote environment must self-configure without any *a priori* information about the network topology or global view. Nodes act in response to environmental events and relay collected and possibly aggregated information through the dynamically formed multi-hop wireless network in accordance with desired system functionality. These networks can form the basis for many types of smart environments such as smart hospitals, battlefields, earthquake response systems, and learning environments. A set of applications, such as biomedicine, hazardous environment exploration, environmental monitoring, military tracking and reconnaissance surveillance are the key motivations for many recent research efforts in this area.

Low-cost deployment is one acclaimed advantage of sensor networks, which imply that the resources available to individual nodes are severely limited. Limited processor bandwidth and small memory are two arguable constraints in sensor networks, which will disappear with the development of fabrication techniques. However, the energy constraint is unlikely to be relieved quickly due to slow progress in developing battery capacity. Moreover, the untended nature of sensor nodes and hazardous sensing environments preclude battery replacement as a feasible solution. On the other hand, the surveillance nature of sensor network applications requires a long lifetime; therefore, it is a very important research issue to provide a form of energy-efficient surveillance service for a geographic area.

Previous research focuses on how to provide full or partial sensing coverage in the context of energy conservation. In such an approach, nodes are put into a dormant state as long as their neighbors can provide sensing coverage for them. These solutions regard the sensing coverage to a certain geographic area as binary, either it provides coverage or not. However, we argue that, in most scenarios such as battlefields, there are certain geographic sections such as the general command center that are much more security-sensitive than others. Based on the fact that individual sensor nodes are not reliable and subject to failure and single sensing readings can be easily distorted by background noise and cause false alarms, it is simply not sufficient to rely on a single sensor to safeguard a critical area. In this case, it is desired to provide higher degree of coverage in which multiple sensors monitor the same location at the same time in order to obtain high confidence in detection. On the other hand, it is overkill and energy consuming to support the same high degree of coverage for some non-critical area. Based on such observations, this paper leverages previous solutions and addresses the problem of providing a differentiated surveillance service for sensor networks. By differentiated surveillance, we mean providing different degrees of sensing coverage for a sensor network according to different requirements. Two major goals are achieved in this paper. First, our new sensing coverage algorithm outperforms other state-of-the-art solutions in terms of energy conservation, energy balance and communication overhead when it runs in non-differentiated mode. Second, this paper is the first to propose differentiated surveillance for sensor networks with minimal overhead.

The remainder of the paper is organized as follows: Section 2 discusses previous research related to the sensing coverage problem found in sensor networks. Section 3 describes the design of our sensing coverage protocol without differentiation. Section 4 extends the design to provide differentiated surveillance. Section 5 gives a brief description of baselines to which we compare our work, including two other state-of-the-art sensing coverage protocols. Section 6 provides a detailed performance evaluation and comparison. We conclude the paper in Section 7.

2. RELATED WORK

A set of research issues needs to be addressed before surveillance-based applications such as military tracking and environmental monitoring become technically feasible and economically practical. Recently, several schemes are proposed to address the sensing coverage problem in sensor networks. In [18], full surveillance coverage is support by a node-scheduling scheme based on off-duty eligibility rules, which allows nodes to turn themselves off as long as the neighboring nodes can cover the area for them. This rule guarantees 100% sensing coverage as long as no void exists. However, this rule underestimates the area that the neighbor nodes can cover, which leads to excess energy consumption. In [22], surveillance coverage is achieved by a probing mechanism. In this solution, after a sleeping node wakes up, it broadcasts a probing message within a certain range and waits for a reply. If no rely is received within a timeout, it will take the responsibility of surveillance until it depletes its energy. In this solution, the probing range and wakeup rate can be adjusted to affect the degree of coverage indirectly. However, this probing-based approach has no guarantee on sensing coverage and blind points can occur. Our solution is different from these solutions in the sense that it can not only *guarantee* sensing coverage to a certain geographic area, but it can also adapt the degree of coverage to that area, up to the limitation imposed by the number of sensor nodes present.

Energy balancing is another research issue addressed in this paper. [18] and [22] only consider the metric in terms of the total amount of energy consumed regardless of the distribution of the energy among the nodes. We argue that unbalanced energy dissipation causes some nodes to die much faster than others do, therefore, the half-life of the network is dramatically reduced in the un-balanced approach. Research has addressed the energy balance issue from different aspects of sensor networks. SPEED [10] balances the traffic by non-deterministically forwarding the packet through multiple routes. GAF [20] performs leader rotation among the nodes inside a virtual grid, in order to balance energy consumption.

Research on network topology control such as SPAN [4], LEACH [10] and GAF [20] addresses the problem of providing communication coverage within an energy conservation context. These have a similar flavor as the surveillance coverage problem. For example, LEACH [10] partitions a network into clusters and randomly rotates the cluster leader in order to evenly distribute the energy consumption among the sensors. SPAN [4] is another randomized algorithm where nodes make local decisions on whether to sleep or to join a backbone network in order to reduce energy consumption. The major difference between those and this work is that communication coverage considers only connectivity between the nodes. In contrast, surveillance (sensing) coverage addresses the coverage problem to every physical point in the terrain. As a result, new scheduling is required to force some nodes stay awake for surveillance purposes even though they are not participating in data forwarding.

Besides the aforementioned work in communication and sensing coverage that conserves energy, work in energy conservation for general sensor networks has been considered at various levels of the communication stack. From the bottom to the top, special hardware [15] is designed with multiple energy dissipation settings. MAC layer protocols developed for energy savings mostly take advantage of overhearing and scheduling to allow nodes to sleep while they are not transmitting or receiving messages [7][11]. At the network and routing layers, solutions are diversified. Data placement schemes [3] minimize energy along the transmission path through data caching. In [16], R. Ramanathan et. al. adjust communication range dynamically based on the node density to conserve energy consumed in transmission. MFR [17] by Takagi et. al. uses a minimal hop path to reduce the total number of transmissions. [21] sets routes according to the energy remaining at nodes along that path, and [10] uses mechanisms to save energy through the distribution of messages among various paths from source to destination. At the application layer, the protocols incorporate routing semantics to form groups and rotate leadership responsibilities, allowing non-leader nodes to sleep and conserve their energy [4]. Finally, data aggregation techniques inside the network and application layers also provide energy conservation features in both application independent [8] and application dependent fashions [12][13]. All of these protocols tackle the energy issue from different aspects of sensor networks, thus we consider them complementary to our work.

Recently, research has started to address QoS issues to support differentiated services for sensor networks. SWAN [1] uses feedback information from the MAC layer to regulate the transmission rate of non-real-time traffic in order to sustain real-time traffic. RAP [14] uses velocity monotonic scheduling to prioritize real-time traffic and enforces such prioritization through a differentiated MAC Layer. In [2], the delivery ratios of packets at different priority levels are affected by the forwarding probability at intermediate nodes. To date, to the best of our knowledge, no algorithm has been specifically designed to address how to provide differentiated surveillance for sensor networks. Due to the surveillance nature of most sensor network applications, we deem that this service is essential.

3. BASIC PROTOCOL DESIGN

In this section and section 4, we introduce the design of an adaptive sensing coverage scheme for sensor networks. The basic design without differentiation is introduced first in section 3. This is then the basis for the extension for differentiated surveillance to support scenarios in which a partial coverage (<100%) is sufficient or a high degree of coverage (>100%) is desired (section 4).

3.1 Design Goals

The basic design goal for this work is to provide energy efficient sensing coverage for a geographic area covered by sensor nodes. Though it has been addressed in previous work [18] [22], our work distinguishes itself from previous solutions in following sub-goals that we achieve.

- Reduce total amount of energy consumed.
- Reduce energy variation among nodes.
- Reduce communication overhead in establishing nodes' work schedules.
- Support a new extension to the service for differentiated surveillance with small overhead.
- Provide communication connectivity as an auxiliary benefit.

In the remaining parts of the paper, we support these claims with analytical analysis and simulation results.

3.2 Assumptions

First, we assume that each node knows its own location [9] and nodes are not moving. The node location information does not need to be precise when we are using conservative sensing ranges to decide the schedules (see 3.4). These are common assumptions for many sensor network applications. For simplicity and convenience of protocol description in the rest of the paper, we refer to the sensing area of a node as a circle with a nominal radius *r* centered at the location of the node itself. We will later show that we can also deal with irregular and/or non-uniform sensing areas as long as the neighboring nodes are aware of each other's sensing area. In the protocol description, we deploy the sensor nodes in a two-dimensional Euclidean plane. However, the protocol can be extended to a three-dimensional space or a curved surface without much difficulty. We also assume that the neighboring nodes should be roughly time synchronized on the order of seconds, which can be easily achieved by current millisecond-level synchronization solutions such as found in [6]. Also, the tolerance for small synchronization skews (see 3.4.2) allows the re-synchronization process to happen even less frequently so that the cost can be neglected.

The last assumption we make is that nodes can directly communicate with the neighboring nodes within a radius larger than $2r$ (r is nominal sensing radius). This is a typical case in all systems we experienced. For example, MICA II [5] has a communication range of about 1000 feet, while the sensing range is within a hundred feet even with long-range motion sensors. Actually, this is an optional assumption. The protocol works so long as the nodes are able to communicate directly or indirectly with each other within the distance of 2r and the communication range need not be regular. We make this assumption for simplicity in the protocol description and to avoid routing overhead for any two nodes that can sense a common area. This assumption also helps provide a connectivity guarantee when full sensing coverage is achieved.

3.3 Basic Design without Differentiation

Each node in the sensor network is either in sleeping mode or in working mode. Our goal for the basic design without differentiation is to have as many nodes as possible go to sleep to save energy and extend the lifetime of the sensor network while guaranteeing 100% sensing coverage to the target area.

3.3.1 A Node's Working Schedule

The lifetime of a sensor network is composed of an initialization phase and a sensing phase. During the initialization phase, each sensor node finds its own position [9] and synchronizes [6] time with neighboring nodes. After that, nodes enter into a sensing phase and start to sense environmental events. The sensing phase of nodes is divided into rounds with equal duration and are synchronized. Each node will establish a working schedule through our algorithm, which tells it when to sleep and when to work for each round. When a node goes to sleep, its sensing, communication and computation components can all be asleep and only a timer needs to work and wake up all components according to its schedule.

We denote the duration of each round as *T*. The schedule for a node is determined by a tuple with four parameters: (T, Ref, T_{front}, T_{end}). As shown in Figure 1, *Ref* is a random time reference point chosen by a node within [0, *T*). T_{front} is the duration of time prior to the reference point *Ref* and T_{end} is the duration of time after reference point *Ref*. We describe how to decide these parameters in section 3.3.2. For any given tuple (T, Ref, T_{front}, T_{end}), a node executes the following working schedule:

A node wakes up at time ($T \times i + Ref - T_{front}$) and goes to sleep at time ($T \times i + Ref + T_{end}$)

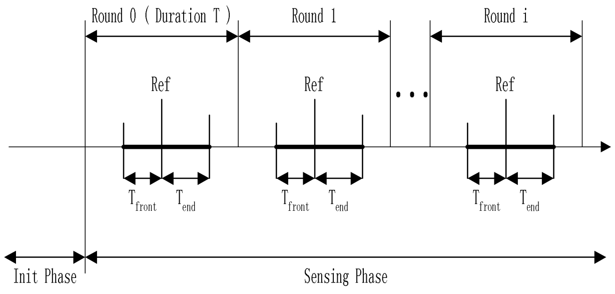


Figure 1. Node Working Schedule

It should be noted that the following two rules hold for the schedule:

1) The tuple (T, Ref, T_{front}, T_{end}) for each node is chosen during the initialization phase and does not change during the whole sensing phase, unless rescheduling is required for fault-tolerance purposes.
2) Time duration T_{front} and T_{end} should be in the interval [0, *T*) and furthermore, the sum of T_{front} and T_{end}, which is the time duration for a node to work during each round, should be less than or equal to *T*.

*3.3.2 Setting Tuple Parameters (*T, Ref, T_{front}, T_{end}*)*

In our approach, we cover the target area with a virtual square grid. We describe how to choose the unit grid size later in section 3.4.1. Now we describe the protocol that decides the schedule for each sensor node to guarantee that each grid point in the target area is covered by at least one working node at any time and with minimum energy consumption. Here we only consider the energy consumed by the sensing function, so the energy consumed by a node is proportional to its working time for each round, which is ($T_{front} + T_{end}$). Parameter *T,* which is the duration of each round, is pre-determined and kept constant across all the nodes inside the sensor network. We discuss how to set this parameter in more detail in section 3.4.2.

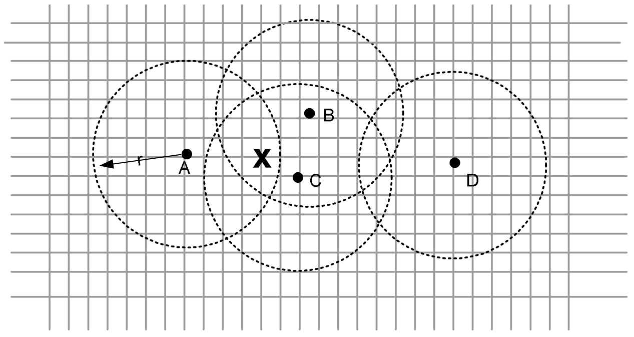


Figure 2. Sensing Coverage to Single Grid Point X

After the nodes complete localization and time synchronization, they each broadcast a beacon (a random jitter in transmission time is added to avoid collision). This beacon has the information of its location and a randomly chosen time reference point *Ref*. The reference time should be uniformly chosen among [0, T) and follows an identical independent distribution (iid). According to the assumption of direct communication within 2r, each node within the distance of 2r of the sending node should receive the beacon. For example in Figure 2, nodes A, B and C will hear each other. After every node beacons the time reference point *Ref*, each node maintains a neighbor table with the information of locations and time reference points of its neighbors within 2r's distance. For example, node A will keep information about nodes B, C and itself in a table. D's beacon is ignored since it is outside the 2r's distance of A.

Before discussing the sensing coverage of the entire area, we first focus on how to provide sensing coverage for a single grid point. Here we consider node A and grid point x shown in Figure 2. Obviously, all the sensor nodes whose sensing areas cover grid point x are in the 2r's distance of node A. Therefore, node A has the information of their locations and time reference points in its neighbor table. With such information node A sorts those time reference points in ascending order and lays out all *N* reference times including its own reference time point (see Figure 3). (Suppose that the total number of sensor nodes which can cover grid point x is *N*).

Without loss of generality, suppose node A has the ith reference time *Ref*(i), node B has the (i-1)th reference time *Ref*(i-1) and node C has the (i+1)th reference time *Ref*(i+1). Then the time interval T_{front} and the time interval T_{end} for grid point x are set by our algorithm to [*Ref*(i) - *Ref*(i-1)]/2 to [*Ref*(i+1) - *Ref*(i)]/2, respectively. There are two special cases for the T_{end} and T_{front} calculations:

1) If node A has the smallest reference time among all *N* nodes that can cover grid point X, then the time interval T_{front} is (T+ *Ref*(1)-*Ref(N)*)/2 and the time interval T_{end} is (*Ref*(2)-*Ref(1)*)/2.

2) If node A has the largest reference time among all *N* nodes that can cover grid point X, then the time interval T_{front} is (*Ref*(N)-*Ref(N-1)*)/2 and the time interval T_{end} is (T+ *Ref*(1)-*Ref(N)*)/2.

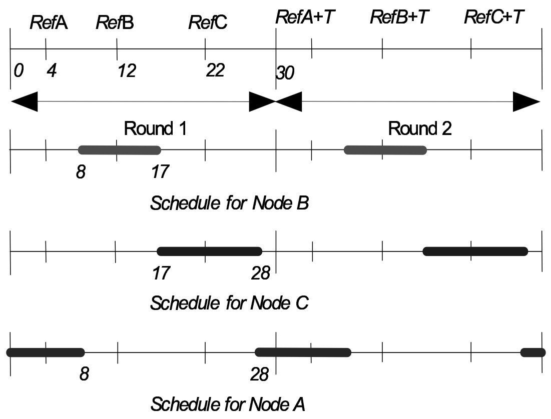


Figure 3. Calculating node schedules for a single grid point

To understand the procedure better, we explain how to decide node schedules for a certain grid point x through an example shown in Figure 3. Suppose the duration of each round *T* is 30 minutes and only nodes A, B and C can provide coverage to grid point x. Nodes A, B and C choose reference time *Ref* values 4, 12 and 22, respectively. According to our algorithm, node B will set the T_{front} value to (12-4)/2 = 4 and set the T_{end} value to (22-12)/2 = 5 to cover the half gap between the reference time points. Consequently, the parameter tuple (*T*, *Ref*, T_{front}, T_{end}) for B is (30, 12, 4, 5), which means node B stays awake for (4+5) minutes out of the 30 minutes period (4 minutes before and 5 minutes after time reference point 12). Similarly, the parameter tuples for A and C are (30, 4, 6, 4) and (30, 22, 5, 6).

It should be noted that such a schedule for a single grid point is optimal in terms of energy consumption, since the total awake time for all three nodes is equal to the duration of the round (30 minutes in the example). Any other schedule, which has less total awake time, will introduce a blind time on that grid point. Since there is no overlap among the node's working schedules for grid point x and total coverage time equals the duration of the round, it is easy to conclude that at any time, grid point x is covered by at least one node's schedule.

The same procedure is carried out for every grid point. In the end, every node has a set of schedules for all the grid points it can cover. Since it is impossible for a node to execute multiple schedules simultaneously, it is necessary to create a single integrated schedule. Now we describe how a node combines this set of *schedules for grid points* into a single *schedule for the area* it can cover.

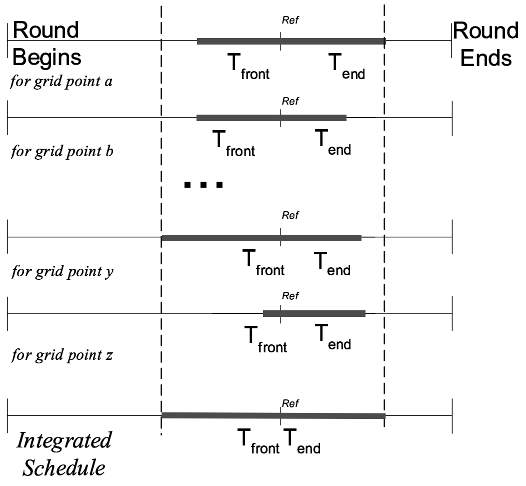


Figure 4. The Process of Schedule Integration on a node

After each node calculates its schedules for all grid points it can cover, it creates an integrated schedule (T, Ref, T_{front}, T_{end}). The integrated T_{front} of node A is the largest one among all T_{front} time intervals of node A for all the grid points that node A can cover. Similarly, the integrated time interval T_{end} of node A is the largest one among the T_{end} time intervals of node A for all grid points that node A can cover. Therefore, the integrated schedule of each node is the union of its schedules for all the grid points it can cover. Previously, we have concluded that a given grid point is covered fully in time by at least one neighboring node's schedule. Thus with the calculated integrated schedules of all the sensor nodes, at any time any grid point is covered by at least one awaken sensor node. To clarify this integration process further, Figure 4 shows how to calculate the integrated schedule on a node according to its schedules for a set of grid points.

Note that many sensor networks are deployed with a reasonable density so that when every node is working, the whole target area is sensed by the sensing nodes without blind holes and our approach guarantees a 100% sensing coverage at any time during the sensing phase. Another noteworthy point is that, in order to prevent the nodes at the edge of the network from working all the time, when we specify the target area, we need to exclude the edge of the network. Then with enough density, it is highly probable that any grid point in the specified target area can be sensed by at least two nodes when all nodes are awake.

One might argue that intuitively such integration through the union of schedules for each grid point would lead to a pessimistic result with very long working duration. However, this is not the case. In fact, a node usually has the same schedule for grid points near to each other, which will not increase the length of integrated T_{front} and T_{end} when the union operation takes place. This has been shown in the simulation.

Our approach has several major advantages for sensor network deployment:

1) Communication overhead in our approach is minimized. For the approach described above, each node needs only to beacon one message without consideration of collisions. Compared to approaches that a node beacons each time it wakes up [22] or for each round [18], the energy consumed by our approach's communication is much less.
2) The random time reference point with uniform distribution contributes to the energy balance among the nodes. An optimization on this (discussed later) can enhance this energy-balancing feature even more.
3) Due to the optimal energy consumption property for a single grid point[1], the total energy consumption of our approach increases very slowly with the increase of node density. As a result, the system lifetime increases proportionally with the increase in node density. We show this in the evaluation section.

Although the length of working time for each node may differ due to randomness, the protocol is still fair in the sense that no node is given priority to work longer or shorter.

It can be easily seen from the above description that even when at the initialization phase some node is not discovered by its neighbors, once there are enough nodes to cover the target area, the 100% sensing coverage is still guaranteed. However, it may cause other nodes to work unnecessarily longer.

The protocol just described provides a guarantee of 100% sensing coverage at any time for the target area. Acknowledging that in some situations, full coverage by single sensor nodes is not enough to obtain high confidence in detection, our basic design can be easily extended to provide a *guaranteed* degree of coverage for a geographic area with very small overhead (see section 4).

3.4 Design Issues

This section completes our approach with several design issues concerning grid granularity, clock skew, irregular sensing patterns and node failures.

3.4.1 Dealing with Possible Blind Points in Space due to the Large Granularity of the Grid Size

One may argue that in our protocol description, there can be small sensing holes because we only guarantee grid point coverage, instead of a grid area. Here we will explain why this approach still makes sense. We denote the unit grid size as *d*. We also define a conservative sensing area of a sensor node as a smaller concentric circle than its actual sensing area in order to handle this problem and other issues.

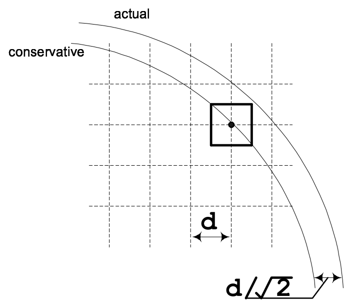


Figure 5. Grid Size Granularity Issues

[1] However, the integrated schedule is sub-optimal due to working schedule redundancy among nodes.

From figure 5, it can be easily seen that the target area is fully covered by the sensor nodes' actual sensing areas as long as 1) each grid point is inside the conservative sensing area of at least one sensor node, and 2) the difference between the actual sensing radius r_{actual} and the conservative sensing radius $r_{conservative}$ is greater than $d/\sqrt{2}$. Then the problem of covering the whole target area with actual sensing areas reduces to that of covering the grid points with conservative sensing areas. With such observations, we decide the upper bound of the grid size according to the following rules.

1) If we use the actual sensing radius to calculate the nodes' working schedules, the grid size *d* should be smaller than the size of the objects we want to detect, so that these objects will not fit in the gap among the grid points.

2) If we use a conservative sensing radius, the grid size *d* should be smaller than $\sqrt{2}(r_{actual} - r_{conservative})$.

At the same time, a lower bound for the grid size should be set with consideration of computational cost (see 3.7.1).

Other factors like the imprecision of location information and irregularity of the sensing areas should also be considered when we decide the conservative sensing radius.

In the remaining protocol description, we do not distinguish between concepts of grid point coverage and area coverage.

3.4.2 Dealing with Possible Blind Points in Time due to Synchronization Skew

Due to the precision limits of synchronization algorithms for real sensor network systems, there is always some synchronization skew. When nodes try to trigger some events at the same time, the actual time the event is triggered on each node will be different due to the synchronization skew. Our protocol is based on synchronization, so we must compensate for the synchronization skew to make the schedules seamless. We define the maximum skew of the whole sensor network as T_s. In order to bridge the gaps caused by the synchronization skew, we only need to extend the schedule of each node for $T_s/2$ in both directions. This compensation affects the performance of the protocol. In order to minimize the impact of this performance degradation, it is favorable to increase the time duration *T* for each round. The reason is that when T_s is fixed, the longer a round is, the lower is the compensation overhead compared to the duration of a round or the schedules for a round. For example, suppose $T_{front}+T_{end}$ is 30 minutes, if we allow synchronization skew to be 1 second, the performance will be degraded less than 0.1%. However, *T* should not be very large, which may lead to unbalanced energy consumption.

3.4.3 Dealing with Possible Blind Points in Space due to Irregularity in Sensing Patterns

In the description of the protocol, we assume that the sensing area of each node is uniformly a regular circle, and the nodes are deployed in a two-dimensional Euclidean plane. However, if we review the protocol description, it is clear that these assumptions are not mandatory. We define a node's sensing neighbors as the nodes that can sense at least a sub-area of its sensing area. Our protocol works as long as the sensing neighbors are aware of each other's sensing area, through pre-existing knowledge or communication with small overhead, no matter where the nodes are deployed and whether the sensing areas of nodes are regular or uniform. Even without precise information about sensing areas or node locations, we can use a conservative sensing range to make sure of sensing coverage.

3.4.4 Reschedule for Fault Tolerance

In this subsection, we propose a mechanism to provide fault tolerance with the basic protocol that guarantees a 100% coverage. We assume any two nodes do not fail at the same time and when some node fails, it just stops working silently. After each node calculates its integrated schedule, it broadcast this schedule to its 2r neighbors. When a node is working, it sends a heartbeat signal periodically to its working neighbors within the 2r range. Each working node knows the schedules of its 2r neighbors and expects heartbeat signals from the working ones among such neighbors. After a node detects the failure of one of its neighbors through heartbeat timeout, it wakes up all its neighbors that are also the 2r neighbors of the failed node. The wakeup mechanism can be achieved through either a hardware wakeup circuit or a software solution. Then these 2r neighbors recalculate the schedules with the knowledge that the problematic node failed. With the new schedules, the target area is guaranteed to be fully covered by working nodes' sensing areas. In exchange for the fault tolerance feature, the communication overhead increases significantly due to the heartbeat signals.

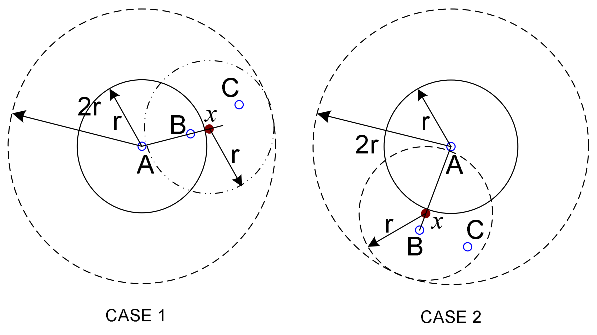


Figure 6. A Wakeup Example for Rescheduling

Figure 6 gives a simple proof to show that all sleeping 2r neighbor of a failed node *A* can be awakened up by at least one working node that is also node *A*'s 2r neighbor. There are two cases: 1) suppose a sleeping node *B* is inside the sensing area of node *A*. We extend the line segment that connects *A* and *B*. A point *x* on the extended line segment is very close, but outside of node *A*'s sensing area. There must be some working node *C* that is sensing point *x*. It can be easily seen that node *C* is in the 2r neighborhood of node *A*, and the distance of *C* and *B* is less than 2r so that node *C* can wake up node *B*. 2) Suppose a sleeping node *B* is outside of node *A*'s sensing area, we connect *A* and *B* with a line segment. We choose point *x* so that *x* is on the line segment and is very close, but outside of *A*'s sensing coverage. There must be some working node *C* that covers point *x*. It can be easily seen that *C* is in the 2r neighborhood of *A* and the distance of *C* and *B* is less than 2r so that *C* can wake up *B*.

3.5 Optimizations and Extensions

This section discusses optimizations and extensions built upon our basic design. These techniques achieve better performance in exchange for more beacons required.

3.5.1 Second Pass Optimization for Lower Energy Consumption

Each node's integrated schedule is the super set of its schedules for many grid points. It is possible that the integrated schedules are more than sufficient to provide the coverage guarantee. Therefore, we make a second pass optimization to reduce this kind of redundancy. In order to simplify the description, we only describe the optimization for a 100% degree of coverage. After a sensor node calculates the integrated schedule, it sends this integrated schedule to its neighbors within the distance of 2r. When some node finds out that it is the one with the longest schedule, which has the largest $T_{front}+T_{end}$ value compared to its 2r neighbors, it will recalculate a new schedule. The new schedule shrinks T_{front} and T_{end} values as much as possible, while still guaranteeing 100% coverage for all the grid points the node is able to sense. Then it beacons the new schedule to its neighbors within 2r's distance. Each node that has the longest schedule among the nodes in the 2r range that have not updated their schedules recalculates the schedule and beacons its new schedule. When there is a conflict of beacons, the conflicting nodes make use of a random back-off scheme to send the beacons again and avoid conflicts. In the evaluation section, we show that the system performance improves with this optimization.

3.5.2 Multi-Round Extension for Energy Balance

A sensor network's lifetime is affected by both the total energy consumption of all the sensor nodes and the variation of energy consumption per unit time by each node. In a sensor network that has higher variance in energy consumption, some nodes run out of energy much earlier than others do. Then blind holes will appear in the target area, or another scheduling process should be applied to eliminate the blind holes. Given the same total energy consumed, the less is the variation of energy consumption per unit time, the longer the network lasts with full sensing coverage.

Energy consumption variance in our protocol can be attributed to at least two reasons. First, due to the randomness of node deployment, some nodes may have fewer neighbors than others in the range of 2r, so they have to work longer than others do. This unbalance is intrinsic for a certain deployment and little can be done to deal with this problem. The second reason is due to the randomness of reference time. As an example, if several sensor nodes can all sense a grid point, and their reference times are selected very close to each other, there must be an extraordinarily long schedule of one of the nodes for this grid point compared to others. Here we propose an extension to smooth the energy consumption variation caused by the randomness of reference time selection. Instead of calculating a single schedule, we calculate M schedules for each node according to M independently selected random reference times *Ref* for each node. At the initialization phase, each node beacons its M reference times *Ref* to its 2r neighbors. The ith (i is between 1 and M) schedule is calculated with the ith reference time of each node. Then at each round in the sensing phase, the nodes choose one schedule consecutively. We will show in the evaluation section that this extension decreases the variation in energy consumption.

3.6 Effect on Communication Connectivity

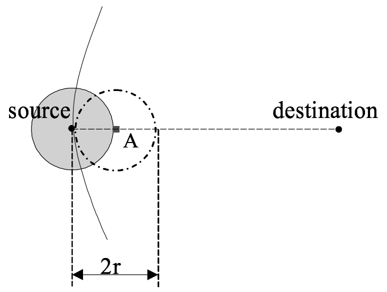


Figure 7. Communication connectivity

Sensing coverage and communication connectivity are two major targets for energy-efficient approaches for sensor networks. For quite a few routing protocols like [10], the communication connectivity is based on the existence of a next hop closer to the destination. Say we want to send a message from a source to a destination and the source cannot reach the destination in one hop (Figure 7) in a sensor network whose 100% sensing coverage is guaranteed with our protocol. Let us consider the location A that is very close to, but out of the sensing circle of the source node and on the line segment connecting the source and the destination. In order to cover this location, there must be another node in the dashed circle with the radius r. We can see that any node inside the dashed circle is closer to the destination than the source node so that it can serve as the next hop. Thus, the existence of a next hop closer to the destination is guaranteed when a 100% sensing coverage is provided, which is a good feature that supports communication connectivity.

3.7 Protocol Analysis

This section provides the theoretical analysis of our algorithm for a better understanding of the performance issues.

3.7.1 Complexity Analysis

The major complexity involved in the proposed protocol is the computational complexity. In the following discussion, we analyze the steps the protocol takes to calculate the integrated schedule for a single node.

The first step is to calculate the schedules for each grid point. We know that the number of grid points in the node's sensing area is about $\pi r^2/d^2$, and suppose that a single grid point can be sensed by *N* nearby nodes on average, which depends on node density. For each grid point, a node should decide if it is in the sensing area of each node in the 2r range, which involves $4N\pi r^2/d^2$ distance computations. For each grid point, all these nodes' reference times are sorted, which takes *cNlogN* basic steps, where c is a small constant. The schedule for a grid point is decided in negligible time. Last, it takes $2\pi r^2/d^2$ comparison to get the integrated schedule for this node. For instance, in a certain sensor network, we choose *d* as *r/10* and *N* as 10, the basic steps it takes to calculate the integrated schedule is in the order of magnitude of ten thousand. In a typical hardware platform [5], the energy consumed for tens of thousands of basic calculations is in the same order of magnitude of transmitting a single bit, so it is negligible compared to communication overhead.

The memory space occupied for calculating the integrated schedule is composed of 1) *4N* entries on average for the neighborhood table, 2) *N* memory units for sorting the reference times on average and 3) $2\pi r^2/d^2$ memory units for storing the schedules for grid points. Typically, the memory usage of the protocol on each node is in the order of magnitude of Kbytes, which is suitable for resource bounded sensor nodes like [5].

3.7.2 Communication Overhead Analysis

In the basic design, each node only beacons once (without consideration of collisions) in the initialization phase, disseminating the information about its reference time and location. During the sensing phase, there is no need to send extra messages for coordination. In the second round optimization, each node beacons three messages indicating the reference time, the integrated schedule and the updated schedule. Typically, as mentioned, the energy consumption for sending or receiving a single bit is on the same order of magnitude as running thousands of instructions. In real sensor networks, the retransmissions used to deal with contention make the energy consumption for sending a message even higher. Therefore, the very few number of messages transmitted in our protocol is preferable for minimizing energy consumption and extending system lifetime. Any extra energy consumed by complicated computation is negligible compared to the energy saved by decreasing communication overhead.

4. ENHANCED DESIGN WITH DIFFERENTIATION

It is easy to modify our basic protocol to deal with differentiated sensing coverage when there are different requirements, statically or dynamically. In the basic algorithm, the middle point between two reference times is selected as the start/end point of a schedule for a grid point. That is to say, T_{front} and T_{end} are chosen as half of the intervals between two reference times, respectively. If we want to adjust the degree of sensing coverage to an arbitrary degree α, the only change in the protocol required is to extend or shrink T_{front} and T_{end} proportionally, i.e., T_{front} and T_{end} should be multiplied by α. It can be easily seen that with all the schedules for each grid point multiplied by α, the aggregate schedule of a sensor node is also multiplied by α. There is one constraint for the extension: if the sum of the intervals multiplied by α is greater than the round duration T, we should shrink the multiplier so that $T_{front}+T_{end}$ does not exceed the round interval.

More formally, a node's working schedule for differentiated coverage is determined by the tuple (T, Ref, T_{front}, T_{end}, α), and a node executes the following working schedule:

A node wakes up at time ($T \times i + Ref - T_{front} \times \alpha$) *and it goes to sleep at time* ($T \times i + Ref + T_{end} \times \alpha$)

The intended degree of coverage α provides a guarantee that the average number of working nodes that sense each grid point in each round is at least α, up to the limitation imposed by the number of sensor nodes present. Specifically, when α is 2, at any time, each grid point is sensed by at least 2 working nodes. When any single node fails, the target area is still fully covered. For an α value other than 1 and 2, the number of working nodes that sense a specific grid point is only guaranteed in the average sense. This feature is of great importance for a highly secured area. In fact, this differentiation mechanism can be generalized to the situations where α is smaller than 1. By doing so, our solution will provide partial coverage in time to the target area and further reduces the total energy consumed.

It should be noted that this differentiation mechanism can be applied with very small overhead. After the sensor network is deployed, the sensor nodes are programmed so that any grid point of the target area is covered. When an emergency event happens, sensor nodes in the whole sensor network or part of the network should raise the degree of coverage immediately. For example, when a sensor network used for surveillance detects an intruder, more nodes should be awakened to achieve a more precise and reliable tracking of the intruder. The obvious advantage of our scheme is that *no rescheduling through beaconing is required*. One can disseminate the degree of coverage α to the target area and nodes simply expand their schedules according to this parameter to provide higher degree of coverage without further beacon exchange.

5. BASELINES FOR COMPARISON

To enable a detailed evaluation, theoretical baselines and baselines from recent publications [18] [22] are considered in our evaluation.

5.1 Theoretical Upper Bound and Lower Bound

Here we give the upper bound and lower bound of energy consumption per unit time for a 100% coverage of a target area. We use them as baselines. The upper bound is trivial and happens when all the sensor nodes are working all the time. For different hardware platforms, the energy consumed by a working node per unit time varies significantly. For generality, we denote it as one unit of energy consumption U_e. So the upper bound of energy consumption by a sensor network composed of N nodes is trivially NU_e. The lower bound of energy consumption of a sensor network differs according to the deployment of the sensor nodes. However, we are able to obtain a theoretical lower bound although it may not be reachable by a certain sensor network. The lower bound is based on the following theorem:

When we cover a unit area with equivalent circles, the lower bound of the number of circles used is $2/\sqrt{27}r^2$. The corresponding lower bound for energy consumption is $2AU_e/\sqrt{27}r^2$, where A is the size of the target area.

Details of this theorem can be found in [19].

5.2 Related Work for Comparisons

[18] and [22] are the first works to address the sensing coverage problem in sensor networks. [22] proposes a mechanism based on probing for energy-efficient sensing coverage for sensor networks. In the mechanism, sleeping nodes wake up according to a dynamically changing wakeup rate. When a node wakes up, it broadcasts a probing message. If there is any working node in a radius of r centered at the wakeup node, the working node will broadcast a reply and the awakened node will receive it. If the awakened node does not receive any reply in a certain amount of time, the node switches into working mode. The node in working mode continuously senses its sensing area until it fails or runs out of energy. [18] shows that such a probing-based mechanism is not able to ensure the original sensing coverage and avoid blind holes. It also points out that although by decreasing the probing area, the probing based mechanism can minimize the area of blind holes; it keeps more nodes working compared to the sponsor coverage-based scheme in [18]. Hence, more energy consumption

is required in [22]. Therefore, in the evaluation section we only compare our protocol with [18].

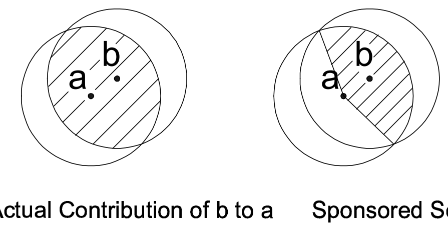


Figure 8. Underestimation of sensor coverage by neighboring nodes

[18] is referred to as the *sponsored coverage scheme* in this paper. It presents a node-scheduling mechanism based on an eligibility rule. At the beginning of each round, each node calculates its eligibility for going to sleep. A back-off mechanism is used to avoid sensing holes caused by simultaneous actions of multiple nodes. Our approach has two advantages over this one. First, the sensor nodes only need to send a few messages for scheduling in the initialization phase, while the scheme in paper [18] requires broadcasting to the neighbors at the beginning of each round. Although the exact results depend on the system settings, it can be easily concluded that the per-round based broadcast messages cause more overhead than ours. Second, when calculating eligibility for going to sleep, the contribution of a neighbor node's sensing area to the sensing area of the node itself is simplified to a "sponsored sector", which is much smaller than the actual contribution, especially when the neighbor node is very close to the node itself (figure 8). So compared with our protocol, theirs needs more nodes to be awake; therefore, more energy consumption occurs.

6. EVALUATION

In this evaluation section, we demonstrate the improved performance generated by our scheme in terms of 1) total amount of energy consumed, 2) energy variation among nodes, 3) half-life of the network and 4) sensing coverage over time. Moreover, we will also demonstrate a set of new features found in our enhanced differentiated service that is not supported by previous schemes.

In the evaluation we do not include communication cost due to data transfer because it is highly application specific. Also for some applications as intruder detection, data transfer only happens when some rare events are issued. For such systems, energy spent on data transfer is relatively insignificant.

6.1 Simulation Configuration

We run our basic protocol and extensions on a special purpose simulator. In our simulation, the sensor nodes are distributed in a 160mX160m square field. The sensing range is 10m and the communication range is 25m. The sensor nodes are deployed with a uniform distribution into the square field. For our protocol, the target area is the 140mX140m square in the center of the square field to prevent the nodes at the edge from working all the time. We only do statistics on the central 100mX100m field to eliminate the edge effect. All experiments are repeated 10 times with different random seeds and different node deployments. The 95% confidence intervals of the results are about 5~10% of the means.

6.2 Total Energy Conservation

In this experiment, we investigate the energy conservation performance in term of total energy consumed per unit of time. We collect results from our basic design, the second pass optimization of our design and [18]'s sponsored coverage scheme. We also compare the simulation results with the lower bound and the upper bounds. The following figure shows the simulation results.

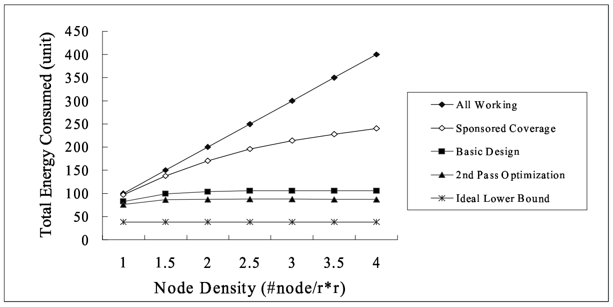


Figure 9. Total Energy Consumption per Unit of Time

From Figure 9, we can see that our protocol consumes much less energy than [18]'s sponsored coverage scheme. This is due to the fact that in the sponsored sector based scheme, the contribution of one working node to another node's sensing coverage is not fully considered. To be more specific, the sponsored sector is much less than the actual contribution (see 5.2). While in our protocol, the full contribution is considered so the redundancy is reduced. Another distinguished feature that our protocol provides is that with the increase of node density, total energy consumed by our protocol changes very little. In contrast, the sponsored sector based scheme suffers when density increases. This is because the closer nodes are to each other, the more severe an underestimation of sensor coverage by neighboring nodes is. As a result, our scheme outperforms sponsored coverage scheme by as much as a 50% reduction in energy consumption when the density is about 3 nodes per r-square. Our scheme ensures that the system lifetime will increase nearly proportionally with the node density of the sensor network, which is desirable for long-life-time surveillance applications.

6.3 Balancing the Energy Consumption

In this simulation, we investigate performance of energy balance. We measure the standard deviation of energy consumed by each node in our basic design and in the multiple round extension with M=10. Results are compared with the sponsored coverage scheme.

Figure 10 shows that in both the sponsored coverage scheme and our scheme, the average energy consumption for a single node reduces when the network density increases. However, our scheme reduces at a much faster rate than the sector-based scheme. When the node density is reasonably high (> 2 node/ r-square), our protocol outperforms the sponsored vector based scheme with regards to energy consumption balance among nodes. From Figure 10, we can also see that the multiple round extension can effectively reduce a certain amount of the variation in energy consumption.

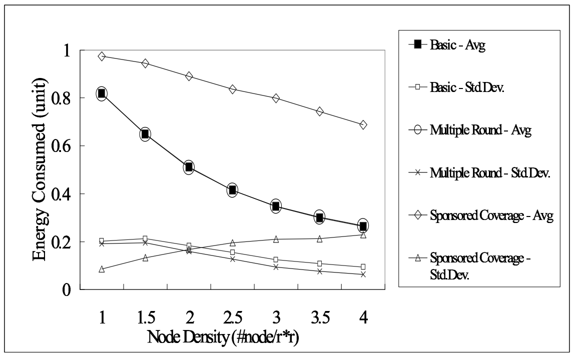


Figure 10. Single Node Energy Consumption: Average and Standard Deviation

6.4 Half-Life of the Network

In order to measure the system lifetime extension due to our protocol, we define the half-life of a sensor network as the time from the beginning of the deployment until exactly half of the sensor nodes are still alive. We assume that the lifetime of a node when always working is 1000 minutes. The round duration we used in simulation is 10 minutes.

We see from the Figure 11 that the distinguishing feature of our approach is that the system half-life increases nearly linearly as the node density increases, while the sponsored approach increases slowly when the node density increases. For example, our approach increases the half-life of the network by 130% when node density is 4 per r-square. There are two reasons contributing to this phenomenon. First, the sponsored coverage approach consumes more energy on average than our approach. Second, the standard deviation among nodes in the sponsored approach increases significantly as node density increases (Figure 11), which leads to some nodes dying faster than others.

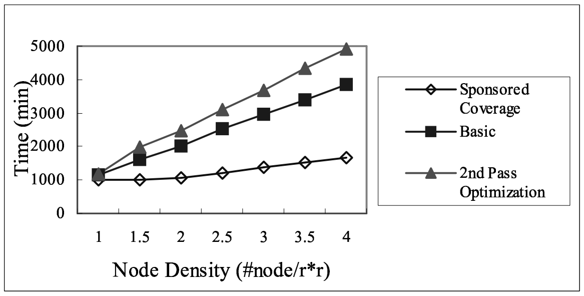


Figure 11. Half-Life of the network

6.5 Coverage over Time

One of the most important things that the designers of sensor networks are interested in is, after a certain amount of time from when the network is deployed, how much of the target area is covered by the working sensors. In order to answer this question, we simulate the percentage of coverage after certain time intervals. The percentage of coverage takes care of the coverage in both time and space. For each grid point in the target area, we choose several sampling times during a round. If for some (grid point, time) pair, the grid point is sensed by some working node, we call this pair to be "valid". We count the ratio of the valid pairs among all the pairs with both our proposed protocol and the sponsored sector based scheme described in [18]. Different curves shown in Figure 12 are obtained under different node density. The curves with the hollow markers are the results from the sponsored coverage scheme and the curves with the solid markers are the results form our scheme.

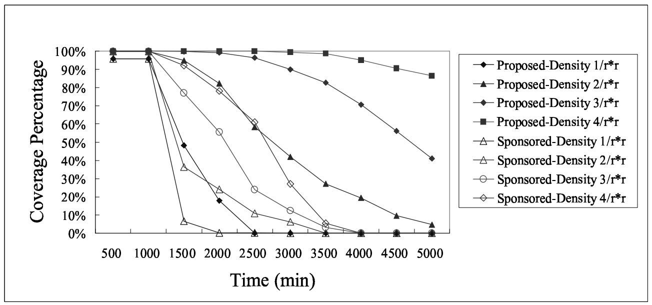


Figure 12. Sensing Coverage over Time

From the simulation results, we see that over time with the same node density, the target area is covered more with our protocol than with the sponsored coverage scheme. For example, at density of 2 per r-square, it take about 5000 minutes for our scheme to degrade to zero sensing coverage, while the sponsored coverage scheme degrades to zero sensing coverage in just 3500 minutes.

6.6 Total Energy Consumed for Differentiated Surveillance

Unlike previous schemes, our basic design can be easily extended to provide differentiated surveillance by proportionally increasing the T_{front} and T_{end} interval values. Figure 13 shows the simulation results of differentiated coverage with two different node densities.

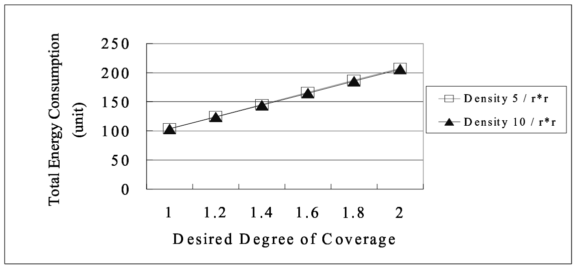


Figure 13. Total Energy Consumed for Differentiated Surveillance

From the results, we conclude that 1) the energy consumed per unit time increases linearly when the parameter of desired degree of coverage α increases and 2) the energy consumed under different node densities changes very little when the node densities are high enough. This is shown in Figure 13 where the two curves nearly overlap each other. This independence between the physical node density and actual energy consumed is quite desirable.

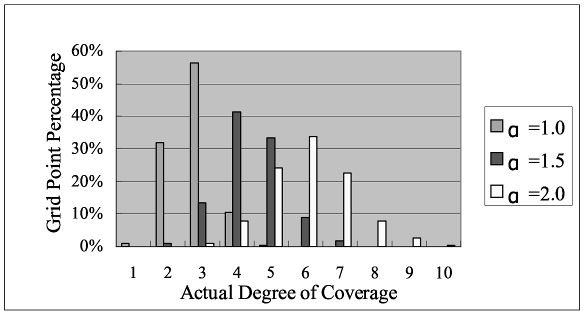


Figure 14. Actual Degree of Coverage for Differentiated Surveillance

6.7 Actual Degree of Coverage for Differentiated Surveillance

In this simulation, we set the node density to be 5 per r^2, and sample the actual degree of coverage for 100X100 grid points in the center. In addition to the proof from section 4, Figure 14 shows that our scheme can guarantee a degree of coverage to the desired value α for the target area. For example, if α is 2, then we guarantee that all grid points are covered by more than two sensor nodes and a single failure node will not affect the full coverage. However, the guaranteed α is the minimum possible degree of coverage for each grid point. Due to the protocol's redundancy, most grid points are covered by a higher degree than the desired degree of coverage α. Figure 14 shows the distribution of actual degree of coverage for the grid points when we apply different desired degrees of coverage α. For example when α = 2, we guarantee that no grid point has an actual degree of coverage of less than 2. We acknowledge that the current solution only guarantees a sensor coverage lower bound and future work will continue to investigate the possibility to provide a sensor coverage upper bound for better energy conservation.

7. CONCLUSION

In this paper, we introduce an adaptable sensing coverage mechanism for differentiated surveillance in sensor networks. Unlike previous schemes, our scheme guarantees not only full sensing coverage to a certain geographic area, but also the degree of coverage up to the limit imposed by the density of nodes available. The distinguishing advantages of our scheme are a much longer network half-life through energy conservation and balancing among sensor nodes, and a small communication overhead required to establish a working duty schedule among nodes. Several optimizations and extensions are proposed to provide even better performance. Simulations show that our protocol accomplishes differentiated surveillance with low energy consumption and outperforms other state-of-art schemes by as much as a 50% reduction in energy consumption and as much as a 130% increase in the half-life of the network.

8. ACKNOWLEDGEMENTS

This work was supported in part by NSF grant CCR-0098269, the MURI award N00014-01-1-0576 from ONR, and the DARPA IXO offices under the NEST project (grant number F336615-01-C-1905).

9. REFERENCES

[1] G.H. Ahn, A. T. Campbell, A. Veres, and L. H. Sun, "SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks," *IEEE INFOCOM'2002*, New York, June 2002.

[2] S. Bhatnagar, B. Deb, and B. Nath, "Service Differentiation in Sensor Networks," *Fourth International Symposium on Wireless Personal Multimedia Communications*, September 2001.

[3] S. Bhattacharya, H. Kim, S. Prabh, and T. Abdelzaher, "Energy-Conserving Data Placement and Asynchronous Multicast in Wireless Sensor Networks," *First International Conference on Mobile Systems, Applications, and Services (MobiSys)*, San Francisco, CA, May 2003.

[4] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks," *Proceedings of the 6th ACM MOBICOM Conference*, Rome, Italy, July 2001.

[5] Crossbow Technology, Inc., *http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0042-01_A_MICA2.pdf.*

[6] J. Elson, L. Girod, and D. Estrin, "Fine-Grained Network Time Synchronization using Reference Broadcasts," *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Boston, MA.

[7] C. Guo, L. C. Zhong, and J. M. Rabaey, "Low Power Distributed MAC for Ad Hoc Sensor Radio Networks," *Proceedings of IEEE GlobeCom 2001*, San Antonio, November 25-29, 2001.

[8] T. He, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher, "AIDA: Application Independent Data Aggregation in Wireless Sensor Networks," *ACM Transactions on Embedded Computing System*, 2003, to appear.

[9] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher, "Range-Free Localization Schemes in Large Scale Sensor Networks," *MobiCom 2003*, September 2003.

[10] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks," *International Conference on Distributed Computing Systems (ICDCS 2003)*, Providence, RI, May 2003.

[11] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," *HICSS '00*, January 2000.

[12] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks," *Proceedings of the 22nd International Conference on Distributed Computing Systems*, Vienna, Austria, IEEE. July 2002.

[13] B. Krishnamachari, D. Estrin, and S. Wicker, "Impact of Data Aggregation in Wireless Sensor Networks," *International Workshop on Distributed Event-Based Systems*, Vienna, Austria, July 2002.

[14] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He, “RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks,” *IEEE RTAS 2002*, San Jose, CA, September 2002.

[15] R. Min, M. Bhardwaj, S. H. Cho, A. Sinha, E. Shih, A. Wang, and A. Chandrakasan, “An Architecture for a Power-Aware Distributed Microsensor Node,” *2000 IEEE Workshop on Signal Processing Systems (SiPS '00),* October 2000.

[16] R. Ramanathan and R. Rosales-Hain, “Topology control of multihop wireless networks using transmit power adjustment,” *Proc. IEEE lnfocom 2000*, March 2000.

[17] H. Takagi and L. Kleinrock, “Optimal Transmission Ranges For Randomly Distributed Packet Radio Terminals,” *IEEE Trans. on Communications*, 32(3):246-257, March 1984.

[18] D. Tian and N. D. Georganas, “A Node Scheduling Scheme for Energy Conservation in Large Wireless Sensor Networks,” *Wireless Communications and Mobile Computing Journal*, May 2003.

[19] R. Williams, *Geometrical Foundation of Natural Structure: A Source Book of Design*, Dover Publications Inc, New York, pp. 51-52, 1979.

[20] Y. Xu, J. Heidemann, and D. Estrin, “Geography-informed Energy Conservation for Ad Hoc Routing,” *ACM MOBICOM 2001*, Rome, Italy, July 2001.

[21] Y. Xue and B. Li, “A Location-aided Power-aware Routing Protocol in Mobile Ad Hoc Networks,” *Proceedings of IEEE Globecom 2001,* Vol. 5, pp. 2837-2841, San Antonio, Texas, November 25-29, 2001.

[22] F. Ye, G. Zhong, S. Lu, and L. Zhang, “Energy Efficient Robust Sensing Coverage in Large Sensor Networks,” *UCLA Technical Report 2002*.

EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks [*]

Abdelzaher T., Blum B., Cao Q., Chen Y., Evans D., George J., George S., Gu L., He T., Krishnamurthy S., Luo L., Son S., Stankovic J., Stoleru R., Wood A.
Department of Computer Science, University of Virginia, Charlottesville, VA 22904

Abstract

Distributed sensor networks are quickly gaining recognition as viable embedded computing platforms. Current techniques for programming sensor networks are cumbersome, inflexible, and low-level. This paper introduces EnviroTrack, an object-based distributed middleware system that raises the level of programming abstraction by providing a convenient and powerful interface to the application developer geared towards tracking the physical environment. EnviroTrack is novel in its seamless integration of objects that live in physical time and space into the computational environment of the application. The performance of an initial implementation of the system is evaluated on an actual sensor network based on MICA motes. Results demonstrate the ability of the middleware to track realistic targets.

Keywords: sensor networks, programming paradigms, tracking, QoS, distributed systems

1 Introduction

The work reported in this paper is prompted by the increasing importance of large-scale wireless sensor networks [15] as a future platform for a growing number of applications such as habitat monitoring [7, 21], intrusion detection [29], defense, and scientific exploration. Advances in hardware miniaturization [10] have made it economically viable to develop embedded systems of massively distributed disposable sensor nodes, characterized by coordination of a very large number of tiny wireless computing elements. A great impediment to rapid deployment of such systems lies in the lack of distributed software and programming support for sensor network applications. A new distributed computing paradigm is needed that exports appropriate abstractions and implements efficient information management protocols in large-scale sensor networks. EnviroTrack is an attempt to develop such a paradigm.

EnviroTrack is a middleware layer that exports a new address space in the sensor network. In this space, physical events in the external environment are the addressable entities. This type of addressing is convenient for applications that need to monitor environmental events. For example, a surveillance application that monitors vehicle movement behind enemy lines may assign unique labels to individual vehicles. Their state can then be addressed by reference to these labels. Moreover, computing or actuation objects can be attached to individual addresses in much the same way computation is assigned to IP hosts in an Internet-like environment. Such attached computation or actuation is then performed in the physical neighborhood of the named entity. Hence, for example, a microphone could be turned on at some network address (e.g., one that names a vehicle in the external environment) to listen-in on the corresponding environmental object. As the named vehicle moves, the middleware will turn on the appropriate nearby node microphones such that a non-interrupted audio stream is delivered to the receiver despite the mobile nature of the source. Communication can also occur between two mobile endpoints. For example, a walking soldier with a PDA may track the position of a suspect vehicle detected elsewhere in the network. In short, we (i) export a novel logical address space in which external environmental objects are the labeled entities, and (ii) allow arbitrary data, computation, or actuation to be attached to such logical network addresses. These data, computation, and actuation are encapsulated in an abstraction we call *tracking objects*.

The EnviroTrack middleware library implements a set of protocols that off-load from an application developer the details of inter-object communication, object mobility, as well as the maintenance of tracking objects and their state. It abstracts away the fact that computation associated with the object may be distributed and performed by all sensor nodes in the vicinity of the tracked physical entity. As the tracked entity moves, the identity and location of the sensor nodes in its neighborhood change, but the tracking object representing it remains the same. The programmer thus interacts with a changing group of sensor nodes through a simple, uniquely addressable, object interface.

EnviroTrack has been implemented and tested on a popular sensor network platform based on MICA motes [16]. Our initial implementation of this infrastructure uses compiled NesC [13] programs on TinyOS [15], an operating system for

[*]The work reported in this paper was supported in part by the National Science Foundation grants EHS-0208769, CCR-0205327, and CCR-00-92945, DARPA grant F33615-01-C-1905, and MURI grant N00014-01-1-0576.

sensor networks. Recent advances in programming support for sensor networks, such as the development of a virtual machine [19], will significantly simplify the code development and dissemination effort in the future. We present evaluation results, which illustrate how typical sensor-network applications that use EnviroTrack will perform on the current hardware platform.

The rest of this paper is organized as follows. Section 2 defines the tracking problem in more detail, describes our programming system architecture, and elaborates on the main abstractions provided by EnviroTrack. Section 3 illustrates how a sample tracking application can be written in EnviroTrack. Section 4 provides implementation details. Section 5 presents a performance evaluation. An overview of related work is presented in Section 6. The paper concludes with Section 7.

2 Programming Model

The programmer's view of an application written in EnviroTrack is depicted in Figure 1. Sensors which detect certain user-defined entities in the physical environment form groups, one around each entity. A network abstraction layer associates a *context label* with each such group to represent the corresponding tracked entity in the computing system. Context labels can be thought of as logical addresses of virtual hosts (contexts) which follow the external tracked entity around in the physical environment. In the following, we use contexts and context labels interchangeably. Objects can be attached to context labels to perform context-specific computation. These attached objects are called *tracking objects*. They are executed on the sensor group of the context label. Since the actual location of the tracking object is the nodes in the physical vicinity of the target, the object can perform local sensing and actuation to interact directly with the target's locale. For completeness, EnviroTrack also supports conventional static objects that are not attached to context labels.

Context labels have types depending on the entity tracked. For example, a context label of type CAR is created wherever a car is observed. To declare a context label of some type e (named after the tracked event type), the programmer must supply three pieces of information. First, the programmer supplies a function $sense_e()$ that describes the sensory signature identifying the tracked environmental target. For example, if the context type is to identify moving vehicles, $sense_e()$ might be a function of magnetometer and motion sensor readings. The middleware watches for the specified sensory pattern in the environment and creates a sensor group around the detected target when the pattern occurs. This function is also used to maintain the membership of the sensor group around the tracked target when the target moves. Group membership, in this case, is restricted to those nodes that sense the given target (i.e., for which $sense_e()$ is true).

Second, the programmer declares what constitutes the environmental state to be encapsulated in the context label. This

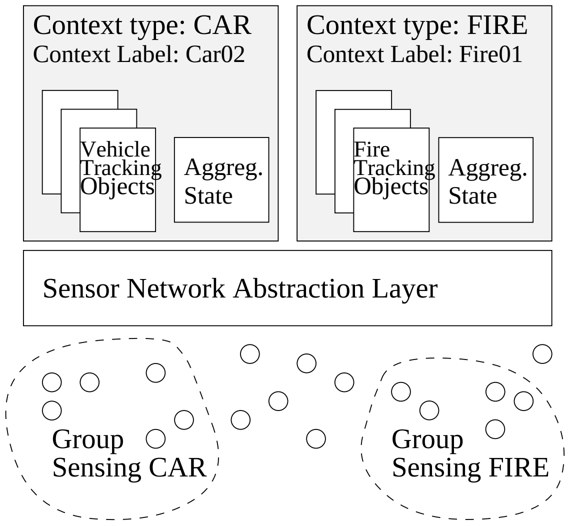


Figure 1. Programming Model

state is shared by all tracking objects attached to this label. State is declared by defining an aggregation function $state_e()$ that acts on the readings of all sensors for which $sense_e()$ is true or was true within a recent past defined by a freshness constraint. The aggregation is carried out locally by a sensor node that acts as the group leader of all sensors sensing the named target. The aggregation function can also include a critical mass constraint that specifies the minimum number of sensors that must be involved in the aggregation for the result to be statistically meaningful. EnviroTrack provides a library of the most common distributed aggregation functions to choose from, such as addition, averaging, and median computation. These functions can also be location-aware, for example, to compute the center of gravity of the measurements. The underlying infrastructure includes a data collection protocol executed by the leader to collect, timestamp, and log sensor data (i.e., the arguments for the $state_e()$ function) from sensor group members satisfying $sense_e()$. The $state_e()$ function is then applied on the collected data in a way that satisfies the freshness and critical mass conditions. Finally, the programmer specifies which objects are to be attached to the context label. Attached object code can reference the aggregate state maintained by the leader in this context.

In the following, we describe in more detail the network abstraction layer, tracking objects, and aggregate state.

2.1 Network Abstraction Layer

Context labels abstract sensor groups for the programmer. The programmer is aware that a distributed computation, associated with the context label, is executed on multiple sensors in the vicinity of a tracked entity. The programmer, however, is not involved in managing the membership, leader election, and leader handoff in the sensor group.

A sensor node joins the sensor group of a particular context when its local sensor readings satisfy the boolean condition $sense_e()$. It leaves the group when this condition is no longer satisfied.[1] A sensor node can be part of multiple groups at one time. Programs running for different groups are effectively independent. The sensor group associated with a context label maintains two invariants. First, all members of a group at time *t* satisfy the condition $sense_e()$. Second, the group is not partitioned. All members of a sensor group can communicate with each other possibly using multiple hops through other members of the same group. This physical continuity constraint is introduced to ensure that groups formed around different entities of the same type remain distinct and do not merge as long as the tracked entities are physically separated.

2.2 Tracking Objects

The tracking objects attached to a context label consist of methods that are invoked either by the passage of time (time-triggered), or by the arrival of messages that carry method invocation requests. Object code is executed on a single node. In the current implementation, this node is the sensor group leader of the enclosing context. Object code may make references to the aggregate state maintained by the enclosing context, returned by the $state_e()$ function. This state is collected by a distributed data collection protocol which constitutes the distributed part of the objects' computation. Note that the code is independent of the number and identity of participants of the distributed data collection protocol. It can assume, however, that the aggregation results always satisfy the semantics of aggregate state (i.e., they are in accordance with the specified freshness and critical mass requirements).

2.3 Aggregate State

The function $state_e()$ is configured by declaring aggregate state variables for context e. The definition of a state variable in the enclosing context specifies three important pieces of information:

- Aggregation function. Aggregation functions produce scalar values from sets of sensor readings. Several aggregation functions are provided in a library that can be extended by the programmer.

- Freshness L_e. The freshness threshold tells the system how long sensor readings can be used before they are considered stale. Only readings taken within the prescribed freshness time are used to compute the value of an aggregate state variable.

- Critical mass N_e. The critical mass is an integer that denotes the minimum number of sensor nodes that should be involved in the aggregation for the returned value to be valid. Only readings produced within the freshness threshold can contribute to the critical mass threshold.

Since freshness is decided at configuration time, nodes that join the group associated with a particular context label periodically send to the leader their measurements at a period $P_e = L_e - d$, where d is an estimate of maximum message delay and processing time within the group. This ensures that the results of aggregation are always based on sensor readings that are not older than L_e. The leader maintains approximate aggregate state by performing the aggregation function periodically on all the messages received within a sliding window of P_e time units. The state is tagged valid (using a *valid* flag) if more than N_e messages were received within the window. The application code running on the leader, can perform asynchronous *read* operations on aggregate state variables, which return their current value and validity status.

Figure 2 shows the overall internal structure of the middleware, illustrating both member and leader code. As seen in figure, the main function of members is to report their readings periodically to the group leader. The leader computes the aggregate state and runs the application, which may communicate with remote contexts using a message transport protocol. A distributed group management protocol keeps track of group membership and leader election. Observe that each sensor node has both member and leader code. The role taken by the node is chosen by the group management protocol.

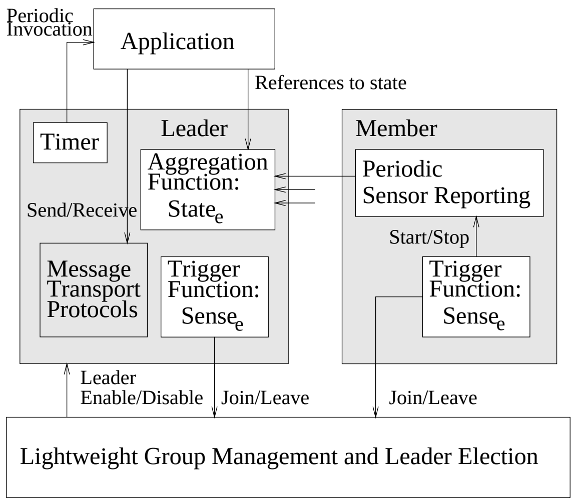

Periodic Invocation
Application
References to state
Leader
Timer
Aggregation Function: State$_e$
Send/Receive
Message Transport Protocols
Trigger Function: Sense$_e$
Member
Periodic Sensor Reporting
Start/Stop
Trigger Function: Sense$_e$
Leader Enable/Disable
Join/Leave
Join/Leave
Lightweight Group Management and Leader Election

Figure 2. Middleware Architecture

3 Language Features and Application Example

To facilitate the use of our middleware, we developed simple language support for declaring context labels and aggregate

[1]Alternatively, a separate deactivation condition may be written.

```
(1)  begin context tracker
(2)    activation: magnetic_sensor_reading()
(3)    location : avg (position) confidence=2, freshness=1s
(4)
(5)    begin object reporter
(6)      invocation: TIMER(5s)
(7)      report_function() {
(8)        MySend (pursuer, self.label, location);
(9)      }
(10)   end
(11) end context
```

Figure 3. Sample EnviroTrack Code

state variables. A preprocessor uses the stated declaration to emit appropriate code that initializes the middleware and configures the $state_e()$ and $sense_e()$ functions. The preprocessor then configures the trigger conditions for membership in particular contexts, and replaces references to the aggregate state variables by middleware function calls that evaluate and return them at runtime.

An EnviroTrack program consists of a list of context declarations such as the one shown in Figure 3. Each context declaration includes an *activation* statement specifying the $sense_e()$ condition for creating new instances of the declared context type. The activation statement is followed by aggregate state declaration for the created context. This declaration consists of a list of variables, each with its own freshness and critical mass constraints. The declared aggregate state variables are computed for the context at run-time as described in Section 2.3. This computation is performed independently of application code. Finally a list of objects is attached. Each object may have NesC functions with optional *invocation* conditions. Invocation conditions may be written in terms of aggregate state variables defined in the enclosing context. They state when the particular method is to be invoked. All static objects are declared separately within the *default* context type.

We illustrate our programming syntax by an application example. A typical sensor network application is one in which a dense network of motes is deployed to track the location of moving vehicles. For simplicity of illustration, we assume that the presence of the vehicle is determined using a magnetic sensor. In our application, sensors that detect magnetic distortion caused by the vehicle form a group abstracted by a context label. Note that several context labels may be instantiated, depending on the number of vehicles sensed. In each context label, the attached object periodically reports the vehicle's location to a preselected mote interfaced to a mobile pursuer. The pursuer (a laptop) monitors all vehicles at all times and records their tracks. The program in Figure 3 shows how the vehicle-tracking context is defined. Pursuer code is not shown.

The example in the figure defines a context of type *tracker*. Line 2 specifies that the activation condition, $sense_e()$, for this context type is encoded by the boolean function *magnetic_sensor_reading()*. This function is written in NesC. It returns a true value when a vehicle is detected. Line 3 defines one aggregate variable, namely, the average position $location$. It specifies that the value of $location$ returned upon a reference must represent the average of at least 2 sensor node readings measured no earlier than 1 second ago. Hence, $N_e = 2$ and $L_e = 1$.

Lines 5-10 describe the attached computation. Line 6 specifies when the computation is invoked. It dictates that the report function be invoked periodically with a period of 5 seconds. This is followed by the code of the function. This code simply makes a call to *MySend()* which in turn calls the routing layer to send the message to the pursuer. Two parameters are passed in the message, a handle of the originating context label obtained using $self.label$ and the aggregate state variable $location$ indicating the average position of all sensors currently detecting the reported vehicle (i.e., the estimated position of the vehicle).

The above code will generate multiple instances of the tracker if multiple vehicles are present. Further, even though the vehicles move and the sensor nodes comprising their corresponding contexts will change, the context labels will not. This significantly simplifies the programmer's interaction with the varying sensor group tracking each vehicle.

4 Implementation

In this section, we describe implementation issues in EnviroTrack. Our implementation is built on TinyOS [15], an operating system kernel developed exclusively for sensor nodes. TinyOS provides support for communication, multitasking, and code modularity. Geared towards communication-intensive applications, it exports the abstraction of components, which can be integrated into structures similar to a protocol graph. Each component consists of command handlers, event handlers and simple tasks. Communication protocols can be constructed easily in a modular manner by developing the appropriate handlers independently of others. The implementation of the EnviroTrack programming system consists of the following main modules:

- The EnviroTrack preprocessor: This preprocessor translates EnviroTrack declarations such as the one shown in Section 3 into NesC code which calls run-time libraries implementing group management, data aggregation and communication.
- The group management protocol: This protocol maintains the membership of the sensor group associated with a single context label.
- Routing services: These services implement a communication protocol between different context labels.

These modules are described next.

4.1 The EnviroTrack Preprocessor

The input to the EnviroTrack preprocessor is the context description file, such as the one shown in Section 3. The preprocessor patches a set of NesC program templates using the information gathered from the context description file to produce the target NesC modules such as those implementing the $sense_e()$ and $state_e()$ functions. The programs are then compiled using the provided TinyOS development tools.

The outer loop of our TinyOS program template code is implemented as a timer handler. This handler is invoked on the sensor group leader periodically and executes one iteration per invocation. The handler maintains an array of contexts. Each entry represents one context and provides access (via function pointers) to that context's activation condition, $sense_e()$, and object code, as well as its status. The generic handler in the template simply goes through this array checking if any context satisfies the activation condition. The compiler emits an `initContextStructures()` function that sets up this array based on the context description file. At run-time, sensor devices remain in this time-triggered mode until an appropriate condition is sensed. Activation conditions of different contexts are expressed in terms of boolean NesC functions which access local sensory measurements. These functions are sensor dependent. They can be written by the developer or chosen from a common library.

When an activation condition, $sense_e()$ is satisfied for a context of type e, group management services are activated on the motes sensing that condition. The execution of these services creates a context label (of type e) and maintains its approximate aggregate state, $state_e()$, on the current group leader. Subsequent invocations of the timer handler check for method invocation conditions defined in terms of this aggregate state, and post TinyOS tasks to execute methods whose invocation conditions are satisfied.

In the current implementation, objects are permanently attached to contexts. Each of the methods attached to a context is emitted with their names mangled (by adding the context name). The contents of each function are also parsed to replace references to aggregate variables with function calls that return the aggregate variable's value in accordance with its specified tracking QoS. Every possible aggregation for every sensor value is available as a function call. The naming of these functions is done based on a known scheme so as to allow the compiler to generate the correct call. Each aggregate variable is associated with attributes of freshness and critical mass. The functions (that return aggregate values) themselves are patched with the right value of freshness and confidence to produce the specified QoS.

4.2 Group Management Services

Group management services, shown at the bottom of Figure 2 maintain coherence of context labels. That is, they ensure that a group of sensors identifying the *same entity* in the environment produce a *single* context label. This label must persist and remain unique even as the membership of this sensor group changes. Ideally, to maintain context label coherence, at any point in time, nodes sensing the same external entity maintain a single "majority" leader.

Contexts are created when a node first senses condition $sense_e()$. The node immediately starts a leader election process in which it randomly chooses a small timeout value. A node which times out first sends a message informing its neighbors that it is leader. Upon receipt of this message, other nodes sensing the same $sense_e()$ condition become members. We require that a node's communication radius be larger than twice its sensing radius such that all nodes sensing the same target are within each other's communication range.

An elected group leader sends periodic heartbeats, which are received by all group members. Leader heartbeats have three purposes. First, they inform current members that the leader is alive. Should the leader die, a new leader election is started after a timeout. Second, they carry application state that must persist across leader handoffs. This state is recorded by all member nodes. This mechanism allows new leaders to continue computations of failed leaders from the last state received. An application can explicitly create persistent state using a $setState()$ primitive and read it using $getState()$. Finally, heartbeats are overheard past the group's perimeter thus informing neighboring nodes of the existence of context label e. Nodes that cannot sense the target themselves but know of its existence from nearby leader heartbeats are called *group followers*. If these nodes subsequently sense the condition $sense_e()$, they join the present group instead of forming a new context label. The mechanism ensures that multiple spurious context labels do not emerge around the same target. When the leader gets out of sensory range from the target, it sends a leader handoff message which initiates a new leader election. The resulting behavior is that a group with a unique leader is created around each target. Membership changes and leader (and state) handoffs occur automatically as the target moves.

A detailed simulation study of the above protocol appeared in [4] in which particular attention was paid to various failure and message loss scenarios that result in election of spurious leaders. It was shown that while spurious leaders do emerge, very simple techniques can substantially reduce their effect on system behavior. For example, in the presence of message loss, a leader handoff may produce two nodes both of whom claim to be leaders of the same context label. However, since these nodes are within each other's communication range, the one with the higher node identifier can eventually force the other to relinquish leadership. The same applies if a node elects itself as leader of a new context label for a target that is already being tracked by another. The effect of such spurious context labels is reduced by letting nodes that hear two nearby leaders ignore the one with the smaller *weight*. Each new context label is initially created with a leader weight of zero. Leaders of existing context labels accrue a weight equal to the number of messages received by the leader from members to date. This weight is passed during leadership handoffs. Hence, leaders of spurious context labels will generally be ignored. Consequently, the abstraction of a single context label per target is adequately maintained.

The mechanism described above opens several important questions for future research. One is what do when multiple targets cross paths. In the present scheme a violation of context label coherence may occur. For example, the "younger" context label may disintegrate (be absorbed in the group of the "older") and later emerge as a different label when the targets separate. Such anomalies should be dealt with at the application layer. It may be impossible to solve them in middleware without complex signal processing as it may be impossible, say, for a magnetic sensor to identify which of two nearby targets is responsible for its magnetic reading. From the application's perspective, the sensor network has a notion of granularity which defines the resolution of target detection and is related to the communication radius of nodes. If multiple targets fall within the same granule, they become indistinguishable. When they separate, they again become distinct targets.

4.3 Routing Services

To route among different context labels, we use an algorithm similar to landmark routing [22]. Nodes are assumed to know their location such that geographic routing can be used. Leaders of established context labels who wish to communicate broadcast their existence and report their location to a landmark. Other nodes route packets to the landmark, which in turn forwards them to the leader of the context label. Upon leader handoffs (the location of) the new leader is reported to the landmark. In addition, a forwarding pointer is inserted at the previous leader to forward packets that are in transit. On top of the routing layer a simple demultiplexor is implemented that dimultiplexes incoming messages at the destination and forwards them to one of several application modules. This allows implementing remote method invocation. The destination address of the remove method contains the name of the context label and the method identifier. The latter is used by the demultiplexor to identify the module implementing the needed method.

5 Performance

In this section, we evaluate the performance of an actual implementation of the presented tracking middleware service. The implementation is on MICA motes running TinyOS. While some simulation studies have been performed on the group management protocol [4] as mentioned in Section 4.2, this is the first detailed report on the performance of an actual implemented prototype of the complete service. In the context of performance evaluation, it is interesting to node that the programming interface imposed on top of our middleware does not interfere with its run-time performance. In fact, this interface was written by the authors after the tracking middleware was developed. It simply automates the process of configuring the middleware for tracking. Once the preprocessor has parsed the user's context declarations and emitted the configured code, the middleware looks the same as if it was hand-coded. No performance penalty is associated with the improved level of abstraction.

With the above observation in mind, we now present the experimental performance of tracking. We first establish a case for the viability of our middleware for tracking in practice. We then proceed with stress-testing EnviroTrack to explore the limitations of the current implemented prototype.

5.1 A Case for Tracking

Our case-study target is the T-72 tank (made in Russia), moving in an off-road sensor field. This particular tank weighs 44 tons and has a maximum off-road speed of around 45 km/hr [12]. Sensors in the field are equipped with magnetometers. Honeywell advertises magnetic traffic monitoring sensors which can detect moving vehicles from a range of up to 30 meters [20]. These sensors operate by detecting slight disturbances to the Earth magnetic field caused by ferrous objects. The magnitude of this disturbance depends on the amount of the ferrous material in the tracked object. Since the T-72 tank weights about 40 times the average vehicle in ferrous matter, its presence could be detected at a much larger distance than 30 meters. Magnetic effects are attenuated with the cube of the distance. Hence, we set the magnetic detection radius for the tank to approximately $30 * 40^{1/3}$ which amounts to about 100 meters. It is easy to show geometrically that if the tank can be detected 100 meters away, it is guaranteed that it is always within range from at least one sensor as long as sensors are put on a grid about 140 meters apart. We thus assume a rectangular grid of sensors with a per-hop distance of 140 meters. Note that covering a border area of say 70 km x 5 km at this spacing would require roughly 18,000 sensor devices, which is about the right size for the envisioned sensor networks. Moving at its maximum speed, a T-72 tank will cover one hop every 11.2 seconds.

We developed a testbed which provides a scaled down, 1000:1, model of this scenario. To experiment with variable sensor range more readily, we replaced magnetic sensors with light sensors installed on MICA motes. The magnetic field of the target was emulated by moving a round object of a corresponding radius above the sensor field to block a strong light source from the appropriate sensors. The field was arranged into a rectangular grid. In our first experiment, the tracked object was moved at a speed of 10 seconds/hop and 15 seconds/hop, which corresponds to an emulated speed of 50 km/hr and 33 km/hr, respectively. A single context type was defined, whose declaration is similar to Figure 3. At run-time a context label was generated. Group management maintained a leader for the context label. The leader sent to a base station the average position reported by nodes sensing the target at the current time. After each run, logs on individual motes were inspected to produce message loss and total throughput statistics. Message loss was computed by counting the number of messages sent but never received on any other mote.

Figure 4 shows the real and tracked object trajectory (reported to the base station) in a representative run. The motes were put at integer (x, y) coordinates. The horizontal line at $y = 0.5$ is the real target trajectory. The tracking error occurs because our sensors have no notion of proximity to the target. Moreover, direction anomalies occur due to message loss which causes sensor position aggregation to use a subset of reporting sensors only. An application receiving this trajectory can presumably improve the results by applying filtering to the reported raw data. Results could be further improved if sensor nodes could perform ranging to estimate target proximity.

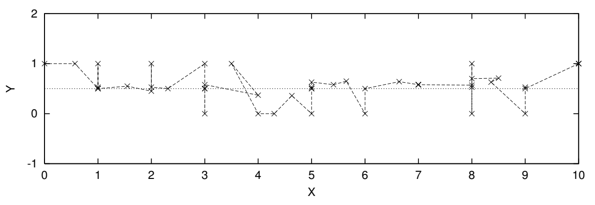


Figure 4. Tracked Tank Trajectory

Figure 5 shows the percentage of successful context label handovers for two target speeds and two settings of group management parameters. A successful handover means that the context label successfully follows tank location by virtue of leadership handoff from one member node to another along the target's path. An unsuccessful handover means a different context label is spawned at the new tank's location, not realizing that it refers to the same tank as the current context label. This case violates context label coherence.

In the first group management parameter setting, leader heartbeats are not propagated past the sensing radius. As expected, in this case it is more likely that multiple context labels are generated for the same target since nodes which sense the target for the first time might not be aware of the existing context label. Figure 5 shows that a fraction of handovers will fail in this case unless target speed is slow. In the second setting, the sensing and communication ranges are such that leader heartbeats are propagated beyond the sensing radius. In this case, all handovers are successful at both emulated tank speeds. This is in agreement with expectations since the group management algorithm in Section 4.2 requires that the communication range be larger than the sensing range. The experiment demonstrates the importance of setting these ranges correctly not to violate the group management assumptions.

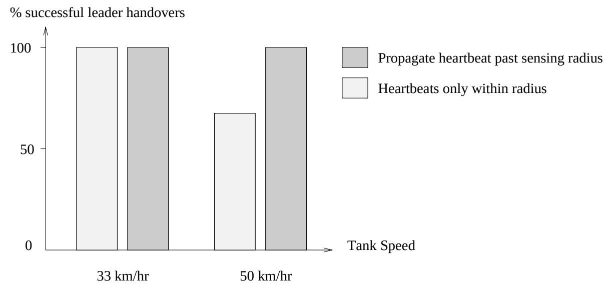


Figure 5. Successful Handovers

Finally, Table 1 shows sample communication data collected during our experiments for the second (correct) case above. Each point is averaged over three independent runs. In particular, we show the measured percentage of lost leader heartbeats (HB loss), lost sensor messages incurred during data aggregation (Msg loss), and the average useful link utilization (Link Util). To compute the latter, we divided the total number of bits sent per second by the total link capacity (50kbs for MICA motes). Hence, this is a worst case estimate, since it assumes a broadcast model in which no two messages could be sent concurrently.

The table demonstrates four important points. First, our system operates correctly in the presence of message loss, which is necessary in sensor network applications. Second, message loss is not caused by link utilization, but rather by the unreliability of the wireless medium (no reliability is implemented in the MAC layer of the MICA motes). Note that the effect of collisions increases with target speed. Third, our communication requirements constitute only a tiny fraction of available link capacity. Hence, we have not yet stressed the limits of the system's capabilities. Fourth, link utilization increases only slightly with tank speed. Hence, the bandwidth requirements of the algorithm have potential to scale well with tracking difficulty.

Speed	**% HB loss**	**% Msg loss**	**% Link Util**
33 km/hr	7.08	3.05	2.54
50 km/hr	22.69	17.05	2.88

Table 1. Communication Performance Data

The aforementioned proof-of-concept results show that the severe limitations on the memory, CPU, and network bandwidth of the MICA motes do not prevent them from performing communication protocol stack processing, group management, leader handoff, and aggregate state computation associated with maintaining our context label abstraction. Moreover, with appropriate sensor selection and parameter settings, realistic targets can be successfully tracked. Next, we stress-test the architecture to determine the maximum trackable target speed as a function of various parameter settings of the middleware.

5.2 Testing the Maximum Trackable Speed

The maximum trackable speed refers to the maximum speed a target can have without causing violations of context label coherence. If a target moves too fast it can be detected by nodes who have not yet heard of it, which results in creation of spurious context labels. The most important parameter which affects the maximum trackable target speed in our architecture is the heartbeat period of the group leader. In the experiments conducted, the timeout associated with failed leader detection (due to absence of heartbeats) is set to 2.1 the heartbeat period. In other words, we wait for two consecutive missing heartbeats before initializing leader re-election.

The maximum trackable speed is computed for the worst-case scenario, which is the case when the current leader fails causing leadership takeover to take place. In this case, a slow heartbeat period will allow the target to escape tracking during the leadership takeover. Consequently, several disconnected groups will be formed (as the target is rediscovered independently at different points along its track). The maximum trackable speed (the highest target speed at which the single group abstraction is maintained) observed in the experiment is shown in Figure 6 as a function of heartbeat period for two events: a narrow siganture event (outer bars), and a wide signature event (inner bars). The figure also shows the trackable speed during normal operation in which each leader willingly relinquishes leadership to another as the target moves out of its sensor range. This case is labeled “relinquish” in the figure and shows a maximum trackable speed that is independent of the heartbeat period.

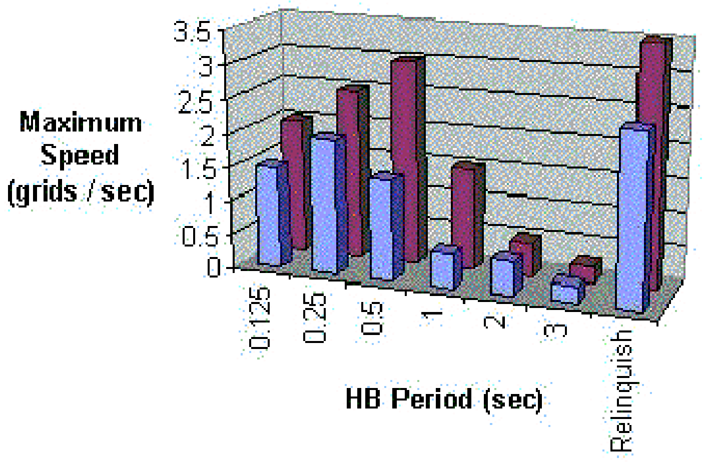


Figure 6. Effect of Timers on Maximum Trackable Speed

Several points can be made from this graph. First, for a large range of parameter settings, the maximum trackable speed is 1-3 hops/s, which is 10-30 times faster than the speed of the tank presented in the previous section. Thus, very fast targets can be tracked, or alternatively, sensors with a much smaller sensing radius can be successfully used to track realistic targets.

Second, we see that events with a larger sensory signature (expressed in figure in terms of multiples of average node separation, or *grids*) can be tracked at higher speeds. This may seem intuitive, as larger targets should be easier to track.

Third, we see that as the heartbeat period is reduced (sending out more frequent heartbeats) faster targets can be tracked. This is intuitive as faster heartbeat makes the group management mechanism more responsive. Realizing that heartbeats are bandwidth-consuming messages and that both CPU and communication bandwidth are limited in our experiments, we stress tested the heartbeat period to determine where overload occurs.

To determine the identity of the bottleneck resource that causes the decline in the maximum trackable speed at small heartbeat periods, we repeated the above experiment in the presence of a substantial amount of cross traffic. The cross traffic was exchanged between motes that do not participate in the EnviroTrack protocol but rather generate “background noise”. The shape of Figure 6 in the presence of this cross traffic remained largely unaffected. We therefore conclude that communication bandwidth is not the bottleneck. The bottleneck appears to lie in CPU processing.

In our next experiment, we test the effect of varying the ratio between the communication radius (CR) and the sensing radius (SR) on the trackable target speed. We use explicit leadership handoffs in this experiment (as opposed to handoffs due to leader failures). The results are shown in Figure 7. From this figure, the most important point to note is that for a given CR:SR ratio (which may or may not be a controllable parameter by system designers), larger events are trackable at faster speeds. The direct cause of this is the number of leadership handovers that occur. For a constant speed, when an event is larger, the average time between handovers decreases (as a single leader can sense the target longer) requiring fewer messages to be processed. The lower communication overhead results in a higher trackable speeds. The other point to note is that our tracking architecture breaks down when the CR:SR ratio falls below 1. This occurs because nodes outside of communication range from the leader also sense the event and concurrently form spurious groups thus violating context label coherence. The performance improves as the ratio increases as two nodes that sense the same target are less likely to be outside each other’s range.

6 Related Work

A growing challenge facing the distributed systems community is to develop programming paradigms and run-time sup-

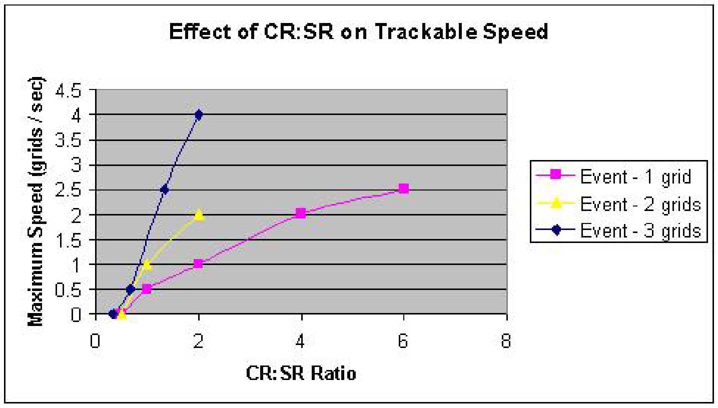


Figure 7. Effect of Sensory Radius on Maximum Trackable Speed

port for the operation of large-scale embedded sensor networks. Classical distributed programming paradigms and middleware such as CORBA [28], group communication [8], remote procedure calls [3], and distributed shared memory [6, 25] share in common the fact that their programming abstractions exist in a logical space that does not represent or interact with objects and activities in the physical world. Their main goal is to abstract distributed communication rather than facilitate distributed sensory interactions with an external physical environment. In contrast, a new paradigm tailored for sensor should be centered around environmentally-driven abstractions aimed at simplifying the coding of interactions with the physical world that arise in distributed deeply embedded systems.

The work reported in this paper is related to several recent projects, such as Cricket [23], Sentient Computing [1] and Cooltown [9], that propose high-level paradigms in which an embedded distributed computing system is able to share perceptions of the physical world. These systems allow the location of entities in the external environment to be tracked. One major difference of these systems from EnviroTrack is that they assume cooperative users who, for example, can wear beaconing devices that interact with location services in the infrastructure for the purposes of localization and tracking [23, 1]. Our interest, in contrast, is in situations where no cooperation is assumed from the tracked entity.

In the absence of cooperation, several research efforts proposed alternative addressing schemes that do not rely on having destinations with specific identities, but rather contact sensor nodes in the vicinity of a phenomenon of interest based on the attributes of data they sense. For example, DataSpace [17] exports abstractions of physical volumes addressable by their locations. Similarly, directed diffusion [18, 14] and the intentional naming system [2] provide addressing and routing based on data interests [18, 14]. Attributed-based naming is also related to the notion of content-addressable networks [24] proposed for an Internet environment, which allows queries to be routed depending on the requested content rather than on the identity of the target machine. We adopt a form of attribute-based naming we call *context labels*. In our architecture, however, context labels are *active* elements. Not only do they provide a mechanism for *addressing* nodes that sense specific environmental conditions, but also they can *host context-specific computation* that tracks the target entity in the environment.

Recent research on system software for sensor networks has seen the introduction of distributed virtual machines designed to provide convenient high-level abstractions to application programmers, while implementing low-level distributed protocols transparently in an efficient manner [27]. This approach is taken in MagnetOS [11], which exports the illusion of a single Java virtual machine on top of a distributed sensor network. The application programmer writes a single Java program. The run-time system is responsible for code partitioning, placement, and automatic migration such that total energy consumption is minimized. Maté [19] is another example of a virtual machine developed for sensor networks. It implements its own bytecode interpreter, built on top of TinyOS. The interpreter provides high-level instructions (such as an atomic message send) which the machine can interpret and execute. Each virtual machine instruction executes in its own TinyOS task.

A somewhat different approach of providing high-level programming abstractions is to view the sensor network as a distributed database, in which sensors produce series of data values and signal processing functions generate abstract data types. The database management engine replaces the virtual machine in that it accepts a query language that allows applications to perform arbitrarily complex monitoring functions. This approach is implemented in the COUGAR sensor network database [5]. A middleware implementation of the same general abstraction is also found in SINA [26], a sensor information networking architecture that abstracts the sensor network into a collection of distributed objects.

Our system is different in that it is geared for environmental tracking applications. To the authors' knowledge, EnviroTrack is the first programming support for sensor networks that explicitly facilitates the coding of tracking applications. Its novel abstractions and underlying mechanisms are well-suited for monitoring targets that move in the physical world. EnviroTrack therefore can have a major impact on application development for sensor networks.

7 Conclusions

This paper introduced the design, implementation, and experimental evaluation of a new distributed programming paradigm and experimental prototype for sensor network applications. The paradigm differs from existing distributed computing models in its central focus on abstracting interactions with a *physical environment* produced by a large array of distributed sensors and actuators. The key advantage of this paradigm lies in its considerable potential to reduce development costs of deeply embedded systems. This reduction comes from offloading from the application developer the details of managing low-level communication, mobility, and group management issues in groups of redundant sensor nodes in tracking applications. Performance results show that in addition to convenient abstractions, efficient implementation is possible in our architecture, in that target tracking is successful at practical target speeds.

This paper might be a first step towards a predictable sensor network "virtual machine" for writing distributed deeply-embedded applications. Such a layer should export reliable behavior and well-defined semantics, implemented on an unreliable, unpredictable, and resource constrained hardware and communication infrastructure. The virtual machine would hide the complexity of sensor network programming from the application developer, making a new more robust and more dynamic realm of sensor network applications attainable to impact future defense, surveillance, habitat monitoring, and disaster management systems.

References

[1] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper. Implementing a sentient computing system. *IEEE Computer*, 34(8):50–56, August 2001.

[2] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The design and implementation of an intentional naming system. In *ACM Symposium on Operating Systems Principles*, Kiawah Island, SC, December 1999.

[3] A. Birrel and B. Nelson. Implementing remote procedure calls. *ACM Transactions on Computer Systems*, 2(1), February 1984.

[4] B. Blum, P. Nagaraddi, A. Wood, T. Abdelzaher, S. Son, and J. Stankovic. An entity maintenance and connection service for sensor networks. In *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, San Francisco, CA, May 2003.

[5] P. Bonnet, J. Gehrke, and P. Seshardi. Towards sensor database systems. In *2nd International Conference on Mobile Data Management*, pages 3–14, Hong Kong, January 2001.

[6] J. Carter, J. Bennet, and W. Zwaenepoel. Implementation and performance of munin. In *ACM Symposium on Operating Systems Principles*, pages 151–164, October 1991.

[7] A. Cerp, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communication technology. In *ACM Sigcomm Workshop on Data Communication*, San Jose, Costa Rica, April 2001.

[8] G. V. Chockler, I. Keidar, and R. Vitenberg. Group communication specifications: A comprehensive study. *ACM Computing Surveys*, 33(4):427–469, December 2001.

[9] P. Debaty and D. Caswell. Uniform web presence architecture for people, places, and things. *IEEE Personal Communications*, 8(4):46–51, August 2001.

[10] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Mobile networking for smart dust. In *ACM MOBICOM*, Seattle, WA, August 1999.

[11] R. B. et al. On the need for system-level support for ad hoc and sensor networks. *Operating System Review*, 36(2):1–5, April 2002.

[12] Federation of American Scientists Military Analysis Network. http://www.fas.org/man/dod-101/sys/land/row/t72tank.htm.

[13] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to network embedded systems. In *ACM SIGPLAN Conference on Programming Language Design and Implementation*, San Diego, CA, June 2003.

[14] J. Heideman, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. *Operating Systems Review*, 35(5):146–159, December 2001.

[15] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. In *ASPLOS*, Cambridge, MA, November 2000.

[16] M. Horton, D. Culler, K. Pister, J. Hill, R. Szewczyk, and A. Woo. Mica: The commercialization of microsensor motes. *Sensors Online*, 19(4), April 2002. http://www.sensorsmag.com/articles/0402/index.htm.

[17] T. Imielinski and S. Goel. Dataspace - querying and monitoring deeply networked collections in physical space. *IEEE Personal Communications*, 7(5):4–9, October 2000.

[18] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *ACM MOBICOM*, Boston, Massachusetts, August 2000.

[19] P. Levis and D. Culler. Mate: A tiny virtual machine for sensor networks. In *ASPLOS*, San Jose, CA, October 2002.

[20] Magnetic Sensors. http://www.magneticsensors.com/mark_det.html.

[21] A. Mainwaring, J. Polastre, R. Szewczyk, and D. Culler. Wireless sensor networks for habitat monitoring. In *First ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, September 2002.

[22] G. Pei, M. Gerla, and X. Hong. Lanmar: Landmark routing for large scale wireless ad hoc networks with group mobility. In *MobiHoc*, Boston, Massachusetts, August 2000.

[23] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *ACM MOBICOM*, Boston, MA, August 2000.

[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Sigcomm*, San Diego, CA, August 2001.

[25] D. J. Scales and K. Gharachorloo. Towards transparent and efficient software distributed shared memory. In *ACM Symposium on Operating System Principles*, Saint Malo, France, October 1997.

[26] C.-C. Shen, C. Srisathapornphat, and C. Jaikeo. Sensor information networking architecture and applications. *IEEE Personal Communications*, 8(4):52–59, August 2001.

[27] E. Sirer, R. Grimm, A. Gregory, and B. Bershad. 'design and implementation of a distributed virtual machine for networked computers. In *ACM Symposium on Operating System Principles*, pages 202–216, Kiawah Island, SC, December 1999.

[28] S. Vinoski. Corba: Integrating diverse applications within distributed heterogeneous environments. *IEEE Communications Magazine*, 14(2), February 1997.

[29] A. Wood and J. A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10), October 2002.

RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks

Chenyang Lu Brian M. Blum Tarek F. Abdelzaher John A. Stankovic Tian He

Department of Computer Science
University of Virginia
Charlottesville, VA 22903
{chenyang, bmb5v, zaher, stankovic, th7c}@cs.virginia.edu

Abstract

Large-scale wireless sensor networks represent a new generation of real-time embedded systems with significantly different communication constraints from traditional networked systems. This paper presents RAP, a new real-time communication architecture for large-scale sensor networks. RAP provides convenient, high-level query and event services for distributed micro-sensing applications. Novel location-addressed communication models are supported by a scalable and light-weight network stack. We present and evaluate a new packet scheduling policy called velocity monotonic scheduling that inherently accounts for both time and distance constraints. We show that this policy is particularly suitable for communication scheduling in sensor networks in which a large number of wireless devices are seamlessly integrated into a physical space to perform real-time monitoring and control. Detailed simulations of representative sensor network environments demonstrate that RAP significantly reduces the end-to-end deadline miss ratio in the sensor network.

1. Introduction

With the advances in MEMS devices and embedded processors and radios, it will soon be feasible to deploy large-scale sensor networks to perform distributed micro-sensing and control of physical environments [11]. For example, a surveillance system may use a large network of acoustic sensors to detect and track vehicles in a security area. Similarly, biometric sensors can be deployed in airports to detect harmful bio-agents and issue alarms to command and control centers during potential bio-attacks. These smart sensors and actuators are equipped with low-power processors and short-range radio transceivers [9]. They will automatically form multi-hop ad hoc networks to communicate both among themselves and to remote base stations (e.g., PDA's).

Because distributed micro-sensing involves direct interaction with a physical environment, data communication in sensor networks often has timing constraints in the form of *end-to-end deadlines*. Surveillance may require the position of an intruder be reported to a command center within 15 sec so that pursuing actions can be initiated in time. Data in a system may have different deadlines due to different validity intervals. The validity intervals (and hence, update deadlines) of the locations of different intruders such as pedestrians and motor vehicles may depend on their movement speeds. For example, locations of tanks have shorter update deadlines than those of pedestrians. Similarly, the location of an intruder should have a shorter update deadline than the temperature measurement of a region because the former can change faster than the latter. Sensor network protocols should support *real-time* communication by minimizing the packet *deadline miss ratio*, i.e., the percentage of packets that meet their end-to-end deadlines.

While sensor networks share the notion of timing constraints with more traditional embedded systems, they differ in two respects. First, individual sensors are typically very small in size and resource capacity. Hence, the philosophy of sensor networks relies on resource dedication rather than sharing. In other words, individual sensor devices and nodes are likely to be dedicated for individual tasks, thereby eliminating much of the need for sophisticated CPU scheduling in a multitask environment.

Second, it is envisioned that sensor nodes will operate in groups, since individual nodes are too limited and unreliable to perform useful activities from the application's perspective. Group activities require coordination and communication among member nodes. Sensing results of groups need to be sent back to base stations through multi-hop communication. Thus, the main schedulable resource becomes the wireless communication channel. Progress of user-level activities and their ability to meet end-end deadlines are therefore determined by scheduling of the communication medium rather than scheduling of the processor. Towards that end, new real-time communication architectures are required for ad hoc wireless environments.

Multi-hop wireless communication scheduling differs from CPU scheduling in that it has an inherent notion of distance. In sensor networks, the distance is determined by the physical locations of source and destination. These locations impose distance constraints on messages, in addition to time constraints, calling for communication scheduling policies that are cognizant of both time and space.

The first contribution of this paper is RAP, a real-time communication architecture for large-scale wireless sensor networks. RAP provides a set of convenient, high-level query and event services to real-time distributed micro-sensing applications. Query and event services are based on novel location-addressed communication models supported by a scalable and light-weight network stack.

The second contribution of RAP is a novel *Velocity Monotonic Scheduling* (VMS) policy suitable for packet scheduling in sensor networks. VMS is based on a notion of packet *requested velocity*. Each packet is expected to make its end-to-end deadline if it can move toward the destination at its requested velocity, which reflects its local urgency. Compared with non-prioritized packet scheduling, VMS improves the deadline miss ratios of sensor networks by giving higher priority to packets with higher requested velocities. VMS can outperform deadline-based packet scheduling because velocity more accurately reflects the local urgency at each hop when packets with the same deadline have different distances to their destinations. Assuming that each sensor knows its own location (using GPS or other location services [8]), the requested velocity can be determined locally. This property enables VMS to scale well in large-scale sensor networks.

The final contribution of this paper is a detailed simulation study of the real-time performance of several routing protocols and packet scheduling algorithms in a typical sensor network scenario. Our simulation experiments demonstrate that, for sensors far away from their base station, RAP reduces the deadline miss ratio from 90.0% to 17.9%, compared to existing wireless communication protocols (DSR over 802.11b). To our best knowledge, ours is the first detailed performance study on deadline issues in multi-hop wireless sensor network settings under overload conditions.

In the following sections, we discuss the key characteristics of sensor networks, present the design of RAP, report a set of simulation experiments with sensor network configurations, and conclude the paper by summarizing our key results and future work.

2. Real-time Communication in Sensor Networks

In this section, we describe the characteristics of sensor networks and communication models on sensor networks. This analysis serves as a basis for our design of real-time communication protocols.

Sensor networks are an instance of mobile ad hoc networks (MANET) [10] that have recently attracted a lot of interest and visibility due to their flexibility and the feasibility of their deployment at low costs. In general, mobile ad hoc networks depend on peer-to-peer communication protocols that do not require a fixed infrastructure such as centralized servers and access points. Sensor networks are different from their traditional ad hoc wireless counterparts (composed of laptops and PDA's) in that they have a larger scale, higher density, smaller devices, and a tighter interaction with a physical environment. Energy conservation is critical in sensor networks because of their long lifetime and form-factor constraints, which preclude the use of large batteries or power supplies.

In most envisioned sensor network applications, a large number of sensors are deployed in an area and a small number of more powerful nodes (such as PDA's with Internet connections) form possibly mobile interface stations which serve as the entry points to the sensor network. In the following, we shall call such interface stations, *base stations*. A user may query the physical environment through such base stations. Alternatively she may also register for an *event*. The occurrence of the event will automatically trigger a specified query. A query can specify timing requirements including rate, start time, duration, and end-to-end deadlines. For example, a user can register for a virus_found event in a rectangular area with coordinates (10,10,20,20), and specify a query on the event to report the density of the detected virus. If a virus is found, the density of the viruses should be reported to the base station from where they are found every 1.5 sec for a duration of 30 min. Every reading should reach the base station within an end-to-end deadline of 5 sec.

Communication in a sensor network can be divided into two categories: *local coordination* and *sensor-base communication*. Before sending information to the base station, sensors within the local area coordinate among themselves to aggregate data and generate a reliable result. For example, acoustic sensors may need triangulation among multiple nodes to decide the location of a tank. Local coordination often occurs within a distance of one or a few radio radii. Sensor-base communication is responsible for reporting the aggregated data to the base station, which often spans many (e.g., tens of) hops. Consider a communication radius of 30 m of short-range radios transceivers, it is conceivable to have more than 10,000 nodes and tens of hops of communication in a coverage area of several square kilometers. Since sensor-base communication typically travels a much larger number of hops than local coordination messages, in this paper we focus on the former type of communication.

Unlike IP networks, sensor-base communication directly uses *location* as the target address. Instead of querying a sensor with ID 1002, a user or application queries a geographic region. The identities of sensors that happen to be located in that region are not important. Any sensors in that region that receive the query may initiate local coordination to aggregate the requested data. A leader may be elected to send the query result back to the base station. If continuous monitoring is required, the query may report the desired measurement periodically through the multi-hop ad hoc network. The base station can attach its location to the query message so that the query results can also be addressed by location (assuming no two base stations are at a same location).

Communication in sensor networks can suffer from "hot regions", i.e., areas where the network is seriously congested. Hot regions are often caused by numerous related events that synchronously trigger a large number of data flows toward the base station. Examples of related events include correlated measurement of the same environmental activity, or correlated environmental activities such as a group of new targets simultaneously entering a security area, or a bio-attack on a part of an airport. Maximizing the number of packets that make their deadlines in overload conditions is critical in sensor networks.

3. Design of RAP

We now present the design of RAP to support real-time communication in large-scale sensor networks. Given the unique characteristics of sensor networks, the goal of RAP includes the following:

- Provide general service APIs that are suitable for distributed micro-sensing and control in sensor networks
- Maximize the number of packets meeting their end-to-end deadlines
- Scale well with large number of nodes and hops
- Introduce minimum communication and processing overhead.

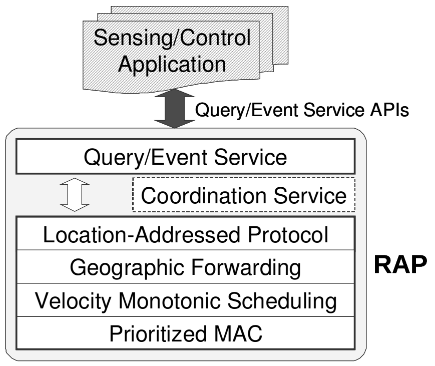


Figure 1 The RAP communication architecture

The architecture of RAP is shown in 1. Sensing and control applications interact with RAP through a set of Query/Event Service APIs. A Query/Event Service layer submits the query or event registration to an area. The Query/Event Service at the sensors in that area then (periodically or aperiodically) sends query results back to the base station. If an event is registered, the query is started only if the registered event happens. The sensor-base communication is supported by a network stack including a transport-layer Location-Addressed Protocol (LAP), a Geographic Forwarding (GF) routing protocol, a Velocity Monotonic (packet) Scheduling (VMS) layer, and a prioritized MAC. This network stack embodies a set of efficient and localized algorithms to reduce the end-to-end deadline miss ratio of sensor-base communication. This network stack is the focus of this paper.

The coordination service is responsible of dynamic group management and data aggregation among sensors (e.g., multiple sensors coordinate to determine the location of a target through triangulation). The coordination services are part of our on-going research and not addressed in this paper.

We now describe the Query/Event service APIs and the network protocol stack in detail in the following subsections.

3.1. Query/Event Service APIs

Applications may submit queries or register for events through a set of query/event service APIs. The API provides a high-level abstraction to applications by hiding the specific location and status of each individual node. These APIs allow applications to specify the timing constraints of queries. The underlying layers of RAP are responsible for orchestrating the sensing and communications of relevant sensors to accomplish all query and event services.

RAP provides the following query/event service APIs.

- **query{attribute_list, area, timing_constraints, querier_loc}**

Issue a query for a list of attributes in an area. A query has timing constraints. If a period is specified for a command, query results will be automatically sent from an area to the issuer of the query in every period. For example, the following query requires the average density of the viruses in an rectangular area (10,10,12,12) be reported to the base station of the querier every 1.5 sec. Every reading should reach the base station within an end-to-end deadline of 5 sec. The query includes the location of its base station so that query results can be sent back using LAP. In this paper we assume the location of the base stations are fixed.

```
query {
    virus.count,
    area=(10,10,12,12),
    period=1.5,deadline=5,
    base=(100,100)
}
```

- **register_event{event, area, query}**

Register for an event. A query is triggered once an event occurs. For example, the following API call registers a virus_count query for a virus_found event. If any viruses are found in a rectangular area with coordinates (0,0,100,100), returns the average density of the viruses of the 2×2 square area centered at the event location (X_{event},Y_{event}) every 1.5 sec. Every reading should reach the base station within an end-to-end deadline of 5 sec.

```
register_event {
    virusFound(0,0,100,100),
    query {
        virus.count,
        area=(X_event-1,Y_event-1,X_event+1,Y_event+1),
        period=1.5, deadline=5,
        base=(100,100)
};
```

A query or event is sent to every node in the specified area. Query results are sent back to the base station based on its location provided by the query or event registration.

3.2. Location-Addressed Protocol

LAP is a connectionless transport layer in the network stack. LAP is similar to UDP except that all messages are addressed by location instead of IP address. Three types of communication are supported by LAP: unicast, area multicast, and area anycast.

- *Unicast* delivers a message to a node that is closest to the destination location. Unicast can be used by sensors to send query results to base stations.
- *Area multicast* delivers a message to every node in a specified area. Area multicast can be used to register for an event or send a query to an area, for coordination among nodes in a local group.
- *Area anycast* delivers a message to at least one node in a specified area. Area anycast can also be used for sending a query to a node in an area. The node can initiate group formation and coordination in that area.

Since this paper is concerned with real-time issues in overload conditions, in the rest of this paper we focus on unicast from sensors to base stations because this form of communication contributes to most of the real-time traffic in sensor networks.

3.3. Geographic Forwarding

Since communication destinations are identified by geographic location, we assume the routing layer is aware of physical geography. A router can determine the physical location of the destination relative to itself and forward the packet in the general direction of the destination. Geographic forwarding (GF) [16] has been proposed in earlier wireless literature and evaluated in traditional MANET environments.

More precisely, GF makes a greedy decision to forward a packet to a neighbor if 1) it has the shortest geographic distance to the packet's destination among all immediate neighbors; and 2) it is closer to the destination than the forwarding node. When such nodes do not exist, the GPSR protocol [16] can be used to route packets around the perimeter of the void region. The only state on each node maintained by GF and GPSR is a table of the locations of immediate neighbors. Because GF uses immediate neighborhood information to make localized routing decisions, it is highly scalable with regard to the number of nodes, network diameter, and the rate of change in topology [16]. GF works best in sensor networks that usually have high node densities and support location-addressed communication. Location addressed communication means that GF can be used without a location directory service, which introduces extra management and communication overhead. High node density causes two desirable properties of GF in sensor networks. First, the greedy forwarding algorithm described above has a high success probability in finding a good path from source to destination resulting in efficient communication. Second, the number of hops is approximately proportional to the distance that a packet has to travel. Hence, the distance between a node and a packet's destination can serve as an indication of the packet's hop count.

3.4. Velocity Monotonic Scheduling

A key component of real-time communication architectures is the packet scheduling policy which determines the order in which incoming packets at a node are forwarded to an outgoing link. In existing ad hoc networks, packets are typically forwarded in FCFS order. FCFS scheduling does not work well in real-time networks where packets have different end-to-end deadlines and distance constraints. Instead, competing packets should be prioritized based on their local urgency. In the context of sensor networks, packet scheduling should be both *deadline-aware* and *distance-aware*. Deadline-aware means that a packet's priority should relate to its deadline. The shorter the deadline, the higher the packet priority. Distance-aware means that a packet's priority should relate to its distance from the destination. The longer the distance, the higher the packet priority.

An example is shown in Figure 2. In scenario 1, both sensors A and B send periodic flows to a base station C. Packets from A and B compete at nodes D, E, and F because of possible collision of transmissions from B, F and D. They should also be prioritized in the network-layer queues in node E. Similarly, in scenario 2 flows from A and B will compete at nodes E, F, G, and H. Assume that both flows share a same deadline in each graph, then A's packets should have higher priorities than B's packets because A's packets have to travel farther than packets from B, and therefore should move faster in the competing regions.

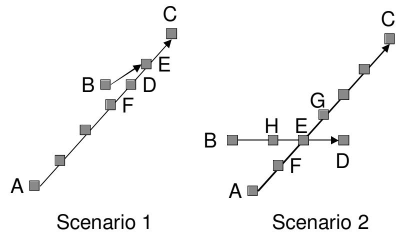


Figure 2 Scenarios of distance-aware scheduling

Since packet priority should be decided based on both distance and deadlines, we propose *Velocity Monotonic Scheduling* (VMS). VMS assigns the priority of a packet based on its *requested velocity*. A packet with a higher requested velocity is assigned a higher priority. VMS improves the number of packets that meet their deadlines because it assigns the "right" priorities to packets based on their different urgencies on the current hop. VMS also solves the fairness problem described in [18] in sensor networks because packets that are far away from the base station will tend to have higher priorities when it competes against other packets that are closer to the destination.

We investigate two priority assignment policies: Static Velocity Monotonic (SVM) and Dynamic Velocity Monotonic (DVM), depending on whether the requested velocity of a packet is updated dynamically in intermediate nodes.

3.4.1. Static Velocity Monotonic

SVM computes a fixed requested velocity at the sender of each packet. Assume a packet is sent from a sender at (x_0,y_0) to a destination at (x_d,y_d), and has an end-to-end deadline D sec, then SVM sets its requested velocity to:

$$V = dis(x_0,y_0,x_d,y_d)/D \quad (1)$$

where $dis(x_0,y_0,x_d,y_d)$ is the geographic distance between (x_0,y_0) and (x_d,y_d). The requested velocity of a packet is fixed on each hop.

3.4.2. Dynamic Velocity Monotonic

DVM *dynamically* re-calculates the requested velocity of a packet upon its arrival at each intermediate node. Assume a packet arrives at a node at location (x_i,y_i); its destination is at (x_d,y_d); it has an end-to-end deadline D sec, and its elapsed time, i.e., the time it has been in the network, is T_i sec; its requested velocity V_i at (x_i,y_i) is set to:

$$V_i = dis(x_i,y_i,x_d,y_d)/(D-T_i) \quad (2)$$

At the sender node (x_0,y_0), the elapsed time $T_0=0$ and the requested velocity is initialized to $V=dis(x_0,y_0,x_d,y_d)/D$. The requested velocity of a packet will be adjusted based on its actual progress (i.e., actual velocity). A packet's requested velocity increases by re-applying eq. 2 at subsequent nodes if its previous progress towards to the destination is slower (e.g., due to a hot region) than its previous requested velocity. On the other hand, its requested velocity decreases if it moves faster than its previous requested velocity. This is so this packet can give way to other more urgent packets.

Note that although clock synchronization simplifies its implementation, DVS can be implemented without clock synchronization. To do this, each packet contains a field as its elapse time counter. Each node increases the counter by the time the packet stays in it plus the transmission and propagation time.

3.4.3. Priority Queue

Each packet is assigned a priority based on its requested velocity and queued at the network layer when there are multiple outstanding packets. Several options are available for implementing priority queues. One approach is to insert all packets into a single queue ordered by priority. When the queue us full, higher priority incoming packets overwrite lower priority ones. The benefit of this solution is that it accurately reflects the order of requested velocities, and allows all packets to share the same buffer regardless of their priority. The approach, however, requires implementing a data structure whose insertion time, in the worst case, grows logarithmically in the number of packets.

To bound the queue insertion overhead, another approach currently used in our simulation is to maintain multiple FIFO queues each corresponding to a fixed priority level. Each priority corresponds to a range of requested velocities. A packet is first mapped to a priority, and then inserted into the FIFO queue that corresponds to its priority. This approach is more efficient because no ordering needs to be performed for every incoming packet. The per-packet overhead is logarithmic only in the number of priority levels, not the number packets. To further reduce overhead, after a packet has been inserted in a priority queue, its requested velocity and priority is not updated based on eq. 2 until it reaches the next node.

Assuming that packets that miss their deadlines are useless, priority queues actively drop packets that have missed their deadlines to avoid wasting bandwidth.

3.5. MAC-layer prioritization

Local prioritization at each individual node is not sufficient in wireless networks because packets from different senders can compete against each other for a shared radio communication channel. To enforce packet priorities, MAC protocols should provide *distributed prioritization* on packets from different nodes. Extensions (e.g., [1][15]) of the IEEE 802.11 wireless LAN protocol [18] have been investigated to provide distributed prioritization. Recently EDCF has been specified in the proposed 802.11e standard to provide different transmission priorities [19].

In this paper we implement two extensions proposed by Aad and Castelluccia [1]. We modified two components of the standard 802.11 implementation: the initial wait time after the channel becomes idle, and the backoff window increase function. These mechanisms are chosen because they introduce minimal overhead and can be ported to light-weight CSMA/CA protocols [23] that are more suitable to sensor networks than 802.11. We now briefly describe the mechanisms. The detailed description and analysis of these mechanisms are available in [1].

3.5.1. Initial Wait Time after Idle

802.11 sets a DIFS counter once the communication channel has become idle. Before sending an RTS (Request To Send) packet, a node will wait a random period of time between 0 and DIFS. To prioritize this process we set the DIFS parameter based on the packet priority:

DIFS = BASE_DIFS * PRIORITY

Packets with a higher priority (corresponding to a smaller PRIORITY value) on average choose a smaller waiting period.

3.5.2. Backoff Increase Function

802.11 doubles its backoff window, CW, to extend a node's waiting period when a transmission collision occurs. We modified 802.11 to increase CW in accordance with the packet priority[1]:

CW=CW*(2+(PRIORITY-1)/MAX_PRIORITY)

MAX_PRIORITY is the maximum value of priority (corresponding to the lowest priority). The backoff counter of a node with a pending lower priority packet increases faster than a node with a pending packet with a higher priority.

The above two mechanisms give high priority packets high probability to get the channel in both the contention avoidance and contention phases.

In summary, RAP integrates a set of light-weight protocols to satisfy the following key requirements of large-scale sensor networks.

[1] The backoff function is slightly changed from the original extension to mitigate its stability problem observed in [1].

- RAP provides general query and event service APIs as a convenient high-level service abstraction suited for distributed micro-sensing applications.
- RAP increases the number of packets meeting their end-to-end deadlines by prioritizing the transmission of contending packets based on their requested velocities.
- RAP scales well in large-scale sensor networks because it is composed of efficient and localized protocols and algorithms at every layer. The only states GF maintains are the locations of immediate neighbors. VMS determines a packet's priority only based on locally available information. No per-flow state is maintained inside the network.

4. Experimentation

We ran a set of simulation experiments to evaluate the aforementioned real-time packet scheduling and prioritization protocols on sensor networks for a biometric sensing application. We implemented GF, VMS, and the 802.11 extensions on the GloMoSim wireless network simulator [4] developed by UCLA.

4.1. Network configuration

We tuned the network parameters in reference of the Berkeley motes [9], a state-of-the art network sensors. We generated a square region of 136×136 m^2 divided into 100 13.6×13.6 m^2 grids. 100 nodes were simulated with one node randomly placed in each grid.

The other network parameters are listed as follows:

- Radio communication radius: 30.5 m
- Packet size: 32 - 160 B
- Bandwidth: 200 kbps. Current version of MICA motes available to us supports a bandwidth of 50kbps. Future versions are expected to have a higher capacity. Due to limitations of the GloMoSim simulator, we had to send data flows on top of the UDP/IP stack that contribute to 28 B overhead. In a real implementation we expect to eliminate the UDP/IP headers.

4.2. Application Workloads

We simulate a bio-sensor application that monitors viruses in an area. Users can register for events and query bio-sensors, which generate periodic data flows to a base station. Data flows have different rate and timing requirements. We assume that a base-station sends two different queries: count and detail to various locations.

Count:

```
registerEvent {
    virusFound(0,0,136,136),
    query {
        virus.count,
        area=(Xevent-1,Yevent-1,Xevent+1,Yevent+1),
        period=Pc, deadline=Dc
        base = (134.07, 128.06)
    };
};
```

Detail:

```
query {
    detail,
    area=(x-1,y-1,x+1,y+1),
    period=Pd, deadline=Dd
    base=(134.07, 128.06)
};
```

A user registers a *count* query with a virusFound event in the whole 136×136 m^2 squared area. A virusFound event is generated when a grid detects a specified virus at location (X_{event},Y_{event}). This event triggers a query virus_count, which periodically reports the density of the detected virus in the area (X_{event}-1,Y_{event}-1,X_{event}+1,Y_{event}+1) to the base station.

A user can also directly submit a *detail* query to get more detailed data collected at a location. *Detail* generates periodic flows that send detailed information about a grid to the base station for further analysis. While a large number of *count* flows may be generated (e.g., during a bio-attack), the user may only query the *details* of a small number of important grids. We assume that packets (called *count packets*) returned by *count* queries have longer deadlines than packets (called *detail packets*) returned by *detail* queries. The sizes of count and detail packets are 32 B and 160B, respectively.

We simulate a scenario that correlated events (i.e., a bio-attack) result in two hot regions each covering approximately a square of 54.4×54.4 m^2. A hot region locates at the southwest corner. The other hot region is close to the center of the region. The two hot regions are on a same diagonal to the base station to generate a worst-case congestion situation. Each hot region generates multiple flows to a base station on the northeast corner of the region. In addition, a small number of other flows are generated from other randomly picked locations. A total of 31 nodes send CBR flows representing count flows, with a subset of 15 of these nodes also sending CBR flows representing detail flows. All flows are started with a uniformly randomized time within a window of 5 sec to simulate synchronous events common in sensor networks.

We varied the rates and deadlines between the count and detail flows to better understand the effect of these parameters on different protocols. The table below lists the configurations that we tested in our simulations.

rate (1/s) count : detail	deadline (s) count : detail
0.67 : 0.30	50 : 5
0.76 : 0.35	25 : 5
0.80 : 0.36	10 : 5

The rates and number of flows were chosen such that the network is close to its breaking point where packets start to miss their deadlines.

4.3. Implementation of Protocols

Before we investigate packet scheduling algorithms, an important design decision in RAP is the routing protocol. Our investigations focus on two routing protocols, Dynamic Source Routing (DSR) [12] and GF [16]. DSR is an ad hoc routing protocol designed for traditional ID-based MANET. Previous performance studies [5] showed that DSR outperforms other major ID-based routing protocols in term of packet delivery ratio. GF is a location-based routing protocol suitable for location-addressed communication. While DSR and GF are not new, they have not been previously studied in term of deadline miss ratio in sensor networks. We compare their deadline miss ratios in Section 4.4.

At the packet scheduling layer we compare SVM and DVM against two baselines: FCFS and a deadline-based scheduling algorithm that we call DS. DS assigns a fixed priority to packets based on their end-to-end deadlines. In our workload, all count packets are assigned priority 3 (the lowest), and all detail packets are assigned priority 1 (the highest). At the MAC layer, 802.11 and its extensions were used in combination with other protocols. We now list the combination of protocols in the following table.

	Routing	Scheduling	MAC
DSR/FCFS	DSR	FCFS	802.11
GF/FCFS	GF	FCFS	802.11
GF/DS	GF	DS	802.11 extension
GF/SVM	GF	SVM	802.11 extension
GF/DVM	GF	DVM	802.11 extension

The first column contains the acronyms that are used to represent the combinations in the same row.

DS, SVM, and DVM actively dropped packets that already missed their deadlines, while DSR and GF did not actively drop packets to be consistent with original specifications. Only greedy forwarding is implemented for GF. We did not implement GPSR. The beacon period of GF is 5 sec.

The network-layer queues can hold a total of 300 packets for each configuration. DSR and GF had a single FIFO queue with 300 entries, while DS, SVM, and DVM each had three FIFO queues corresponding to different priorities. The mapping from a velocity to a priority is shown in the following table.

Priority	Velocity Range (m/s)	
	SVM	DVM
1	(10, ∞)	(40, ∞)
2	(5, 10]	(10, 40]
3	(0, 5]	(0, 10]

The velocity ranges in SVM are chosen to balance the number of flows in each priority level. The velocity ranges in DVM initially assigned priority 2 or 3 to all flows. This allowed raising some packets' priorities to priority 1 if they are delayed.

Six repeated runs were made for each of the nine combinations of the rates and deadlines. The main performance metric is the deadline *miss ratio*, i.e., the percentage of generated packets that are received by the base station within their deadlines. Each data point in Figures 3-6 represents the mean miss ratio of six runs. The 90% confidence interval is also shown for each mean.

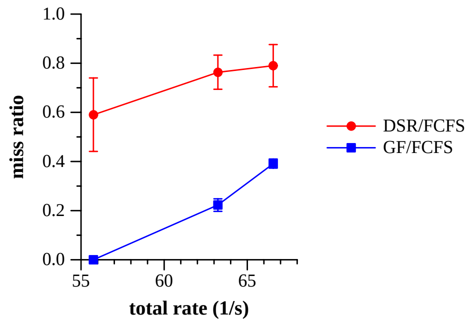


Figure 3 Overall deadline miss ratio of DSR and GF with deadlines (5,10)

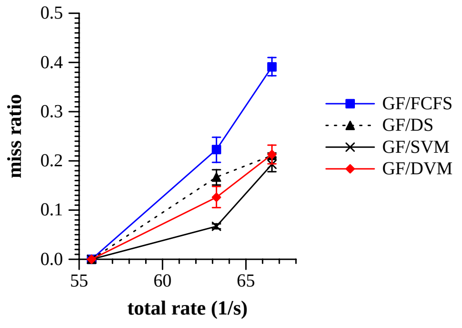
4.4. Routing: DSR and GF

First we compare the overall deadline miss ratio of DSR/FCFS and GF/FCFS (see Figure 3). DSR has a significantly higher miss ratio than GF. We found that DSR lost a large number of packets due to queue overflow, while GF lost no packets due to queue overflow. The high percentage of packet drop in DSR is caused by its aggressive route-caching. In DSR, each node caches overheard routes. When a node receives a route discovery packet from another node, it checks its route cache and informs the sender of the requested route if it is available in its route cache. In the hot regions in our network, only the first flow needs to flood the network to acquire a route to the base station. All the later flows from the same region will be informed of the existing route causing most packets from the hot region to go through the *same* route (close to the diagonal line in our network). In overload conditions, nodes on the shared route ran out of queuing space and lost packets. This problem can be common in sensor networks because of their correlated traffic patterns. In such networks, related events (i.e., a bio-attack) can start a large number of flows from a same region almost synchronously. GF does not have the overflow problem because it delivers a packet through a straight line from its source to the base station. Packets from different sensors are routed through different nodes because the source sensors have different directions toward the base station. This an important reason that GF is more suitable than DSR in sensor networks.

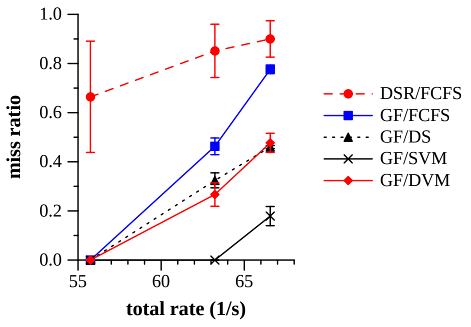
4.5. Packet scheduling

Now we compare different packet scheduling algorithms. The overall deadline miss ratios for deadlines of (5,10) s are presented in Figure 4a. The overall deadline miss ratio of DSR/FCFS is not shown in this figure because it is significantly higher than all other protocols and cannot fit in the scale (the maximum miss ratio of 0.5) of the graph. Since packets close to the base station are more likely to meet their deadlines and tend to "dilute" the difference between different algorithms, we also present in Figure 4b the miss ratio of the subset of flows from the farther hot region at the southwest corner.

From both figures, all prioritization-based packet scheduling (DS, SVM, and DVM) achieved miss ratios that were significantly lower than the protocols using FCFS. In particular GF/SVM achieved a significantly lower deadline miss ratio than all other protocols. As shown in Figure 4b, when the highest overall rate was 66.6 packets/s, only 17.9±3.9% of all packets from the farther hot region missed their deadlines for GF/SVM, compared with a miss ratio of 77.6±1.7% for GF/FCFS, 46.0±0.6% for GF/DS. This result demonstrates SVM's advantage of considering both distance and deadlines in packet prioritization.



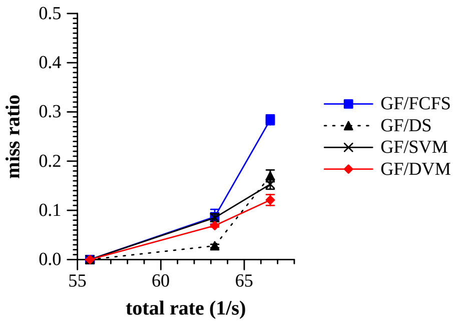
(a) Overall Deadline Miss Ratios



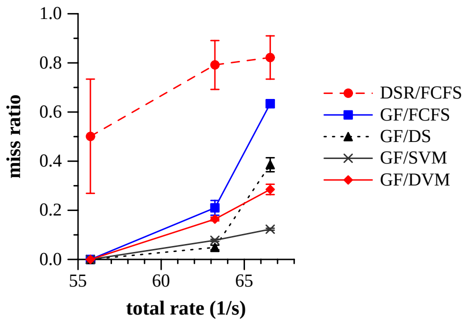
(b) Deadline Miss Ratios of Flows from the far corner

Figure 4 Deadline miss ratio with deadlines (5,10)

The miss ratios of all flows and the flows from the remote hot regions with deadlines (5, 25) s and (5, 50) s are presented in Figure 5ab and Figure 6ab, respectively. All velocity-based and deadline-based packet scheduling still significantly outperform GF and DSR with FCFS scheduling. Moreover, SVM consistently achieves the lowest miss ratio in both cases. The difference between SVM and DVM decreases as the difference between deadlines of the two types of flows is increased. DS and SVM perform almost identically when deadline is (5, 50) s. This conforms to our intuition. While the distances from each sensor to the base station stays the same, the bigger differences between the deadlines of detail and count flows become a dominant factor in requested velocity.

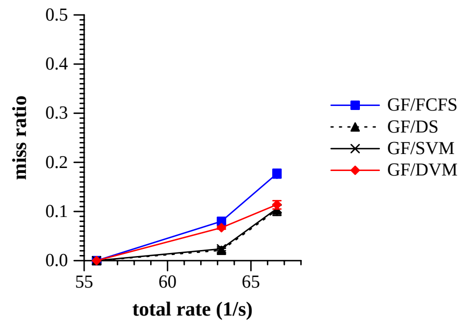


(a) Overall Deadline Miss Ratios

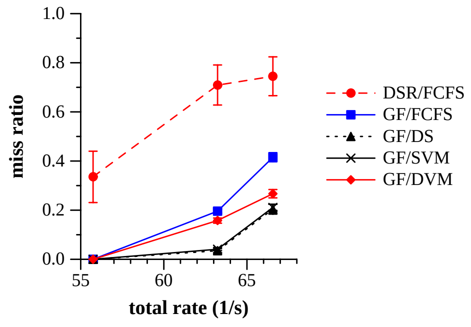


(b) Deadline Miss Ratios of Flows from the far corner

Figure 5 Deadline miss ratio with deadlines (5,25)



(a) Overall Deadline Miss Ratios



(b) Deadline Miss Ratios of Flows from the far corner

Figure 6 Deadline miss ratio with deadlines (5,50)

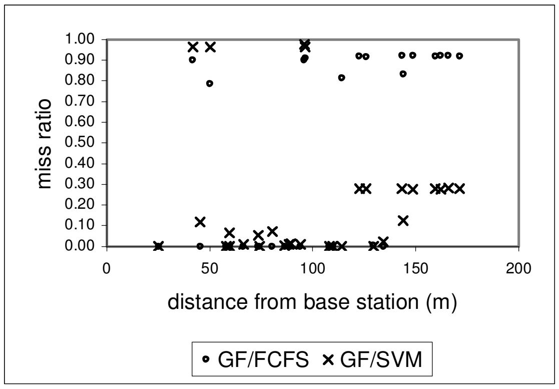


Figure 7 Miss ratio vs distance between source and destination (Deadline: (5:10) s; Rates: (0.8, 0.36)/s)

It is interesting to note that DVM did not perform as well as SVM. Although the implementation of only three priority levels reduced the flexibility of DVM, we found that it was not the cause of the unsatisfactory performance of DVM. Replacing the three priority queues with a single packet queue ordered by velocity had similar performance results in our additional experiments (not included in this paper due to space limitations). DVM's performance may be caused by the particular workloads in our experiments that all far-away flows travel through the same hot regions, and hence there is little need for priority adjustment in intermediate nodes. We plan to further investigate DVM in environments where flows suffer from different degrees of congestions.

4.5.1. Distance fairness

We find that SVM achieves better *fairness* to flows from sensors far away from the base station. This form of fairness is important to sensor networks because it affects how well sensor networks can scale. A sensor network cannot provide sufficient service if all remote sensors cannot report to the base station in time! We show the fairness by plotting the miss ratio of packets as a function of its sender's distance from the base station in a typical run with the highest rate in Figure 7. GF/FCFS significantly discriminates against remote sensors. Almost all packets that are from sensors more than 120 m away from the base station miss their deadlines. In contrast, SVM reduces the deadline miss ratio of remote sensors to about 30%. SVM is also fairer than DS and DVM that both achieved a miss ratio of about 60% for those packets (not shown due to space limitations).

In summary, SVM consistently achieves lower deadline miss ratios than both FCFS and the deadline based scheduling policy in all experiments. The performance improvement of SVM is especially significant for data flows generated by sensors far away from their base station. Compared to FCFS and DS, SVM reduces the deadline miss ratio of far-away flows from 90.0% and 46.0%, respectively, to only 17.9% with the maximum tested load. By providing fairer service to remote sensors, SVM can scale significantly better than FCFS and DS in large sensor networks.

5. Related Work

There are significant research results on real-time communications on single-hop wired LANs (e.g., [24][25]), multi-hop wired LANs (e.g., [14]), ATM (e.g., [17][18]), and the Internet (e.g., [13][22][21]). A good survey about real-time network architecture for packet-switched network is [3]. However, there have been few published works on real-time multi-hop sensor networks, which has significant different constraints from previous real-time networks.

Directed diffusion [11] is a data-driven communication paradigm for sensor networks. Users can broadcast interests to sensor networks. Sensors whose data match an interest report their data to the node that posts the interest. Our event service is similar to the interests in directed diffusion. The difference is that RAP allows users to specify the deadlines of queries on events.

RAP's network protocol stack priorities the transmission of packets based on their requested velocities. In contrast, directed diffusion does not prioritize transmission. It does not directly support location-addressed communication.

There has been significant research on routing protocols targeted at traditional MANET systems with a smaller scale than sensor networks. Broch et. al. [5] presented detailed simulation results of four representative routing protocols in small MANET with radio communication similar to wireless LAN cards. Their results showed that reactive routing protocols including DSR [12] and AODV [21] introduce less overhead packets and achieve higher data throughput. However, DSR and AODV flood the network to establish a route. This may introduce high overhead for large-scale sensor networks. Flooding can be partly avoided through aggressive caching of overheard routes on each node, but it can cause the queue overflow problem as described in Section 4.4. DSR writes the IDs of every node on the route to the packet header, which can cause significant overhead in sensor networks with many hops. Karp and Kung [16] presented geographic forwarding protocols and demonstrated that they scale better than DSR in term of network diameters and moving speed.

At the MAC layer Woo and Culler [23] proposed a MAC protocol with adaptive rate control to achieve fairness among nodes regardless of their distance from the base station in a sensor network. However, their MAC does not provide prioritization for packets with different velocities. Timing constraints are not considered in their protocol.

Several prioritization and real-time architectures of wireless LANs have been proposed in the literature. In [2] Adamou et. al. presented a fair scheduling algorithm on Wireless LAN. Choi and Shin [6] proposed a Time-Division Duplexed LAN architecture for both real-time and non-real time communication. These solutions are not designed for multi-hop networks. Kanodia et. al. [15] proposed a MAC-layer prioritization mechanism for 802.11. Their solution depends on the RTS/CTS mechanism and requires all nodes to overhear to RTS/CTS even when they are not sending or receiving data. The overhearing requirement prevents nodes from sleeping, which can be vital for improving the power efficiency in sensor networks [23].

Our work on VMS is inspired by coordinated multi-hop scheduling [15] developed by Kanodia et. al. They proposed three priority index assignment policies for multi-hop wireless networks. The Time-To-Live (TTL) policy assigns priority to a packet based on its TTL counter, while each node decreases TTL by the time it spent in that node. The TTL-based priority can dynamically adapt packet priorities based on its progress. We note that TTL-based priority may not handle scenario 2 in Figure 2 well because A and B's packets may have a similar TTL despite the fact that they have different distances to travel after E. The fixed per-node allocation decreases the priority index on each node by a per-node constant. The uniform delay budget (UDB) allocation assigns a fixed priority index to a packet based on its end-to-end deadline divided by the end-to-end hop count. UDB essentially utilizes *per-hop* velocity computed based on end-to-end hop count, while VMS is based on *geographic* velocity computed based on the geographic distance to the destination. UDB requires routing protocols to provide the end-to-end hop count for each flow at the cost of route discovery and maintenance overhead. UDB cannot work with GF, which does not provide hop count. In comparison, VMS does not require hop count and is a perfect match with GF.

6. Conclusions

Real-time communication is a critical service for future sensor networks to provide distributed micro-sensing in physical environments. We present RAP, a new real-time communication architecture for large-scale sensor networks. RAP provides convenient, high-level query and event services for distributed micro-sensing applications. Novel location-addressed communication models are supported by a scalable and light-weight network stack. We exploit the notion of velocity in real-time communication protocols on sensor networks. Velocity reflects the local urgency of a packet by capturing both key constraints in sensor networks, namely, the end-to-end deadline and the communication distance. We present Velocity-Monotonic Scheduling as a suitable scheduling policy to minimize deadline miss ratios in multi-hop sensor networks. Detailed simulations of sensor network environments demonstrate that RAP significantly reduces both the end-to-end deadline miss ratio in the sensor network. In the future we will investigate the schedulability analysis and admission control algorithms for VMS in order to provide deadline guarantees. Security and reliability aspects of the protocols are also important research directions. We will develop coordination protocols in sensor networks and implement RAP on Berkeley motes [9].

Acknowledgements

We thank Sang Son and Chengzhi Li for their helpful discussions. The work reported in this paper was supported, in part, by the DARPA NEST program under grant F33615-01-C-1905, and the DoD MURI program administered by the ONR under Grant N00014-01-1-0576.

7. References

[1] I. Aad and C. Castelluccia, “Differentiation Mechanisms for IEEE 802.11,” *IEEE INFOCOM 2001*, Anchorage, Alaska, April 2001.

[2] M. Adamou, S. Khanna, I. Lee, I. Shin, S. Zhou, “Fair Real-time Traffic Scheduling over A Wireless LAN,” *Proceedings of the 22nd IEEE Real-Time Systems Symposium*, RTSS 2001, London, UK, December 3-6, 2001

[3] C. M. Aras, J. F. Kurose, D. S. Reeves, H. Schulzrinne. "Real-Time Communication in Packet-Switched Networks", Proc. of the IEEE, Vol. 82 No. 1, Jan. 1994, pp. 122--139.

[4] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla. "GloMoSim: A Scalable Network Simulation Environment," *UCLA Computer Science Department Technical Report 990027*, May 1999.

[5] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, and J. Jetcheva. “A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols.” In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 1998)*, ACM, Dallas, TX, October 1998.

[6] S. Choi and K. G. Shin, “A Unified Wireless LAN Architecture for Real-Time and Non-Real-Time Communication Services,” IEEE/ACM Transactions on Networking, 8(1), February 2000.

[7] Crossbow, “MICA measurement systems”, http://www.xbow.com/Products/Product_pdf_files/MICA%20data%20sheet.pdf.

[8] J. Hightower and G. Borriello, "Location Systems for Ubiquitous Computing," *Computer*, 34(8), pp. 57-66, IEEE Computer Society Press, August 2001.

[9] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, “System Architecture Directions for Network Sensors,” ASPLOS 2000.

[10] IETF Working Group on Mobile Ad Hoc Networks, http://www.ietf.org/html.charters/manet-charter.html.

[11] C. Intanagonwiwat, R. Govindan and D. Estrin, “Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks,” *the Sixth Annual International Conference on Mobile Computing and Networks (MobiCOM 2000)*, August 2000, Boston, MA.

[12] D. B. Johnson and D. A. Maltz. “Dynamic Source Routing in Ad Hoc Wireless Networks.” In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, Chapter 5, pages 153-181, Kluwer Academic Publishers, 1996.

[13] J. Liebeherr, D. E. Wrege, and D. Ferrari, “Exact Admission Control in Networks with Bounded Delay Services,” IEEE/ACM Transactions on Networking, 1996.

[14] D. Kandlur, K. G. Shin, and D. Ferrari, “Real-time Communication in Multi-hop Networks,” *IEEE Trans. on Parallel and Distributed Systems*, October 1994, pp. 1044-1056.

[15] V. Kanodia, C. Li, A. Sabharwal, B. Sadeghi, and E. W. Knightly, “Distributed Multi-Hop Scheduling and Medium Access with Delay and Throughput Constraints,” *International Conference on Mobile Computing and Networking (MobiCom 2001)*, Rome, Italy, July 2001.

[16] Karp, B. and Kung, H.T., “Greedy Perimeter Stateless Routing for Wireless Networks,” in *Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000)*, Boston, MA, August, 2000, pp. 243-254.

[17] S. Kweon and K. G. Shin, “Providing Deterministic Delay Guarantees in ATM Networks,” *IEEE/ACM Transactions on Networking*, 6(6), December 1998.

[18] C. Li, R. Bettati, and W. Zhao, “Static Priority Scheduling for ATM Networks,” in *Proc. of IEEE Real-Time Systems Symposium*, Dec. 1997.

[19] S. Mangold, S. Choi, P. May, O. Klein, G. Hiertz, L. Stibor: IEEE 802.11e Wireless LAN for Quality of Service (invited paper). Proceedings of the European Wireless, Vol. 1, pp. 32-39, Florence, Italy, February 2002.

[20] B. O’Hara and A. Petrick, *IEEE 802.11 Handbook - a Designer’s Companion*, IEEE Press, 1999.

[21] I. Stoica and H. Zhang, “Providing Guaranteed Services Without Per Flow Management,” *SIGCOMM*, 1999.

[22] S. Wang, D. Xuan, R. Bettati, and W. Zhao, “Providing Absolute Differentiated Services for Real-Time Applications in Static-Priority Scheduling Networks,” *IEEE INFOCOM 2001*.

[23] Woo and D. Culler. “A Transmission Control Scheme for Media Access in Sensor Networks,” *ACM/IEEE International Conference on Mobile Computing and Networks (MobiCOM 2001)*, Rome, Italy, July 2001.

[24] W. Zhao, J. A. Stankovic , K. Ramamritham, “A Window Protocol for Transmission of Time-Constrained Messages,” *IEEE Transactions on Computers*, 39(9), p.1186-1203, Sept. 1990.

[25] K. M. Zuberi, K. G. Shin, “Design and Implementation of Efficient Message Scheduling for Controller Area Network,” 49(2), *IEEE Transactions on Computers* February 2000.

SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks

Tian He[a] John A Stankovic[a] Chenyang Lu[b] Tarek Abdelzaher[a]

[a] *Department of Computer Science*
University of Virginia
{tianhe, stankovic, zaher}@cs.virginia.edu

[b] *Department of Computer Science & Engineering*
Washington University in St Louis
lu@cs.wustl.edu

Abstract

In this paper, we present a real-time communication protocol for sensor networks, called SPEED. The protocol provides three types of real-time communication services, namely, real-time unicast, real-time area-multicast and real-time area-anycast. SPEED is specifically tailored to be a stateless, localized algorithm with minimal control overhead. End-to-end soft real-time communication is achieved by maintaining a desired delivery speed across the sensor network through a novel combination of feedback control and non-deterministic geographic forwarding. SPEED is a highly efficient and scalable protocol for sensor networks where the resources of each node are scarce. Theoretical analysis, simulation experiments and a real implementation on Berkeley motes are provided to validate our claims.

1. Introduction

Many exciting results have been recently developed for large-scale sensor networks. These networks can form the basis for many types of smart environments such as smart hospitals, battlefields, earthquake response systems, and learning environments. While these potential applications remain diverse, one commonality they all share is the need for an efficient and robust routing protocol.

The main function of sensor networks is data delivery. We distinguish three types of communication patterns associated with the delivery of data in such networks. First, it is often the case that one part of a network detects some activity that needs to be reported to a remote base station. This type of communication is called unicast. Alternatively, a base station may issue a command or query to an area in the sensor networks. For example, it may ask all sensors in the region of a damaged nuclear plant to report radiation readings, or command all lights in a given area to turn on. This type of communication motivates a different routing service where one end-point of the route may be an area rather than an individual node. We call this area-multicast. Finally, since sensors often measure highly redundant information, in some situations it may be sufficient to have any node in an area respond. We call a routing service that provides such capability, area-anycast. SPEED provides the aforementioned three types of communication services.

Since sensor networks deal with real world, it is often necessary for communication to meet real-time constraints. In surveillance systems, for example, communication delays within sensing and actuating loops directly affect the quality of tracking. To date, few results exist for sensor networks that adequately address real-time requirements. In this paper we develop a protocol SPEED that supports soft real-time communication based on feedback control and stateless algorithms for large-scale sensor networks. We evaluate SPEED via simulation using GloMoSim [15] and compare it to five other ad hoc routing protocols: DSR [5], AODV [10], GF [13] and two scaled down versions of SPEED. The performance results show that SPEED 1) reduces the number of packets that miss their end-to-end deadlines, 2) reacts to transient congestion in the most stable manner, and 3) efficiently handles voids [6] with minimal control overhead. We also implement SPEED on the Berkeley motes [4]. The results show that SPEED helps balance the traffic load to increase the system lifetime.

2. State of the Art

Several routing protocols have been developed for ad hoc wireless networks. Sensor networks can be regarded as a sub-category of such networks, but with a number of different requirements.

In sensor networks, location is more important than a specific node's ID. For example, tracking applications only care where a target is located, not the ID of the reporting node. In sensor networks, such location-awareness is necessary to make the sensor data meaningful. Therefore, it is natural to utilize location-aware routing. A set of location based routing algorithms have been proposed. Finn [2] proposed a greedy geographic forwarding protocol with limited flooding to circumvent the voids inside the network. GPSR [6] by Karp and Kung use perimeter forwarding to get around voids. Geographic distance routing (GEDIR) [13] guarantees loop-free delivery in a collision-free network. LAR [7] by Young-Bae Ko improves the efficiency of the on-demand routing algorithms by restricting routing packet flooding in a specified "request zone."

SPEED also utilizes geographic location to make localized routing decisions. The difference is that SPEED is designed to handle congestion and provide a soft real-time communication service, which are not the main goals of previous location-based routing protocols. Moreover, SPEED provides an alternative solution to handle voids other than approaches based on planar graph traversal [6] and limited flooding [2].

Several real-time protocols have been proposed for sensor networks. SWAN [1] uses feedback information from the MAC layer to regulate the transmission rate of non-real-time TCP traffic in order to sustain real-time UDP traffic. RAP [9] uses velocity monotonic scheduling to prioritize real-time traffic and enforces such prioritization through a differentiated MAC Layer. Woo and Culler [14] proposed an adaptive MAC layer rate control to achieve fairness among nodes with different distances to the base station. All of these algorithms work well by locally degrading a certain portion of the traffic. However, this kind of local MAC layer adaptation cannot handle long-term congestion where routing assistance is necessary to divert traffic away from any hotspot. SPEED provides a combination of MAC layer and network layer adaptation that effectively deals with such issues. To the best of our knowledge, no routing algorithm has been specifically designed to provide soft real-time guarantees for sensor networks.

Reactive routing algorithms such as AODV [10] and DSR [5] maintain routing information for a small subset of possible destinations, namely those currently in use. If no route is available for a new destination, a route discovery process is invoked. Route discovery broadcasts can lead to significant delays in a sensor network with a large network diameter. This limitation makes on-demand algorithms less suitable for real-time applications.

3. Design Goals

Our design is inspired by the observation that unlike wired networks, where the delay is independent of the physical distance between the source and destination, in multi-hop wireless sensor networks, the end-to-end delay depends on not only single hop delay, but also on the distance a packet travels. In view of this, the key design goal of the SPEED algorithm is to support a soft real-time communication service with a desired delivery *speed* across the sensor network, so that end-to-end delay is proportional to the distance between the source and destination. It should be noted that delivery speed refers to the approaching rate along a straight line from the source toward the destination. Unless the packet is routed exactly along that straight line, delivery speed is smaller than the actual speed of the packet in the network. For example, if the packet is routed in the opposite direction from the destination, its speed is negative. Our algorithm ensures that this condition never occurs.

Upon this soft real-time delivery service, SPEED provides three types of real-time communication services, namely, real-time unicast, real-time area-multicast and real-time area-anycast, for sensor networks. In doing so, SPEED satisfies the following design objectives.

1. **Stateless Architecture**. The physical limitations of sensor networks, such as large scale, high failure rate, and constrained memory capacity necessitate a stateless approach. SPEED only maintains immediate neighbor information. It doesn't require a routing table as in DSDV [11] nor per-destination states as in AODV [10]. Thus, its memory requirements are minimal.
2. **Soft Real-Time.** Sensor networks are commonly used to monitor and control the physical world. SPEED provides a uniform delivery speed across the sensor network to meet the requirement of real-time applications such as disaster and emergency surveillance in sensor networks.
3. **Minimum MAC Layer Support.** SPEED doesn't require real-time or QoS aware MAC support. The feedback control scheme employed in SPEED allows it to be compatible with all existing best effort MAC layers.
4. **QoS Routing and Congestion Management.** Most reactive routing protocols can find routes that avoid network hot spots during the route acquisition phase. Such protocols work well when traffic patterns don't fluctuate during a session. However, these protocols (e.g. [5]) are less successful when congestion patterns change rapidly compared to the session lifetime. When a route becomes congested, such protocols either suffer a delay or initiate another round of route discovery. As a solution, SPEED uses a novel backpressure re-routing scheme to re-route packets around large-delay links with minimum control overhead.
5. **Traffic Load Balancing.** In sensor networks, the bandwidth and energy are scarce resources compared to a wired network. Because of this, it is valuable to utilize several simultaneous paths to carry packets from the source to the destination. SPEED uses non-deterministic forwarding to balance each flow among multiple concurrent routes.
6. **Localized Behavior.** Pure localized algorithms are those in which any action invoked by a node should not affect the system as a whole. In algorithms such as AODV, DSR and TORA, this is not the case. In these protocols a node uses flooding to discover new paths. In sensor networks where thousands of nodes communicate with each other, broadcast storms may result in significant power consumption and possibly a network meltdown. To avoid that, all distributed operations in SPEED are localized to achieve high scalability.
7. **Void Avoidance.** In some scenarios, pure greedy geographic forwarding may fail to find a greedy path to the destination, even when one actually exists. SPEED handles the void the same way as it handles congested areas and guarantees that if there is a greedy route between the source and destination, it will discover it.

Note, while SPEED does not use routing tables, SPEED does utilize location information to carry out routing. Because of this, we assume that each node is location-aware.

4. SPEED Protocol

SPEED maintains a desired delivery speed across sensor networks by both diverting traffic at the networking layer and locally regulating packets sent to the MAC layer. It consists of the following components:

- An API
- A neighbor beacon exchange scheme
- A delay estimation scheme
- The Stateless Non-deterministic Geographic Forwarding algorithm (SNGF)
- A Neighborhood Feedback Loop (NFL)
- Backpressure Rerouting
- Last mile processing

As shown in Figure 1, SNGF is the routing module responsible for choosing the next hop candidate that can support the desired delivery speed. NFL and Backpressure Rerouting are two modules to reduce or divert traffic when congestion occurs, so that SNGF has available candidates to choose from. The last mile process is provided to support the three communication semantics mentioned before. Delay estimation is the mechanism by which a node determines whether or not congestion has occurred. And beacon exchange provides geographic location of the neighbors so that SNGF can do geographic based routing. The details of these components are discussed in the subsequent sections, respectively.

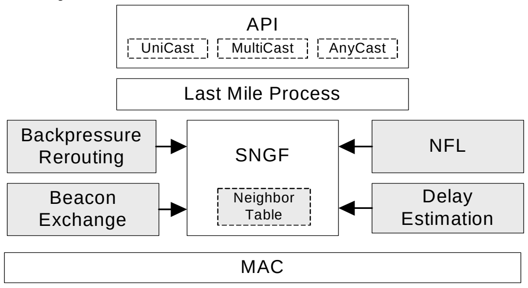


Figure 1. SPEED Protocol

4.1. Application API and Packet Format

The SPEED protocol provides four application-level API calls:

- AreaMulticastSend (position, radius, packet): This service identifies a destination area by its center position and radius. It sends a copy of the packet to every node inside the specified area with a speed above a certain desired value.
- AreaAnyCastSend (position, radius, packet): This service sends a copy of the packet to at least one node inside the specified area with a speed above a certain desired value.
- UnicastSend(Global_ID, packet): In this service the node identified by Global_ID will receive the packet with a speed above a certain desired value.
- SpeedReceive(): this primitive permits nodes to accept packets targeted to them.

Though SPEED is a real-time protocol, we don't use deadline as a parameter in our API. SPEED aims at providing a uniform packet delivery speed across the sensor network, so that the end-to-end delay of a packet is proportional to the distance between the source and destination. With this service, real-time applications can estimate end-to-end delay before making admission decisions. Delay differentiation for different classes of packets is left as future work.

There is a single data packet format for the SPEED protocol, which contains the following major fields:

- PacketType: the type of communication: Area Multicast, AreaAnyCast or Unicast.
- Global_ID: only used in Unicast communication to identify a destination node.
- Destination Area: Describes a three-dimensional space with a center point and radius in which the packets are destined.
- TTL: Time To Live field is the hop limit used for last mile processing.
- Payload.

4.2. Neighbor Beacon Exchange

Similar to other geographic routing algorithms, every node in SPEED periodically broadcasts a beacon packet to its neighbors. This periodic beaconing is only used for exchanging location information between neighbors. We argue that the beaconing rate can be very low when nodes inside the sensor network are stationary or slow moving. Moreover, piggybacking [6] methods can also be exploited to reduce this beacon overhead.

In addition to periodic beaconing, SPEED uses two types of on-demand beacons, namely a delay estimation beacon and a backpressure beacon, to quickly identify the traffic changes inside the network. The functionality of two beacons will be discussed in section 4.3 and 4.6, respectively. As shown in the evaluation (section 5.4), our on-demand beacon scheme introduces only a small overhead in exchange for a fast response to congestion.

In SPEED, each node keeps a neighbor table to store information passed by the beaconing. Each entry inside the table has the following fields: (NeighborID, Position, SendToDelay, ExpireTime). The ExpireTime is used to timeout this entry. If a neighbor entry is not refreshed after a certain timeout, it will be removed from the neighbor table. SendToDelay is a delay estimation to the neighbor node identified by the NeighborID field. The details of setting this value are discussed in the next section.

4.3. Delay Estimation

We use single hop delay as the metric to approximate the load of a node. We notice that the delays experienced by broadcast packets and unicast packets are quite different due to different handling inside the MAC layer and that unicast packet delay is more appropriate for making routing decisions. In a scarce bandwidth environment, we cannot afford to use probing packets to estimate the single hop delay. Instead we use the data packets passing this node to perform this measurement. Delay is measured at the sender, which timestamps the packet entering the network output queue and calculates the round trip single hop delay for this packet when receiving the ACK. At the receiver side, the duration for processing an ACK is put into the ACK packet. The single-trip time is calculated by subtracting receiver side processing time from the round trip delay experienced by the sender. We compute the current delay estimation by combining the newly measured delay with previous delays via the exponential weighted moving average (EWMA) [8]. Propagation delay is ignored. We argue that this delay estimation is a better metric than average queue size for representing the congestion level of the wireless network, because the shared media nature of the wireless network allows the network to be congested even if queue sizes are small.

4.4. Stateless Non-deterministic Geographic Forwarding (SNGF)

Before elaborating on SGNF, we introduce three definitions:

- The Neighbor Set of Node *i*: NS_i is the set of nodes that are inside the radio range of node *i*. Note, we do not assume that the communication radius is a perfect circle. SPEED works with irregular radio patterns.

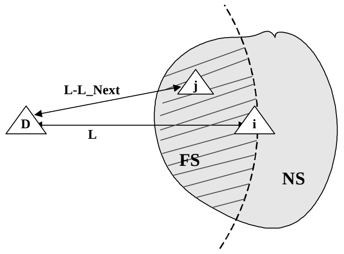


Figure 2. NS and FS definitions

- The Forwarding Candidate Set of Node *i*: A set of nodes that belong to NS_i and are closer to the destination. Formally, $FS_i\,(Destination) = \{node \in NS_i \mid L - L\_next > 0\}$ where L is the distance from node *i* to the destination and *L_next* is the distance from the next hop forwarding candidate to the destination. These nodes are inside the cross-hatched shaded area as shown in Figure 2. We can easily obtain $FS_i\,(Destination)$ by scanning the *NS* set of nodes once.

  It is worth noticing that the membership of the neighbor set only depends on the radio range, but the membership of the forwarding set also depends on destination area.
- Relay Speed. Relay speed is calculated by dividing the advance in distance from the next hop node j by the estimated delay to forward a packet to node j. Formally,
  $$Speed_i^j\,(Destination) = \frac{L - L\_next}{HopDelay_i^j}.$$

Since in SPEED, nodes keep the Neighbor Set (NS), but don't keep a routing table or flow information, the memory requirements are only proportional to the number of neighbors.

Based on the destination of the packet and the current FS, the Stateless Non-deterministic Geographic Forwarding (SNGF) portion of our protocol routes the packets according to the following rules:

1. Packets are forwarded only to the nodes that belong to the FS_i *(Destination)*. If there is no node inside the FS_i *(Destination)*, packets are dropped and a backpressure beacon is issued to upstream nodes to prevent further drops (see 4.7). To reduce the chance of such drops, we deduce a lower bound of node density that can virtually eliminate these drops (appendix A).
2. SPEED divides the neighbor nodes inside FS_i *(Destination)* into two groups. One group contains the nodes that have relay speeds larger than a certain desired speed $S_{setpoint}$, the other contains the nodes that cannot sustain such desired speed. The $S_{setpoint}$ is a system parameter that depends on the communication capability of the nodes and desired traffic workload a sensor network should support.
3. The forwarding candidate is chosen from the first group, and the neighbor node with highest relay speed has a higher probability to be chosen as the forwarding node. In our approach, we use a discrete exponential distribution to trade off between load balancing and optimal path length.
4. If there are no nodes belonging to the first group, a relay ratio is calculated based on the Neighborhood Feedback Loop (NFL), which is discussed in more detail in section 4.5. Whether a packet drop will really happen depends on whether a randomly generated number between (0,1) is bigger than the relay ratio. In SPEED a packet is dropped only when no downstream node can guarantee the single hop speed set point $S_{set\text{-}point}$ and dropping packets must be peformed to reduce the congestion. Though one can consider buffering packets as an alternative to the dropping, however, we argue that under real-time and small memory constrains, dropping is often a better choice.

SNGF provides two nice properties to help meet our design goals. First, since SNGF sends packets to the downstream node capable of maintaining the desired delivery speed, soft real-time end-to-end delivery is achieved with a theoretical delay bound: *Delay Bound* = $L_{e2e}/S_{setpoint}$, where L_{e2e} is the distance between the source and destination. $S_{setpoint}$ is the uniform speed to be maintained across the sensor network. Second, SNGF can balance traffic and reduce congestion by dispersing packets into a large relay area. This load balancing is valuable in a sensor network where the density of nodes is high and the communication bandwidth is scarce and shared. Load balancing also balances the power consumption inside the sensor networks to prevent some nodes from dying faster than others.

SNGF provides MAC layer adaptation and reduces the congestion by locally dropping (or optionally buffering) packets. This adaptation is good enough to deal with transient overshoot inside the sensor networks. But if such congestion remains for a relatively long time, network layer adaptation is desired to redirect traffic to a less congested area, which is discuss further in section 4.6.

4.5. Neighborhood Feedback Loop (NFL)

The Neighborhood Feedback Loop (NFL) is the key component in maintaining the single hop relay speed. The NFL is an effective approach to maintaining system performance at a desired value. This has been shown in [12], where a low miss ratio of real-time tasks and a high utilization of the computational nodes are simultaneously achieved. Here we want to maintain a single hop relay speed above a certain value $S_{setpoint}$, a performance goal desired by the system designer.

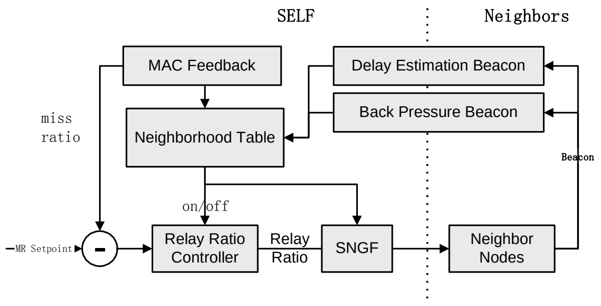


Figure 3. Neighborhood Feedback Loop (NFL)

We deem it a miss when a packet delivered to a certain neighbor node has a relay speed less than $S_{setpoint}$, or if there is a loss due to collision. The percentage of such misses is called this neighbor's miss ratio. The responsibility of the NFL is to force the miss ratios of the neighbors to converge to a set point, namely zero.

As shown in Figure 3, the MAC layer collects miss information and feeds it back to the Relay Ratio controller. The Relay Ratio controller calculates the relay ratio and feeds that into the SNGF where a drop or relay action is made. The Relay Ratio controller currently implemented is a multiple inputs single output (MISO) proportional controller that takes the miss ratios of its neighbors as inputs and proportionally calculates the relay ratio as the output to the SNGF. Formally it is described by the following formulas.

$$u = 1 - K \frac{\sum e_i}{N} \quad if \ \ \forall e_i > 0$$

$$u = 1 \quad if \ \ \exists e_i = 0$$

where e_i is the miss ratio of the neighbor i inside the FS set, N is size of the FS set. u is the output (relay ratio) to SNGF. And K is the proportional gain.

It should be noted that the Relay Ratio controller will be activated only when all nodes inside the forwarding set (FS) cannot maintain the desired single hop relay speed $S_{setpoint}$ and a drop is absolutely necessary to maintain the single hop delay. Such a scheme ensures that re-routing has a higher priority than dropping. In other words, SPEED will not drop a packet as long as there is another path that can meet the delay requirements.

By reducing the sending rate to the downstream nodes, the neighborhood feedback loop can maintain a single hop relay speed. However, this MAC layer adaptation can't solve the hotspot problem, if the upstream nodes, which are unaware of the congestion, keep sending packets into this area. In this case, backpressure rerouting (network layer adaptation) is necessary to reduce the traffic injected into the congested area.

4.6. Back-Pressure Rerouting

Backpressure re-routing is naturally generated from the collaboration of neighbor feedback loop (NFL) routines as well as the stateless non-deterministic geographic forwarding (SNGF). To be more explicit, we introduce this scheme with an example (Figure 4).

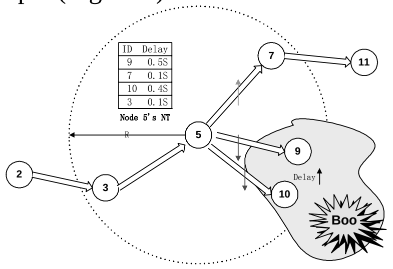


Figure 4. Backpressure rerouting case one

Suppose in the lower-right area, heavy traffic appears, which leads to a lower relay speed in nodes 9 and 10. Through the MAC layer feedback, node 5 will detect that nodes 9 and 10 are congested. Since SNGF will reduce the probability of selecting nodes 9 and 10 as forwarding candidates and route more packets to node 7, it will reduce the congestion around nodes 9 and 10. Since all neighbors of 9 and 10 will react the same way as node 5, eventually nodes 9 and 10 will be able to relay packets above the desired speed.

A more severe case could occur when all the forwarding neighbors of node 5 are also congested as shown in Figure 5.

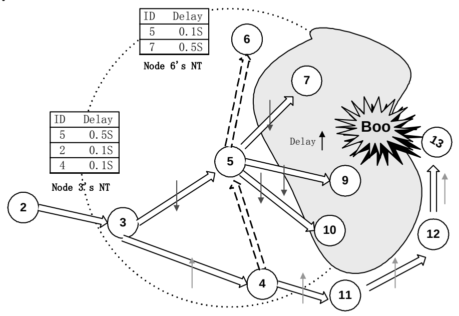


Figure 5. Backpressure rerouting case two

In this case, the neighborhood feedback loop is activated to assist backpressure re-routing. In node 5, a certain percent of packets will be dropped in order to reduce the traffic injected into the congested area. At the same time, an on-demand backpressure beacon is issued by node 5 with the following fields.

(ID, Destination, AvgSendToDelay)

AvgSendToDelay is the average SendToDelay of all nodes inside FS_{ID}(Destination). In our example, when the destination is at node 13, AvgSendToDelay is the average delay from node 5 to nodes 7, 9 and 10.

When a neighbor receives the back-pressure beacon from node 5, it determines whether node 5 belongs to its FS(Destination). If node 5 does, this neighbor modifies the SendToDelay for node 5 according to the AvgSendToDelay. For example only node 3 will consider node 5 as a next hop forwarding candidate to the destination where node 13 resides. If node 5 is not in the FS(Destination), then this neighbor ignores the backpressure beacon. This backpressure mechanism can reduce the chance of “false congestion indication”, to ensure that traffic from node 4 to node 6 will not be affected by the backpressure beacon.

If, unfortunately, node 3 is in the same situation as node 5, further backpressure will be imposed on node 2. In the extreme case, the whole network is congested and the backpressure will proceed upstream until it reaches the source, where the source will quench the traffic flow to that destination.

Backpressure rerouting is a network layer adaptation used by SPEED to reduce the congestion inside the network. In this case no packet needs to be sacrificed. Network layer adaptation has a higher priority than MAC layer adaptation used by SNGF and NFL. A drop via the feedback loop is only necessary when the situation becomes so congested and there is no alternative to maintaining a single hop speed other than dropping packets.

4.7. Void Avoidance

Greedy geographic based algorithms have many advantages over the traditional MANET routing algorithms for real-time sensor network applications. They do not suffer route discovery delay and tend to choose the shortest path to the destination. Moreover without flooding, they have relatively low control packet overhead. Unfortunately, they also have a serious drawback. In many cases, they may fail to find a path even though one does exist. To overcome this, SPEED deals with a void the same way it deals with congestion. As shown in the Figure 6, if there is no downstream node to relay packets from node 2 to node 5, node 2 will send out a backpressure beacon containing fields: (ID, Destination, ∞). The upstream node 1 that needs node 2 to relay the packets to that destination will set the SendToDelay for node 2 to infinity and stop sending packets to node 2. If node 3 doesn’t exist, further backpressure will occur until a new route is found. It should be admitted that our scheme of void avoidance isn’t guaranteed to find a path if there is one as in GPSR[6], but it is guaranteed to find a *greedy* path if one exists. To maintain real-time properties, we don’t allow backtracking to violate our desired speed setpoint. However, as we can see from the evaluation section 5.6, such a simple scheme can significantly reduce packet loss due to voids in high-density sensor networks.

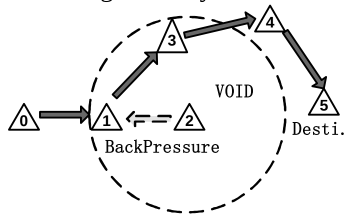


Figure 6. Void avoidance scheme

4.8. Last Mile Process

Since SPEED is targeted at sensor networks where the ID of a sensor node is not important, SPEED only cares about the location where sensor data is generated.

The last mile process is so called because only when the packet enters into the destination area will such a function be activated. The SNGF module aforementioned controls all previous packet relays.

The last mile process provides two novel services that fit the scenario of sensor networks: Area-multicast and Area-anycast. The area in this case is defined by a center-point (x,y,z) and a radius, in essence a sphere. More complex area definitions can be made without jeopardizing the design of this last mile process.

Nodes can differentiate the packet type by the PacketType field mentioned in section 4.1. If it’s an anycast packet, the nodes inside the destination area will deliver the packet to the transport layer without relaying it onward. If it’s a multicast packet, the nodes inside the destination area which first receive the packet coming from the outside of the destination area will set a TTL. This allows the packet to survive within the diameter of the destination area and be broadcast within a specified radius. Other nodes inside this destination area will keep a copy of the packet and rebroadcast it. The nodes that are outside the destination area will just ignore it. The last mile process for unicast is nearly the same as multicast, except the node with a specified global_ID will deliver the packet to the transport layer. If the location directory service is precise, we can expect the additional flooding overhead for the unicast packets to be small. The current implementation of the last mile process is relatively simple. More efficient and robust techniques are desired for future research.

5. Experimentation and Evaluation

We simulate SPEED on GloMoSim [15], a scalable discrete-event simulator developed by UCLA. This software provides a high fidelity simulation for wireless communication with detailed propagation, radio and MAC layers. Table 1 describes the detailed setup for our simulator. The communication parameters are mostly chosen in reference to the Berkeley mote specification.

Routing	AODV, DSR, GF, SPEED, SPEED-S, SPEED-T
MAC Layer	802.11 (Simplified DCF)
Radio Layer	RADIO-ACCNOISE
Propagation model	TWO-RAY
Bandwidth	200Kb/s
Payload size	32 Byte
TERRAIN	(200m, 200m)
Node number	100
Node placement	Uniform
Radio Range	40m

Table 1. Simulation settings

In our evaluation, we compare the performance of six different routing algorithms: AODV [10], DSR [5], GF [13], SPEED, SPEED-S, SPEED-T.

GF forwards a packet to the node that makes the most progress toward the destination. SPEED-S and SPEED-T are reduced versions of SPEED. SPEED-S replaces the SNGF with a MAX-SPEED routing algorithm that geographically forwards the packets to nodes that can provide a max single hop relay speed. SPEED-T replaces the SNGF with a MIN-DELAY routing algorithm that geographically forwards packets to nodes that have a minimum single hop delay. Both reduced versions have no backpressure rerouting mechanisms.

In our evaluation, we present the following set of results: 1) end-to-end delay under different congestion levels, 2) miss ratio, 3) control overhead, 4) communication energy consumption, and 5) packet delivery ratio under different node densities. All experiments are repeated 16 times with different random seeds and different random node topologies. We also implement SPEED on the Berkeley motes [4]. The results obtained from this testbed show a load balance feature of SPEED protocol (see section 5.7).

5.1. Sensor Network Traffic Pattern

There are two typical traffic patterns in sensor networks: a base station pattern and a peer-to-peer pattern. The base station pattern is the most representative one inside sensor networks. For example, in surveillance systems, multiple sensors detect and report the location of an intruder to the control center. In tracking systems, a base station issues multiple tracking commands to a group of pursuers. In a different respect, the peer-to-peer pattern is usually used for data aggregation and consensus in a small area where a team of nearby motes interact with each other. The end-to-end delay in the base station pattern is the major part of delay for the sensing-actuation loop, and is therefore, the focus of our evaluation.

5.2. Congestion Avoidance

In a sensor network, where node density is high and bandwidth is scarce, traffic hot spots are easily created. In turn, such hot spots may interfere with real-time guarantees of critical traffic in the network. In SPEED, We apply a combined network and MAC layer congestion control scheme to alleviate this problem.

To test the congestion avoidance capabilities, we use a base station scenario, where 6 nodes, randomly chosen from the left side of the terrain, send periodic data to the base station at the middle of the right side of the terrain. The average hop count between the node and base station is about 8~9 hops. Each node generates 1 CBR flow with a rate of 1 packet/second. To create congestion, at time 80 seconds, we create a flow between two randomly chosen nodes in the middle of the terrain. This flow then disappears at time 150 seconds into the run. This flow introduces a step change into the system, which is an abrupt change that stress-tests SPEED's adaptation capabilities to reveal its transient-state response. In order to evaluate the congestion avoidance capability under different congestion levels, we increase the rate of this flow step by step from 0 to 100 packets/second over several simulations

Figure 7 and Figure 8 plot the end-to-end (E2E) delay for the six different routing algorithms. At each point, we average the E2E delays of all the packets from the 96 flows (16 runs with 6 flows each). The 90% confidence interval is within 2~15% of the mean, which is not plotted for the sake of legibility.

Under the no or light congested situations, Figure 7 and Figure 8 show that all geographic based routing algorithms have short average end-to-end delay in comparison to AODV and DSR. There are several factors accounting for this outcome. First, the route acquisition phase in AODV and DSR leads to significant delays for the first few packets, while geographic based routing doesn't suffer from this. We argue that without an initial delay cost, geographic based routing is more suitable for real-time applications like target tracking where the base station sends the actuation commands to the sensor group, which is dynamically changing as the target moves. In such a scenario, DSR and AODV need to perform route acquisition repeatedly in order to track the target. Second, the route discovered through flooding and path reversal has relatively more hops than greedy geographic forwarding. The reason for even higher delay in AODV than DSR is that DSR implementation intensively uses a route cache to reduce route discovery and maintenance cost. As shown in Figure 8, SPEED-T has higher delay than GF, SPEED-S and SPEED, because SPEED-T only uses hop delay to make routing decision and disregards the progress each hop makes, which leads to more hops to the destination in wireless multi-hop networks. Instead, under lightly congested situation, GF, SPEED-S and SPEED tend to forward a packet at each step as close to the destination as possible, thereby reducing the number of hops and the end-to-end delay.

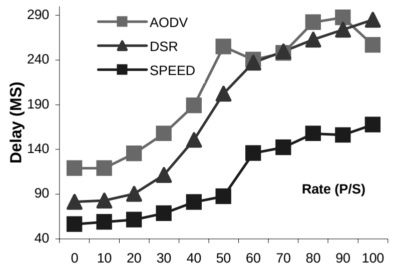


Figure 7. E2E Delay Under Different Congestion

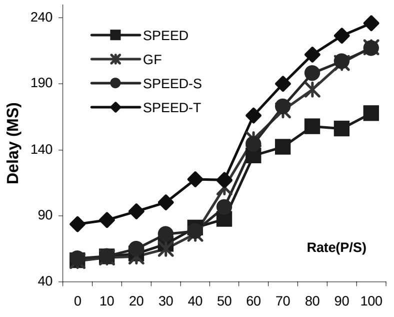


Figure 8. E2E Delay Under Different Congestion

Under the heavy congested situations (Figure 7 and Figure 8), each routing algorithm responds differently. SPEED performs best. For example, SPEED reduces the average end-to-end delay by 30%~40% in the face of heavy congestion in comparison to the other algorithms considered. The key reasons for SPEED's better performance are 1) DSR, AODV and GF only respond to severe congestion, which leads to link failures (i.e., when multiple retransmissions fail at the MAC layer). They are insensitive to long delays as long as no link failures occur. 2) DSR, AODV and GF routing decisions are not based on the link delays, and therefore may cause congestion at a particular receiver even though it has long delays. 3) DSR and AODV flood the network to rediscover a new route when the network is already congested. 4) SPEED-T and SPEED-S don't provide traffic adaptation. When all downstream nodes are congested, SPEED-T and SPEED-S cannot reduce or redirect the traffic to uncongested routes. 5) SPEED not only locally reduces the traffic through a combination of SNGF and Neighborhood Feedback loops in order to maintain the desired speed, but also diverts the traffic into a large area through its backpressure rerouting mechanism. This combination leads to lower end-to-end delay.

5.3. E2E Deadline Miss Ratio

The deadline miss ratio is the most important metric in soft real-time systems. We set the desired delivery speed $S_{setpoint}$ to 1km/s, which leads to an end-to-end deadline of 200 milliseconds. In the simulation, some packets are lost due to congestion or forced-drops. We also consider this situation as a deadline miss. The results shown in Figure 9 and Figure 10 are the summary of 16 randomized runs.

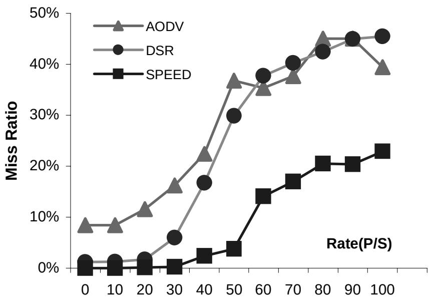


Figure 9. MissRatio Under Different Congestion

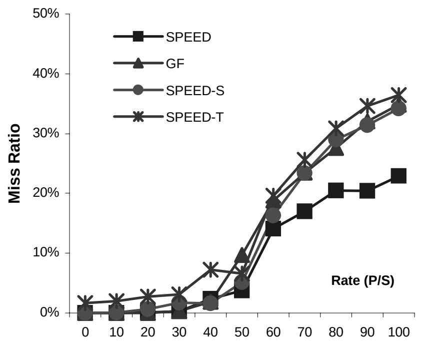


Figure 10. MissRatio Under Different Congestion

AODV and DSR don't perform well in the face of congestion because both algorithms flood the network in order to discover a new path when congestion leads to link failure. This flooding just serves to increase the congestion. GF only switches the route when there are link failures caused by heavy congestion. The routing decision is based solely on distance and does not consider delay. SPEED-T only considers the single hop delay and doesn't take distance (progress) into account, which leads to a longer route. SPEED-S provides no adaptation to the congestion and cannot prevent packets from entering the congestion area. Only SPEED tries to maintain a desired delivery speed through MAC and network layer adaptations, and therefore has a much less miss ratio than other algorithms. Due to its transient behavior, SPEED still has about a 20% miss ratio when the network is heavily congested. Future work is needed to reduce the convergence time in order to improve the performance.

Comparing Figure 9 and Figure 10, we argue that purely localized algorithms without flooding outperform other algorithms when traffic congestion increases. Generally, the less state information a routing algorithm depends on, the more robust it is in the face of packet loss and congestion.

5.4. Control Packet Comparison

Except for AODV, all other routing algorithms studied use a relatively low number of control packets. Most control packets in DSR and AODV are used in route acquisition. Because AODV initiates route discovery (flooding) whenever a link breaks due to congestion, it requires a large number of control packets. DSR uses a route cache extensively, so it can do route discovery and maintenance with a much lower cost than AODV. The only control packets used in GF, SPEED-S and SPEED-T (Figure 11) are periodic beacons, whose number is constant at 750 under different congestion levels. In addition to periodic beacons, SPEED uses two types of on-demand beacons to notify neighbors of the congestion. This costs SPEED more control packets than the other three geographic based routing algorithms (Figure 11).

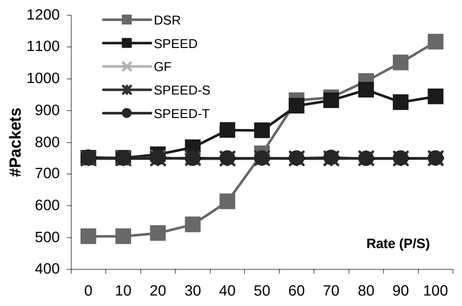


Figure 11. Control packet overhead comparison

5.5. Energy Consumption

Under energy constraints, it is vital for sensor nodes to minimize energy consumption in radio communication to extend the lifetime of sensor networks. From the results shown in Figure 12, we argue that geographic based routing tends to reduce the number of hops in the route, thus reducing the energy consumed for transmission. AODV performs the worst as a consequence of sending out many control packets during congestion. DSR has larger average hop counts and more control packets than other geographic base routing algorithms. SPEED-T only takes delay into account, which leads to longer routes. Figure 12 shows that SPEED has nearly the same power consumption as GF and SPEED-S when the network is not congested. Under such situations, SPEED tends to choose the shortest route and does not require any on-demand beacons. Under heavy congestion, SPEED has slightly higher energy consumption than GF and SPEED-S, mainly because SPEED delivers more packets to the destination than the other protocols when heavily congested.

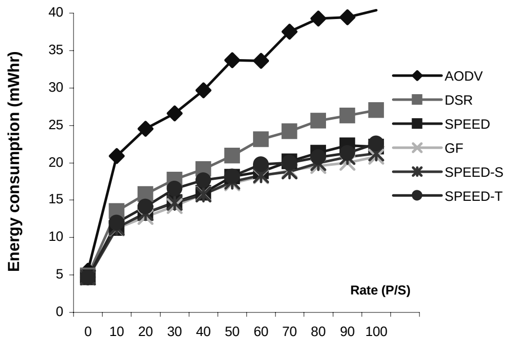


Figure 12. Energy Consumption for transmission

5.6. Void Avoidance

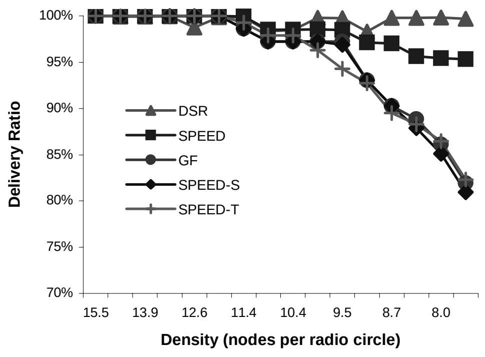


Figure 13. Deliver ratio under different density

This experiment tries to evaluate the end-to-end delivery ratio of all routing algorithms under different node densities. To eliminate packet loss due to the congestion, we only use four flows with a rate of 0.5 packets/second, these flows go from the left side of the terrain to the base station at the right side of the terrain. To change the density of the network, instead of increasing the number of nodes in the terrain, we keep the number of nodes constant at 100, and increase the side length of the square terrain in steps of 50 meters. It is no surprise that DSR performs best in the delivery ratio since it is a flooding based route discovery algorithm. Theoretically, DSR should have 100% delivery ratio (Figure 13) as long as the network is not partitioned. All other geographic based algorithms have 100% delivery ratio when the network has high density (>12 nodes / per radio range). However, when the network density is reduced below 9 nodes/ per radio circle, GF, SPEED-S and SPEED-T degrade performance rapidly. Only SPEED can manage to deliver 95% of its packets to the destination. However, SPEED drops 5% of its packets, because those packets need backtracking in order reach the destination. If backtracking, those packets would have a negative delivery speed, which is not allowed by SPEED for the sake of maintaining the real-time properties. It should be pointed out that GPSR[6], another well known geographic based routing algorithm, permits backtracking and can achieve 100% delivery rate as long as the network is not partitioned.

5.7. Implementation on Motes

We have implemented the SPEED protocol on Berkeley motes platform with a code size of 6036 bytes (code is available at [3]). Three applications including data placement, target tracking and CBR are built on top of SPEED. Due to space limitation, we only present partial results here. In the experiment, we use 25 motes to form a 5 by 5 grid. To evaluate the load balance capability of the SPEED, we send a CBR flow from node 24 to node 0 which is the base station. We collect the number of packets relayed by intermediate motes (1~23) and compare this with the result obtained from GF protocol which we also implemented on the motes.

GF tends to relay packets via a fixed route which leads to unbalance traffic, for example, in Figure 14, node 14 sends out 98 packets while node 13 doesn't sent out any packets. SPEED uses non-deterministic forwarding, which can balance energy consumption. We argue that in sensor networks, balanced energy consumption can prevent some nodes from dying faster than others, therefore increasing the network lifetime.

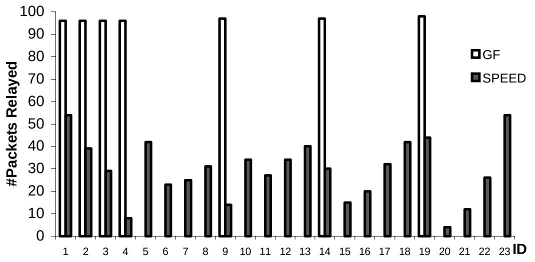


Figure 14. Traffic Balance

6. Conclusion

Many excellent protocols have been developed for ad hoc networks. However, sensor networks have additional requirements that were not specifically addressed. These include real-time requirements and nodes which are severely constrained in computing power, bandwidth, and memory. SPEED maintains a desired delivery speed across the network through a novel combination of feedback control and non-deterministic QoS-aware geographic forwarding. This combination of MAC and network layer adaptation improves the end-to-end delay and provides good response to congestion and voids. Our simulations on GloMoSim and implementation on Berkeley motes demonstrate SPEED's improved performance compared to DSR, AODV, GF, SPEED-S and SPEED-T. SPEED is a new protocol that meets the requirements of sensor networks in real-time situations.

7. Acknowledgment

This work was supported in part by the DAPRPA IXO offices under the NEST project (grant number F336615-01-C-1905), the MURI award N00014-01-1-0576 from ONR and NSF grant CCR-0098269.

References

[1] G. S. Ahn, A. T. Campbell, A. Veres and L.H. Sun. SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks, In *Proc. IEEE INFOCOM'2002*, June 2002.

[2] G. G. Finn. Routing and Addressing Problems in Large Metropolitan-scale Internetworks. *ISI/RR-87-180, USC/ISI*, March 1987.

[3] T. He, L. Gu, B.Blum, Jun Xie. Nest Project Source Code http://sourceforge.net/projects/vert/

[4] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System Architecture Directions for Network Sensors. *ASPLOS 2000*.

[5] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, Chapter 5, pages 153-181, Kluwer Academic Publishers, 1996.

[6] B. Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *IEEE MobiCom*, August 2000.

[7] Y.B. Ko and N. H. Vaidya. Location-Aided Routing(LAR) in Mobile Ad Hoc Networks. In *IEEE MobiCom 1998*, October 1998.

[8] J. F. Kurose, K. W. Ross. Computer Networking A Top-Down Approach Featuring the internet. ISBN 0-201-47711-4 Addison Wesley Longman Inc.

[9] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He. RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks, *In IEEE RTAS 2002*, September 2002.

[10] C. E. Perkins and E. M. Royer. Ad-hoc On Demand Distance Vector Routing. In *WMCSA'99*, February 1999.

[11] C. E. Perkins and P. Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for Mobile Computers, in *SIGCOMM Symposium on Communications Architectures and Protocols*, pp. 212-225, September. 1994.

[12] J. A. Stankovic, T. He, T. F. Abdelzaher, M. Marley, G. Tao, S. Son, and C. Lu. Feedback Control Scheduling in Distributed Systems, *IEEE RTSS*, December 2001.

[13] I. Stojmenovic and X. Lin. GEDIR: Loop-Free Location Based Routing in Wireless Networks, *IASTED Int. Conf. on Parallel and Distributed Computing and Systems*, November 3-6, 1999.

[14] A. Woo and D. Culler. A Transmission Control Scheme for Media Access in Sensor Networks, In *IEEE MobiCOM 2001*, July 2001.

[15] X. Zeng, Rajive Bagrodia, and Mario Gerla. GloMoSim: a Library for Parallel Simulation of Large-scale Wireless Networks. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulations -- PADS '98*, May 26-29, 1998.

MAC Layer Abstraction for Simulation Scalability Improvements in Large-scale Sensor Networks

Tian He[†], Brain Blum[*], Yvan Pointurier[*], Chenyang Lu[‡], John A. Stankovic[*], Sang Son[*]
[*]Department of Computer Science, University of Virginia
[†]Department of Computer Science and Engineering, University of Minnesota
[‡]Department of Computer Science and Engineering, Washington University in St. Louis

Abstract— **It is risky to implement a large-scale wireless sensor system without predicating its behavior beforehand through simulation, an indispensable step toward the final system. Since in wireless simulation, over 90% simulation time is spent on the characterization of the MAC layer behaviors, abstraction at the MAC layer becomes the key to the scalability of wireless sensor network simulations. In this work, we employ an abstract simulation mode, established by online data collection, to describe the MAC behaviors efficiently. Observing that the abstract simulation mode becomes inaccurate when network traffic changes significantly, we adopt a hybrid approach. We switch between the abstract simulation mode and full simulation mode in response to the traffic changes. Statistics are collected online under full simulation mode to update the abstract simulation mode to be used. In evaluation, we demonstrate that our approach is 9 times faster than the original simulation with a loss of accuracy less than 10% in the end-to-end delay and 1% in the number of messages successfully transmitted across the network.**

I. Introduction

With the advance of sensor network research, numerous protocols have been proposed recently. Validation and analysis of these protocols are very important for system designers to make right decisions to meet application requirements. One method available for validation and analysis is mathematical modeling. Unfortunately work in mathematics is limited due to the complex, dynamic, and unpredictable nature of these systems as they scale to thousands of devices. One can also explore new protocols for sensor networks directly with the physical devices (e.g., mica2). Problems occur again with the complexity and scale of these networks. The sensor devices that exist today provide a limited subset of the functionality and scale. Due to the practical limitations of mathematical analysis or physical network implementations, simulation is an important tool for researchers in this field, at least during the initial stage of the system design. Aside from just providing a means of representing models where it was previously impossible or just not practical, simulation also allows fast evolution of protocols without the significant effort of reworking a mathematical analysis or reloading code onto thousands of physical motes. The need for simulators capable of modeling these large, complex, and seemingly limitless networks grows every day. Although many simulators have been built to date (ns-2 [8], GloMoSim [20], SSF [18], SensorSim [16], TOSSIM [10], SENSE [3], J-Sim [6], ATEMU [15], SENS [19], TOSSF [14]), the scalability of these tools is not satisfactory. For example ns-2, currently the most popular network simulator, suffers an order of magnitude performance penalty by invoking the Tcl interpreter during a simulation run and occupies approximately gigabytes memory to simulate thousands of nodes. This limitation in scalability is a problem as networks being tested grow and simulators are left with the task of modeling the incredibly complex behavior of these large systems.

One method of improving the scalability of network simulators is through model abstraction. *The goal of model abstraction is to sufficiently reduce the complexity of a model, without suffering (too great) a loss in accuracy*. With this thought in mind, our work is an attempt to study the feasibility of using abstraction as a solution to the problem of scalability. Specifically, we are targeting to MAC layer abstraction, because based on our empirical profiling results, over 90% simulation time is spent on the characterization of the MAC layer behaviors.

The originality of our approach lies in three aspects: First, to the best of our knowledge, all prior work on data collection for abstraction occurs off-line prior to simulation. In contrast, our abstraction is established on-line by analyzing the data collected from a full simulation mode. This approach provides a high fidelity than off-line static abstractions. Second, we dynamically switch the granularity of the simulation based on the feedback from the traffic rate monitor. This approach embraces the accuracy gain from the detailed simulation and speedup from high-level abstraction. Third, our approach is independent of upper layer implementations. Different network protocols and applications can run on top of our implementation seamlessly. we demonstrate that our approach is 9 times faster than the original simulation with a loss of accuracy less than 10% in the end-to-end delay and 1% in the number of messages successfully transmitted across the network.

The rest of paper is organized as follows. In section II, we discuss research to date that has pursued similar goals. Specifically we look at different solutions to scalability and identify the uniqueness of our solution. In section III we discuss the rationale behind our abstraction and the basic ideas behind our solution. Section IV provides detail on our implementation within the GloMoSim framework and section V follows with results and analysis. Section VI inspects the unresolved issues that needs to be addressed in the future work. We conclude in section VII.

II. State Of The Art

In search of a solution to augmenting scalability in wireless network simulators, we looked at prior work in two general areas of network simulation research. The first, achieving simulation scalability, gave us insight into prior strategy and potential solutions to help guide us towards our ultimate attempt at a MAC layer abstraction. The second, simulation validation, helped us understand the important aspects and techniques to validate our ultimate solution. Prior research in both of these areas are discussed in this section.

A. Scalability

To solve the scalability problem, modern research on network simulation has taken two major approaches: parallel simulation and simulation abstraction. The first, parallel simulation, has been implemented in architectures such as DaSSF [18]and GloMoSim [20] and has proven adequate in speeding up simulations on a limited scale. [11], [13] and [20] discuss the application and results of implementing parallel and distributed simulation across several processes to achieve speedup. The biggest advantage of parallel simulation is that speedup is gained without sacrificing the accuracy and granularity of the simulation. This research on parallel simulation environments and techniques is complementary to our work.

The second approach, called *model abstraction*, is the focus of our work. [1], [4] and [17] outline fundamental goals and tradeoffs to consider when simulating a network layer abstraction. Additionally these papers provide important insights into understanding the effects of model abstraction and testing for possible simulator invalidation. [5] looks at two abstraction techniques: centralized computation and abstract packet distribution. Centralized computation saves memory consumption and time by centrally computing protocol states to reduce the workload and complexity of performing these computations for every simulated node. Abstract packet distribution avoids link-by-link packet transmission by scheduling packets directly at the receiving end. This second technique is similar to our work in that it eliminates potentially unnecessary details during packet transfer. In our work we implement a similar, but less aggressive technique that results in a more detailed and hence more accurate abstraction. Hybrid simulation, as implemented in SensorSim [16], is another approach to achieving speedup through abstraction. This approach utilized data collected from a real system to apply statistical data to simulation models. Our approach is different from this technique in the sense that we use adaptive simulation granularity from fine detailed packet-level simulation and apply it to coarse abstract-level simulation. [1] uses a fluid-based approach to speedup simulation. In this solution, speedup is achieved by coarsening the representation of network traffic from a packet level granularity to a flow level granularity where closely related packets are substituted with a single packet. [12] abstracts details of an 802.11 MAC Protocol by substituting probabilistic packet arrival data and statistical packet arrival times to remove the details of the 802.11 MAC protocol contention phase. This research most closely resembles our own with the exception that data collection occurs off-line. This limitation prevents researchers from applying statistical data to situations that have not been previously encountered and makes dynamic traffic patterns difficult to model. Our research addresses this limitation by performing online data collection by monitoring traffic patterns to determine when simulated abstraction is appropriate.

B. Validation

Aside from the work designing and implementing our abstraction, we also studied model validation to better understand the overall effect and validity of our research. Validation tests such as the chi-square goodness of fit test [9] can be used to test whether the simulated model reflects the real situation. [7] and [4] discuss issues and techniques for model validation and analysis. Although various methods and techniques are proposed and discussed in these papers, our primary focus is on comparing the simulated results of our abstraction with the results of the high granularity simulation [12]. Since different networking scenarios have different impacts on simulated results, a limited set of case comparisons are not enough to claim the validation of any model. Rigorous model validation, specifically as it applies to abstraction, is left as an area for future research.

III. Methodology Overview

Accurately modeling wireless networks on the scale of hundreds of thousands of nodes has been and will remain a challenging problem for research. Despite the effects of Moore's law it remains impractical to handle simulations of large-scale networks over reasonable periods of simulation time to collect precise experimental data.

A. Design Goals

The array of simulators developed to date provide varying levels of detail, accuracy, scalability, modularity (flexibility), etc. Unfortunately the more detailed a simulation becomes, the less capable it is of scaling to large complex networks. It is crucial to decide what level of detail can be sacrificed in exchange for simulation speed and scalability. Since in the wireless scenario, nearly $90 \sim 95\%$ of simulation time is spent modeling details of the MAC and below, abstracting these layers can provide a large speedup while preserving the accuracy of upper-layer behavior. The higher level protocols are ideal for our abstraction because they are not concerned with MAC layer detail so long as the packet transmission delay remains consistent with that of a detailed model's behavior. More specifically, this work attempts to solve the speedup problem by abstracting out MAC layer implementation detail for the MAC protocols (e.g.MACAW [2]) in GloMoSim [20]. While the specific abstraction we implement is under GloMoSim, our concept of MAC layer abstraction is generic and can be applied across other simulators.

The goals of our work are four fold. 1) Primarily we want to increase the speed of the simulation, in turn increasing the potential for scaling to very large network models. 2) We also need to validate the results of our abstracted implementation against a more detailed and correctly implemented simulator. 3) Slightly less important but along the lines of validation is the need to fully understand the effects of our abstraction on the model being simulated. The importance of understanding both the positive and negative impacts of our abstraction allows us to continue to refine our technique and potentially apply it to other layers or on other MAC layer protocols in future research. 4) Finally our fourth goal is to identify additional bottlenecks in simulations in hopes of addressing these bottlenecks in future work.

B. Model Assumptions

We assume that the current wireless simulators (e.g. GloMoSim [20], ns-2 [8]), prior to our modifications, is a valid representation of the MAC layer and therefore the baseline simulation is used as a means of comparison. Although the success of our abstraction does not hinge on a valid initial implementation, we assume a valid model so that similar implementations on other validated simulators can expect similar results. For our design we also impose the assumption of relatively consistent network traffic. This assumption is made to ensure that enough time is spent abstracting MAC layer behavior. We discuss the relaxation of this assumption later. Currently our model assumes that the simulated network is static or with a low mobility. With a high mobility, it is difficult to characterize the dynamics of radio propagation and its effect on our abstraction, so further investigation on high mobility is left for future work.

C. Overview of Detailed MAC Layer Simulation

Before discussing specifically how we added abstraction to the MAC layer simulation, it is important to understand certain aspects of the MAC layer simulation. To illustrate the simulation process, here we use GlmoSim [20], a typical discrete wireless simulator, as an example. We note this paper is described in the context of GloMoSim, however the design idea can be applied in other discrete simulators as well. GloMoSim was developed as a modular library of components that contribute to an extensible, robust, and dynamic simulator for wireless networks. By isolating nodes' communication layers into independent modules, GloMoSim allows the researcher to "plug and play"different protocols (i.e. protocols that they develop and implement) without concern for the inner workings of other architectural layers. To handle the overall organization of the simulator, GloMoSim implements a main module responsible for instantiating and organizing nodes, scheduling messages between modules, and tracking the exchange of messages within the simulation. The most important responsibility of the main component is to invoke calls to specific modules as appropriate to control the overall sequential flow of events throughout the simulation. This flow of control is handled by scheduling and passing *messages* which represent events in the simulation. A message, as defined by GloMoSim, has a multitude of purposes. For example, a message used for simulating communication between the network and radio layer not only encompasses the application payload, but also contains any header information that previous layers (application, transport, network, etc..) have attached to this payload. For communication messages, this mimics the way packets are packaged in the OSI seven-layer architecture. Aside from their obvious use for communication, messages are used to schedule node timeouts, handle mobility, or provide any form of communication between modules during simulation. To better understand the use of messages within GloMoSim we provide a simplified example of a packet being sent into the simulated environment. When node A's network layer has a packet to send over a single hop, the network layer schedules a message (encompassing the packet and upper layers' header information) to the MAC layer. The simulator core stamps this message with a time of delivery and returns to the main sequential flow in the main module. When the simulation time for this message arrives, the main module looks at its type and passes it to the appropriate node's MAC layer. Upon receiving the message, the MAC layer further decodes the type of MAC layer message so that it can be handled appropriately. In this situation, the MAC layer recognizes that it is the beginning of a new exchange to the network, so it initiates several more messages. These messages are also scheduled to take place at specific simulation times and are later handled by the main module when it becomes appropriate. The subsequent message scheduled is a notification of message propagation from the Radio layer. This message results in *every node* within the simulator receiving the propagated message and scheduling messages for overhearing, responding when appropriate, backing off, etc. As you can see from this significantly simplified example, due to the broadcast nature of wireless communication, a packet being sent out into the network involves the scheduling of two to possibly 20 or more GloMoSim internal messages per simulated node. This overhead in terms of simulation time escalates when a network has hundreds of thousands of nodes.

D. Architectural Solution

To improve simulation scalability for this described architecture, we attempt to abstract a layer of the previously defined GloMoSim architecture to reduce the number of messages exchanged during a simulation. Due to the fact that many sensor network simulations attempt to understand network, transport, or even application layer behavior, our abstraction argues that the details of how MAC layer message exchange takes place are not quite important so long as the overall transmission behavior of the model is maintained. Specifically this allows us to abstract the MAC layer and therefore reduce the MAC layer messages being modeled to speed up the overall simulation time. As shown in Figure 1, our solution adds four essential modules and one orthogonal module to the GloMoSim simulator. The four essential modules are a Data Collector module, an Abstraction Model Builder, a Traffic Monitor and A Mode Switcher. The orthogonal module is simply the addition of data collection for model validation. To better understand these changes, Figure 1 shows an architectural schematic of how our changes fit into a simulator architecture.

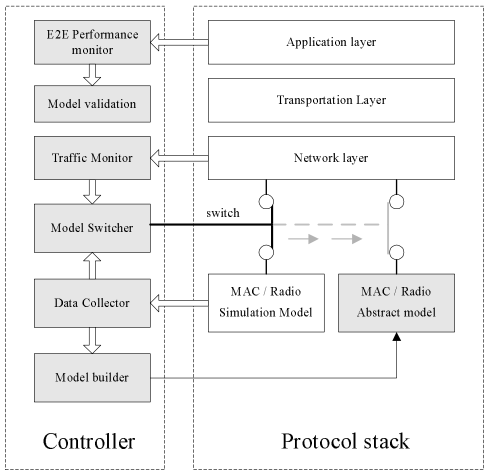


Fig. 1. Architecture for MAC layer Abstraction

The description of this architecture is as follows.

- **Data Collector:** In Figure 1, the Data Collector module is used to monitor and collect information on the detailed and complete exchange of packets as specified in the protocol. The Data Collector is actively collecting data when simulator is in the Full Simulation Mode. During this mode the simulator functions as previously implemented by the GloMoSim developers. By collecting MAC layer message data during the simulation, we attempt to use this collected data to apply a statistical model that removes this MAC layer message exchange.
- **Model Builder:** Upon switching to Abstract Simulation Mode, the Model Builder utilizes information previously collected by the Data Collector to determine an applicable statistical model for our abstraction. Currently the Model Builder implements a simple statistical history-index model, described in section IV-B. Once a switch is made, the Model Abstract Simulation Mode utilizes the statistical model to abstract the message exchange with a predicted delay and corresponding probability of successful packet arrival at the receiving node. This abstraction significantly reduces the simulation time.
- **The Traffic Monitor and Mode Switcher:** they are provided to actively monitor and respond to simulated traffic flow during a simulation. These modules monitor packet rates in the network and utilize a threshold-based method of deciding when to switch from Full Simulation Mode to Abstract Simulation Mode and back. The combined effort of these modules is referred to as *Toggling*.
- **Model Validation Module:** The fifth module in our implementation, data collection for model validation, is implemented across the End-to-End Performance Monitor and the Model Validation Unit. Currently our End-to-End Performance Monitor collects statistics on end-to-end message delay and the Model Validation unit has not been fully implemented. The current solution is discussed further in section IV-D and work with online Model Validation and the addition of a feedback mechanism for dynamic optimization of abstraction parameter is left for future work.

It is important to note that our MAC layer abstraction subsequently abstracts the details of the radio layer for all packets. This happens as a result of the packets no longer being sent as a radio signal. Instead, packets are directly passed to the receiving nodes' routing layer avoiding both the MAC and radio layer altogether.

E. Abstraction Parameters

The mode switch can be tuned by changing up to three abstraction parameters. 1) The *initial stride* is the number of packets that are monitored at the beginning of a simulation during Simulation Mode. When this number of packets has been simulated, the mode switches to Abstract Simulation Mode (on figure 1 the switch position would be to the right). 2) The *heartbeat period* controls the frequency of traffic monitoring. The "packet arrival rate"is the rate at which (unicast) packets are transmitted from the network layer to the MAC layer for all simulated nodes. Each time a heartbeat is issued, we compare the arrival rate in the time period that just finished with the arrival rate in the previous time period. If (the absolute value of) this ratio exceeds a *threshold* (our third abstraction parameter), the packet arrival rate is considered fresh and we no longer consider past data applicable to the current abstraction and a switch back to full simulation mode is needed. The abstraction parameters: (i) initial stride, (ii) heartbeat period and (iii) threshold, are static during a simulation run and easily modifiable. Intuitively, a longer initial stride, shorter heartbeat, and smaller threshold result in a more accurate, and therefore longer simulation. This tradeoff is discussed in section V-A.

IV. Implementation

Our initial approach to abstracting a MAC layer protocol expose us to the design and code of various simulators. After looking into ns-2 [8], GloMoSim [20], SSF [18], and the TinyOs simulator [10], we eventually choose GloMoSim due to its ease of understanding, relatively good performance, and modularization of layers.

As stated before, our MAC layer abstraction toggles between two different modes during a simulation. The Full Simulation mode implements the detailed packet exchange as previously implemented by GloMoSim. The only change to the previous implementation is the collection of data to be used in the Abstract Simulation Mode. The Abstract Simulation Mode reduces the number of messages exchanged during simulation and therefore has a vast impact on the overall simulation time.

To switch between modes we implemented what we call a Toggle feature. The final piece of our implementation involves data collection for analyzing the effect of our abstraction on end-to-end delay and packet loss for the models chosen. These four components are described in detail below.

A. Data Collection

In Full Simulation mode the simulation runs exactly as it had previously been implemented by the GloMoSim designers. For the purposes of this paper we are going to assume that this original design was correct and can therefore be used as the baseline for our analysis. The only difference in Full Simulation mode is that we have inserted several data structures in combination with a fair amount of logic to eavesdrop and collect data for the detailed simulation as it runs. This eavesdropping takes place in both the MAC layer, Radio layer. For example, the main body of the simulation ruling MACAW protocol [2] goes as follows : When the network layer has a packet to be sent to its next hop, the MAC layer is notified of this packet and a RTS is scheduled. At this point we collect the time in which the RTS was initiated and the node in which the RTS is destined and we set the sending nodes state as SENDING RTS and the receiving nodes state as AWAITING RTS. From here on the Data Collection component tracks the exchange of packets between nodes, updates the node's state accordingly, and sets variables within the Data Collection structure that records how long the exchange took and whether or not the exchange was successful. Any collisions with other RTS, CTS, Data packets or other noise are handled by the simulation and recorded as appropriate. Finally when this MAC layer exchange has completed the nodes state is reset and its collection data index is updated as appropriate.

B. Model Abstraction

At some point during the Data Collection portion of our simulation, the toggle feature of our implementation realizes that enough data has been collected and it is time to switch to abstraction. The implementation of the toggle feature follows this description of the Abstract Simulation model. The Abstract Simulation Mode is used to significantly reduce the number of messages sent during the simulation and therefore its goal is to reduce the overall simulation time. We do this by applying the data collected during the Full Simulation mode to provide a statistical approach to generalize data flow through the network. This is possible under fairly regular traffic patterns or sending intervals per node. An example application is a temperature sensing application where nodes report temperature readings to a base station every second. We statistically determine the send times and whether or not a packet successfully arrives by generating a random number in the range of the number of data exchanges previously collected. Using this random number, we index the data structure where this information resides. If the index into our data structure points to a successful exchange then we use the time of this exchange and directly send our current packet to the network layer of the receiver with a set delay appropriate to the time it would have taken (statistically speaking) for this packet to arrive. This time includes the radio transmission time, radio propagation time, overhead time incurred during message exchange between layers, and any back-off that occurred due to collisions in the network. If our index points to a data packet that had previously been dropped, we drop this packet accordingly. These dropped packets statistically model packet loss in the model. While our abstraction methodology is fairly simple, it works for several reasons. Primarily we assume fairly constant traffic which allows us to look at behavior from the past to determine current behavior. Although this assumption seems stringent, it is relaxed by applying strict monitoring of aggregate packet rate by our toggle component which can be modified by changing the abstraction parameters. As you will see, the worst case scenario for our simulation is when the smooth traffic assumption is relaxed and nodes sporadically send messages changing the networks aggregate packet rate. In this scenario our model realizes that the changing rate requires fresh data and continues to toggle back to Full simulation mode with Data Collection. When this happens the simulation runs as it had prior to our implementation with the exception that we incur the additional cost of data collection and therefore see slight performance loss.

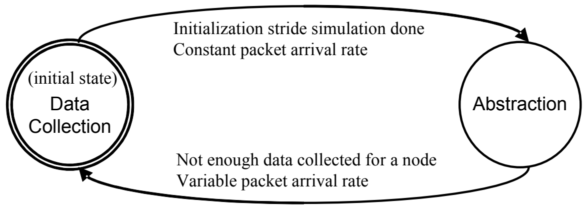

Initialization stride simulation done
Constant packet arrival rate
(initial state)
Data
Collection
Abstraction
Not enough data collected for a node
Variable packet arrival rate

Fig. 2. Phase Transition

C. Toggling Between Modes

The problem of switching between Full Simulation mode and Abstract Simulation Mode is handled by a toggle component that is responsible for monitoring traffic flow and data collection. The state transition diagram for our Toggling component is shown in figure 2 and a visual representation of the Toggling behavior is provided in figure 3. Specifically the toggle component functions as follows. While in Full Simulation mode the toggle component monitors the total number of data packets collected. When the total packets collected reaches some threshold, (the abstraction parameter previously discussed) the toggle component modifies a state parameter which tells the MAC layer to use our Abstract Simulation Model as implemented.

Once the simulation is running in Abstract Simulation Mode, the toggle component begins to monitor the aggregate packet transfer rate provided by the network layer of every node in the network. The heartbeat counter is used to periodically check the aggregate rate of messages sent by comparing the total number of messages sent between the current time and the last heartbeat with the total number of messages sent between the two prior heartbeats. Because the time between heartbeats remains constant this value provides a rate. At each heartbeat these two rates are compared and if the difference between them exceeds some threshold (one of the abstraction variables previously mentioned) then the aggregate send behavior is determined to have changed significantly enough to warrant switching back to Full Simulation mode. Another case where our mode switches from Abstract Simulation Mode to Full Simulation Mode is when the model is in Abstract Simulation Mode and a node for which we do not have enough statistical data wants to send a packet. At this point we switch modes so that statistics for that node can be appropriately collected. Note that this deeply impacts the performance of the simulation as all nodes switch to Full Simulation Mode simultaneously. In our design we choose to implement the Toggling feature as a network parameter (as apposed to a node parameter) since nodes in abstraction do not send packets and as a result would invalidate data collected for other nodes simultaneously. One additional challenge we face in our implementation is when the toggling mechanism is invoked during a packet exchange in Full Simulation mode. Due to the complexity of a more thorough solution, our current implementation simply drops the packet in transit. For our simulation, these lost packets are rare and as a result do not have significant impact on our results.

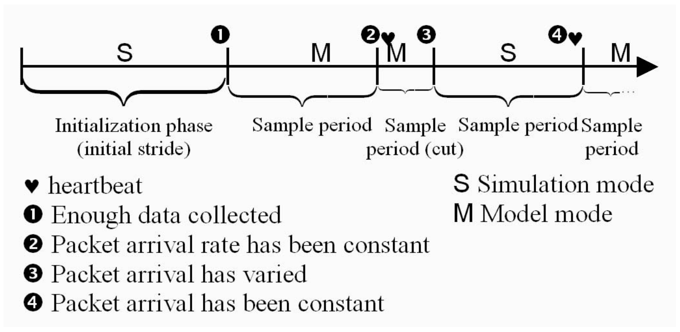

S
❶
M
❷♥
M
❸
S
❹♥
M
Initialization phase (initial stride)
Sample period
Sample period (cut)
Sample period
Sample period
♥ heartbeat
❶ Enough data collected
❷ Packet arrival rate has been constant
❸ Packet arrival has varied
❹ Packet arrival has been constant
S Simulation mode
M Model mode

Fig. 3. A hybrid Approach with Toggling

D. End-to-End Data and Model Validation

The final modification to GloMoSim is building a mechanism to collect information for comparison and validation purposes. GloMoSim provides information specific to each layer implemented in the simulation but unfortunately this information is not detailed enough to satisfy our needs. Our modification involves an addition to a component of the simulator that collects the end-to-end delay for all packets successfully sent through the network. By collecting the end-to-end delay for each packet between specific nodes, we are not only able to get the average and standard deviation of the end-to-end transfer time, but the node to node data loss (for our CBR implementation) allowing us to ensure that the MAC layer appropriately handles collisions and therefore congestion in the network. Since the end-to-end delay statistics are a necessary component for validating our abstraction, but are not necessary for the abstraction itself, we take measure to ensure that this additional code does not affect our speedup analysis. Fortunately since this code is necessary for our comparison between models, we implement these changes in both the original and abstracted simulators to provide consistent results.

V. Evaluation

In the evaluation, nodes in a sample network are uniformly placed in a field of 1000m x 1000m, in which 10 CBR flows between 30 nodes are transmitted during 900 seconds. Because these flows have random start times (0 to 2 seconds), random starting and end points (fixed for the duration of the simulation), and random rates (2 to 1000 packets per second), some parts of the network are more active than others and the load on each node changes with time. By using several CBR flows with randomness in their parameters, we create a more general load for the network. To test the effects of our abstracted simulator on the model previously described we run the model several times with each simulated run taking place on different architectures of the GloMoSim simulator. We maintain a seed value of 1, allowing pseudo random numbers generated during the simulation to be consistent between each architecture simulated. The simulation times for the architectures described below are compared in Figure 4.

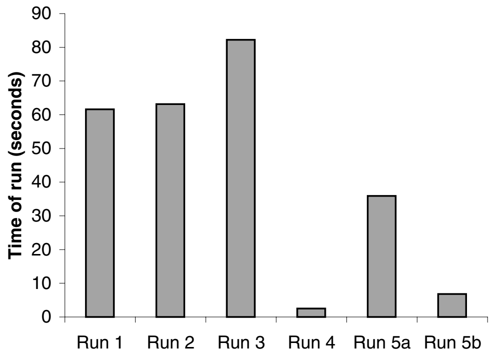


Fig. 4. Simulation Time under Different Runs

In a first run, we test the original GloMoSim architecture and measure the overall simulation time [1] We use this value as our reference time. For the remainder of our paper, we refer to this and the subsequently described runs as run 1, run 2, etc. In a second run, we use a modified version of the GloMoSim architecture that includes end-to-end delay data collection for all packets correctly transmitted at the application level. This architecture is also addressed as the "original simulation". In a third run, we use our fully modified version of GloMoSim with Toggling deactivated so that the simulator never enters our Abstract Simulation Mode. This simulation is used to determine the cost of online data collection. In a fourth run, we use our fully modified version of GloMoSim with Toggling and Data Collection deactivated and the simulator initially

[1]the simulations were run on a PC AMD Athlon 1.3 GHz with 768 MB of memory.

set to Abstract Simulation Mode. This architecture provides data on the maximum possible speedup. Finally in a fifth run (run 5a and run 5b), we use our fully modified version of GloMoSim as we have intended it to function with two different thresholds. We gather end-to-end delays for all packets correctly transmitted at the application level. We also call this run the "abstracted run".

A. *Abstraction vs. implementation trade-off*

To understand the effect of our abstraction on simulation time, we have run the five architectures previously described to obtain time comparisons for each architecture. As we can see from Figure 4, if the overhead of the end-to-end delay collector (run 2) remains low (2%), the overhead due to the data collector (run 3) is relatively high (30%). Fortunately, our assumption that most of the time spent in simulation is a result of MAC and radio layer details proves to be true. If messages from these layers were delivered directly from the network layer of a node to the network layer of the receiving node, simulation time would be 24 times faster. As a result, the overhead introduced by our implementation can be counterbalanced by the speedup incurred through abstraction. For our abstraction the simulation (run 5b) was 9 times faster than the original simulation and only 40% slower than the fastest possible run (run 4). This speedup occurred with a loss of less than 10% accuracy in the end-to-end delay and a 1% loss in the number of messages successfully transmitted across the network.

B. *Accuracy vs. time trade-off*

We assess the accuracy of our abstraction by monitoring a flow consisting of several hops through the simulated network. We choose to use multiple hops due to the fact that it allots more room for discrepancy between our simulated abstraction and the original GloMoSim architecture. In addition to assessing simulation times between architectures, we also run our model over varied threshold values as previously described. In these cases the initialization stride is 2000 packets and the heartbeat period is 2 seconds with the threshold varied at 2% (case 5a) and 5% (case 5b).

C. *Threshold Comparison*

Figure 5 shows us end-to-end delays, the number of messages that arrive along our chosen flow, and the overall simulation time for both the original simulation and our abstracted run (run 2, run 5a, run 5b). We can see that the end-to-end delays viewed by the application layer help to validate our abstraction (less than a 10% difference). From this figure we can also see a tradeoff between end-to-end delay accuracy and simulation time as a result of the different values for our threshold parameter. This is the result of an increased threshold allowing more variance in traffic rate prior to switching back to Full Simulation mode. This greater threshold results in less switching and therefore a slight decrease in accuracy with a significant increase in speedup. Figure 6 and Figure 7 depict this behavior by plotting end-to-end packet delay as a function of the packet sent for both threshold values. The packet sent can also be looked at as a time parameter for this figure.

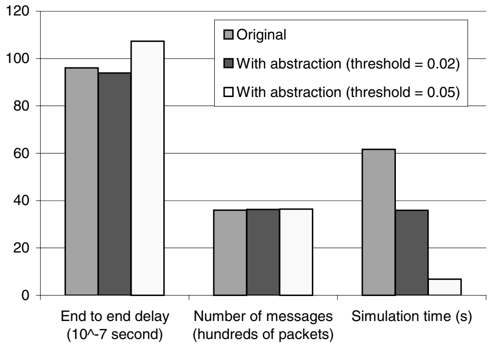


Fig. 5. Model Comparison (run 2, run 5a, run 5b)

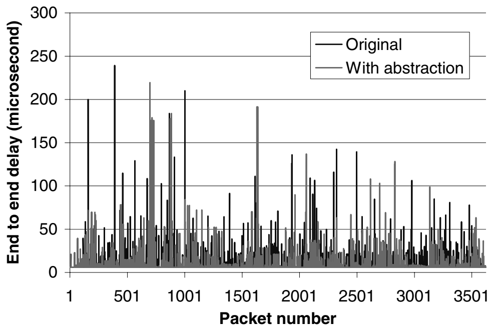


Fig. 6. End-to-End Delay for Threshold = 0.02

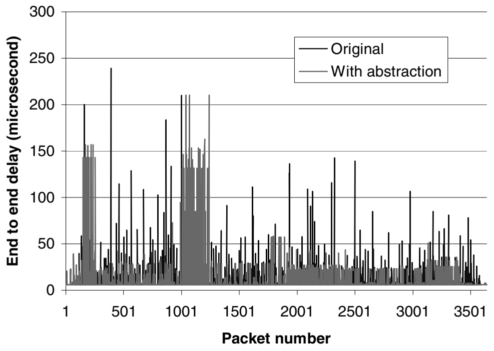


Fig. 7. End-to-End Delay for Threshold = 0.05

From these results we can see the importance of fine tuning our three abstraction parameters. These parameters allow for the adaptation of the simulator to more appropriately fit the model in simulation. Although we currently modify these parameters off-line, this issue could be re-solved by profiling the simulator or implementing an auto-profiling and auto-adaptation mechanism to handle parameter modification online. To do this the simulator would run an initial phase during which it tries to find optimal parameter values. The simulator could then change these parameters dynamically to adjust the quality of the abstraction. Ultimately, the end user could provide a single parameter (i.e. a value between 0 and 10) with 0 meaning optimize for speed and 10 meaning optimize for accuracy at the expense of speed. This could be implemented using a feedback control loop in the simulator with the speed and accuracy references as inputs. Note this is a second level of auto-adaptation as we already provide a mechanism for the modified simulator to respond automatically to load variations. Further discussion of online parameter optimization is left for future work.

VI. Future Work

Due to space constraints, several directions remain both unexplored and under explored. The following ideas and issues are left for future work:

- In our implementation we have only tested/validated our abstraction for small models implemented on MACA using CBR (constant bit rate). For further validation and to test the true success of our model we must continue to run tests on larger, more complex, and more dynamic models to better understand the effects of our abstraction.
- For our first Abstract Simulation Model we simply used a random index into an array of collected data to determine send time and the probability of a successful data exchange. This simple model abstracts data when traffic flow remains fairly constant and switches to data collection when traffic flow changes. The loss of speedup incurred by switching back to data collection could be reduced by applying a system identification model that more aptly handles dynamic traffic flow in our abstraction.
- Automatic model validation is another area left open for future work. In our implementation we used simple mathematical validation and manual control of abstraction parameters to determine the accuracy and effect of our abstraction. It is conceivable that feedback control and dynamic adjustments could be implemented to manipulate the abstraction parameters and optimize speedup online.
- Finally, our first iteration involved extending the MACA protocol as implemented by GloMoSim. Future work could involve similar abstraction to other MAC layer protocols such as 802.11 or MACAW to determine the extensibility of our design.

VII. Conclusion

Research on protocols for wireless ad-hoc sensor networks continue to reach new limits. To fully understand the impact of these protocols researchers must be able to simulate networks of hundreds of thousands to potentially millions of nodes. A major problem of current wireless network simulators is their inability to simulate networks of this scale. Research to date has taken several approaches to solving this problem. In this paper we present the novel design, implementation, and results of an online algorithm to switch the simulator between a *Full Simulation Mode* during which data are collected, and an *Abstract Simulation Mode*, which uses the collected data to abstract the simulator's MAC layer. This innovative design is implemented on the GloMoSim simulator to specifically abstract the MACA protocol and can easily be ported to work across other protocol layers on a variety of simulators. Our design can be utilized to speed up future simulations studying transport, network, or application layer protocols. In the evaluation, we achieved 9X speedup at the cost of only 10% loss in end-to-end delay accuracy combined with a negligible difference in the number of messages correctly exchanged through our abstraction. While these results are promising they are by no means the limit of this design. Future work on optimizing our design includes implementing alternate abstract simulation models and extending our work to include mobility and the more adequate handling of dynamic network behavior.

References

[1] J. Ahn and P. Danzig. Packet network simulation: Speedup and accuracy versus timing granularity. In *IEEE/ACM Transactions on Networking*, pages 743–757, 1996.

[2] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. Macaw: A media access protocol for wireless lan. In *the SIGCOMM '94 Conference on Communications Architectures*, 1994.

[3] G. Chen, J. Branch, M. J. Pflug, L. Zhu, and B. Szymanski. SENSE: A Sensor Network Simulator. *Advances in Pervasive Computing and Networking,Springer: 249-267*, 2004.

[4] J. Heidemann and et. al. Effects of Detail in Wireless Net-work Simulation. In *SCS Multiconfer-ence on Distributed Simulation*, pages 3–11, 2001.

[5] P. Huang, D. Estrin, and J. Heidemann. Enabling Large-scale Simulations: Selective Abstraction Approach to the Study of Multicast Protocols. In *USC/ISI, USC*, 1998.

[6] J-SIM. *The J-SIM Project.* Available at `http://www.j-sim.org/`.

[7] D. B. Johnson. Validation of Wireless and Mobile Network Models and Simulation. In *DARPA/NIST Workshop on Validation of Large-Scale Network Models and Simulation*, 1999.

[8] Kevin Fall and Kannan Varadhan. *The ns Manual.* Available at `http://www.isi.edu/nsnam/ns/`.

[9] J. L.Devore and et.al. Probability and Statistics for Engi-neering and the Sciences. In *5th Edition ISBN 0-534-37281-3 Duxbury Press New York, NY*, 2003.

[10] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, November 2003.

[11] J. Liu. Performance Prediction of a Parallel Simulator. In *PADS'99*, 1999.

[12] J. Liu, D. M. Nicol, L. F. Perrone, , and M. Liljenstam. Towards high performance modeling of the 802.11 wireless protocol. In *In WSC 2001*, 2001.

[13] T. Mineo and et. al. Impact of Channel Models on Simu-lation of Large Scale Wireless Networks. In *MSWiM*, 1999.

[14] L. F. Perrone and D. Nicol. A Scalable Simulator for TinyOS Applications. In *Proceedings of the Winter Simulation Conference, 2002*, 2002.

[15] J. Polley, D. Blazakis, J. McGee, D. Rusk, J. S. Baras, and M. Karir. ATEMU: A Fine-grained Sensor Network Simulator. In *SECON'04*, 2004.

[16] A. S. S. Park and M. B. Srivastava. SensorSim: A Simulation Framework for Sensor Networks. In *MSWiM 2000*, 2000.

[17] A. F. Sisti and S. D. Farr. Model Abstraction Tech-niques: An Intuitive Overview. In *Air Force Research Laboratory/IFSB*, 2000.

[18] ssfnet. *The SSFNET Manual.* Available at `http://www.ssfnet.org`.

[19] S. Sundresh, W. Kim, , and G. Agha. SENS: A Sensor, Environment and Network Simulator. In *Proceedings of 37th Annual Simulation Symposium, 2004*, 2004.

[20] X. Zeng, R. Bagrodia, and M. Gerla. Glomosim: a library for parallel simulation of large-scale wireless networks. In *the 12th Workshop on Parallel and Distributed Simulations*, 1998.

Gradient-Driven Target Acquisition in Mobile Wireless Sensor Networks

Qingquan Zhang[1], Gerald Sobelman[1], Tian He[2]

[1]Department of Electrical and Computer Engineering, University of Minnesota, Twin cities, USA
[2]Department of Computer Science and Engineering, University of Minnesota, Twin cities, USA
{zhan0511,sobelman}@umn.edu
{tianhe}@cs.umn.edu

Abstract. Navigation of mobile wireless sensor networks and fast target acquisition without a map are two challenging problems in search and rescue applications. In this paper, we propose and evaluate a novel Gradient Driven method, called GraDrive. Our approach integrates per-node prediction with global collaborative prediction to estimate the position of a stationary target and to direct mobile nodes towards the target along the shortest path. We demonstrate that a high accuracy in localization can be achieved much faster than other random work models without any assistance from stationary sensor networks. We evaluate our model through a light-intensity matching experiment in MicaZ motes, which indicates that our model works well in a wireless sensor network environment. Through simulation, we demonstrate almost a 40% reduction in the target acquisition time, compared to a random walk model, while obtaining less than 2 unit error in target position estimation.

Key words: Wireless Sensor Network, Navigation, Localization, Probabilistic Model, Rescue

1 Introduction

Wireless sensor networks have gained extensive attention in many applications such as tracking, differentiated surveillance, and environment monitoring [1–3]. Moreover, the hybrid systems of mobile objects (e.g. Robots) and sensor networks create new frontiers for civilian and military applications, such as search and rescue missions in which the background environments are inaccessible to humans. A heterogeneous searching team consisting of robots and a wireless sensor network has greater advantage, considering its distributed computation and navigation capability achieved through the cooperation of embedded wireless sensor networks.

Although the applications of mobile sensor networks keep diversifying, several underlying capabilities remain fundamental and critical. In this work, we focus on the target acquisition – finding the locations of stationary targets using mobile sensor nodes. The challenging problem we address in this work is *to navigate a team of mobile sensor nodes toward the stationary targets fast and accurately while consuming the least amount if energy and other resources.* In this emerging research arena, most research groups employed static wireless sensor networks to navigate the mobile sensor nodes. Tan in [4] used distributed static sensor networks to collect the data and execute local calculations to generate a path for a mobile sensor network to move toward the goal. Although the in-network calculation implemented in that project was quite efficient in creating the shortest routing path, the additional requirement of a stationary distributed sensor network sets a barrier for rescue applications, because of the high cost to cover a large geographic area with a large number of sensors. Other research groups [5] proposed gradient methods in which the mobile wireless sensor nodes move toward the gradient direction assuming that targets carried the most intensive strength of interested signals. However, in all of their implementations, the assistance of a stationary wireless sensor network was assumed to be available in generating a local signal distribution map. A probabilistic navigation algorithm is presented in [6], where a discrete probability distribution of vertex is introduced to point to the moving direction. This algorithm computes the utilities for every state and then picks the actions that yield a path toward the goal with maximum expected utility. The shortcoming of this method is that it requires the arrival of a mobile sensor node to localize the target position and significant communication overhead is introduced by the iteration process.

2 Contribution

In this paper, we propose to compensate those deficiencies by incorporating a prediction model of real-time processes into a mobile sensor network sensing and navigation architecture. We are interested in the mutually beneficial collaboration of the algorithms described above but seek to reduce the costs and provide faster target localization. *The novelty of our approach is the seamless integration of a per-node prediction model with a global prediction model.* The per-node prediction model guarantees that a mobile node can acquire the position of a target alone, while the global prediction significantly reduces the navigation overhead and time, if collaboration among the nodes is available. Specifically, the main contributions of our prediction models are:

- Our model provides more meaningful description of individual sensor readings in term of accuracy and confidence.
- Our model works with a single mobile sensor node as well as a swarm of mobile sensor nodes. In the latter case, the sensor nodes have the ability to share local information in order to draw a global picture, which helps each sensor node to acquire the target along a significantly shorter path.
- The in-network prediction algorithm enables faster yet accurate target position acquisition: sensor nodes would be required to reach the target only

when the model prediction is not accurate enough to satisfy the requirement with an acceptable confidence. This allows a significant reduction in navigation energy.

The remainder of this paper is organized as follows. Section 3 defines the assumptions. Section 4 overviews the design. In Section 5, we present the in-network per-node prediction model. Section 6 describes target acquisition in the context of global prediction and the corresponding mobile sensor node navigation protocol. Section 7 presents empirical data obtained from the MICAZ platform as well as simulation results. Finally, in Section 8, we present our conclusions and future works.

3 ASSUMPTIONS

Our design is based on two assumptions: network connectivity and the self-localization of mobile wireless sensors.

- **Connectivity:** First, wireless sensor nodes in the network are assumed to be able to ensure connectivity. Individual mobile sensor nodes deployed in large area is likely to lose connection to a central base station, if the routing information is not updated. Therefore, it is desirable to maintain connections across a team of mobile sensor nodes while minimizing power consumption and allowing the sensor nodes to achieve their individual goals.
- **Node Self-Localization:** The second assumption hinges on the localization availability for a mobile wireless sensor network. If a mobile sensor node enters an unknown area, it must be able to specify its own location dynamically without a map. This location can be obtained either through GPS such as used in ZebraNet [7] and VigilNet [3]. It can also use a dynamic localization scheme [8] that adjusts the estimated location of a node periodically based on the recent observed motion.

4 OVERVIEW OF PREDICTION MODEL

The objective of our GraDrive target acquisition scheme is to predict the location of stationary targets within allowable uncertainty (or a confidence level) dictated by a rescue plan. To illustrate the design of GraDrive, we start our description with a rescue scenario shown in Fig. 1. Here we note that our method is independent of this rescue application and can be applied in other scenarios as well.

- **Objective:** The control center (base) disseminates a search objective to a mobile sensor network with two parameters, *error tolerances* and *confidence level* of the target, specifying the quality of target acquisition. For example, the objective would be locating a target within 2 meters with at least 95% confidence. The tolerance levels for each mobile sensor nodes can vary correspondingly in case different nodes are designed for different purposes.

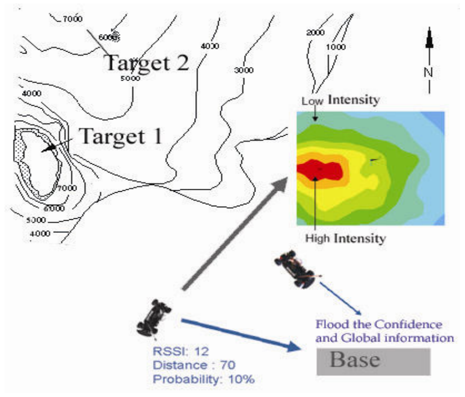


Fig. 1. The Architecture schematic of GraDrive

- **Individual Prediction Model:** Once the search objectives are received by the mobile nodes, individual node decides their most efficient way to locate the potential target with the requested confidence *individually*, using the per-node prediction model. It starts to move toward the direction in which it anticipates the fastest path to reach the confidence.
- **Collaborative Prediction Model:** In addition to its own plan and navigation, sensor nodes also report back to a base station, where all the individual nodes' readings and plans are collected and computed to create a global map and an uncertainty area. If computation results show probability increase by certain interval, e.g.5% to its previous state computation, the base station will disseminate the global prediction value over the network so that each sensor nodes in network can update their model. In other words, the prediction result based on collaborative information overrules the results from the individual prediction model.
  As demonstrated in Fig. 2 from (A) to (D), the individual sensor node continuously predicts the target position with increasing probability and move toward the target, the uncertainty area where the target is located shrinks through collaboration among mobile sensor nodes. If collaborative probability calculated reaches the dictated objective, a success of rescue plan is achieved. The position it reports is the exact target position specified. Compared to other static sensor node navigation plans, the prediction results computed by our model still provide considerably more information than MobileRobot [6] and SafeRobot [9].

5 GRADRIVE MODEL DETAILS

In this section, we formally describe our per-node prediction model to estimate the position of a stationary target with certain confidence. This per-node prediction model forms the basis for global collaborative prediction described in Section 6. We note even though we consider an unknown area with multiple

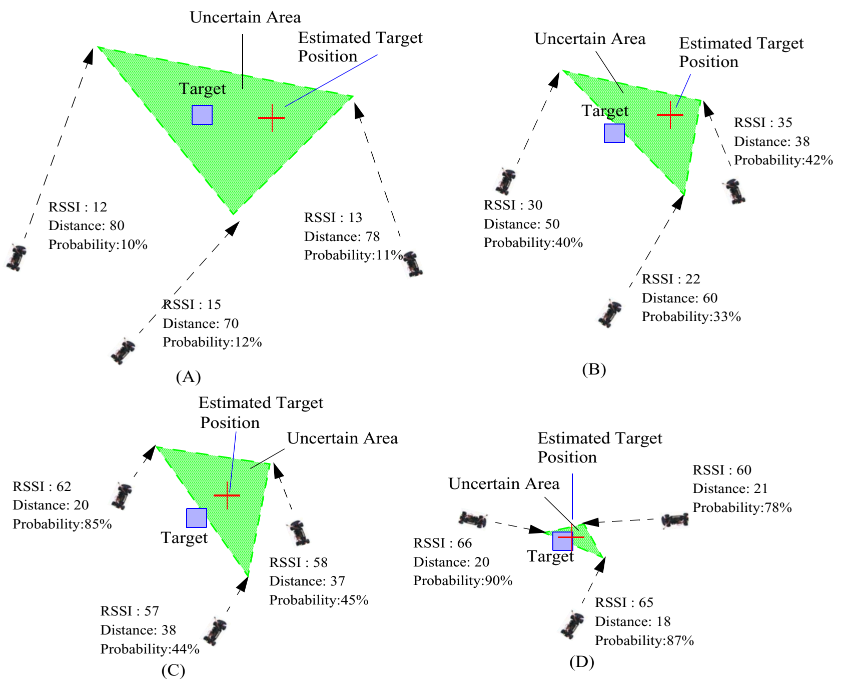


Fig. 2. Collaborative Prediction Scheme of GraDrive

targets, the searching for separate targets is independent to each other as long as the field (RSSI) generated by one target doesn't overwhelm that generated by others. Therefore in the remaining of paper, we focus on only single target acquisition problem.

5.1 Prediction Problem Formulation

Conventionally, we begin with a value-prediction problem, which creates a Received Signal Strength Indicator $F(\theta)$ over a parameter set θ. For example, if $\theta = (d, t, v)$, RSSI is related to d, the distance between a mobile sensor node and the target, t, the time of sampling, and v, the speed of mobile sensor nodes. This model can be established by getting consecutive sensing readings (system states) when a mobile sensor node moves. Typically, the number of parameters in θ is much less than the number of states collected and changing one parameter changes the estimated value of many states. To approximate our model appropriately, we seek to minimize the mean squared error over some distribution, P, of the inputs.There are generally far more states than components in θ. The flexibility of the function estimator is thus a scarce resource. Better approximation at some states can be gained generally only at the expense of worse approximation at other states.

5.2 Distance Prediction Model

In GraDrive, we extend the familiar one-dimensional normal probability density function known as Gaussian distribution to two variants multidimensional distribution. The predicted distance from sensor nodes' current position to predicted target position can be queried or estimated from the model. The multidimensional Gaussian distribution function over two attributes, trust interval and RSSI, can be expressed as a function of two parameters: a 2-tuple vector of means, μ, and a 2×2 matrix of covariances,Σ. Further, we assume the trust interval set by a rescue team is independent of the RSSI received, which means the trust interval of the predicted distance estimation T_i to the mean of historic results μ doesn't change dynamically along the searching process. The two dimensional distribution can be separated for description purposes. Without loss of generality, it is assumed that the predicted distance d is disproportional to RSSI, that can be expressed as $d = r_1/RSSI + r_2$, where r_1 and r_2 are two adaptive parameters that can be determined before the searching process. We note here other RSSI attenuation models can be used here as well without invaliding our approach. We then use historical data or experience data to construct the models, providing r_1 and r_2 at each RSSI value appropriately. Besides offering the predicted distance, a probability model associated the d is also constructed to provide confidence of the prediction, e.g. given a predicted distance of 2 feet, the confidence for this prediction is 95%. The models must be trained before it can be used, a general limitation for probabilistic model. The accuracy of the model, therefore, relies on the accuracy of data used to train it. Once the initial model is constructed, each sensor nodes can query the predicted distance map from saving model and come up with a confidence value. One distribution of the distance d against the confidence p over one RSSI is a Gaussian distribution. Suppose that rescue team have set a trust interval of T_i, given the distribution of distance over one RSSI, we can get the points d_i that satisfied that $P(d_i) - P(u) <= T_i$. Here we emphases that if the trust interval is too small, the amount of data needed to train the model will increase exponentially.

5.3 Signal Strength Distribution Prediction Model

Besides obtaining the distance d information based on measured $RSSI$, we can further refine the RSSI distribution Model. This distribution model can then be used to navigate the mobile sensor network toward the target at a shortest path. The central element in our approach is to construct a prediction model that represents attributes as accurate as possible in a mobile sensor network. As we discuss above, if the predicted RSSI distribution function depends on parameters including distance d and confidence or probability p, the function can be expressed as $F(d, p)$ considering d and p's distribution are independent. If we do the Tylor expansion on function F, a polynomial function of attributes d and p is achieved, shown as

$$F(d, p) = f(d_0, d_1, d_2...)f(p) \tag{1}$$

where d_i is the function of distance variable d. To reduce the computation energy consumption, only second order polynomial is considered in our case, which offers a 3-tuple vector of $D = [d_0, d_1, d_2]$:

$$\begin{aligned} d_0 &= \mathrm{c}_0 \\ d_1 &= 1/(d + c_1) \\ d_2 &= 1/(d^2 + c_2) \end{aligned} \tag{2}$$

where c_1, c_2, c_3 are constants used to avoid singularity when $d = 0$. Now we can define our gradient distribution function into a simple format as:

$$F = D \bullet A \bullet p \;\; where \; A = [a_0, a_1, a_2] \tag{3}$$

Equation 3 is our probabilistic gradient distribution prediction function for attributes of d and p. suppose that each sensor nodes observe the value of attribute D_j to be d_j, we now input sensing reading into a vector of D_j. Thus the vector D is extended as a matrix.

If enough sensing samplings are provided, we can apply non-linear Least Square Fitting to estimate the parameters A. For nonlinear least squares fitting to our undetermined parameters, linear least squares fitting may be applied iteratively to a literalized form of the function until convergence is achieved. Since we can anticipate the power type of fit and have decided initial parameters chosen for our models, the nonlinear fitting has good convergence properties.

In general, the computation of the matrix does cost a large amount of the wireless nodes' energy. The solution in GraDrive is to simplify the prediction distribution function as above, given that prediction function computation can be distributed over the network with collaboration of its neighbors or the data to be delivered back to a base station where stronger computation ability and energy are normally not limitations. If this is the case, the base station creates a gradient distribution map globally using a weighted average method as a function of probability and predicted distribution. This kind of global information is sent back to each individual node involved in application.

6 TARGET LOCALIZATION USING THE COLLABORATIVE PREDICTION MODEL

Based on the per-node prediction model, the mobile sensor nodes can infer the position of target (x, y) and the associated confidence value p. This information is then used to perform global predictions. Specifically, we propose to use a probability-weighted average model for global collaborative prediction, due to its high efficiency and low cost characteristics. The simple rational behind our method is that the sensor nodes having a higher probability are much closer to the intended target.

Generally, if the predicted target location provided by sensor nodes $n_1, n_2, ..., n_k$, are $(x_1, y_1), (x_2, y_2), ..., (x_k, y_k)$ associated with probability value $p_1, p_2, ...p_k$. The estimated position of the target is given as:

$$X = \frac{\sum_{i=1}^{k} p_k x_k}{\sum_{i=1}^{k} p_k} \; Y = \frac{\sum_{i=1}^{k} p_k y_k}{\sum_{i=1}^{k} p_k} \tag{4}$$

6.1 Collaborative Navigation and Prediction Protocol

With per-node and global prediction models established, we are now ready to describe how the sensor nodes navigate using these two models.

Initially, sensor nodes enter an intended region with certain moving speeds, moving directions and trust intervals. It should be noted that different mobile sensor nodes could have different moving speed or initial moving direction. After the entrance, the mobile sensor nodes continue to detect the RSSI in its sensing range. The detected RSSI readings are an important input for training the model it is assigned initially. Thus they use a default navigation plan, which is to keep moving forward unless they detect a smaller sensing reading. During the moving process, nodes themselves perform per-node prediction calculation to construct the local RSSI map as described in Section 5.3. Meanwhile, the sensor nodes estimate their distance to the target position according to the sensing RSSI, randomly pick one prediction within its trust interval. The predicted target location information is forwarded back to a base station. To prevent excessive energy consumption in communication, the frequency of updates can be specified in advance. As long as the global picture is not available, individual sensor nodes navigate according to the per-node prediction model. However, if the base station notifies the sensor nodes that it has constructed a global RSSI distribution with certain confidence, each sensor node will combine the information with its current model together and change its direction toward the gradient direction. This process will be repeated until the target position has been discovered locally or at the base station within acceptable confidence.

6.2 Default Navigation Plan when Global Prediction Unavailable

If initially there is no global picture constructed by the base station with acceptable confidence, or if there is only one separated node in the network for rescue plan, or if the network is partitioned or unable to deliver the data, the mobile sensor nodes fall back to the per-node prediction model. Given its current sensing reading, it compares with previous readings stored in memory at each motion step. After getting a smaller sensing reading, it rotates 90 degrees clockwise. The reason for that is that the target position is most likely located perpendicularly to its previous moving direction.

7 EXPERIMENTS AND SIMULATION

7.1 Model Matching Experiment

In order to verify the feasibility of the proposed prediction model and parameter-fitting algorithms, we have prototyped a light sensing system based on Berkeley MICAZ modes. Even though it is stationary, the prediction model and parameter-fitting algorithms can still be verified at the base station site which can be transferred to individual sensor nodes and implemented. Light signal strength is used as an example of RSSI to feed the model. One laptop equipped with motherboard acts as the base station. A lamp works as a target and a series of sensor nodes are deployed as shown in Fig. 3. The sensor nodes detect the sensing reading and exchange the readings to their neighbors. The base station calculates the parameters for the sensor nodes by using the least square fitting method. Fig. 4 shows one set of data fitted by the prediction model. The distance between two adjacent sensor nodes is equal and unified for matching purpose. Since the received signal strength is not an accurate measurement, probability approximation model comes into play. From the matching results, it is shown that the least square method tries to reduce the deviation among the sensing data collected. Other sets of data can also be collected and used to train the model before it can be applied into the mobile sensor scenario.

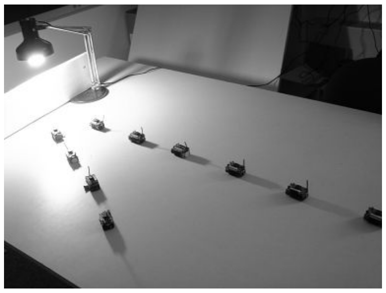


Fig. 3. Model fitting experiment with light as source of signal and using Micaz nodes in array to sense the signal strength

7.2 Simulation Setup

We have developed a program to verify the advantage of using our prediction model to locate the target in a faster approach. In our simulation, a 200×200 m^2 area is regarded as an unknown space with a target located at the center and a distribution along the diameter is defined. Essentially, it would be any random distribution that having a gradient toward the center. Each distance unit is represented as the smallest unit that the mobile sensor nodes can travel each time during simulation. The navigation algorithm is used to simulate the mobility of objects. Initially, the mobile nodes are located at the edges of the area. The initial direction is randomly picked by each mobile sensor node. If some sensor nodes move outside the simulation region, they bounce their moving direction back into simulation area. Under simulation, each mobile sensor node moves at a constant speed in integer multiples of 1m/s. After each time unit (1 second in our case), a node determines their next moving direction according to our algorithm.

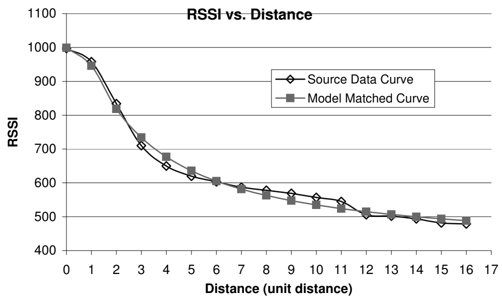


Fig. 4. The predicted model with real sensing data

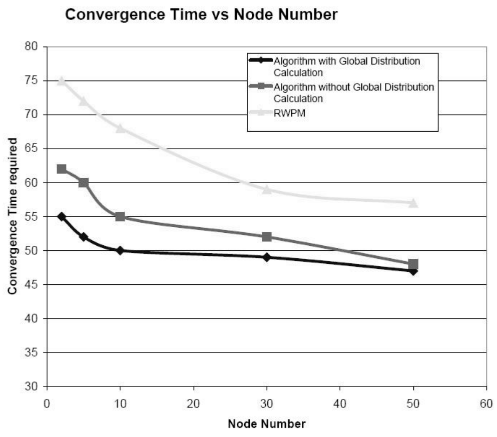


Fig. 5. Convergence time with node number for different models

7.3 Delay in Target Acquisition

We first experiment on comparing our algorithm (w/o global distribution calculation option) against Random Way Point Model. The simulation results (Fig. 5) suggest that even without a global distribution calculation mode turning on, our default algorithm (rotating 90 degree counterclockwise) still provides 30% faster estimation than the random way method. If global calculation mode is on, then initially the sensor nodes still use default plan, but if the global signal strength distribution is available, it moves faster than the default algorithm.

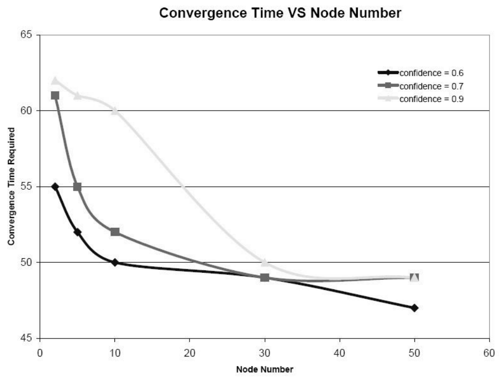


Fig. 6. Convergence time with Node number under different required confidence level

7.4 Impact of the Confidence p

We also compare the impact of different required confidence level on the convergence time as shown in Fig. 6. It is clear that if the required confidence level goes beyond 90%, it will take much longer to simulate simply because it requires at least 2 nodes to get very close to target position. It is reasonable to choose a relative high confidence level e.g. 80% in order to balance accuracy and time cost. 0

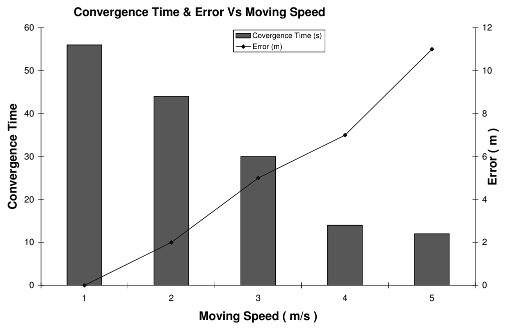


Fig. 7. Convergence time and Accuracy with different moving speeds of mobile sensor nodes

7.5 Impact of the Target Speed

In Fig. 7, we further investigate the relationship between the moving speed of sensor nodes and prediction accuracy of target location. The convergence time correlated directly with moving speed of each sensor node since the average time for sensor nodes to get closer to target is reduced. However, the accuracy of prediction gets worse if the speed increases because the minimum deviation for the prediction is increased as well. Therefore the error continues to grow in the prediction as node moves faster from its original location. In the situation of high speed, accuracy error larger than 10 units is shown. To protect against inaccuracies in the prediction model of mobile sensor nodes, a user must set a limit for moving speed of sensor nodes.

8 CONCLUSION AND FUTURE WORK

In this paper we present a probabilistic prediction model for dynamic target localization and evaluation of the localization algorithm. Our model does not require any known map to determine the positions of potential targets. Also the proposed gradient driven algorithm leads to a 40% reduction in time compared to that of a random working model. The relationship between sensor density and convergence time can be used as a reference of consideration for doing planning of such a mobile sensor network. Even though the computation power could be large, the error of the predicted target position can reach to almost zero and in a short time (about only 47sec). As future work, we would like to implement our algorithm on off-the-shelf hardware platforms. We would also need to design a speed self-adjusting algorithm so that the sensor node has the ability to trade off performance and cost.

References

1. A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita, " A Wireless Sensor Network for Target Detection, Classification, and Tracking," *Computer Networks (Elsevier)*, 2004.
2. J. Liu, J. Reich, and F. Zhao, "Collaborative In-Network Processing for Target Tracking," *J. on Applied Signal Processing*, March 2003.
3. T. He, S. Krishnamurthy, J. A. Stankovic, and T. Abdelzaher, "An Energy-Efficient Surveillance System Using Wireless Sensor Networks," in *MobiSys'04*, June 2004.
4. J. Tan, A. Verma, and H. Sawant, "Selection and navigation of mobile sensor nodes using a sensor network," 2006.
5. I. Chatzigiannakis, S. Nikoletseas, and P. Spirakis, "Distributed communication algorithms for ad hoc mobile networks," *J. Parallel Distrib. Comput.*, vol. 63, no. 1, pp. 58–74, 2003.
6. M. Batalin, G. Sukhatme, and M. Hattig, "Mobile robot navigation using a sensor network," in *IEEE International Conference on Robotics and Automation*, 2004.
7. P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein, "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet," in *Proc. of ASPLOS-X*, October 2002.
8. S. Tilak and et al., "Dynamic localization control for mobile sensor networks," in *Conference Proceedings of the 2005 IEEE International Performance, Computing and Communications Conference*, 2005.
9. H. H. Gonzalez-Banos and J. C. Latombe, "Robot navigation for automatic model construction using safe regions," in *ISER '00: Experimental Robotics VII*. London, UK: Springer-Verlag, 2001, pp. 405–415.

Feedback Control Scheduling in Distributed Real-Time Systems[*]

John A. Stankovic, Tian He, Tarek Abdelzaher, Mike Marley, Gang Tao, Sang Son and Chenyang Lu
Department of Computer Science, University of Virginia
Charlottesville, VA 22903
e-mail: { stankovic, tianhe, zaher, son, chenyang }@cs.virginia.edu, gt9s@virginia.edu

Abstract

Distributed soft real-time systems are becoming increasingly unpredictable due to several important factors such as the increasing use of commercial-off-the-shelf components, the trend towards open systems, and the proliferation of data-driven applications whose execution parameters vary significantly with input data. Such systems are less amenable to traditional worst-case real-time analysis. Instead, system-wide feedback control is needed to meet performance requirements. In this paper, we extend our previous work on developing software control algorithms based on a theory of feedback control to distributed systems. Our approach makes three important contributions. First, it allows the designer for a distributed real-time application to specify the desired temporal behavior of system adaptation, such as the speed of convergence to desired performance upon load or resource changes. This is in contrast to specifying only steady-state metrics, e.g., deadline miss ratio. Second, unlike QoS optimization approaches, our solution meets performance guarantees without accurate knowledge of task execution parameters - a key advantage in an unpredictable environment. Third, in contrast to ad hoc algorithms based on intuition and testing, our solution has a basis in the theory and practice of feedback control scheduling. Performance evaluation reveals that the solution not only has excellent steady state behavior, but also meets stability, overshoot, and settling time requirements. We also show that the solution outperforms several other algorithms available in the literature.

1. Introduction

Many soft real-time systems such as smart spaces, financial markets, processing audio and video, and the world wide web are not amenable to traditional worst-case real-time analysis. Most of these systems are distributed. They operate in open environments where both load and available resources are difficult to predict. Monitoring and feedback control are needed to meet performance constraints. Several difficulties are observed in meeting performance constraints in these systems. One main difficulty lies in their data-dependent resource requirements, which cannot be predicted without interpreting input data. For example, the execution time of an information server (a web or database server) heavily depends on the content of requests, such as the particular web page requested. A second major challenge is that these systems have highly uncertain arrival workloads; it is not clear how many users will request some resource in the www or how many users might walk into a smart space. A third challenge involves the complex interactions among many distributed sites, often across an environment with poor or unpredictable timing behavior. Consequently, developing certain types of future real-time systems will involve techniques for modeling the unpredictability of the environment, handling imprecise or incomplete knowledge, reacting to overload and unexpected failures (i.e., those not expressed by design-time failure hypotheses), and achieving the required performance levels and temporal behavior.

We envision a trend in the theory of real-time computing that aims at providing performance guarantees without the requirement of fine-grained task execution models such as those involving individual task execution times. Instead, we shall see the emergence of coarse-grained models that describe the aggregate behavior of the system. Coarse-grained models are easier to obtain and they need not be accurately computed. These models are more appropriate for system analysis in the presence of uncertainty regarding load and resources. In this paper, we explore one such model based on difference equations. Unlike the more familiar queuing-theory models of aggregate behavior, difference equation models do not make assumptions regarding the statistics of the load arrival process. Thus, while both types of models describe queuing dynamics, difference equation models are independent of load assumptions and consequently more suitable for systems where load statistics are difficult to obtain or where the load does not follow a distribution that is easy to handle analytically. The latter is the case, for example, with web traffic, which cannot be modeled by a Poisson

[*] This work was supported, in part by NSF grants CCR-0098269, the MURI award N00014-01-1-0576 from ONR, and DAPRPA ITO office under the NEST project (grant number F336615-01-C-1905).

distribution. Differential equation models include input load as a measured variable. Hence, they are particularly suitable for feedback control architectures in which input load is measured at run-time.

In this paper, we show that our solution outperforms several other algorithms available in the literature. Some of our key performance results are the ability to adapt to unpredictable environments and better transient and steady state response. In addition to improvements in performance, our solution has a basis in the theory and practice of feedback control scheduling. This is in contrast to the more common *ad hoc* algorithms based on intuition and testing where it is very difficult to characterize the aggregate performance of the system and where major overloads and/or anomalous behavior can occur since the algorithms are not developed to avoid these problems.

2. DFCS Framework Overview

Early research on real-time computing was concerned with guaranteeing avoidance of undesirable effects such as overload and deadline misses. Solutions were predicated on knowing worst-case resource requirements a priori. In contrast, in highly uncertain environments, the main concern is to design adaptation capabilities that handle uncertain effects dynamically and in an analytically predictable manner. To address this issue, we propose a framework called Distributed Feedback Control Real-time Scheduling (DFCS). The framework is based on feedback control that incrementally corrects system performance to achieve its target in the absence of initial load and resource assumptions. One main performance metric of such a system is the quality of performance-convergence to the desired level. In our framework, the desired convergence attributes may be specified and enforced using mechanisms borrowed from control theory. These mechanisms are very robust and have been applied successfully for decades in physical process-control systems that are often non-linear and subject to random external disturbances. Before establishing our DFCS framework, we give an overview of the software system being controlled and describe the feedback-control mechanism involved.

We assume that there are N computing nodes connected via a network. Tasks arrive at nodes in unknown patterns. Each task is served by a periodically invoked schedulable entity (such as a thread) with each instance having a soft deadline equal to its period. The periodicity constraint is motivated by the requirements of real-time applications such as process control and streaming media. It is also motivated by recent trends in real-time operating system design, such as temporal isolation of independent applications. Note that temporal isolation is usually achieved using operating system constructs such as capacity reserves, hierarchical schedulers, and resource shares, all of which rely on periodic scheduling in the kernel.

We abstract a typical adaptive system by two sets of performance metrics. The primary set represents metrics to be maintained at specified levels, for example, the optimal utilization of a server, or the desired altitude of an airplane. The secondary set represents negotiable metrics such as service quality. The objective of adaptation is to incur minimum degradation in secondary metrics while maintaining the primary metrics at their desired values. To represent multiple levels of degradation in secondary metrics, we assume that each task has several service levels of different quality. For example, a task can execute for varying amounts of time with the quality of the result improving with greater execution time.

The goal of our DFCS architecture is to maintain the primary performance metrics around their targets. A key intuition that affects the architecture of the feedback loops is that the dynamics of a distributed system manifest themselves on two different time-scales. Fast dynamics are observed on individual nodes. These dynamics arise from local load changes due to individual task arrivals and terminations. Slower dynamics are observed globally on the scale of the entire system. These dynamics arise from changes in aggregate load distribution. Hence, our feedback architecture necessarily involves two sets of loops, local and distributed ones, each tuned to the dynamics of the corresponding scale.

Each node in the distributed system (Figure 1) has a local feedback control system (LFC) and a distributed feedback control system (DFC). The distributed feedback controller is responsible for maintaining the appropriate QoS balance between nodes. The local feedback controller is responsible for tracking the global QoS set point set by distributed controller and ensuring that tasks that are admitted to this node have a minimum miss ratio (MR) and the node remains fully utilized. It is important to note that these two types of controllers form the main parts of the distributed scheduling in the system, but they are not the entire algorithm. In this paper we develop the distributed controller, rely on a local controller (developed earlier and presented in a previous paper [10][11][12]), and then embed these two controllers into the complete distributed scheduling algorithm (see section 3.7). We test the complete algorithm.

Now consider a few more details about the controllers' structure. The distributed controller commands the local controller via the QoS set point. The entire control system of the local node becomes an actuator for the distributed controller to control the state of one local node. The local controller manipulates its actuators in the LFC to achieve the target QoS set point. While the framework is general enough to accommodate different definitions of QoS, in this paper, we let the primary performance metric be the deadline miss ratio (MR). Since performance metrics of admitted tasks can be trivially satisfied if the admitted task set is empty, it is especially important to quantify the loss of QoS due to task rejection to avoid trivial solutions. For this reason, we use two different miss ratio measurements, MR and inNode-MR. The former is the global miss ratio of all submitted tasks (whether they are admitted or rejected). The latter is the miss ratio of admitted tasks only. The distributed controller is responsible for bounding the overall miss ratio, MR, of the system. The local controller is responsible for controlling the inNode miss ratio of locally admitted tasks as dictated by the distributed controller. These design choices lead to the control system in Figure 1. Note that each of the local controller and the distributed controller has two similar parts, a miss ratio controller and a utilization controller. The miss ratio controller is active during overload. The utilization controller is active at underutilization when no deadline misses are observed. Its purpose is to keep the system sufficiently utilized. These are key parts of our model developed in the next section. The LFC also has a service level ratio controller (SLR) to address our secondary metric.

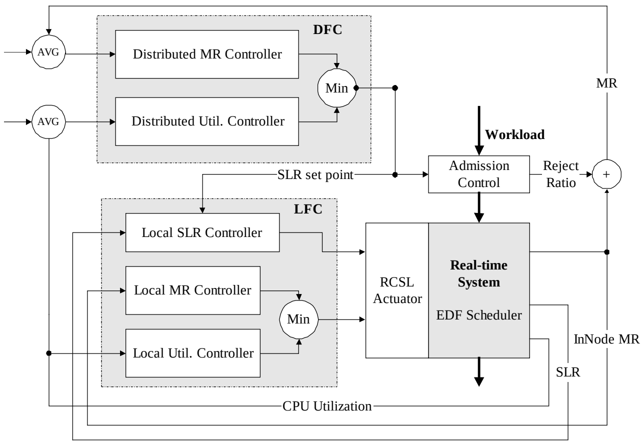


Figure 1: A Typical Node in DFCS.

Admission control (Fig. 1) is based on estimated CPU utilization and the global service level set point and decides to admit or reject tasks from the outside. If one task is rejected, it will be offloaded to another node based on a certain routing policy. Figure 1 also show a Reject Control and Service Level Actuator (RCSL): To track the service level set point from the DFC, the service level actuator modifies the service level of tasks in the system to the appropriate level, then the reject control actuator aborts tasks or notifies AC to admit more if necessary. Finally, the real-time system is the plant processing the request from the users. We can plug various scheduling algorithms into the RTS based on different requirements. Here, we use EDF in our design.

3. DFCS Modeling and Design

The design of DFC control policies requires two components: a task model and difference equations describing the dynamics of the DFC in underutilization and overload situations, respectively. The design process proceeds as follows:

First, we specify the task model. Second, we specify the desired dynamic behavior using both transient and steady-state performance metrics. This step requires a mapping from the performance metrics of adaptive real-time systems to the dynamic response metrics of control systems used in control theory. Third we establish a mathematical model of the system for the purposes of feedback control. We take a simple approach where our model aggregates the overall performance of the system in a single model. We show by our performance study that this model works well, in spite of its simplicity. Finally, based on the performance specifications and the system model from steps 2 and 3, we apply the mathematical techniques of control theory to design the controller that gives analytic guarantees on the desired transient and steady-state behavior at run-time. We map the resultant controller to various nodes in the system depending on the network structure being studied. This step is similar to the process that a control engineer uses to design a controller for a feedback-control system to achieve desired dynamic responses.

3.1. Task Model

For each task T_i, there are N QoS service levels (N > 1). Task T_i running at Service Level q ($0 \le q < N$) has a deadline $D_i[q]$ and an execution-time $C_i[q]$ that is unknown to the scheduler. The requested CPU utilization, $J_i(q) = C_i[q]/D_i[q]$, of the task is a monotonically increasing function of the service level q, which means that a higher QoS will require more CPU utilization. Let the average CPU utilization needed for a task set at level 0 be U_b. Without loss of generality, the average CPU utilization for a task set at level q is f(q)U_b, where f(q) is a polynomial representing the Taylor's series expansion of the relation between CPU utilization and QoS level. Here we use the first order approximation of this relation to define average requested CPU utilization $J(q)$ of a task set:

$$J(q) = (Aq + 1)U_b$$

where $q \in [\,0, \text{MaxLevel}]$ and MaxLevl = N–1.

We make use of this approximation in the rest of the paper to derive the system model. Note that if this approximation is not appropriate, higher order ones can be used. The design process remains the same.

3.2. System Specification and Metrics

To design adaptive systems, it is necessary to devise specifications and performance metrics for the adaptation process itself. Following the successful practice of the control community in specifying and evaluating the performance of control loops, we have proposed a series of canonic benchmarks that test software adaptation capabilities. These benchmarks generate a set of simple load profiles adapted from control theory; namely, the step load and the ramp load. The step load represents a worst-case load variation: one that occurs in zero time. The ramp load represents a more gradual variation that features a slower rate of change. By experimenting with different rates of change, it is possible to assess the convergence of an adaptive system to the desired performance upon perturbations caused by changes in the environment. Effects of different "speeds" of environmental variation can be analyzed. If the rate of change of the environment is bounded, this analysis can yield guarantees on convergence time and worst-case performance deviation.

We measure system load in percentage of the system capacity. The load corresponding to the full system capacity is said to be 100%. An overload is a system load that is higher than 100%. A load profile *L(t)* is the system load as a function of time. In practice, this load is translated into system-specific parameters for evaluation purposes. For example, a 500% system load can be translated to the request rate of 8,000 request/sec in a specific web server (assuming a fixed requested file type/size distribution). We have shown in [10][12] that these load profiles can be used as benchmarks in adaptive systems to provide a common test-suite for quantifying and comparing the speed of adaptation to load changes of different adaptive systems.

Consider a time window *[(k-1)W,kW]*, where W is called the sampling *period* and k is called the *sampling instant*. During this window, let *M(k)* be the number of task instances that miss their deadline, let *T(k)* be the total number of task instances, and let *MR(k)* be the miss ratio *M(k)/T(k)*. The following metrics are used to describe the quality of adaptation:

- Overshoot M_o: the maximum amount by which *M(k)* exceeds its reference value, expressed as a percentage of the reference value.
- Settling time T_s: The time it takes the miss-ratio to enter a steady state after a load change occurs.
- Steady-state error E_s: It is the difference between *M(k)* and its set point when no disturbance happens and after system transients have decayed. It indicates the DFC's ability to regulate the controlled variable near the reference value in the long term.

Using these metrics, we can compare the effectiveness of feedback-control to other adaptive real-time scheduling policies. We can also specify the desired behavior of the adaptation process in terms of these metrics to guide the control loop design. The critical part towards the enforcement of these metrics is a good aggregate model of the system. This is described in the next section.

3.3. DFCS Modeling

Let the utilization *U(k)* be the fraction of time the CPU is busy in some sampling window *k*. Let *S(k)* be the service level ratio defined as:

$$S(k) = \frac{Avg.\ service\ level}{MaxLevel} = \frac{\sum_{q=0}^{MaxLevel} (Num.\ of\ Tasks\ completed\ at\ level\ q\ ) \times q}{(T(k) - M(k)) \times MaxLevel}$$

Before applying control theory to design a controller from specifications of adaptive behavior, it is necessary to model the controlled DFCS mathematically. By this we mean deriving the relation between utilization, service level ratio and the resulting miss ratio. Equivalently, we can relate utilization and service level ratio to the number of missed deadlines, which is the approach we take below. When the number of missed deadlines is known, the miss ratio can be trivially obtained. In each time window $[(k-1)W, kW]$, CPU utilization is proportional to the number of tasks that finish successfully. This relationship can be modeled as:

$$U(k) = c(k) \times (T(k) - M(k)) \times J(q) \qquad (1)$$

where *c(k)* is the percentage of the arrived tasks that finish in the same sampling window. For example if *c(k)=1*, all tasks arrive and finish in the same period. From the perspective of control theory, worst-case conditions for convergence stability are those when system gain is maximum, i.e., when the system is most sensitive to changes in its inputs. The maximum gain condition corresponds to *c(k)=1*, in which case equation (1) can be simplified as:

$$U(k) = (T(k) - M(k)) \times J(q) \qquad (2)$$

From definitions of $J(q)$ in section 3.1 and $S(k)$ above, we can derive the following formula:

$$J(q) = (1 + A \times S(k) \times MaxLevel) \times U_b \qquad (3)$$

Combining Equations (2) and (3), we get

$$U(k) = (T(k) - M(k)) \times (1 + A \times S(k) \times MaxLevel) \times U_b \qquad (4)$$

Two important subcases arise in modeling the system: namely, overload and underutilization. They are modeled separately in the two subsequent subsections, respectively.

3.4. System Dynamics at Overload

When the DFCS is overloaded and tasks begin to miss their deadlines, there are two approaches to tackle the situation: Admission control and QoS adjustment. Admission control reduces a node's local miss-ratio by rejecting incoming requests. QoS adjustment tries to accommodate more tasks by degrading their service levels. In the DFCS design, we deem task rejection the same as missing the task's deadline. Hence, we favor QoS adjustment to adjust miss control. Here we get a difference equation that describes how QoS adjustment affects the number of misses when the system is overloaded ($M(k) > 0$). From Equation (4), we obtain

$$M(k) = T(k) - \frac{U(k)}{U_b(1 + A \times S(k) \times MaxLevel)} \qquad (5)$$

Since we assume EDF scheduling, when deadline misses occur it must be that the CPU utilization *U(k)* is 100%. Substituting this in Equation (5) and obtaining its differential, we get the linearized small-signal model of the system:

$\Delta M(k) = G_M \times \Delta S(k) + \Delta T(k)$ where $\Delta M(k) = M(k) - M(k-1)$,

$\Delta S(k) = S(k) - S(k-1)$, $\Delta T(k) = T(k) - T(k-1)$

$$G_M = \frac{A \times MaxLevel}{U_b(1 + A \times MaxLevel \times S(k-1))(1 + A \times MaxLevel \times S(k))} \quad (6)$$

We use Equation (6) as the model for the purpose of controller design in overload situations.

3.5. System Dynamics at Under Utilization

When DFC is underutilized, we model the number of missed deadlines as zero. Hence, this is not an appropriate measurement for control purposes. Instead, we switch to utilization measurements. We can increase the QoS of the task set when the utilization is low to improve our service to the user. Here we obtain a difference equation that describes how QoS adjustment affects the CPU utilization when the system is underutilized (*U(k)* <100%). We set the number of misses *M(k) = 0* in Equation (4). The resulting equation is differentiated yielding the following model:

$$\Delta U(k) = G_U \times \Delta S(k) + G_t \times \Delta T(k) \quad (7)$$

where $G_U = (T(k-1) \times A \times MaxLevel \times U_b$

$G_t = (1 + A \times S(k-1) \times MaxLevel) \times U_b$, $\Delta U(k) = U(k) - U(k-1)$

3.6. DFC Loop Design

Having modeled the node dynamics, we now apply control design methods to the distributed feedback control loop. First, we define a set of performance specifications we want to achieve. Based on our knowledge of the DFC model and the performance specifications, we apply a control design method called Root Locus to tune the distributed controller, which is the crucial part of the DFC. Due to space limitations, we do not review local controller design here, which has been intensely studied in our previous work [10][12].

3.6.1. Design of the Control Loop

In the distributed case, we want each node in the DFCS to provide the same QoS to the user. This property is often preferred in many distributed applications. For example, in a web server farm, the QoS of each HTTP request should be independent of where this request is served in the farm. So the major goal of the distributed controller is to calculate the QoS set point for the system, based on the global miss number error $E_M(k)=M_S - M(k)$ and/or the global CPU utilization error $E_U(k)=U_S - U(k)$ as shown above, the DFC can be modeled with two difference equations. One describes the relation between the changes of the service level ratio and the changes of miss number when the whole system is overloaded; the other models the relation between the changes of the service level ratio and the changes of CPU utilization when the system is underutilized. Based on this knowledge, we design the distributed feedback control loop.

Because the external workload is not under our control, we deem *ΔT(k)* as the external disturbance. Let G_U be the gain from *ΔS(z)* to *ΔU(z)* when the system is underutilized and G_M be the gain from *ΔS(z)* to *ΔU(z)* when system is overloaded. We get:

$$M(k) = M(k-1) + \Delta M(k) = M(k-1) + G_M \Delta S(k) \text{ when } M(k\text{-}1)>0 \quad (8)$$

$$U(k) = U(k-1) + \Delta U(k) = U(k-1) + G_U \Delta S(k) \text{ when } U(k\text{-}1)<1 \quad (9)$$

where G_M and G_U are defined in (6) (7), respectively.

We can now draw the block diagrams of the feedback control system. When the system is overloaded, the distributed miss feedback control loop is invoked. It is shown in Fig 2. The components inside the dotted rectangle describe the dynamics of the controlled process with input $\Delta S(z)$ and output $M(z)$, where $C_M(z)$ is the miss controller to be designed and M_S is the miss set-point. Note that while the output of this figure is the number of missed deadlines, the miss ratio is simply that output divided by the total number of tasks. Either metric can be used depending on the chosen set point. More importantly, note that while the controlled system gain does change by a multiplicative factor when the metric used is miss ratio, the overall loop gain remains the same. This is because the designed controller gain in this case will be multiplied by the inverse of that factor. With the above observation in mind, the discussion below applies to both miss ratio and miss number control.

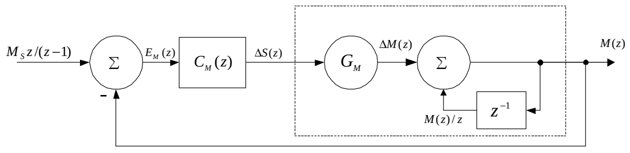


Fig. 2 Miss Feedback Control Loop

When the system is underutilized, we use the distributed utilization feedback control loop shown in Fig 3. $C_U(z)$ is the CPU utilization controller to be designed and U_S is the CPU utilization set point.

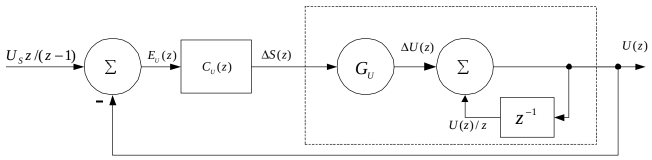


Fig. 3 CPU Utilization Feedback Control Loop

In z-transform notation we have:

$$M(z) = H_M(z)\frac{M_S z}{z-1} \text{ with } H_M(z) = \frac{C_M(z)G_M}{1 - z^{-1} + C_M(z)G_M} \quad (10)$$

$$U(z) = H_U(z)\frac{U_S z}{z-1} \text{ with } H_U(z) = \frac{C_U(z)G_U}{1 - z^{-1} + C_U(z)G_U} \quad (11)$$

Here the gains G_M and G_U are assumed to be set at some fixed values for nominal control design and analysis. Because our system intrinsically has an integral part, it is enough to use only a Proportional controller to design

$C_M(z)$ and $C_U(z)$ to guarantee the stability and zero steady state error. A general form of the digital Proportional controller in the time domain and z-domain is:

$\Delta S(k) = K_p E(k)$ (t-domain) $C(z) = K_P$ (z-domain) (12)

Here we denote K_M as the proportional term for the miss controller and K_U for the utilization controller. These values are substituted in Equations (10) and (11). Setting $C_M(z) = K_M$ and $C_M(z) = K_U$ we get:

$$M(z) = H_M(z)\frac{M_S z}{z-1} \text{ with } H_M(z) = \frac{K_M G_M z}{(1+K_M G_M)z-1} \quad (13)$$

$$U(z) = H_U(z)\frac{U_S z}{z-1} \text{ with } H_U(z) = \frac{K_U G_U z}{(1+K_U G_U)z-1} \quad (14)$$

3.6.2. Stability

According to control theory, system performance is determined by the poles of the closed loop transfer function. From Equations (13) and (14), we get $1/(1+K_M G_M)$ as the pole for $H_M(z)$ and $1/(1+K_U G_U)$ as the pole for $H_U(z)$, which are inside the unit cycle. Hence, for the DFC system, stability is ensured.

3.6.3. Steady State Error

Based on the Final-Value Theorem, the steady state values of $M(k)$ and $U(k)$ are:

$$\lim_{k\to\infty} M(k) = \lim_{z\to 1}(z-1)M(z) = \lim_{z\to 1}\left\{(z-1)\frac{K_M G_M z}{(1+K_M G_M)z-1}\times\frac{M_s z}{z-1}\right\} = M_S$$

$$\lim_{k\to\infty} U(k) = \lim_{z\to 1}(z-1)U(z) = \lim_{z\to 1}\left\{(z-1)\frac{K_U G_U z}{(1+K_U G_U)z-1}\times\frac{U_s z}{z-1}\right\} = U_S$$

This result theoretically proves that the DFC system can bring the miss number and CPU utilization to their set point in steady state with zero error. It can also be verified that for a constant external disturbance $\Delta T(k) = \Delta T$, this asymptotic property still holds.

3.6.4. Settling time

Settling time can be determined by the poles inside the unit cycle. The closer the pole is to the origin, the shorter the settling time. Theoretically, we want $1/(1+K_M G_M)$ and $1/(1+K_U G_U)$ to be as small as possible to reduce settling time. However, there are limitations on how large the gains K_M and K_U can be. For example when system is in overload, if we reduce SLR too much, the system will become underutilized. The limitation can be described as a maximum rate of change:

$$\Delta M(z) \le E_U(z) \text{ and } \Delta U(z) \le E_U(z) \quad (15)$$

From the block diagram we know that $\Delta M(z) = K_M G_M E_M(z)$ and $\Delta U(z) = K_U G_U E_U(z)$. Thus, the following should be satisfied:

$$K_M G_M \le 1 \text{ and } K_U G_U \le 1 \quad (16)$$

While the previous sections establish the theoretic foundation for DFC design, in reality G_M and G_U are time-varying, and the proportional term of the distributed controller is taken in (17) to deal with a worst case situation:

$$K_M = \frac{1}{Max\{G_M\}} \text{ and } K_U = \frac{1}{Max\{G_U\}} \quad (17)$$

If we let $K_M G_M = 1$ and $K_U G_U = 1$, the poles of $H_m(z)$ and $H_U(z)$ will be 0.5. According to control theory, the settling time is determined by the distance of the pole from the origin of the root locus plot. With radius of 0.5, the theoretical settling time is about 8 sampling periods, which guarantees the specification in Table 1. In the experiment, based on the model, the calculated controller settings are 0.82 and 1.22 for the miss and utilization controllers, respectively.

3.6.5. Performance specification of DFC

As an example, here we assume the desired DFCS has the performance specifications listed in table 1. The sampling period is W = 1 sec. The transient and steady state performance requirements are the following: (1) the miss ratio and CPU utilization of DFCS should be stable after a step load of 160%. (2) DFCS should settle down to steady state within 10-sec. (3) Both the miss ratio and CPU utilization of DFCS should remain at their set-point in the steady state.

Load Profile	SL(0,160%)
Sampling Period W	1 sec
Settle Time T_s	< 10 sec
Steady-state miss number M_s	< 1% *T(k)*
Steady-state CPU utilization	>99%

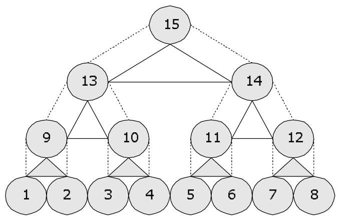
Table 1. Performance Specification

3.7. Network Structures

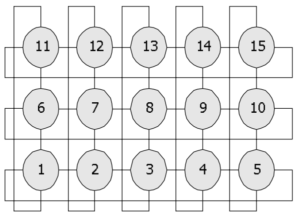
In a distributed system, the interactions between nodes must be considered. These interactions depend on the logical network structure. We investigate two logical network structures (hierarchical and neighborhood) and design distributed real-time scheduling algorithms based on each network structure. The neighborhood structure (a mesh) has good scaling potential and hence is used for comparison with other well-known distributed scheduling algorithms. The hierarchical structure does not scale well, although there are some situations where a hierarchy is valuable.

Hierarchical feedback control scheduling (H-DFCS) allows a large distributed systems to be broken down into a multiple levels, as illustrated in Figure 4.A. By doing so, only the information that is required to coordinate subsystems needs to be exchanged at higher levels. In the H-

DFCS system, any node that has sub-nodes can be considered a coordinator. The full scheduling algorithm for this system operates every sampling period in the following manner. Each node contains the LFC control system, with the exception of the top node in the hierarchy. This node contains the LFC control system along with the DFC control system. The top node will receive the MR and CPU utilization averaged for the entire system and use this as inputs to its distributed controller to determine the new service level set point for the entire system.



A. Hierarchical



B. Neighborhood

Figure 4 Network structures

Load balancing is achieved by migrating tasks between nodes through the network. Each node determines the route by comparing the MR values from the its children, parent and siblings. The MR values are weighted, to describe the number of nodes that they represent. This information is then used to assign a percentage to each entry in the route table, specifying the ratio of the offloaded tasks to be sent along each route. In the hierarchical case that implements a binary tree, there are 4 possible routes from any given node --- two to its two children, one to its parent, and one to its sibling. Routes are unidirectional, and are assigned only if the miss ratio of the other node in question is lower than that of the current node.

In neighborhood feedback control scheduling (N-DFCS) (Figure 4.B) every node contains both a local and distributed controller. This means that a node shares its state information with its direct neighbors and receives state information from these same neighbors. This information that is exchanged is averaged and used to determine the service-level set-point for the node as well as the routes from the node. This prevents a node from being isolated from other nodes, keeps information sharing to local neighbors, and overall creates a more decentralized scheduling system.

Basically, the DFCS is controlling the node based on the state of the node as well as the state of its neighbor nodes. One main difference between the neighborhood solution and the hierarchical solution is that the former works within individual subnets instead of the whole distributed system. The advantage here is that state information is shared in a more decentralized manner.

4. Performance Evaluation

In the first part of our performance evaluation we compare the performance of both hierarchical (HCLOSE) and neighborhood (NCLOSE) feedback control scheduling algorithms with two baseline algorithms: no cooperation and neighborhood open loop.

No Cooperation: Nodes in the system do not interact in any fashion. When a task arrives at a node even if it cannot be scheduled at that node, it remains at that node. This can be considered as a global open loop system without any load balance support. Note: each node still uses local feedback control, but there is no global feedback support.

Neighborhood Open Loop: Neighborhood Open Loop (NOPEN) is a second baseline algorithm, in which a node schedules a task using local feedback, if it can. If it cannot be scheduled, it will select a neighbor with a lower miss ratio. If more than one neighbor has a lower miss ratio, the probability of one neighbor to be selected is proportional to the miss ratio difference between the node and this neighbor. Then the task is offloaded to the selected node. If the task cannot be scheduled at the receiving node, it will be again sent to another selected neighbor, until its deadline expires. It can be considered as a global open loop system with load balance support. The load balance method used in NOPEN is the same one used in the feedback control algorithms so that performance gains are not coming from different load balancing strategies, but from the distributed controllers.

4.1. Experimental Setup

For the experiments, we set up 15 nodes as shown previously in Figure 4 with the network delay modeled as one fifth of the average task period. Each task has two different service levels (best and half), which indicates its CPU requirement. The system is initialized with 100 tasks. This represents ~16% load if all tasks use their highest (best) service level (it represents a ~8% if all task use half of their service request). At time 200, a step load is applied with 1000 tasks, which represent ~160% load if all tasks use their highest service level (80% if half). To test the system reaction to unbalanced workloads, the extra workload is applied uniformly to the first 8 nodes (0~7) and the remaining 7 nodes (8~14) don't accept any extra workload directly from the outside. Of course they can accept tasks offloaded by other busy nodes throughout the network.

Workload Model

In our experiments, each task has just 2 different logical versions. A task is described by a tuple $(P, WCET_i, BCET_i, EET_i, AET_i)$, where i is either 1 or 0 corresponding to the higher or lower service levels, respectively. *P* is the period (the deadline of each task instance equals its period), *WCET* is the worst-case execution time, *BCET* is the best-case execution time, *EET* is the estimated execution time and *AET* is the average execution time.

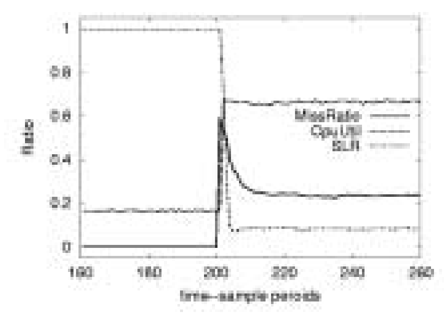
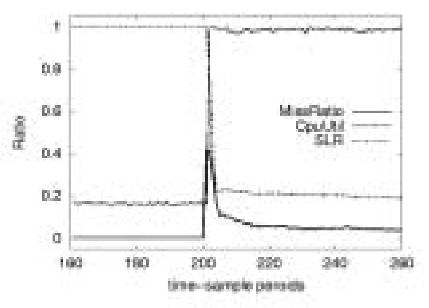
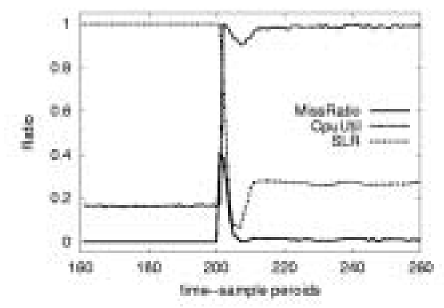
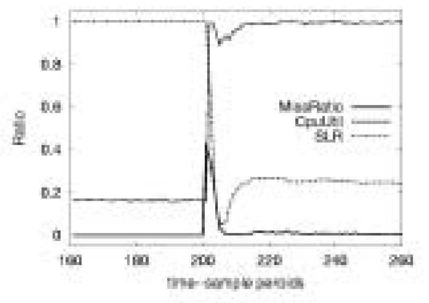
	
a. No cooperation	b. Neighborhood open loop
	
c. Hierarchical closed loop	d. Neighborhood closed loop

Figure 6 Comparison with baseline algorithms using step workload

The actual execution time is computed as a uniform random variable in the interval [AET_i , $WCET_i$] or [$BCET_i$, AET_i], depending on a random *Bernoulli trial* with probability *(AET_i-$BCET_i$)/($WCET_i$-$BCET_i$).* The following equations describe each tuple value as they are used in this experiment:

$$BCET_i = uniform(1{,}10) \qquad WCET_i = 4 \times BCET_i$$

$$EET_i = (WCET_i + BCET_i) \times 0.5 \qquad WCET_1 = 2 \times WCET_0$$

$$P = WCET_1 \times uniform(5{,}50) \qquad AET_i = EET_i \times ETF$$

This workload model is a very general one. For example, task execution times vary from 1 to 40 time units; task periods vary from 20 to 2000 time units. Here ETF is the execution time factor and can be tuned to vary the accuracy of the estimation of the average execution time. For example, an ETF of 1.0 means that the average execution time that is reported to the scheduler will be the average execution time of the task. If the ETF is less than 1.0, this means that the estimation is less than the actual average execution time of the task. If the ETF is greater than 1.0, then the estimation is greater than the actual average execution time of the task. By varying this factor we can determine how the system will react to a pessimistic or optimistic estimation of the required execution time for the tasks in the system. It is important to note that while the workload is described with a $WCET$, the local and distributed controllers have no knowledge of their values! In these experiments, all tests were run 20 times. However, the graphs represent the trace of single run since it shows how the system acts over time. The long term average for CPU utilization and miss ratio were extremely tight (within 1% of the mean) using a 90% confidence interval.

4.2. Performance Comparison

From the results shown in Figure 6, at time 200 all algorithms react to the incoming step load very quickly, because each node has its own local feedback control system, which was previously shown to be very effective in unpredictable situations [10][12]. But we can see that without global feedback both baseline algorithms do not perform well in terms of transient and steady state response. For example, when the step load occurs at time 200, the miss ratio peaks at almost 60% in no cooperation case (Figure 6.a), compared to only 40% for the other three algorithms. Also, for the no cooperation system while it does adapt to the step load, the local controllers do not reach a zero miss ratio after the step load. This is because there is no load sharing between nodes and the tasks have to be executed at the node where it was originally sent. Furthermore, since the loads are not shared between nodes, the CPU resource is not effectively utilized. For example, as shown in Figure 6.a, it turns out that the average CPU utilization is relatively low (around 67%) even when there is a non-zero miss rate (around 22%). The second baseline, the neighborhood open loop, provides a way to share load between nodes in the sense that the tasks that cannot be scheduled will migrate to other nodes. However, each node does not adjust its service level based on the load of other nodes. The node with small number of local tasks will set its service level relatively high, and accept offloaded tasks from overloaded nodes until it reaches 100% utilization. After that, without global feedback control, this node doesn't lower its service level to further accept offloaded tasks. As a result, although NOPEN exploits the computational power of all nodes, NOPEN can't balance the service level between each node, which leads to a higher miss rate (about 8% in Figure 6.b) than found in the closed loop solutions (Figures 6.c and 6.d) where the miss ratio is between 0-1%. Also, the graphs show that the reaction time to the sudden overload (step load) is sluggish in the open loop case in that it takes a relatively long time (about 12 sampling periods) for the system to reach steady state. On the other hand, both closed loop algorithms show satisfactory response in terms of CPU utilization and miss ratio. After the step load at time 200, the local closed loop controller adjusts its output to reduce the miss ratio according to the difference between the actual miss ratio and the miss ratio set point. The global closed loop controller adjusts the global service level set point according to the busyness of the whole system in the HCLOSE case and the busyness of the subsystem in the NCLOSE case. Through the synergy between the local and distributed controllers, the miss ratio drops to zero and the CPU utilization quickly approaches 100% to provide the best service to the tasks. Also, because of the global service level set point, the whole distributed system behaves as a single node to the user, which is preferred in many practical applications.

The above experiments were performed with $ETF = 1$, which means the average execution time of each task is the same as the estimated execution time. Of course, the actual execution time of each task could vary over time. In reality, it is usually difficult to get the accurate estimation of the execution time because of the uncertainty and unpredictability of the environment. Therefore, we have also run the above comparisons for different ETF values. The result further dramatizes the performance difference between the global closed loop case and the two baselines. For example if $ETF = 1.25$ it takes NOPEN about 40 sampling period to settle down, but it only takes NCLOSE about 12 sampling periods. Due to space limitations, we only provide the *ETF*=1 case here. We will show the impact of imprecise execution time estimation (i.e., varying *ETF*) in the next section when comparing our solution to other algorithms in the literature.

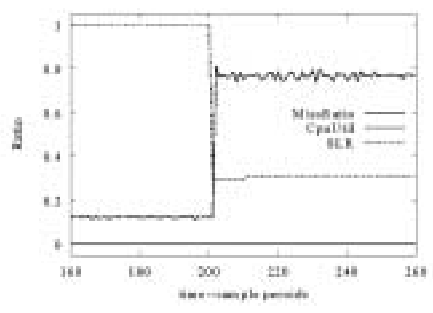
In summary, these experiments demonstrate that feedback control scheduling is a feasible solution for distributed real-time systems. Compared with other baseline algorithms, it can reach zero miss ratio at steady state, effectively share the load between nodes, yield high CPU utilization, and promptly react to load change.

5. Performance Evaluation with Previous Known Algorithms

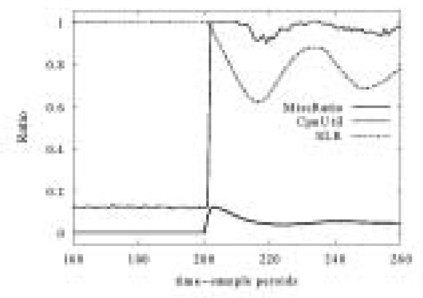
To further evaluate the performance of DFCS, we compare NCLOSE with two other well-known scheduling algorithms, QoS negotiation [1] and Dynamic QoS Management (DQM)[4]. We first present a brief summary of these two algorithms.

QoS Negotiation: The task model in QoS negotiation is similar to our current task model. A distributed system is assumed in which tasks can arrive at any node. There are several QoS levels for each task. These levels are specified upon task arrival as alternative acceptable performance levels for the task. Each QoS level has different resource requirements and a corresponding benefit (called reward) of executing the task at that QoS level. A node can accept a task in which case a QoS contract is said to be signed which promise to execute the task at one of its specified QoS levels. Alternatively, the task may be rejected in which case no contract is signed. There is a penalty if the task is rejected, and a different (higher) penalty if the QoS contract is violated. The latter allows a node to break the contract unilaterally and offer a compensation. The overall objective of the service is to schedule the tasks to yield a maximum global reward taking into account rejection and QoS violation penalties. At each node, there is a local scheduler, which uses a greedy hill-climbing heuristic to schedule the tasks for maximum reward. The heuristic upgrades the task with the largest reward differential between QoS levels when the system is underutilized, and downgrades the task with the smallest reward differential when the system is overloaded. Underutilization is measured by idle time, while overload is measured by deadline misses. When tasks execute at a level that is lower than the maximum QoS level declared in their contract, the system is said to have an unfulfilled potential reward (UPR). A global scheduler runs periodically to migrate tasks from nodes with high UPR to nodes with low UPR to balance the load and gain higher global reward. A hysteresis is used to prevent oscillations in the load balancing processes. The hysteresis reflects the cost of task migration.

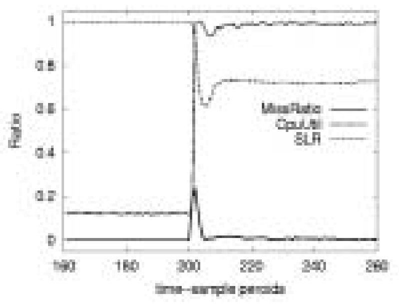
Distributed Dynamic QoS Management: In DQM, each task is associated with a different level of QoS and there are benefits and estimated CPU utilizations for each level. The aim of DQM is to adjust the QoS level of tasks in order to reduce miss ratio and maintain high CPU utilization. More specifically, in the situation of system overload (which is tested by the miss ratio, i.e., if the miss ratio is high, then the system is overloaded), DQM will lower the level of a task's QoS, which has the highest CPU utilization/benefit. In the case of underutilization (which is tested by the percentage of CPU idle cycles), DQM will raise the level of a task's QoS, which has the lowest CPU utilization/benefit. There are two thresholds associated with the algorithm: one for miss ratio and the other for



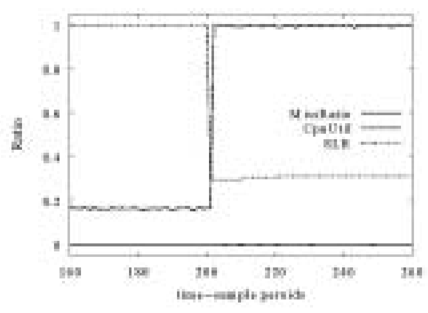
a. QoS Negotiation(ETF=0.75)



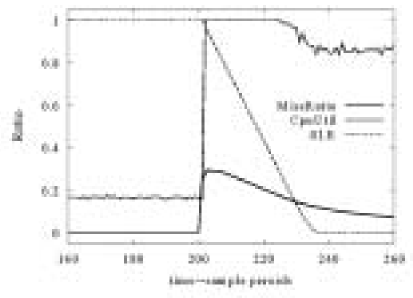
b. DQM(ETF=0.75)



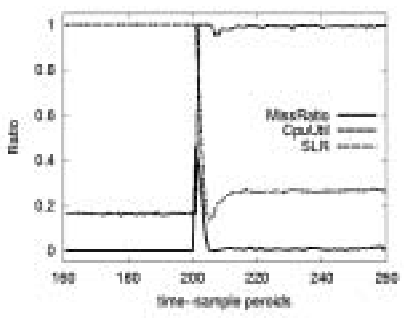
c. NCLOSE (ETF=0.75)



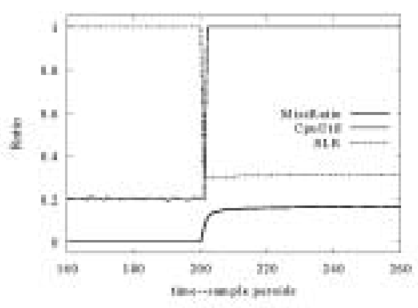
d. QoS Negotiation(ETF=1.00)



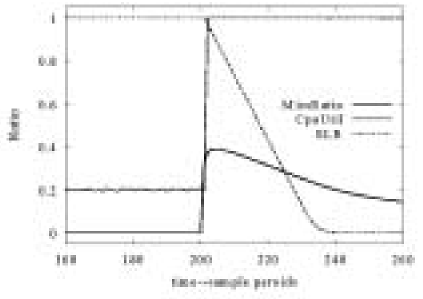
e. DQM(ETF=1.00)



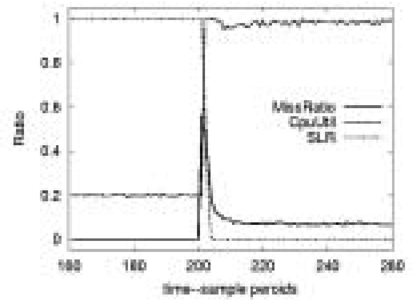
f. NCLOSE (ETF=1.00)



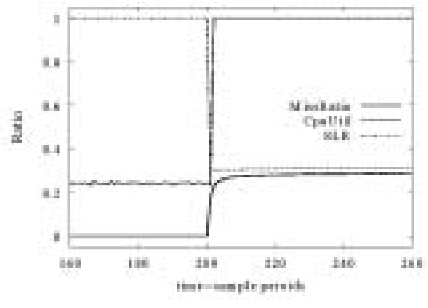
g. QoS Negotiation(ETF=1.25)



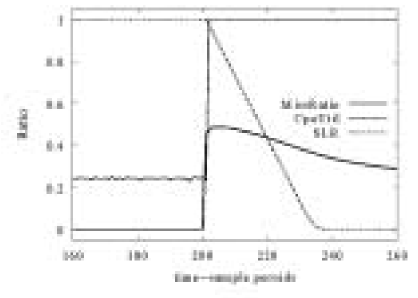
h. DQM(ETF=1.25)



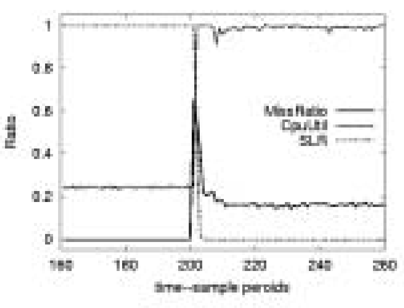
i. NCLOSE (ETF=1.25)



j. QoS Negotiation(ETF=1.50)



k. DQM(ETF=1.50)



l. NCLOSE (ETF=1.50)

Figure 7 System Performance Comparison

CPU utilization. Once the miss ratio is higher than the threshold, the DQM will lower the service level. If the CPU utilization is lower than the threshold, DQM will increase the service level of the system. This is actually a kind of heuristic feedback. The initial algorithm of DQM was designed for a centralized system. To neutralize this limitation, we optimally distributed the total incoming load to N distributed DQM nodes. Because of the way we distribute the load, each DQM node can act independently.
Experimental Setup

In most cases, the same experimental setup as that in Section 4 is used. One important variation in this experiment is to use different values of *ETF* to cover different cases of execution time estimation. As we discussed before, it is usually difficult to provide an accurate estimation of the actual execution time beforehand because of many unknown factors. To make a more realistic comparison, the experiments have evaluated the performance with different estimations:

$$Avg.\ Execution\ time = ETF \times Estimated\ Execution\ time$$

In these experiments, we run the simulation for each algorithm with ETF = 0.75, 1.00, 1.25 and 1.50, to cover the cases of underestimation and overestimation. In the experiment, all tests were run 20 times. However, the graphs represent the trace of single run since it shows how the system acts over time. The long term average for CPU utilization and miss ratio were extremely tight (within 1% of the mean) using a 90% confidence interval.

As shown in Figures 7.d and 7.f, QoS negotiation outperforms NCLOSE by 10% when the estimation is accurate. This should be the case because QoS negotiation is a well-tuned algorithm under the assumption that task execution time are known exactly. Note that its performance depends heavily on the accuracy of the estimation of the execution time. In many real systems it is usually difficult to provide the accurate estimation beforehand.

When the actual execution time is overestimated ($ETF = 0.75$ in Figure 7.a), the QoS negotiation algorithm makes conservative decisions, resulting in lower CPU utilization (76%) and provides low average service level (0.31), compared to 100% CPU utilization and 0.71 average service level of our NCLOSE algorithm ($ETF = 0.75$ in Fig 7.c).

When the actual execution time is underestimated ($ETF > 1$), the performance of QoS negotiation becomes even worse. This is because the scheduling decision is based on inaccurate estimations. In the underestimated situation, the estimated execution time is less that the actual execution time. In that case, the scheduler makes poor decisions by requiring less resource than actually needed by the tasks, causing them to miss their deadlines. The deadline miss ratio reaches 18% for *ETF = 1.25* in Fig 7.g and 28% for *ETF = 1.50* in Fig 7.j, respectively. This is mainly because the algorithm is an open loop algorithm and hence it lacks the mechanism to dynamically adjust its decision according to the actual outcome from the system.

The results also show several deficiencies of DQM. DQM cannot effectively eliminate the error, (i.e., reaching zero miss ratio error) at steady state (Fig 7.b, 7.e, 7.h, 7.k). The problem is that unlike the feedback control in NCLOSE, there is no effective way for DQM to minimize the miss ratio. DQM will lower the service level once the miss ratio is higher than the threshold. One way to lower the error is to choose a smaller threshold. The problem is that if the threshold is very small, the system becomes excessively sensitive to disturbances and becomes unstable. A similar problem exists with the CPU utilization threshold. Furthermore, DQM cannot effectively handle the step load and its reaction to the load change is sluggish. For example Figure 7.e, after 60 sampling periods, it still has 8% miss ratio. Unlike feedback control, which can output a larger control signal to eliminate the error more quickly when the error is large, DQM responds to high miss ratio according to the heuristic, i.e., lower the service level with the highest CPU utilization/benefit. This leads to the slow response, especially when the miss ratio is high. When the estimation is inaccurate, the performance of DQM is even worse, both in transient and steady-state response. The source of these problems is that DQM is a heuristic feedback control algorithm, which adjust the system heuristically based on the output. Without a theoretical basis, such an approach can be inefficient and difficult to tune.

The results show that the feedback-control based approach is very effective. Although the scheduler depends on the estimation to make a decision, the controller can adapt to the inaccuracy of the estimation by monitoring the actual output and adjusting the system according to the difference between the set point and the actual output. When the miss ratio is higher than the set point after the step load, it can adjust service level and therefore reduce the miss ratio. This is also true for the CPU utilization control. This gives NCLOSE the advantage of working in uncertain and unpredictable situations. Furthermore, as we showed in Section 4, based on system modeling, we performed systematic design based on control theory. Given the performance specification such as steady-state error and settling time in Table 1, we designed the controller that satisfies the specification. Through this approach, we have theoretically and experimentally proved the effectiveness of our DFCS framework.

6. Related work

Recently, a new generation of adaptive architectures has emerged. This architecture differs in two respects from early adaptive approaches. First, the performance of the adaptive system is modeled in some coarse-grained manner that represents the relation between aggregate QoS and aggregate resource consumption. This is as opposed to fine-grained models that require knowledge of individual task execution times. Second, feedback was used as a primary mechanism to adjust resource allocation in the absence of *a priori* knowledge of resource supply and demand. This is in contrast to early optimization-based QoS adaptation techniques that assumed accurate models of application resource requirements. Examples of such approaches include [4][7]. In [7], a transaction scheduler called AED monitors the system deadline miss ratio and adjusts task priorities to improve the performance of EDF in overload. The DQM algorithm [4] features a feedback mechanism that changes task QoS levels according to the sampled CPU utilization or deadline misses. These algorithms are based on heuristics rather than theoretical foundations. A particularly interesting approach that belongs in this category is one that uses feedback control theory as the underlying analytical foundation for building feedback-based adaptive systems. This approach uses difference equations to model the aggregate behavior of the system. These equations are then used to design feedback loops with guaranteed properties on both steady state and transient behavior such as speed of convergence. This approach has been adopted by several researchers. For example, Li and Nahrstedt presented a fuzzy-controller on a sender node that dynamically adjusts the sending rate in a distributed visual tracking system [9]. In [5], Buttazzo et.al. presents an elastic task model for adaptive rate control. In [13] integrated control analysis is presented. Hollot et. al. [8] presents control analysis of a congestion control algorithm in Internet routers. In [7], Gandhi et. al. designs a feedback controller to control the queue length of a Locus e-mail server. Eker [6] proposed to integrate on-line CPU scheduling with control performance in embedded control systems. Abdelzaher et. al. [2][3] developed a web server architecture that guarantees desired server utilization under HTTP 1.0. Lu et. al.[10] presented a new feedback architecture to provide relative and absolute delay guarantees in HTTP 1.1 servers. In [11][12], feedback control real-time scheduling algorithms are developed to achieve deadline miss ratio guarantees in uni-processor systems. This paper generalizes the control-theoretical QoS-adaptation approach to distributed systems. Unlike the above solutions that deal with a single controller on a single node, we develop models and algorithms for a group of decentralized controllers that controls the aggregate performance of a distributed system (such as networked embedded systems in smart spaces).

7. Conclusions

We have developed an effective distributed real-time scheduling approach based on feedback control. We have shown that the algorithm is stable and meets transient performance requirements. The algorithm is based both on a novel analytical model and employing standard feedback control design techniques using that model. A performance evaluation study has demonstrated that the algorithm outperforms both baselines and important previous algorithms from the literature in uncertain environments.

8. References

[1] T. F. Adbelzaher, E. Atkins and K. Shin. QoS Negotiation in Real-Time Systems and Its Application to Automated Flight Control. *IEEE transaction on Computers*, 49 (11), Nov. 2000.

[2] T. F. Abdelzaher and N. Bhatti, Adaptive Content Delivery for Web Server QoS. *International Workshop on Quality of Service*, London, UK, June 1999.

[3] T. F. Abdelzaher and K.G. Shin. QoS Provisioning with qContracts in Web and Multimedia Servers. IEEE Real-Time Systems Symposium, Phoenix, Arizona, December 1999.

[4] S.Brandt and G.Nutt. A Dynamic Quality of Service Middleware Agent for Mediating Application Resource Usage. In Real-Time Systems Symposium, pages 307--317, Madrid, Spain,December 1998.

[5] G. Buttazzo, G. Lipari, and L. Abeni, Elastic Task Model for Adaptive Rate Control. IEEE Real-Time Systems Symposium, Madrid, Spain, pp. 286-295, December 1998.

[6] J. Eker. Flexible Embedded Control Systems-Design and Implementation. *PhD-thesis*, Lund Institute of Technology, Dec 1999.

[7] J. R. Haritsa, M. Livny and M. J. Carey. Earliest Deadline Scheduling for Real-Time Database Systems. IEEE Real-Time Systems Symposium, 1991.

[8] C. V. Hollot, V. Misra, D. Towsley, and W. Gong. A Control Theoretic Analysis of RED. IEEE INFOCOM, Anchorage, Alaska, April 2001.

[9] B. Li and K. Nahrstedt. A Control-based Middleware Framework for Quality of Service Adaptations. IEEE Journal of Selected Areas in Communication, Special Issue on Service Enabling Platforms, 17(9), Sept. 1999.

[10] C. Lu, T. F. Abdelzaher, J. A. Stankovic, and S. H. Son.. A Feedback Control Approach for Guaranteeing Relative Delays in Web Servers. IEEE Real-Time Technology and Applications Symposium, Taipei, Taiwan, June 2001.

[11] C.Lu, J.A. Stankovic, G.Tao, and S.H. Son. The Design and Evaluation of a Feedback Control EDF Scheduling Algorithm. In IEEE Real-Time Systems Symposium, Phoenix, Arizona, December 1999.

[12] C.Lu, J.A. Stankovic, T.Abdelzaher, G.Tao, S.H. Son, and M. Marley. Performance Specification and Metrics for Adaptive Real-time Systems. In IEEE Real-Time Systems Symposium, Orlando, Florida, November 2000.

[13] L. Palopoli, L. Abeni, F. Conticelli, M. D. Natale, and G.Buttazzo. Real-Time Control System Analysis: An Integrated Approach. IEEE Real-Time Systems Symposium, Orlando, FL, Dec 2000.

[14] J.Stankovic. Decentralized Decision Making for Task Reallocation in a Hard Real-time System. *IEEE Transactions on Computers*, 38(3):341--355, March 1989.

Achieving Generic Asymmetric Sensing Architecture

Yu Gu and Tian He
{yugu,tianhe}@cs.umn.edu
Department of Computer Science and Engineering
University of Minnesota, Twin Cities

1 Introduction

As a key approach to achieve energy efficiency in sensor networks, sensing coverage has been studied extensively in the literature. Researchers have designed many coverage protocols to provide various kinds of service guarantees on the network lifetime, coverage ratio and detection delay. While the state-of-the-art is encouraging, we believe there are some aspects that need further investigation. First, currently different sensing coverage algorithms focus on different service guarantees (e.g., coverage vs. detection delay). Any single design is not general enough to meet a wide range of sensing requirements under different operating scenarios. Second, lifetime extension in most algorithms is essentially achieved through coordination among neighboring nodes. The local node density, therefore, imposes a theoretical upper bound on the system lifetime. Such a bound can be surpassed through global scheduling. However, the overhead of global scheduling would increase significantly if the coordination among the nodes goes beyond the boundary of neighborhood.

To address these two issues simultaneously, in this abstract, we propose a Unified Sensing Coverage Architecture, called uSense, which features three novel ideas: *Asymmetric Architecture*, *Generic Switching* and *Global Scheduling*. We propose asymmetric architecture based on the conceptual separation of switching from scheduling. Switching is efficiently supported in sensor nodes, while scheduling is done in a separated computational entity, where multiple scheduling algorithms are supported. Such asymmetric architecture enables us to design sophisticated coverage algorithms in an unconstrained design space, represent such intelligence with a lightweight algorithm implemented in sensor nodes and achieve a fast and efficient change of coverage algorithms by disseminating only several parameters. As far as we know, the proposed *generic switching* is the first generic and lightweight switching algorithm for resource-limited sensor nodes. As an instance, we propose a two-level global coverage algorithm, called uScan. At the first level, coverage is scheduled to activate different portions of an area. At the second level, sets of nodes are selected to cover active portions. Interestingly, we show that it is possible to obtain optimal set-cover results in linear time if the layout of areas satisfies certain conditions. The global scheduling, which coordinates nodes within a large area, could lead to significantly better performance compared with localized solutions. Our initial evaluation results indicate that uSense is a promising architecture to support flexible and efficient coverage in sensor networks.

2 uSense Architecture

The uSense design consists of two parts as shown in Figure 1. The switching algorithm is implemented in the sensor nodes, which takes only two scheduling parameters as input: *working schedule bits: S* - an infinite binary string in which 1 denotes the active state and 0 denotes the inactive state, and *switching rate: R* - defines the rate of toggling between states. The computational entity is responsible for generating the scheduling parameters. It takes the schedules decided by sensing coverage algorithms and translates them into the parameters used by the switching algorithm. In order to efficiently disseminate these two parameters to the sensor nodes, we propose to express the *schedule bits*, which is usually periodic and following a certain pattern, in the form of regular expression. To further reduce communication costs, we use a parameterized schedule, which is a binary regular expression that changes with the node's location and enables us to disseminate the schedules through one limited flooding instead of unicasting to every single node.

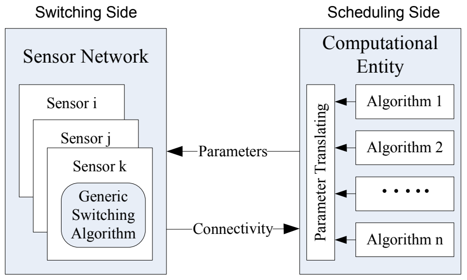


Fig. 1. Overview of uSense Architecture

3 Global Scheduling Algorithms

Conceptually, uSense can support many existing coverage algorithms. Due to its asymmetric architecture, it is especially friendly to the global scheduling algorithms. Since a global scheduling allows many more nodes to activate in turn rather than the localized ones that only schedule the nodes within neighborhood, it leads to a significant energy savings. In this section, we describe a global scheduling algorithm called uScan, which is one of sensing coverage algorithms in Figure 1 that are supported by the uSense architecture.

The outputs of uScan are the scheduling bits *S* and switching rate *R* for individual nodes. uScan is a two-level schedule algorithm, which works as follows: Suppose we provide sensing coverage to a given area using uScan. First, uScan divides the area into small regions, and decides the working schedules for these regions. This level of scheduling is conceptually independent of the deployment of the nodes. At the second-level, we assign nodes to cover the active regions at different time intervals, using a set-cover technique. By combining the first-level schedule and the set-cover assignment, we can decide the schedule bits *S* for individual nodes.

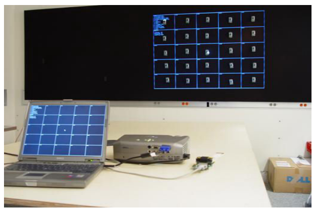


Fig. 2. uSense System Setup

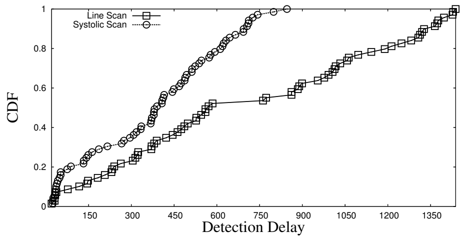


Fig. 3. Detection Delay of Line and Systolic Scan for Static Targets

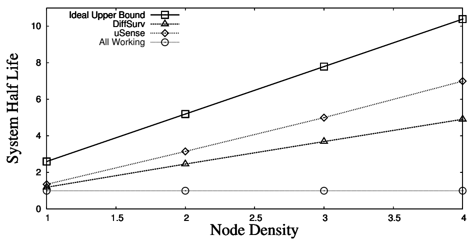


Fig. 4. System Half Life vs. Node Densities

3.1 Level I: Tile Scheduling

In uScan, we partition an area under surveillance into some small regions of the same shape, a process called tessellation. These small regions are called *tiles*, which can be regular triangles, rectangles or regular hexagons in a 2-D space. The size of tiles is set to be smaller than the minimum target size, so that a target is detected as long as a portion of a tile is covered. As a reminder, nodes within a sensor network only support a generic switching algorithm, which has neither the concept of tiles nor the partition information of the tiles. All the complex logic resides in the computational entity. In this section, we discuss two methods for the tile-level scheduling. They differ in the energy consumption rate and the detection delay.

- **Line Scan:** Instead of trying to cover all tiles, we only cover a column/row of tiles in a certain interval of time during one round of scan. The covered columns/rows are increasing or decreasing consecutively.
- **Systolic Scan:** Systolic Scan emulates the cardiac cycles of a beating heart. Over the area under surveillance, we sense the tiles from the inner layer to the outer layer continuously.

Both line scan and systolic scan specify only the set of tiles need to be activated (covered) at a given point of time. The task of covering each tile set is accomplished by the second-level node scheduling, which is described in the next section.

3.2 Level II: Node Scheduling

Tile-level scheduling determines the set of active tiles TS_i at the time interval i. In this section, we describe how we can translate a known tile schedule into a corresponding node schedule bits S, which can be interpreted directly by a generic switching algorithm.

The main idea of our approach is to find the optimal set of nodes which could cover all the tiles that need to be active at time interval i. Before node scheduling, we first map physical node coverage into the coverage bipartite graph according to the coverage relationship. Then we divide node scheduling into two steps. First, for a tile set TS_i, we keep identifying 1-cover set with minimal number of nodes, until the size of 1-cover set is above a certain threshold. Secondly, we create schedules for nodes such that each identified 1-cover set provides coverage to TS_i in a round-robin fashion.

The generic Minimum Set Cover problem has been proven NP-Hard [1]. Fortunately, we find line scan coverage is a special case of the generic set cover problem, because a node only need to cover a continues segment of tiles. By mapping the coverage bipartite graph into a directed acyclic graph with following rules:

1. Map N tiles in TS_i into N vertex $V = \{v_1, ..., v_N\}$ and add one extra vertex v_{N+1}.
2. If a node covers a set of tiles $\{T_i, ..., T_{i+n}\}$, we create n directional edges (v_i, v_j) where $v_j = v_{i+1}, ..., v_{i+n+1}$. Each edge has a unit cost.

We reduce the tile set cover problem to the problem of finding out the shortest paths from v_1 to v_{N+1}, with the overall runtime of $O(|V|) + O(|E|)$.

We note that the proposed polynomial algorithm does not apply to generic tile scheduling, where a tile set does not form a continuous curve or where a node can cover multiple segments of a tile set simultaneously. In these cases, we adopt a greedy set-cover method by choosing the node that covers the most number of tiles first.

In order to support line scan or systolic scan in a 2-D space, we need to identify cover sets for the whole area(not just for a single tile set). Thus a node may need to cover multiple tile sets TS_i. To effectively handle the cases where we have to select cover sets for multiple TS, we designed an algorithm that each node maintains a counter SC which records how many times it has been selected into a unique cover sets. While selecting cover sets for each tile set TS, instead of solely consider the number of nodes in the cover sets, the algorithm calculates the minimum cover set among the nodes whose SC counter values are as small as possible.

After obtaining cover sets for every tile set TS_i, we build the final schedule of node according to all the cover sets it's belonged to, which can be executed directly by our generic switching algorithm.

4 Implementation and Evaluation

We have implemented a complete version of uSense on Berkeley TinyOS/Mote platform, using 30 MicaZ motes as shown in Figure 2. The results showed that uSense is a lightweight and efficient architecture. To reveal the system performance at scale, we have conducted some initial large scale simulations with 10,000-node. Under full coverage mode, we demonstrated that our global scheduling algorithms provide significant energy savings over previous protocols such as DiffSurv [2] under metrics such as Half-life, Coverage Overtime and Node Energy Consumption. In the future, we plan to investigate the performance under partial coverage mode at scale as well, with additional metrics such as Detection Delay for Static Targets and Worst-Case Breach (WCB) for Mobile Targets.

5 References

[1] V. T. Paschos. A Survey of Approximately Optimal Solutions to Some Covering and Packing Problems. In *ACM Computing Surveys*, June 1997.

[2] T. Yan, T. He, and J. A. Stankovic. Differentiated Surveillance Service for Sensor Networks. In *SenSys'03*, November 2003.

An Overview of Data Aggregation Architecture for Real-Time Tracking with Sensor Networks

Tian He[†], Lin Gu[*], Liqian Luo[‡], Ting Yan[*], John A. Stankovic[*], Sang H. Son[*]
[*]Department of Computer Science, University of Virginia
[†]Department of Computer Science and Engineering, University of Minnesota
[‡]Department of Computer Science, University of Illinois, Urbana-Champaign

Abstract— **Since sensor nodes normally have limited resources in terms of energy, bandwidth and computation capability, efficiency is a key design goal in sensor network research. As one of techniques to achieve efficiency, data aggregation has been extensively investigated in recent literature. Previous research on data aggregation has demonstrated its effectiveness in reducing traffic, easing congestion and decreasing the energy consumption. However few are actually designed for a real-world application and implemented in a running system. This paper describes our design and implementation of a physical tracking system, using an aggressive data aggregation architecture as one of building blocks. This architecture can be generally applied to other sensor systems, where communication efficiency is a paramount concern and networking resources are limited.**[1]

I. Introduction

Traditional military surveillance systems, using long-range cameras/radars, are effective in the open terrains where direct line-of-sight is available. While in the urban areas and forests, the effectiveness of these solutions is affected by the obstacles such as tall buildings and trees. To address this issue, the military starts to use wireless sensor networks as an effective surveillance instrument to deal with the non-line-of-sight (NLOS) situations, because that sensor nodes can be deployed anywhere around an environment to provide ubiquitous surveillance. Due to the stealthiness requirement of the military surveillance systems, a tiny form factor is essential. Consequently, sensor nodes have very limited resources, suffering the bandwidth, energy and memory constraints that limit the amount of information that could be transferred. These factors are coupled with unpredictable traffic patterns and dynamic network topologies, making the task of data delivery for such networks difficult. Theoretically, for a given energy budget available in the network, the total amount of bits that can be transmitted is limited. It is desirable to have the capability to deliver more *information* with the same amount of bits over the air. It is often the case that the data represented by these bits is redundant. For example, a series of sensing readings with same values can be concisely described by an average with a zero standard deviation. As one type of data aggregation techniques, this parameterized description of the data distribution can effectively reduce the amount of data transmitted. Since data aggregation can reduce transmissions while still distributing information about the events of interests, it is deemed as a very effective resort to balance the communication needs and energy constraints.

This paper addresses the research challenges related to the data aggregation technique in real-time surveillance applications. We identify the fundamental tradeoffs that can balance the performance within a three-dimensional design space: namely the timeliness, the energy consumption and the information availability. Ideally, we desire to deliver enough information in real-time with minimal energy consumption. Unfortunately, these performance goals are often at odds with each other. For example, a swift delivery prevents a node accumulating sufficient data for energy-efficient data aggregation. It is an interesting research problem to identify a performance surface within this three-dimensional space, so that a system designer can make guided decisions to trade off among the energy, time and data availability, according to the application requirements and system configurations.

To demonstrate our approach to achieve this goal, we employ a typical sensor tracking system, called VigilNet, as a case study. We introduce the data aggregation architecture designed, implemented and integrated in VigilNet and identify the tradeoffs VigilNet provides. Since VigilNet is a typical sensor network system, we believe our studies can render insights for the system designers of similar systems.

The contribution of this work lies in the following aspects: 1) Unlike the previous approaches that mainly focus on the simulation study, we demonstrate how various data aggregation techniques can be designed and implemented practically in a real world application. 2) We reveal the impact of data aggregation on the quality of surveillance, the timeliness and the related overheads. Such an analysis can guide system designers to flexibly change the system configurations in order to accommodate various kinds of operation scenarios. 3) Compared with the solutions without data aggregation, we demonstrate analytically that our approach significantly reduces the amounts of energy consumed.

[1]This work was supported in part by the DAPRPA IXO offices under the NEST project (grant number F336615-01-C-1905), the MURI award N00014-01-1-0576 from ONR and NSF grant CCR-0098269.

II. Related Work

In this paper, we focus on the data aggregation techniques applicable to the wireless sensor tracking systems. We divide our discussion into two categories: data aggregation techniques and sensor tracking systems in general.

A. Data Aggregation Approach

Data aggregation techniques have been widely used in wireless sensor networks. In [13], Intanagonwiwat proposes several basic forms of data aggregation methods, including 1) the Center at the Nearest Source method (CNS), where the source nearest to the destination aggregates the data from other nodes; 2) The shortest Path Trees (SPT) method, where data aggregation happens at the intermediate nodes within a shortest path tree rooted at the sink; and 3) the greedy Incremental Trees (GIT) method, where an aggregation tree is constructed by connecting each destination sequentially to the existing tree via a shortest path. GIT assumes a complete knowledge of global topology information; therefore it provides more opportunities for data aggregation. TAG provides a hierarchical data aggregation scheme at a data collection phase. Using an acquisitional query processor for data collection, TinyDB [17] optimizes the query process to aggregate data with a low energy overhead. Directed Diffusion [14], as a popular data-centric architecture for data acquisition, can be augmented by aggregating data along the reinforced paths from the sources to the sinks. Another type of data aggregation techniques focuses on the data placement of caching services. Bhattacharya et al. [1] investigate the optimal placement of caches between multiple sensor sources and sinks. These caches aggregate the updates from the source nodes and distribute data to leaf sink nodes with minimal requested rates. Since all aforementioned approaches are designed for the systems with no stringent time requirement, the designers of these systems naturally treat the timeliness as a less important issue related to the data aggregation. The closest research related to this work is the AIDA protocol [8]. AIDA takes the timely delivery of messages as well as the protocol overhead into account to adaptively adjust aggregation strategies in accordance with assessed traffic conditions and expected sensor network requirements. Through simulations, it demonstrates the feasibility to reduce the energy consumption and the end-to-end communication delay simultaneously. Our work presented here differs from aforementioned approaches in several aspects: First, this work deals with the real-time issue along with the data aggregation. Second, this work introduces not only the usage of multiple aggregation techniques, but also how these techniques can be intergraded within a tiered architecture. Third, our work is not a simulation study. Instead it is designed for a realistic tracking application with a running implementation.

B. Research on Sensor-based Surveillance and Tracking

Traditional surveillance systems are widely used for decades. Due to space constraints, we only name a few directly related ones. The ASDE system [3] and Secure Perimeter Area Network (SPAN) [4], normally use long-range cameras/radars with a 360-degree view of an area as the instruments for the detection. While these infrastructure protection systems are effective, they are subject to several limitations: First, they can not be emplaced swiftly without infrastructure, which makes such system not suitable for spontaneous military deployment in the remote areas. Second, the large form factor of these systems makes them easily detectable and evadable. Third, the number of surveillance points is small, which makes systems vulnerable to attack. To overcome these limitations, sensor networks is pursued recently by [2], [10], [11], [20], [19] as a more efficient mechanism to accomplish the remote unmanned surveillance missions. Feng et al. [20] design a surveillance and tracking system using a distributed Bayesian estimation technique. Brook et al. [2] propose a distributed surveillance system based on extended Kalman filter techniques. These solutions provide very nice features to improve the surveillance performance in one aspect or another, however ignoring other performance goals. For example, some systems provide high performance at cost of the system lifetime - a critical performance metric for long-term surveillance. The difference between our proposed work and aforementioned approaches is that we adopt our solution in a multiple-dimensional design space, where we consider not only the tracking performance, communication efficiency through data aggregation, but also the timeliness in delivery. This requires a balanced and flexible system design. Another highlight of our approach is the system implementation, which addresses many practical issues hard to capture in the simulated tracking scenarios.

III. Data Aggregation Requirements in VigilNet

The VigilNet is an online surveillance system, which consists of hundreds of tiny sensor nodes. These nodes detect, track and classify incoming targets in a timely and energy efficient manner. The final results are reported to a remote back-end via a long-haul bandwidth-constrained satellite links. In the current hardware platform, each node is equipped with three types of sensors. Magnetic sensors can detect the changes in the magnetic field caused by the movement of ferrous objects. Motion sensors can detect the changes in the infrared radiation caused by warm objects such as personnel. The acoustic sensors detect sound waves. Target signatures can be identified in VigilNet by sampling aforementioned three sensors at the particular rates (e.g. 10bits data at 1000HZ from acoustic sensors). Suppose 100 sensors participate in a tracking process, VigilNet could generate 1Mbit data in one second by using the acoustic sensor alone. The total amount of data to be transmitted multiplies with the number of hops over the network. Given the fact that the current long-haul satellite link only provides

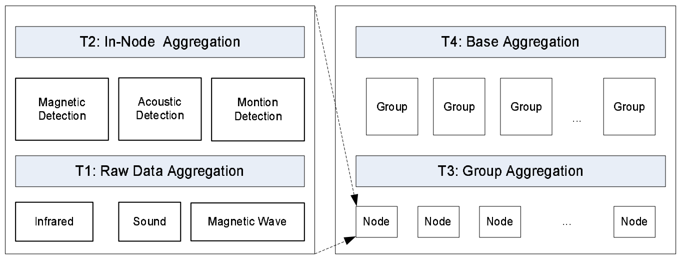


Fig. 1. Four Tier Data Aggregation Architecture

a 1200bps data rate, VigilNet can only send approximately 1-bit aggregated data out of every 1,000 bits of raw sensor readings generated from the network. This requires us to design an aggressive data aggregation architecture in VigilNet.

IV. Four-Tier Data Aggregation Architecture

Normally, data aggregation ratio for a given system is simply the size of the original data divided by the size of the aggregated data. Based on our experience, a single data aggregation strategy is neither sufficient nor flexible to achieve an aggressive a 1000:1 data aggregation ratio. If data aggregation is only done at the node level, information could be lost too early to be useful. If data aggregation is only done at a central site, a sensor network spends too much energy in transmitting the data and possibly suffers severe network congestion and message losses. To balance energy efficiency and data availability, we design and implement a four-tier data aggregation architecture in VigilNet as shown in figure 1.

1) The first layer (T1) is the raw data aggregation layer, which takes the sensor readings form the individual sensors and converts them to the detection confidences, values between zero and one, indicating how confident a detection algorithm of the individual sensor is about the existence of the target. The data aggregation ratio at this layer is largely determined by the slowest raw sampling rate and the frequency in generating the confidence values. VigilNet can achieve approximately a $50 \sim 100 : 1$ ratio at this layer.
2) At the second layer (T2), a node takes the detection confidence values from different sensors to form a single classification vector, which indicates the target type and the corresponding confidence values. With three sensors, this layer achieves a 3:1 aggregation ratio.
3) At the third layer (T3), all nodes that detect the same target join the same logic group to track the target. Each group is represented by a leader to maintain the status of the target by aggregating all the reports from the member nodes. The leader node aggregates not only the location of reporting nodes as well as their confidence vectors together. Periodically, the leader node sends a report, consisting of a time stamp, an aggregation location and a confidence vector, to the base. The sensing density determines the aggregation ratio. In the VigilNet case, we expect an aggregation ratio between 3:1 and 10:1.
4) The fourth layer of aggregation (T4) happens at the base. A base aggregates the individual reports form the same logic group (a group that tracks the same target) together to generate a final report which contains the target type, bearing, speed and detection time-stamp. The aggregation ratio in this layer is determined by the tracking history length. Normally an aggregation ratio between 2:1 and 10:1 can be achieved.

In the next several sections, we give more detail on each layer, and provide the analysis that reflects the tradeoffs between energy efficiency and other properties of the system such as the timeliness.

V. T1: Sensor-Level Data aggregation

The sensor data is the raw input to the network's computation work flow. It provides the foundation of the information processing for tracking events in the network. Data aggregation at this layer should meet following requirements.

- Meet real-time constraints: Because the system deals with transient events, such as fast-moving targets, in the network, the sensor data needs to be processed in a timely manner. If the processing latency is too long, a target would move out of the sensing range before the detection finishes
- Deal with a large volume of inputs: Endeavoring to accomplish reliable, timely, and quality sensing and tracking, a surveillance application often uses multiple sensors and samples them at very high rates. As a result, the combined sampling rate is high, especially, when the acoustic sensing is employed, and the volume of the sensor data input is large.

In the rest of this section, we first introduce different types of sensors and sensor data in the VigilNet, then discuss possible aggregation methods, and finally evaluate the aggregation method used in this layer.

A. *Sensors and sensor data*

VigilNet uses the ExScal motes as sensor nodes. Based on the Mica2 [5] mote design, the ExScal mote, shown in Fig. 2, is designed by CrossBow Inc. and Ohio State University for large-scale surveillance WSNs [6]. The major difference between the ExScal mote and the Berkeley Mica2 mote is that the former integrates a magnetometer (Honeywell HMC1052[12]), a microphone, and 4 PIR sensors on the same circuit board as the processor's.

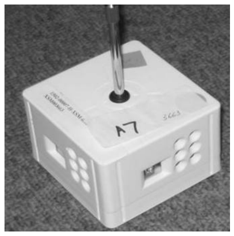


Fig. 2. ExScal mote

Compared to sensors used in other applications, such as the temperature sensors (e.g., the Panasonic ERTJ1VR103J thermistor used on Mica sensor board) and light sensors (e.g., the Clairex CL9P4L photo sensor), the PIR sensors and microphone on ExScal motes track target signals with a relatively high frequency. For example, the microphone can potentially detect a waveform of 16KHz. This leads to a high degree of data availability at the sensor layer. More specifically, we list the sensor and the sensor data used in the following table.

TABLE I
SYSTEM PARAMETERS

Sensor	Type	Sampling rate	Note
Magnetometer	DC	32	8-bit POT 2-axis
PIR	AC	50	
Microphone	AC	1000	

B. Aggregation methods

We call the sensor reading at a specific time on a specific sensor on a specific node a *sample point*. When a sensor network starts operation, each sensor on each node in the network produces a sequence of sample points. All the sample points produced by the network form a set and we call it the *global sample set*.

The global sample set is the complete information about what happens in the network. If all the nodes report their sample points to a base, the base can collect the global sample set and perform computation with it. However, as mentioned in Section 1, due to resource and energy limit, we prefer to deliver same information with fewer bits.

Let's first compute the amount of raw bits generated on one sensor. They correspond to all the sample points generated on one sensor. The magnetic sensor has two axes, each sampling at 32Hz, and the ADC output has 10 bits which are represented by two bytes in the software. Since the ADC values are the relative readings to the reference, one additional 8-bit potentiometer value is needed to record the reference value. Therefore, it takes 24 bits to record one sample point. In each second, 64 sample points are generated, and are represented with 1,536 bits, or 192 bytes. Suppose a sensor node keeps awake for one minute when some events of interest happen. The total number of data for this event is 92,160 bits, or 11,520 bytes. Similarly, we can compute the data generated by the PIR sensor has 48,000 bits, or 6,000 bytes, and the acoustic sensor generates 960,000 bits, or 120,000 bytes. In total, one event generates 1,100,160 bits, or 137,520 bytes.

As we can see, even one event generates a relatively large volume of sensor data. One potential method of aggregate the sensor data is to merge identical data and send out a "summary". However, with noise existing, the ADC seldom generates identical data even with a constant input. Hence, such merging is not effective.

Another way is to compute the differences between consecutive sample points, and record the difference, which is usually in a smaller range and can be represented by fewer bits. This leads to, virtually, a compression scheme. To assess the effectiveness of such compression, we use compression tools on PC to compress the sensor data, and examine the compression ratio. Experiments reveal that the compression ratio on DC signals by using gzip 1.3.3 is 100:37, and 100:31 with bzip2 1.0.2. Hence, even for DC signals, which is a simpler case, on a high-power platform with plenty of resources, the performance of the aggregation using such a method can hardly be satisfactory – it reduces approximately 2/3 of the data, but the remaining volume is still too large for sensor networks. Obviously, there are three key limitations for us to employ a traditional compression techniques: First, the transportation of 1/3 of the global sample set consumes an exorbitant amount of energy; Second, sensor nodes do not sufficient memory to accomplish these compression operations. Third, the latency to collecting and compressing these sample points is too long for a system that must react to the events in real-time.

Therefore, to enhance aggregation's performance to the level we desired, we design and implement a semantics based aggregation. By analyzing sensor data to retrieve its semantics, we achieves a highly efficient aggregation, as we will discuss next.

C. Evaluation of the semantic based aggregation

The reason for collecting sensor data is to form tracking knowledge. Hence, the semantics of the sensor data is the probability of the existence of specific targets. The VigilNet detects four types of targets. Hence, we use a 4-element vector (BV, SV, PF, PS), called confidence vector, to represent the semantics of the sensor data.

In the confidence vector, the elements BV, SV, PF and PS correspond to the four types of targets – big vehicle, small vehicle, person with ferrous objects, and person. The numeric value of these elements are the relative probability of the existence of a target being of the specific type. By using sensor level sensing and classification algorithms, we transform sensor data into confidence vectors [7].

The tracking report rate is a configurable parameter in the VigilNet. Suppose this rate is 2 reports per second, which is a common setting in some deployed systems. We can evaluate the aggregation performance of such a semantics based approach. Each element of the confidence vector is represented by one byte, hence the confidence vector contains 4 bytes. In each second, at most two (zero in case of no even detection) confidence vectors are needed for two tracking reports. Hence, aggregation ratio for the magnetometer is 25:1, for the PIR sensor is 25:2, and for the acoustic sensor is 250:1. Overall, the aggregation ratio for all the sensors is 100:1. Obviously, the semantics based aggregation is highly efficient.

VI. T2: Node-Level Data aggregation

Each sensor node has four PIR sensors, one magnetometer with two axes, and one microphone. The node performs further aggregation after collecting confidence vectors from the sensors. It computes the averages of the sensors' confidence vectors and form a single node-level confidence vector.

Because the three types of sensors form their own confidence vectors per type, the input of the node-level sensing and classification module are three confidence vectors. Hence, the aggregation ratio is 3:1. Combined with the 100:1 aggregation ratio accomplished at the sensor level, the overall node-level aggregation ratio is 300:1.

A. Delay/Energy Analysis w/wo aggregation

To retrieve semantics from sensor data and aggregate sensor level confidence vectors, the sensing algorithms and the node-level classification module need to buffer sensor data for a period of time and then perform their specific processing. This introduces delay into the network. Hence, we need to examine the length of the latency and verify that they are within a reasonable limit.

The delay introduced by the magnetometer is minimal – it buffers only one sample point and updates the confidence vector for each sample point. Hence, there is no delay introduce except for the mandatory sampling interval, which is about 16 milliseconds. Similarly, the PIR sensor introduces little delay because it buffers only a very small set of sample points. The delay introduced is still at the millisecond level. The acoustic sensor, however, buffers sensing data for about 1.2 seconds for processing. Hence, it introduces a latency of 1.2 seconds plus the sampling interval which is negligible in this case. Overall, the latency for accomplishing the best sensing and detection result is the maximum of the three sensors' latencies, i.e., 1.2 seconds.

The benefit, on the other hand, is obvious – with an aggregation ratio of 300:1, we reduces 99.67% of the communication payload. Though the semantics retrieval consumes CPU time and energy, it is a known fact that communication imposes orders of magnitude more energy overhead than computation in a sensor device. Hence, our estimation of energy saving due to data aggregation is still close to 99%.

VII. Group-Level Data aggregation

After forming node-level detection results, VigilNet starts to estimate the current position of the tracked target as well as uniquely and identically represent the target in a logical space. Estimation of target positions is usually done by calculating the weighted average of the locations of nodes reporting detections, using their individual detection confidence values as weights. However, there is a design decision to be made on when and where to conduct such calculations. Representation of targets, as a traditional topic in target tracking, has been widely addressed by either centralized or distributed temporal and spatial correlation algorithms. Our system borrows the distributed group management algorithm from EnviroSuite [15], an programming middleware for tracking and monitoring applications in sensor networks. In the following subsections, we describe in more detail the group aggregation algorithm used in the system and provide a theoretical analysis of how it trades off among information quality, delay and energy consumption.

A. Description of Group Based Data Aggregation

As stated above, tracking of a target consists of two main parts: position estimation and target representation. A simple way to tackle these problems would be to send the detection results and locations of individual nodes that detect targets to a centralized base station. Based on these received node positions [9], [18] , the base is able to estimate current positions of targets, and assign and maintain unique and consistent identities for targets by running temporal and spatial correlation algorithms. However, such a centralized scheme is inefficient both in energy and latency. It incurs excessive power consumption due to communicating multiple reports to a centralized base and may unduly increase latency, especially when targets are far away from the base. In addition, this centralized scheme can easily propagate the false alarms occurred in one part of the network to the base, which could overwhelm the base when the false alarm rate is high.

To avoid these limitations, we adopted a lightweight, distributed solution proposed in our previous work EnviroSuite [15]. Different from centralized solutions, EnviroSuite chooses to process data at or near the location where a target is detected, and sends only aggregates to the base for further processing. Specifically, EnviroSuite contains a set of group management algorithms to 1) instantiate globally addressable objects for targets as their logical representatives, 2)maintain a unique mapping between objects and targets, 3) guarantee the consistency of the mapping despite of target movements, and 4)suppress the false alarms locally.

Original EnviroSuite is completely dynamic, where nodes in the vicinity of a target is dynamically organized into one group, in which, a leader node is dynamically elected to host a corresponding object for the target. Though the dynamics of EnviroSuite enhances its robustness to failures including both message loss and node failures, it suffers a prolonged delay due to the long latency during the leader election, which is undesirable. To address this issue, our system uses a more static version of EnviroSuite, called Lightweight

EnviroSuite [16], where groups are still dynamically formed while leaders are pre-elected in an initialization phase.

The following part explains Lightweight EnviroSuite in more details through a step-by-step description. Note that to be concise, we review only features relevant to this paper. Concrete descriptions of detailed algorithms on leader election and object maintenance can be found in [16]. In the initialization phase, a subset of nodes are elected to be potential leaders with the guarantee that they provide 100% sensing coverage. Later on in the tracking phase, when a target gets detected by local nodes, these nodes immediately report their locations and detection results to their corresponding potential leaders (A potential leader L is claimed to be one of node's potential leaders if and only if L is within 2 times sensing range from that node). These reports are kept by the potential leaders until one of them have collected enough reports and is sensing the target. This potential leader then becomes a real leader, estimates the current position of the target and sends an aggregate report (containing position estimation as well as aggregated confidence vectors) to the base. Upon the reception of the aggregate report, other potential leaders drop their collected data and start over from the beginning again.

B. Delay/Energy Analysis w/wo Aggregation

To aggregate, the leader running EnviroSuite waits for enough reports from members. A configurable parameter *DOA (Degree of Aggregation)* is introduced to measure whether there are enough member reports. All potential leaders withhold their aggregate reports to the base until the number of collected member reports reaches DOA.

Intuitively it is expected that setting DOA to a very small value (e.g., 1) would minimize delays, which, however, is proved incorrect in a realistic, noisy environment the system operates on. Field experiments reveal that in such an environment, false positives of individual nodes are so frequent that, without filtering, excessive traffics between these nodes and the base tend to congest the network and impose large latency to other traffics that contain meaningful data. Therefore, it is critical to use a big DOA to filter out the false alarms of individual nodes, which not only improves the quality of information but also has *positive* affects on latency. Besides, energy consumption is also reduced due to less traffic towards the base. On the other hand, setting DOA to a high value (e.g., 5) is not desirable either, since a high DOA inevitably introduces longer delay. A DOA value higher than the number of nodes within a sensing range should be avoid, otherwise no report will be generated by the leader node. These tradeoffs between information quality, energy consumption and latency require that DOA has to be carefully chosen to satisfy these various aspects of design requirements. Analytical models on energy consumption and group aggregation delay as functions of DOA are built below to provide guidelines for system designers.

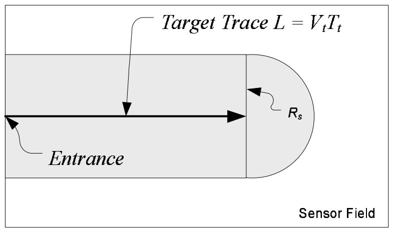


Fig. 3. The Detection Area

1) Energy Gain with Data Aggregation: It is extremely challenging to analyze realistic systems complicated by various factors, e.g., sensing range, target motion model, and node density. However, a rough approximation can be derived by making several simplified assumptions, including circular sensing range (R_s), straight target trajectory with velocity V_t, and uniformly distributed nodes with density d. This approximation gives us some insight on the system performance in general.

Assuming that a target enters the monitored field and moves for time duration T_t as shown in Figure 3, we compares energy consumption with/without aggregation as below. Since all the node that can detect the target are located within the gray rectangle or semi-circle as shown in Figure 3, the total number of reporting nodes is:

$$d(\pi R_s^2/2 + 2R_sV_tT_t) \quad (1)$$

Without group-level aggregation, each node directly sends its detection result to the base, where the expected energy consumption without aggregation(E_{wo_aggr}) is:

$$E_{wo\_aggr} = d(\pi R_s^2/2 + 2R_sV_tT_t)E\bar{n} \quad (2)$$

where E represents the energy consumed to send a one-hop message and $\bar{n}$ represents the average hop count between these detection nodes and the base.

With aggregation, each node sends its detection result only locally and, for every DOA detection nodes, there is an aggregate report sent to the base. Therefore, the expected energy consumption with aggregation (E_{w_aggr}) becomes:

$$\begin{aligned} E_{w\_aggr} &= d(\pi R_s^2/2 + 2R_sV_tT_t)E + \\ &\quad \frac{d(\pi R_s^2/2 + 2R_sV_tT_t)}{DOA}E(\bar{n}-1) \end{aligned} \quad (3)$$

Based on these equations, the percentage of conserved energy (P) due to group aggregation can be calculated as:

$$P = 1 - \frac{E_{w\_aggr}}{E_{wo\_aggr}} = 1 - \frac{1}{\bar{n}} - \frac{1}{DOA} \quad (4)$$

when $\bar{n} \geq 2$ and $DOA \geq 2$. These equations reveal the relation between energy consumption and DOA. As an example, when DOA is set to be 3 and the base is 3 hops away on average, group-level aggregation consumes approximately 33% less energy compared with the no-aggregation scheme.

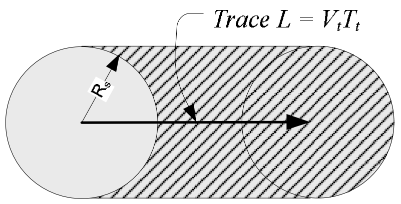


Fig. 4. Movement of a target during time period T

2) Delay Introduced by Aggregation: This section analyzes how tracking report latency is affected by DOA settings. Figure 4 depicts the movement of a target during time period T_t. The white circular and grey circular represent, respectively, the detection area of the target before and after the movement. Nodes located in the diagonally lined area are new detectors of the target and send out member reports for aggregation during this time period, assuming that they have a common potential leader. Let's assume that an aggregate report has been sent out just before the depicted movement, which means that a new aggregate report is not to be sent until the number of reports sent by these new detectors reaches DOA. Therefore, we have the expected delay caused by group-level aggregation T_g:

$$T_g = \frac{DOA}{2R_s V_t d} \tag{5}$$

This equation shows quantitatively the tradeoffs between delay and DOA (indication of information quality). To be more concrete, Figure 5 illustrates different group aggregation delays for varied target velocities and DOA values (setting sensing range to be 10 m and node density to be 1 per 100 m^2).

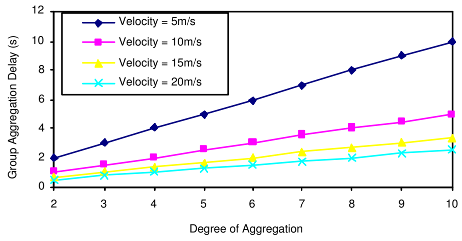


Fig. 5. Group aggregation delays for varied DOA values and target velocities

We note here that as one part of the tier-architecture, the group-level aggregation is not simply another method to further reduce energy consumption. With the inputs from multiple nodes, the group-level aggregation can eliminate the false alarms due to the faulty nodes and improve data quality, which can not be achieved by the node-level aggregation.

VIII. T4: Base-Level Data Aggregation

After receiving the reports from individual leader nodes, the base needs to aggregate the information further, an operation to serve three main objectives:

- Flow control: The long-haul link to the remote back-end could be the bottle neck of the system. A base is required to address the bandwidth mismatch between the sensor network output and the long-haul link capability. This is more likely to happen when the system tracks multiple objects simultaneously.
- Filtering: A base needs to prevent sending the duplicate reports as well as the false alarms to the back-end. To filter out the false alarms, a system needs to correlate the spatiotemporal properties of consecutive reports. Since the base is the only place in the network that has the complete global knowledge of a tracked target, it is at a better position than in-network nodes to filter out the system-wide false alarms.
- Consolidated View: End users is more interested in a consolidated view of the tracked targets instead of the individual sensing reports from the sensor network. Although the group-level aggregation does provide some persistent information such as object ID, it is not effective to keep a long trace history of a target through a sequence of hand-over operations among the leader nodes. To improve the efficiency, a base should be used to create a consolidated view instead.

A. Description of Base-Level Data Aggregation

The base bridges the system back-end and the sensor network. Its aggregation functionalities can be summarized as follows. First, the base takes the input from the in-network group leaders, and creates logical targets according to the spatiotemporal correlations of the input reports. Second, according to the information of the logic targets, the base filters out duplicate reports, messages with long delays, and false alarms, and provides flow control to match the bandwidth of the upstream link. Third, the base makes use of the incoming messages to provide extra information for the logic targets, such as target velocity and target classification information.

More specifically, when the base received a detection report from a certain location, it tries to associate the report with the closest logical target created recently. If the distance from the location of the report to that of the logical target exceeds a predefined threshold, a new logic target is created for the report. If the distance is below the threshold, the report is added into the history of the logical target and the location of the logical target is updated according to the report. An exception is that if the location of the report is the same as one of the last few locations in the history, the report is dropped as a duplicate report. A logical target expires after a certain amount of time if there has been no new report that is added into its history. By this approach the reports are categorized into logical targets based on their spatiotemporal correlations. When the base creates a new logical target, it needs to differentiate valid target reports from false alarms. This is done by accumulating

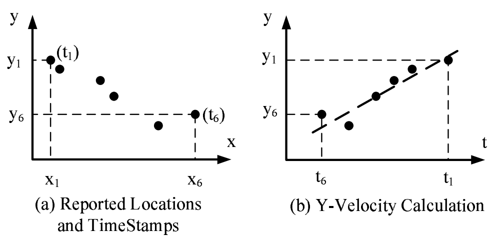


Fig. 6. Velocity Calculation from Reported Locations

spatiotemporal correlated reports up to a certain length before confirming the detection.

Another value that the base-level aggregation adds into the system is velocity calculation for the logical target. Since each message from the network includes a location and a time-stamp, we can use the locations and time-stamps in the logical target's history to infer the velocity of the target. We use a standard linear regression procedure to calculate the velocity. The formula for calculation of the x-axis component of the velocity is

$$v_x = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(t_i - \bar{t})}{\sum_{i=1}^{N}(t_i - \bar{t})^2} \tag{6}$$

in which x_i is the x component of the ith location, and t_i is the ith time-stamp. The x component of the latest location is denoted by x_1, and that of the one right before it is x_2, etc. N denotes the number of reports and time-stamps used for the calculation. The v_y calculation is similar to the Equation 6. As shown in Figure 6, v_y is the slope of the fitting straight line (the thick dashed line shown in Figure 6(b)).

To deal with the bandwidth limit to the system back-end, a flow rate parameter is set during the initial phase of the network operation. The flow rate parameter can be used to calculate the minimum interval between two consecutive reports of a logical target from the base: $D_{base} = 1/Nr$, in which N is the maximum allowable logical targets in the system and r is the flow rate. Obviously, the worst case delay introduced by the base-level aggregation is D_{base}. Supposing the reporting rate of each leader node is R, the aggregation ratio at the base is NR/r. We note that a naive solution is to do flow control at the group-level by setting the reporting rate of each leader node to r/N and the base relays every report to the back-end. In this naive design, a system could save more energy, however less data is available to create a consolidated target view. For example, the velocity estimation becomes less accurate and less fresh with fewer reports.

In summary, the base-level aggregation is essential and can not be replaced or eliminated by the aggregation at other layers, because of its ability to correlate the system-wide reports and resolve the flow rate mismatch between a sensor network and the link to back-end system.

IX. Conclusion

In this paper, we describe a multi-tier data aggregation architecture for target tracking applications. Due to space constraints, we omit the evaluation on this architecture. Details on its performance can be found at [10]. Since sensing data has different semantics at different layers, the pure in-network aggregation leads to low data availability for the high-level aggregation, while the pure centralized aggregation leads to excessive energy consumption. In contrast, the architecture proposed in the work can flexibly achieve the balance between energy, timeliness and data availability.

References

[1] S. Bhattacharya, H. Kim, S. Prabh, and T. Abdelzaher. Energy-Conserving Data Placement and Asynchronous Multicast in Wireless Sensor Networks. In *MobiSys'03*, May 2003.
[2] R. R. Brooks, P. Ramanathan, and A. Sayeed. Distributed Target Tracking and Classsification in Sensor Networks. *Proceedings of the IEEE*, 2002.
[3] S. Corporation. Asde-x brochure. http://www.sensis.com/docs/194//.
[4] T. S. Corporation. Surveillance for airport perimeter security. http://spacecom.grc.nasa.gov/icnsconf/docs/2002/11/Session_E2-5_Barry.pdf.
[5] CrossBow. *Mica2 data sheet*, 2003. at `http://www.xbow.com`.
[6] P. Dutta, M. Grimmer, A. Arora, S. Biby, and D. Culler. Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events. In *IPSN'05*, 2005.
[7] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, A.Tirumala, Q. Cao, J. A. S. T. He, T.Abdelzaher, and B. Krogh. Lightweight Detection and Classification for Wireless Sensor Networks in Realistic Environments. In *SenSys*, 2005.
[8] T. He, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher. AIDA: Adaptive Application Independent Data Aggregation in Wireless Sensor Networks. *ACM Transactions on Embedded Computing Systems, Special issue on Dynamically Adaptable Embedded Systems*, 2004.
[9] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-Free Localization Schemes in Large-Scale Sensor Networks. In *MOBICOM'03*, September 2003.
[10] T. He, S. Krishnamurthy, J. A. Stankovic, and T. Abdelzaher. An Energy-Efficient Surveillance System Using Wireless Sensor Networks. In *MobiSys'04*, June 2004.
[11] T. He, P. Vicaire, T. Yan, Q. Cao, G. Zhou, L. Gu, L. Luo, R. Stoleru, J. A. Stankovic, and T. Abdelzaher. Achieving Long-Term Surveillance in VigilNet. In *InfoCOM*, 2005.
[12] Honeywell. Hmc1052 magnetometers. http://www.ssec.honeywell.com/magnetic/.
[13] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *ICDCS*, 2002.
[14] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *MobiCom'00*, 2000.
[15] L. Luo, T. Abdelzaher, T. He, and J. A. Stankovic. Envirosuite: An environmentally immersive programming framework for sensor networks. In *ACM Transactions on Embedded Computing Systems, to appear.*
[16] L. Luo, T. He, T. Abdelzaher, and J. Stankovic. Design and comparison of lightweight group management strategies in envirosuite. In *DCOSS '05: International Conference on Distributed Computing in Sensor Networks*, June 2005.
[17] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The Design of an Acquisitional Query Processor for Sensor Networks. In *SIGMOD*, June 2003.
[18] R. Stoleru, T. He, J. A. Stankovic, and D. Luebke. A High-Accuarcy, Low-Cost Localization System for Wireless Sensor Networks. In *SenSys'05*, November 2005.
[19] T. Yan, T. He, and J. Stankovic. Differentiated Surveillance Service for Sensor Networks. In *First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, November 2003.
[20] F. Zhao, J. Shin, and J. Reich. Information-Driven Dynamic Sensor Collaboration for Tracking Applications. *IEEE Signal Processing Magazine*, March 2002.

Feedback Control of Data Aggregation in Sensor Networks [*]

Tarek Abdelzaher, Tian He, John Stankovic
Department of Computer Science, University of Virginia, Charlottesville, VA 22903
{zaher,tianhe,stankovic}@cs.virginia.edu

Abstract

Sensor networks have recently emerged as a new paradigm for distributed sensing and actuation. This paper describes fundamental performance trade-offs in sensor networks and the utility of simple feedback control mechanisms for distributed performance optimization. A data communication and aggregation framework is presented that manipulates the degree of data aggregation to maintain specified acceptable latency bounds on data delivery while attempting to minimize energy consumption. An analytic model is constructed to describe the relationships between timeliness, energy, and the degree of aggregation, as well as to quantify constraints that stem from real-time requirements. Feedback control is used to adapt the degree of data aggregation dynamically in response to network load conditions while meeting application deadlines. The results illustrate the usefulness of feedback control in the sensor network domain.

1. Introduction

The work reported in this paper is motivated by the rapid emergence of sensor networks [4] as a new paradigm for writing distributed applications. These networks are composed of a large number of small wireless sensor devices, each equipped with limited processing, communication, and storage capacity. Sensor networks are especially useful in applications involving a poorly accessible, dangerous, or unfriendly environment, where it is difficult to provide a fixed monitoring infrastructure. Instead, a myriad of wireless sensor devices can be deployed (e.g., by air-dropping from a UAV) for remote monitoring and surveillance purposes. Such air-dropped networks are called *ad hoc* sensor networks to distinguish them from other types of sensor networks where nodes are laid out in some fixed predetermined pattern. Ad hoc wireless sensor networks present the most challenge to the research community due to their inherent lack of structure. Example applications include habitat monitoring, defense, border control, and emergency response systems.

[*]The work reported in this paper was supported in part by the National Science Foundation under grants CCR-0093144, ANI-01 05873, EHS-0208769, and MURI N00014-01-1-0576.

This paper introduces fundamental research challenges presented by ad hoc wireless sensor networks from a feedback control perspective. These challenges lie in optimally reconciling the fundamental performance trade-offs that underlie network operation. Hence, the purpose of feedback control in this paper is not to control the dynamics of an external environment, but rather to control network performance itself. At a high level, performance of a sensor network can be viewed as a point in a three-dimensional space. These dimensions are (i) timeliness, (ii) energy consumption, and (iii) information output. It is desired to minimize energy consumption and maximize information output while maintaining timeliness. These requirements are mutually at odds; communicating more information takes more time and consumes more energy.

The nature of the trade-off among the basic sensor network performance requirements depends on current network input, which is the amount of sensory data infused into the network. For example, at low network load, timeliness can be easily achieved together with the other requirements. However, at a higher load, a decision has to be made between timeliness of delivery and the amount of deliverable information. Feedback control loops are needed to trade-off these performance requirements dynamically in a distributed fashion in response to current network conditions, essentially solving a distributed constrained optimization problem. This paper describes an instance of such a feedback control architecture, and derives some results in real-time computing that help quantify the constraints imposed on optimization.

The performance trade-offs mentioned above are fundamentally inherent to ad hoc sensor networks because they invariably arise from the main goal of such networks, namely the collection of sensory data. The most important output of a sensor network is the information it provides to external observers. One of the most limited resources in an ad hoc network is battery capacity. This is partly because advances in battery capacity have developed at a slower rate than advances in processing and communication bandwidth. Moreover, since the network is typically deployed in remote or harsh environments, changing batteries is quite expensive if not infeasible. Hence, maximizing battery lifetime by conserving energy is a predominant concern.

Omni-directional communication is the most energy-consuming operation in a sensor network due to the high degree of signal attenuation and the multipath phenomena that occur when wireless sensors are placed on the ground. Directional communication remains a big challenge since it requires sensors to know to a high degree of accuracy both their own position and orientation, as well as that of their neighbors. The fundamentally high cost of communication results in an important trade-off between the amount of information that the network delivers and its lifetime. A good compromise to conserve battery capacity is to perform appropriate aggregation on collected data to reduce the amount of network communication without much reduction in the information delivered.

Timeliness of delivery is a fundamental performance concern in sensor networks because such networks must react to external phenomena in real-time. An unbounded delay in the loop is unacceptable. From an application's perspective, discovering an intruding target too late is not useful for producing an effective response. Timeliness is generally at odds with energy consumption. If it is possible to delay delivery until more data can be aggregated, overhead can be saved and delivery energy can be reduced. While limited aggregation (or batching) may actually improve the overall timeliness by reducing total traffic, additional aggregation will impair timing performance due the introduced aggregation delay. The break-even point depends on the amount of data currently generated, which is a dynamic quantity that depends on activities in the environment. The timeliness-energy trade-off therefore opens a realm of opportunity for feedback control research in the sensor network domain.

An important consideration in the design of a feedback performance control framework for sensor networks is to quantify the fundamental constraints within which each sensor node operates to solve the global performance optimization problem. In the three-dimensional trade-off space introduced earlier, the basic constraints are on energy, time, and information content. To address the timing constraint, in this paper, we describe important recent results in real-time computing theory that quantify the ability of the network to communicate data in real-time. We relate global timing requirements to the local amount of traffic that can be processed by each node.

As a specific instance of performance control in sensor networks, this paper describes a data communication and aggregation framework that manipulates the degree of data aggregation to maintain specified acceptable latency bounds on data delivery while attempting to minimize energy consumption. An analytic model is constructed to describe the relationships between timeliness, energy, and the degree of aggregation, as well as to quantify constraints that stem from real-time requirements. Feedback control is used to adapt the degree of data aggregation dynamically in response to network load conditions while meeting application deadlines.

The rest of this paper is organized as follows. Section 2 describes the problem statement and the general architecture of our service, which is based on two types of data aggregation; *lossy* and *lossless*. Section 3 derives an expression for real-time system capacity that quantifies the amount of information that can be delivered through the network by the deadlines. This bound is a fundamental design constraint that must be enforced by the feedback control architecture. Section 4 investigates feedback control of lossy aggregation. It describes the conditions under which system capacity is maximized, and describes a feedback scheme that optimizes capacity subject to time constraints by adjusting the degree of aggregation. Section 5 describes local optimization using feedback control of lossless aggregation. Section 6 presents a brief performance evaluation. The paper concludes with final observations and open questions in Section 7.

2. Problem Statement and Architecture

We consider a real-time sensor network where sensory measurements should be delivered to their destinations within specified time constraints. Data is divided into multiple classes. Each class is associated with a bound on delivery time. For example, motion sensor measurements might have to be delivered within 3 seconds to allow real-time tracking of moving targets. In contrast, temperature measurements could be delivered within 30 seconds, since they exhibit slower dynamics. It is desired to deliver all data at the minimum energy cost while satisfying all time constraints. Since the environment is dynamic, the amount of data generated at any time is unpredictable and can vary considerably from time to time. We assume that some sensors report their measurements periodically at all times, while others become active only when triggered by environmental events. For example, flurries of activity in the monitored environment may generate a burst of motion sensor readings. These sensors will be silent when the environment is quiescent. Since the network load is dynamic, overload may occur which can significantly increase communication delay, possibly making it infeasible to deliver data in time. A feedback mechanism is needed to control network delay such that time constraints are met.

The main actuator "knob" that can be manipulated in our system is the degree of data aggregation. In contention-based Medium Access Control (MAC) communication protocols, packing data into larger units reduces the chances of packet collisions, hence reducing energy expenditure and improving delay.[1] Two different types of aggregation are possible; namely, *lossless* aggregation and *lossy* aggregation. Lossless aggregation refers to concatenating individual data items into larger packets, thus amortizing per-packet protocol overhead. In this case, no data is lost. The approach is especially effective in sensor networks where individual sensor readings are small in size, leaving much room for concatenation. Another type of overhead that can be amortized is the local handshake performed ahead of per-hop data transmission to reserve the channel. This handshake is common to contention-based wireless MAC protocols such as 802.11.

Lossless aggregation is effective if the load on the system is not excessive. If the total communication load approaches system capacity, the amount of communicated data must be forcibly reduced. We call the latter case, lossy aggregation. This technique is also useful for energy saving, even when the system is not heavily loaded. The best example of lossy aggregation is the averaging of sensor values. Averaging is a natural choice in many applications. For example, a user may need to know only of the average temperature in a region, as opposed the individual readings of all sensors. Similarly, it may be enough to report only the average estimated location of a target, as opposed to the exact locations of all triggered motion sensors. Lossy aggregation can be either spatial or temporal. In the former, data is averaged from multiple sensors, while in the latter, data from the same sensor is averaged

[1]Due to the difficulty in synchronizing clocks across all nodes in a sensor network, slot-based communication protocols are less practical than contention-based ones.

over time. Both spatial and temporal aggregation incur additional delay waiting for all needed data items to arrive before aggregation is performed.

The service described in this paper adaptively determines the type and amount of aggregation required such that time constraints are met. To maximize information output, lossless aggregation is performed as long as the workload is less than system capacity. Lossy aggregation is invoked only when capacity is exceeded. Two separate control loops are used to determine the amount of aggregation to be applied of each type. Note that, in applications where some degree of lossy aggregation is appropriate even at low load, a lower limit can be imposed on the lossy aggregation controller output. This limit ensures that the desired degree of aggregation is always carried out, even when the system is not overloaded.

A key to the correct operation of the system is to quantify system capacity, such that the correct type of aggregation is used in accordance with load conditions. This quantification is described below, followed by a description of both the lossless and lossy aggregation feedback loops.

3. Real-Time Capacity

The first function of the control system is to decide on the type of aggregation performed (lossy or lossless), depending on whether the network is overloaded or not. In this section, we define a notion of network capacity that is relevant to real-time applications, and relate satisfaction of end-to-end time guarantees to the local state of individual nodes.

3.1 Capacity Definition

Traditional notions of network capacity [2] quantify the amount of information that can be transmitted through the network concurrently at any point in time. In wireless networks, this amount is usually expressed as a product of bytes and meters (called byte-meters) since more data can be transmitted less distance or vice versa. These definitions have no notion of delivery latency and are therefore less suitable for applications where data that arrive after their deadline expiration have little or no value.

In this paper, we define a new notion of capacity we call, *real-time capacity*, denoted C_{RT}. Real-time capacity refers to the total byte-meters that can be delivered *by their deadlines*. To make the capacity expressions independent of the details of the workload (such as the deadline values themselves), we are interested in a normalized capacity expression that quantifies the total byte-meters that can be delivered for per unit of requested latency. It is expected that a network can have a larger byte-meter capacity if deadlines are larger, which makes the aforementioned (normalized) notion of real-time capacity more meaningful.

To illustrate the notion of real-time capacity, consider a network with two data flows, A, and B. Flow A must transfer 1000 bytes a distance of 50 meters (i.e., a total of $50,000$ byte-meters) within 200 seconds. It is said to have a real-time capacity requirement of $50,000/200 = 250$ byte-meters/second. Flow B must transfer 300 bytes a distance of 700 meters within 100 seconds. Its capacity requirement is thus $300 * 700/100 = 2100$ byte-meters/second. Hence, the total real-time capacity needed is $2100 + 250 = 2350$ byte-meters/second. Below, we establish an approximate capacity bound that quantifies the ability of the network to transfer data in time. In particular, all flows meet their deadlines as long as their collective capacity requirements do not exceed the derived capacity bound. This bound will be used to determine whether or not a system is overloaded for the purposes of applying the corresponding data aggregation technique.

3.2 Capacity Derivation and Sampling Rate

Consider a sensor network of n nodes with multiple data sources and a single data sink. The sink could be a monitoring workstation, or a relay that sends the collected data to a user. Packets traverse the network concurrently, each following a multihop path from some source to the sink. Each packet T_i has an arrival time A_i defined as the time at which the sending application injects the packet into the outgoing communication queue of its source node. The packet must be delivered to its destination no later than time $A_i + D_i$, where D_i is called the relative deadline of T_i. Different packets may generally have different deadlines. We call packets that have arrived but whose delivery deadlines have not expired *in-transit* packets. Each packet T_i has an average transmission time C_i that is proportional to its length. Any single path through the network can be thought of as a data pipeline of N stages, where N is the number of hops along the path. In a prior publication [1], we have shown that data traversing a pipeline will meet its end-to-end deadline as long as the following condition holds:

$$\sum_{j=1}^{N} \frac{U_j(1 - U_j/2)}{1 - U_j} < \alpha \qquad (1)$$

where $U_j = \sum_i C_i/D_i$ over all packets T_i in transit through node j. This quantity is called the *synthetic utilization* of node j. The parameter α depends on the scheduling policy used to order outgoing packet transmissions on the link, as discussed in [1]. The bound was derived for nodes with dedicated links. Since, in the case of contention-based protocols, the link is shared with neighboring nodes, the average packet transmission time, C_i must account for this sharing. In particular, with m neighbors, on average, only $1/m$ of link bandwidth can be used by any one node when all nodes are sending. Hence, the average packet transmission time is correspondingly increased m times to account for channel sharing. This is reflected in the values of U_j used in Equation (1). Next, we derive the real-time capacity bound in the presence of lossless aggregation. We use that bound to determine the sensor sampling rate that can be supported during normal operation.

Consider the case where aggregation is lossless. If all traffic congregates on one sink, in the absence of lossy aggregation, the total schedulable traffic generated by all sources is exactly the traffic that can be consumed by that sink. Moreover, at steady state, the sum of synthetic utilizations on all hops some fixed distance j from the sink is no larger than the total synthetic utilization at the sink. This is because the total flow of packets crossing a given perimeter cannot exceed what the destination sees, as shown in Figure 1. Observe that, assuming uniform node density, the number of nodes on a perimeter of radius j (hops) away from the destination increases approximately linearly with j. Hence, the average per-node synthetic utilization decreases linearly with distance from the destination. Assuming the synthetic utilization at the destination is U, and renumbering the hops in ascending order from destination to sources, U_j is proportional to U/j. Thus, from Equation (1):

$$\sum_{j=1}^{N} \frac{U/j(1 - U/2j)}{1 - U/j} < \alpha \tag{2}$$

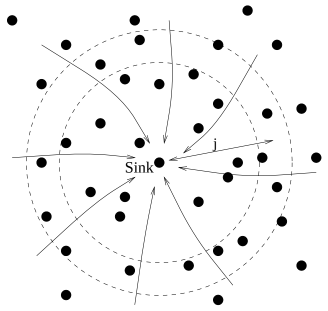


Figure 1. The single sink case

The above equation can be solved for U as a function of the average number of hops N. The equation can be rewritten as:

$$\frac{U}{2}\sum_{j=1}^{N} \frac{1}{j - U} + \frac{U}{2}\sum_{j=1}^{N} \frac{1}{j} < \alpha \tag{3}$$

Since, $U < 1$ (which can be derived from Equation (1)), for large j, $\frac{1}{j-U}$ is approximately equal to $\frac{1}{j}$, and we know that $\sum_{j=1}^{N} 1/j$ is approximately $log\ N$. Thus:

$$U < \alpha/logN \tag{4}$$

Remember that, by definition, $U = \sum_i C_i/D_i$ over all in-transit packets through a node. Since multiplying the packet transmission time, C_i, by the channel transmission speed, W_n, yields packet size, multiplying both sides of the above equation by W_n establishes the average number of bytes that can be transmitted by an average node for each unit of time of the relative deadline. Summing that quantity over the whole network is what defines its real-time capacity (in byte-hops per second). Thus:

$$C_{RT} = W_n \sum_j Uj \tag{5}$$

Since the aggregate synthetic utilization over all nodes distance j from the destination is upper bounded by that at the destination (as explained above) we can sum up the total network capacity by cutting the network into N concentric circles, where N is of the order of the average path length. The traffic through each circle is not less than that at the destination. The total real-time capacity is therefore bounded by W_nUN. Observe that network diameter is generally proportional to network area. Hence, assuming a uniform density network of n nodes, if N is a constant fraction of the diameter, then N is $O(\sqrt{n})$. Assuming that the MAC layer uses an actual transmission rate of W, and that node density is such that on average h nodes typically lie within the range of any one receiver, we can approximately state that $W_n = W/h$. Hence:

$$C_{RT} = \frac{\alpha NW}{hlogN} \tag{6}$$

or:

$$C_{RT} = O(\frac{\sqrt{n}}{log\sqrt{n}})\frac{W}{h} \tag{7}$$

The above expression can be used to set the sampling rate R of periodic sensors. If the size of a single sample is s bytes, its latency constraint is D seconds, and the average sensor distance from the sink is N hops, the real-time capacity requirement of the sensor is sRN/D. Summing up the requirements of all sensors, one must satisfy that the total is less than C_{RT} (given by Equation 6) for the end-to-end time constraints to be satisfied. This imposes a constraint on the maximum sampling rate, R.

Having chosen a sampling rate for periodic sensors, we proceed to the next step of the design problem. In this step, we focus on sensors that are triggered aperiodically by events in the environment. The traffic from such sensors is added to that of the periodic sensors, which may cause system overload when the environment becomes highly dynamic. Lossy aggregation must therefore be performed to maintain timeliness while maximizing information throughput.

4. Control of Lossy Aggregation

When the amount of data generated by the combination of periodic and aperiodic sensors exceeds system capacity, the lossy aggregation feedback loop is activated. The controller of this loop attempts to balance timeliness and information delivered. Its set point can be tuned for better timeliness at the expense of increased aggregation (i.e., more information loss) or lower information loss at the expense of looser timing performance. In particular, the system designer specifies the maximum data path length N for which no deadline misses may occur. The feedback loop must control the degree of aggregation such that information throughput is maximized subject to the above requirement. In the following we derive the local conditions that lead to maximization of global information throughput. We then describe how these conditions are used to design the lossy aggregation feedback loop and compute its per-node set points.

When lossy aggregation is used, the sum of synthetic utilizations of all data sources may exceed that of the sink, since more raw data may be generated than is delivered to the sink. It is desired to devise an aggregation scheme that maximizes total real-time capacity, which is proportional to $W_n \sum_j U_j$ across all nodes in the system, as stated in Equation (5). From the symmetry of the aforementioned summation, as well as the symmetry of the schedulability condition given by Equation (1), the solution that maximizes capacity must be symmetric with respect to synthetic utilization. In other words, U_j must be equal at all nodes. This is called a *load-balanced* network. Since we require that time constraints be met only for paths of length N or less, it is enough to focus on that path length. In a load-balanced network, from Equation (1), the synthetic utilization U of each single node on a communication path of length N must satisfy:

$$\frac{U(1-U/2)}{1-U} < \alpha/N \tag{8}$$

Solving for U, we get:

$$U < 1+\frac{\alpha}{N}-\sqrt{1+(\frac{\alpha}{N})^2} \tag{9}$$

From Equation (5), the capacity of the network is nUW_n byte-hops per unit of relative deadline. Hence, in the optimal case of a load-balanced network, the real-time capacity of the sensor network, denoted C_{RT}, is bounded by:

$$C_{RT} < n(1+\frac{\alpha}{N}-\sqrt{1+(\frac{\alpha}{N})^2}\,)W_n \tag{10}$$

Some interesting observations are apparent. First, rewriting $1 + \alpha/N$ as $\sqrt{1+2\alpha/N+(\alpha/N)^2}$, observe that when N is large, the term $(\alpha/N)^2$ can be neglected leading to:

$$C_{RT} < n(\sqrt{1+\frac{2\alpha}{N}}-1)W_n \tag{11}$$

We know from series expansion that for a small x, the term $\sqrt{1+x}$ is approximately equal to $1 + x/2$. Hence, substituting for the square root in Equation (11) when N is large, and recalling that $W_n = W/h$, we get:

$$C_{RT} < \frac{n\alpha}{Nh}W \tag{12}$$

If path length N is of the order of the square root of the area of the network, which in turn is of the order of the number of nodes, then:

$$C_{RT} = O(\sqrt{n})\frac{W}{h} \tag{13}$$

To maximize real-time information throughput such that the above capacity bound is approached, the local controller at each node attempts to keep its synthetic utilization at the value indicated in the right hand side of inequality 9. Hence, the controller set point, $U_{desired}$, is:

$$U_{desired} = 1+\frac{\alpha}{N}-\sqrt{1+(\frac{\alpha}{N})^2} \tag{14}$$

Choosing a larger N will reduce the utilization, thereby increasing the amount of lossy aggregation. A smaller N will reduce information loss, but increase deadline misses along longer paths. The instantaneous synthetic utilization of a node is $U_{inst} = \sum_i C_i/D_i$, carried out over all outgoing packets. As explained in [1], this value is increased by C_i/D_i when a new packet, T_i, arrives. It is decreased by C_i/D_i only at the delivery deadline of the packet (and is set to zero when the link is idle). Each node maintains an exponential moving average $U_{avg}(k)$ of instantaneous synthetic utilization. This moving average is updated periodically at the controller's sampling interval. The control error $e(k)$ in the k^{th} sampling interval is defined as $e(k) = U_{desired} - U_{avg}$. This error drives an integral regulator of gain K_I whose output m determines the degree of lossy aggregation required, where:

$$\delta m = K_I(U_{desired} - U_{avg}) \tag{15}$$

More specifically, m specifies the ratio of the number of packets after and before aggregation. For example $m = 0.66$ indicates that each 3 incoming packets must be aggregated into 2 (by averaging a pair), where $2/3 = 0.66$. A field in each packet's header keeps track of the number of original raw data items the packet's aggregated value reflects. This allows correct weights to be used when averaging the content of two packets.

Since aggregation can only *reduce* the number of packets, the maximum value of controller output m is 1, indicating that no aggregation is needed. Observe that when the system is underloaded ($U_{avg} < U_{desired}$), the controller eventually saturates at $m = 1$. Anti-windup is then invoked, thus opening the lossy aggregation control loop. Hence, only lossless aggregation is performed in an underloaded system.

Note that the instantaneous synthetic utilization of the system is proportional to m. Hence, the controlled process has a constant gain. If all data deadlines are the same, that gain is unity. The only dynamics in the loop are those that arise from the low-pass filter (i.e., exponential moving averaging) and the controller. The filter is essential to smooth bursts.

5. Control of Lossless Aggregation

When the system operates in the non-overloaded regime, only lossless aggregation is performed to optimize energy consumption and reduce delay. An architecture for application-independent data aggregation is described in [3]. In that regime, a feedback loop measures the average delay incurred to transmit a packet (which includes the contention delay on the wireless medium). This measurement is then used to adapt the degree of lossless data aggregation, called N_{aggr}. When a particular degree of aggregation is indicated, packets are not forwarded to the network device until the corresponding number of them (i.e., at least N_{aggr}) are present in the queue.

The default degree of aggregation N_{aggr} is 1, which occurs at low load. In this case, packets are delivered to the network device for transmission as soon as the device is ready. Note that if more than one packet have accumulated in the queue while the network device was busy, they will be aggregated and sent together. As network traffic builds up and contention delays increase, the feedback loop adjusts the aggregation level, N_{aggr}, to allow a greater minimum degree of aggregation. When the network device is free, packets are sent only as long as at least N_{aggr} of them are present.

Next, we derive a model for data aggregation that will be used to tune our feedback loop. The control loop operates periodically at some appropriately chosen interval, T, measuring the current MAC-layer delay $D(k)$ and adjusting the degree of aggregation, $N_{aggr}(k)$, accordingly. Let the k^{th} sampling interval of the control loop be $[(k-1)T, kT)$. The delay sensor produces its reading, $D(k)$, at the end of each interval. This reading represents the average MAC-layer delay of all packets transmitted in that last sampling interval. The average delay a packet experiences before its transmission is complete is:

$$D(k) = D_{min} + D_{collide} \qquad (16)$$

where D_{min} is the minimum delay experienced when no collisions occur (which is primarily the packet tranmission delay plus some system overhead), and $D_{collide}$ is the average additional delay incurred due to collisions.

Assume that a total of $M(k)$ packets were present in interval k in the combined queues of all nodes sharing the same neighborhood, where only one node can transmit at a time. Given a degree of aggregation, $N_{aggr}(k-1)$, set at the beginning of that interval, at most $M(k)/N_{aggr}(k-1)$ data units will be transmitted on the medium. This is only an approximation, because different nodes may have different $N_{aggr}(k-1)$ values. However, since those nodes share the same medium with the same level of congestion, it is likely that their $N_{aggr}(k-1)$ will be close. Since the probability of collisions grows linearly with the number of data units available for transmission, the expected number of collisions grows with $M(k)/N_{aggr}(k-1)$. Furthermore, since most contention-based MAC-layers exhibit exponential back-off upon a collision, the average contention delay, $D_{collide}$, grows exponentially with the number of collisions. Hence:

$$D(k) = D_{min} + Ae^{bM(k)/N_{aggr}(k-1)} \qquad (17)$$

where A and b are constants. This is clearly a non-linear system. We linearize the system by computing its derivative with respect to the manipulated variable (in this case, $N_{aggr}(k-1)$), which yields the small deviation model:

$$\frac{dD(k)}{dN_{aggr}(k-1)} = -A\frac{M(k)}{N_{aggr}(k-1)^2}e^{bM(k)/N_{aggr}(k-1)} \qquad (18)$$

Hence, if the degree of aggregation is changed by $\delta N_{aggr}(k) = N_{aggr}(k) - N_{aggr}(k-1)$, and assuming a constant workload $M(k+1) = M(k) = M$), it is predicted that:

$$D(k+1) = D(k) - A\frac{M}{N_{aggr}(k-1)^2}e^{bM/N_{aggr}(k-1)}\delta N_{aggr}(k) \qquad (19)$$

The system model contains a nonlinear integral term. A proportional controller can therefore be used to stabilize the system and eliminate steady state error. The gain of the proportional controller can be made dynamic to compensate for part of the system nonlinearity. The controller we use is thus given by:

$$\delta N_{aggr}(k) = PN_{aggr}(k-1)^2 e(k) \qquad (20)$$

where $e(k) = D(k) - D_{desired}$, and P is controller gain.

6. Experimental Evaluation

We simulate our architecture in GloMoSim [5], a scalable discrete-event simulator developed at UCLA. This software provides a high fidelity simulation for wireless communication with detailed propagation, radio, MAC, and network layer components. In our experiments, the communication parameters are chosen in accordance with Berkeley Telos mote specifications, the latest hardware platform on which sensor network research systems are currently deployed for testing.

We evaluate two types of data aggregation techniques discussed in previous sections, namely lossless and lossy aggregation, and compare them with a non-aggregation scheme. During the simulation, we adopt a typical many-to-one traffic pattern, where 10 source nodes send out CBR (Constant Bit Rate) flows to a single sink with average hop length 4 - 6 hops. The end-to-end deadline used in the experiment is 200 ms. To investigate the effectiveness of data aggregation in the presence of congestion, we incrementally increase the sending rate of 10 flows from 1.5 to 3.7 packets/second per flow. Experiments are repeated 30 runs with different seeds such that the 95% confidence intervals are within 2 - 5% of the mean.

Figure 2 demonstrates that both lossless and lossy aggregation can dramatically reduce average packet end-to-end delay in comparison with the non-aggregation scheme when the traffic becomes heavy, thanks to the fact that aggregation techniques can control the amount of information delivered in response to the timeliness requirements.

When the amount of information generated exceeds the real-time capacity, the lossy aggregation demonstrates its excellent capability of achieving low end-to-end deadline miss ratio by aggregating a small percentage of packets together. As shown in Figure 4, miss ratios for the lossy aggregation scheme under different traffic loads are always below 10%, while the lossless aggregation scheme, which doesn't take real-time capacity into account, suffers a high miss ratio penalty when traffic exceeds real-time capacity of the network.

We note that the lossy aggregation does not achieve this excellent performance for free. As shown in Figure 5, it has a non-zero

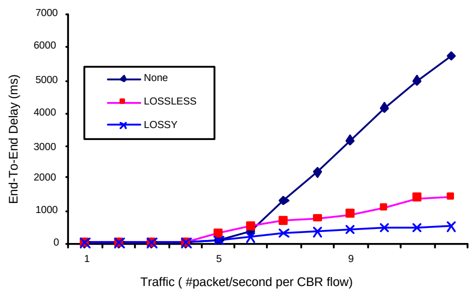


Figure 2. End-to-End Delay Vs. Traffic Load

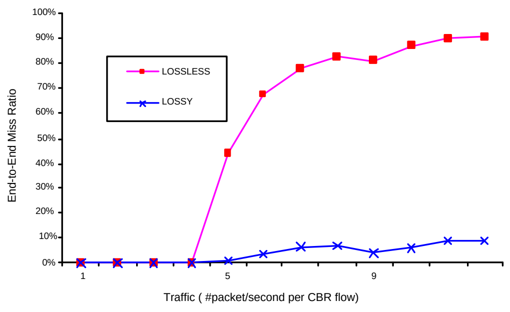


Figure 4. End2End Miss Ratio Vs. Traffic Load

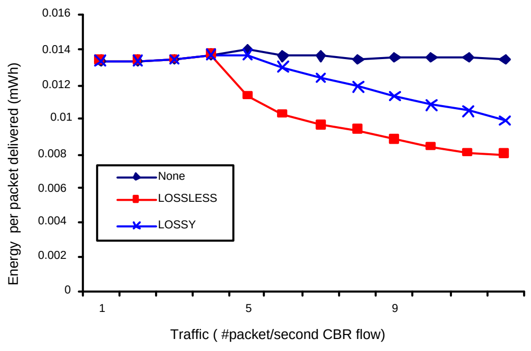


Figure 3. Energy Vs. Traffic Load

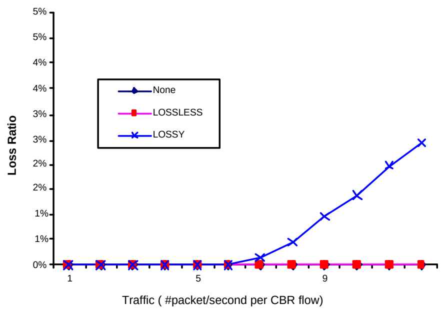


Figure 5. Loss Ratio Vs. Traffic Load

lossy ratio[2] in heavy traffic in exchange for excellent timeliness shown in Figure 4.

In addition, as shown Figure 3, both lossless and lossy aggregations can achieve energy conservation by reducing the number of control messages and the number of retransmissions in the presence of congestion.

7. Conclusions and Future Work

In this paper, we demonstrated the application of control theory to resolve fundamental performance trade-offs in sensor networks. Fundamental limits were presented on real-time network capacity. These limits where then used to derive sensor sampling rates and set points of control loops. Two different mechanisms for data aggregation were presented whose combined effect is to maximize information throughput while maintaining timing constraints and reducing protocol overhead. There are several outstanding issues that the authors hope to address in future interdisciplinary collaborations. For example, how to model non-linearities peculiar to sensor networks? How can these nonlinearities be accounted for in control? How efficient are adaptive control and robust control techniques in dealing with parameter variation and load uncertainty? What other actuators can be applied in addition to aggregation? What is the effect of routing policies? Examples, theoretical foundations, experimental evidence, and practical experience are needed in applying feedback performance control to sensor networks. This is an important focus of our research group at the present time.

[2]Lossy ratio is the percentage of packets that are aggregated with information loss

8. REFERENCES

[1] T. Abdelzaher, G. Thaker, and P. Lardieri. A feasible region for meeting aperiodic end-to-end deadlines in resource pipelines. In *ICDCS*, Tokyo, Japan, March 2004.

[2] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[3] T. He, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher. Aida: Adaptive application independent data aggregation in wireless sensor networks. *ACM Transactions on Embedded Computing Systems*, (to appear) 2004.

[4] J. A. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou. Real-time coomunication and coordination in embedded sensor networks. *Proceedings of the IEEE*, 91(7), July 2003.

[5] X. Zeng, R. Bagrodia, and M. Gerla. GloMoSim: A library for parallel simulation of large-scale wireless networks. In *Workshop on Parallel and Distributed Simulation*, pages 154–161, 1998.

A Communication Architecture and Programming Abstractions for Real-Time Embedded Sensor Networks [*]

T. Abdelzaher, J. Stankovic, S. Son, B. Blum, T. He, A. Wood
Department of Computer Science
University of Virginia
Charlottesville, VA 22904

Chenyang Lu
Department of Computer Science
University of Washington, St. Louis
St. Louis, MO 63130

Abstract

Data distribution in embedded real-time sensor networks requires new protocols and programming environments that achieve time-sensitive message delivery and provide useful abstractions to the application programmer. Attainment of these goals requires changes to multiple layers of the communication protocol stack. In this paper, we review a protocol suite developed by the authors for data communication in embedded sensor networks. It takes into account time constraints and exports attribute-based connections that are tightly integrated with properties of the monitored environment. A programming language is described that allows external physical objects to be represented as first class abstractions in the computing system. The language facilitates writing monitoring applications. The system was implemented on a prototypical sensor network based on MICA motes.

Keywords: sensor networks, programming paradigms, tracking, QoS, distributed systems.

1 Introduction

Ad hoc wireless sensor networks, made possible by advances in communication technology and hardware miniaturization [11], raise the need for a new suite of communication protocols and new programming abstractions for distributed deeply embedded computing. Such sensor networks are especially useful when an inhospitable, poorly accessible, or delicate environment prevents the installation of needed computing infrastructure. An example could be the site of a natural disaster or a target behind enemy lines. Instead, myriads of tiny computationally equipped wireless sensor devices may be dropped to form an ad hoc network that operates autonomously to monitor its surroundings, react to distributed events, or alert appropriate authorities when specific activities are observed.

Sensor networks offer new challenges both from the perspective of building communication protocols and from the perspective of developing appropriate programming models. These challenges arise due to their large scale, autonomous operation, massively parallel interactions with a spatially distributed physical environment, and a more stringent set of resource constraints.

Communication protocols for sensor networks must provide real-time assurances. While ensuring proper timing behavior of systems has been a topic of real-time research for decades, sensor network applications offer physical *space* in addition to time, as a new dimension for interaction with the environment. Hence, while traditional real-time computing research has been concerned with meeting time constraints, a new branch of theory is needed to analyze systems that interact with the their surroundings both in real time and in the real dimensions of physical space. For example, in a network that tracks vehicles through the sensor field, the application must collect sensory measurements in real-time from the actual changing locale in which the vehicle is detected. Message communication must therefore be sensitive to both time and distance constraints, which may depend on external factors such as the physical speed of the monitored vehicle. In this paper, we describe a protocol suite in which both time and distance constraints are addressed.

A new programming paradigm is needed to facilitate the task of sensor network application development. Due to the large scale of sensor networks, programmers should not have to concern themselves with low-level abstractions and functions such as creating and destroying individual connections between pairs of nodes. Instead, the programming environment must offer a conceptual view in which global tasks can be defined in an abstract manner, leaving it for the underlying system to translate them into computational and communication activities on individual sensor nodes. This paper reports on the design of a programming system developed on top of our communication protocol suite, which provides the required high-level abstractions. The language allows external events in the environment to be represented as objects in the computing

[*]The work reported in this paper was supported in part by the National Science Foundation under grants CCR-02-05327 and CCR-00-92945, by DARPA under grant F33615-01-C-1905, and by MURI under grant N00014-01-1-0576.

system facilitating the monitoring of such events by the application. The reported architecture is a part of an ongoing research effort on developing a sensor network virtual machine for future distributed deeply embedded applications.

The rest of this paper is organized as follows. In Section 2, we describe a protocol suite that takes into account time and space constraints, and exports a useful transport-layer abstraction in which logical communication end-points can be associated with tracked objects in the external environment. Section 3 describes a new programming model for sensor networks which builds upon the aforementioned transport protocol to elevate environmental objects into first class programming abstractions. Related work is summarized in Section 4. The paper concludes with Section 5 which describes some of the remaining challenges and directions for future research.

2 A Protocol Suite for Sensor Networks

Communication protocols in sensor networks are the fundamental cornerstone that glues distributed applications together. The deeply embedded nature of sensor networks presents some of the most interesting challenges in the design of their communication protocols. New research topics span all protocol stack layers, primarily motivated by a tighter interaction between the network and its physical environment. At the MAC layer, new protocols are needed that enforce message priorities consistently with time and distance constraints that arise from environmental interactions [22]. Awareness of the physical environment must also be incorporated into the network layer. For example, location should be an essential attribute of addressable networked objects [15]. Location-assisted routing protocols such as LAR [19] and DREAM [4], as well as location services [21] have been described for ad hoc wireless networks. More generally, routing algorithms are needed in which destinations are described implicitly by their environmental attributes. For example, directed diffusion [18, 14] and the intentional naming system [3] provide addressing and routing based on data interests. A fundamental rethinking of basic protocols is required at the transport layer as well. Individual socket-style connections between nodes are too low-level to be a useful abstraction for the programmer. They must be replaced with higher-level alternatives that are more suitable for the main purpose of sensor networks, namely monitoring the external surroundings in which they are embedded.

This section describes our answer to the challenge of incorporating environmental awareness into the design of sensor network communication protocols. Our protocol stack features two important contributions. First, it implements new real-time message scheduling algorithms in which both time and physical distance requirements are observed. Second, it exports a transport-layer address space that associates unique network addresses with external environmental objects. The new addresses serve as connection end-points, thereby raising the level of connection abstraction to entities of direct interest to the application. The layers of our protocol stack are described in the following subsections.

2.1 Real-Time Distance-Aware Scheduling

Message communication in sensor networks must occur in bounded time, for example to prevent delivery of stale data on the status of detected events or intruders. In general, a sensor network may simultaneously carry multiple messages of different urgency, communicated among destinations that are different distances apart. The network has the responsibility of ordering these messages on the communication medium in a way that respects both time and distance constraints.

A protocol that achieves this goal in our architecture is called RAP [22]. It supports a notion of packet velocity and implements velocity monotonic scheduling (VMS) as the default packet scheduling policy on the wireless medium. Observe that for a desired end-to-end latency bound to be met, an in-transit packet must approach its destination at an average velocity given by the ratio of the total distance to be traversed to the requested end-to-end latency bound. RAP prioritizes messages by their required velocity such that higher velocities imply higher priorities. Two flavors of this algorithm are implemented. The first, called *static velocity-monotonic scheduling*, computes packet priority at the source and keeps it fixed thereafter regardless of the actual rate of progress of the packet towards the destination. The second, called *dynamic velocity-monotonic scheduling*, adjusts packet priority *en route* based on the remaining time and the remaining distance to destination. Hence, a packet's priority will increase if it suffers higher delays on its path and decrease if it is ahead of schedule.

To achieve consistent prioritization in the wireless network, not only do we need priority queues at nodes, but also a MAC layer that resolves contention on the wireless medium in a manner consistent with message priorities. We adopt a scheme similar to [1] to prioritize access to the wireless medium. The scheme is based on modifying two 802.11 parameters, namely the DIFS counter and the backoff window, such they are priority-aware. The DIFS counter determines the maximum time a node waits, after the communication channel becomes idle, prior to transmitting an RTS packet. The actual waiting time is randomly chosen between 0 and DIFS. An approximate prioritization effect is achieved by letting the DIFS value depend on the priority of the outgoing packet at the head of the transmission queue. A larger value is given to packets of lower priority. Hence, more urgent packets tend to contend on the medium more aggressively. The back-off window of 802.11 increases the maximum waiting time when collisions occur. To give preferential treatment to higher priority packets, we make this increase dependent on the priority of the head of the queue. A higher increase is incurred for packets of lower priority. Hence, collisions tend to be resolved in favor of higher-priority packets.

A detailed performance evaluation of this scheme can be found in [22]. It is shown that velocity-monotonic scheduling substantially increases the fraction of packets that meet their deadlines taking into consideration distance constraints. More accurate schemes for medium access prioritization remain an open research topic. An interesting related topic is that of schedulability analysis of velocity-monotonic scheduling. Ideally, such an analysis should allow a source node to determine whether a particular desired velocity is attainable between a source-destination pair given current network conditions. While an analytic expression for velocity feasibility is still an open problem, in the following, we describe a feedback-based technique that enforces velocity constraints dynamically by applying back-pressure to slow down the sources when such constraints are violated.

2.2 Enforcement of Velocity Constraints

Consider a network that supports multiple predefined velocities. An application can choose a velocity level for each message. The network guarantees that the chosen message velocity is observed with a very high probability as long as the message is accepted from the application. A network-layer protocol with the above property, called SPEED [13], has recently been developed by the authors. The protocol defines the velocity of an in-transit message as the rate of decrease of its straight-line distance to its final destination. Hence, for example, if the message is forwarded away from the destination, its velocity at that hop is negative.

The main idea of SPEED is as follows. Each node i in the sensor network maintains a neighborhood table that enumerates the set of its one-hop neighbors. For each neighbor, j, and each priority level, P, the node keeps a history of the average recently recorded local packet delay, $D_{ij}(P)$. Delay $D_{ij}(P)$ is defined as the average time that a packet of priority P spends on the local hop i before it is successfully forwarded to the next-hop neighbor j. Given a packet with some velocity constraint, V, node i determines the subset of all its neighbors that are closer to the packet's destination. If L_{ij} is the distance by which neighbor j is closer to the destination than i, the velocity constraint of the packet is satisfied at node i if there exists some priority level P and neighbor j such that $L_{ij}/D_{ij}(P) \geq V$. The packet is forwarded to one such neighbor non-deterministically. If the condition is satisfied at multiple priority levels, the lowest priority level is chosen. If no neighbor satisfies the velocity constraint, we say that a local deadline miss occurs.

A table at node i keeps track of the number of local deadline misses observed for each velocity level V. This table is exchanged between neighboring nodes. Nodes use this information in their forwarding decisions to favor more appropriate downstream hops among all options that satisfy the velocity constraint of a given packet. No messages are forwarded in the direction of nodes with a high miss-ratio. The mechanism exerts back-pressure on nodes upstream from congested areas. Congestion increases the local miss-ratio in its vicinity, preventing messages from being forwarded in that direction. Messages that cannot be forwarded are dropped thus increasing the local miss-ratio upstream. The effect percolates towards the source until a node is found with an alternative (non-congested) path towards the destination, or the source is reached and informed to slow down. The mentioned scheme is therefore effective in exerting congestion control and performing packet rerouting that guarantee the satisfaction of all velocity constraints in the network at steady state [13]. The protocol is of great value to real-time applications where different latency bounds must be associated with messages of different priority.

2.3 Entity-Aware Transport

Although RAP and SPEED allow velocity constraints to be met, the abstractions provided by them are too low-level for application programmers. We develop a transport layer whose main responsibility is to elevate the degree of abstraction to a level suitable for the application. In particular, we propose a transport layer in which connection end-points are directly associated with events in the physical environment. Events represent continuous external activities, such as the passage of a vehicle or the progress of a fire, which is precisely what an application might be interested in. By virtue of this layer, the programmer can describe events of interest and logically assign "virtual hosts" to them. Such hosts export communication ports and execute programs at the locations of the corresponding events. The programmer is isolated from the details of how these hosts and ports are implemented. When an external event (e.g., a vehicle) moves, the corresponding virtual host migrates with it transparently to the programmer.

We call the virtual host associated with an external event of interest an *entity*. Sensor nodes that can sense the event are called *entity members*. Members elect an *entity leader* that uniquely represents the entity and manages its state. Hence, an entity appears indivisible to the rest of the network. The fact that it is composed of multiple nodes with a changing membership is abstracted away.

When the external event moves outside the sensing horizon of the current entity leader, the leader hands-off leadership to another member. Connection state is handed off as well allowing communication with the entity to remain uninterrupted. To ensure unique representation of external events within the computational environment, a unique entity must be associated with each event. The transport protocol meets this constraint by announcing the existence of the entity to nearby nodes that cannot yet sense the event. These announcements are sent periodically by the entity leader and are called *heartbeats*. Nodes that hear a heartbeat but cannot sense the event are called *entity followers*. They are said to be within the *awareness horizon* of the named entity. Upon receiving a heartbeat, such nodes set an *entity timeout timer*. Upon timer expiration, their sta-

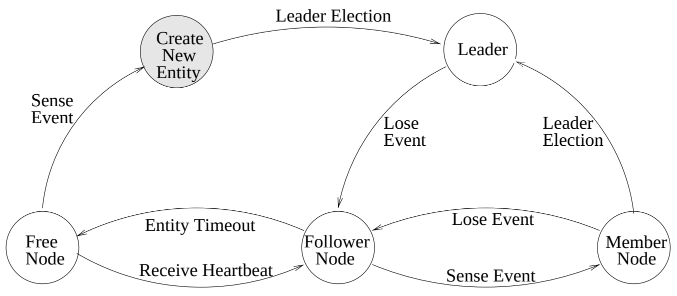


Figure 1. Node state transition diagram

tus as followers expires. The timer is reset to zero every time a new heartbeat is received. When the event enters the sensing horizon of a follower node, the node becomes a member of the entity it is following. If the node is not a follower, it recognizes that a new entity must be created. The node sets a random timer upon expiration of which it claims leadership of the new entity. If it receives a leadership claim message from another node prior to timer expiration, it clears the timer and becomes an entity member. The algorithm ensures that a newly sensed event is represented by a single entity and that current events do not spawn spurious entities as they move from one location to another. Figure 1 depicts the node state transition diagram between follower, member, and leader states, as well as the free state in which a node is not cognizant of any entities.

An evaluation of this architecture reveals that entity uniqueness is maintained as long as the target event moves in the environment at a speed slower than half the nodes' communication radius per second [7]. For example, if sensor nodes can communicate within a 200 meter radius, the transport layer can correctly maintain endpoints attached to targets that move as fast as 100 m/s (i.e., 360 km/hr). The combination of this transport layer and the guaranteed velocity protocols described earlier provides invaluable support to real-time applications. For example, communication regarding moving targets can be made to proceed in the network at a velocity that depends on target velocity itself. Hence, positions of faster targets, for example, can be reported quicker than those of slower ones. To the authors' knowledge no other protocols in sensor networks have explicitly addressed message timing constraints.

3 A Sensor-Network Programming Model

The transport layer described above gives rise to a programming model that elevates tracked activities in the physical environment into first class programming abstractions. In this model, the application developer specifies events to be monitored. The system automatically detects such events and instantiates a so called *context* every time an instance of an event is detected in the environment. From the programmer's perspective, the application is composed of a dynamic set of contexts, each representing a particular event. Objects can be attached to contexts. These objects will logically execute in the locale of the monitored event. Contexts have unique identifiers called *context labels*. Objects attached to a context can be addressed using the context label and object name. They can communicate remotely by remote method invocation. The programmer's view of the application is depicted in Figure 2.

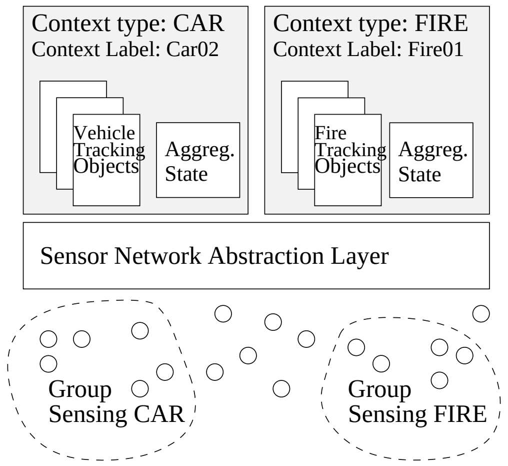


Figure 2. Programming model

A context label around some event, e, is completely defined by two elements, namely (i) the function $sense_e()$ which specifies an environmental condition that spawns the context label, and (ii) the function $state_e()$ which describes the environmental state to be encapsulated in the context label. The former function, for example, might dictate that a label is to be created if magnetic distortion (e.g., the presence of a vehicle) is sensed. The state function returns a set of aggregate variables, each computed using outputs of at least N_e nodes for which $sense_e()$ was true in the last L_e time units. We call N_e and L_e the critical mass and freshness constraints, respectively. For example, to obtain the approximate position of a vehicle we may define $state_e()$ to be the average coordinates of at least 5 nodes that have sensed the vehicle within the last 2 seconds. We define environmental tracking of event e as the process of maintaining the state of this event subject to given freshness and critical mass constraints.

Syntactically, an application consists of a list of context declarations, each specifying an activation condition $sense_e()$, a set of state variables $state_e()$, and a list of attached objects. An example declaration is shown in Figure 3. The example defines a context of type *tracker*, specifies its activation condition, $sense_e()$, as an appropriate magnetometer reading (presumably caused by a nearby vehicle), and defines $state_e()$ as the average $location$ of the tracked target. It specifies that $location$ must represent the average of at least 2 sensor readings measured no earlier than 1 second ago. The attached object is invoked periodically to report the current location of the vehicle to a virtual base station object. It passes the originating

```
begin context tracker
  activation: MAGNETOMETER == ON
  $location$ : avg (position) mass=2, freshness=1s
  begin object reporter
    invocation: PERIOD(0.5s)
    report_function() {
      BaseStation.reportLocation ($self.label$, $location$);
    }
  end
end context
```

Figure 3. Sample code

context label as the identity of the reported vehicle. If there are several vehicles in the field, multiple reporter objects will be automatically instantiated. The programmer does not need to worry about instantiating these objects. Object execution and maintenance of aggregate state occurs automatically. Details of the underlying communication, group membership management, leader handoff, and mobility are handled transparently. Hence, the programmer's interaction with the sensor network is significantly simplified.

We have described real-time communication protocols and programming abstractions motivated by a tighter interaction between sensor networks and their physical environment. Our architecture might be a first step towards a comprehensive vision for next-generation programming systems supporting future real-time deeply embedded distributed sensor network applications.

4 Related Work

Classical distributed programming paradigms and middleware such as CORBA [27], group communication (e.g., ISIS [5]), remote procedure calls (RPC [6]), and distributed shared memory (e.g., MUNIN [9]) share in common the fact that their programming abstractions exist in a logical space that does not represent or interact with objects and activities in the physical world. Their main goal is to abstract distributed communication rather than facilitate distributed sensory interactions with an external physical environment. In contrast, sensor network applications call for a paradigm that revolves around "environmentally-inspired" abstractions aimed at simplifying the coding of interactions with the physical world that arise in distributed deeply embedded systems.

The work reported in this paper is closely related to several recent projects, such as Cricket [23], Sentient Computing [2] and Cooltown [10], which propose high-level paradigms in which an embedded distributed computing system is able to share humans' perceptions of the physical world. These systems allow the location of entities in the external environment to be tracked. One major difference is that they assume cooperative users who, for example, can wear beaconing devices that interact with location services in the infrastructure for the purposes of localization and tracking [23, 2]. Our interest, in contrast, is in situations where no cooperation is assumed from the tracked entity.

In the absence of cooperation, several research efforts proposed alternative addressing schemes that do not rely on having destinations with specific identities, but rather contact sensor nodes in the vicinity of a phenomenon of interest based on the attributes of data they sense. For example, DataSpace [17] exports abstractions of physical volumes addressable by their locations. Similarly, directed diffusion [18, 14] and the intentional naming system [3] provide addressing and routing based on data interests [18, 14]. Attributed-based naming is also related to the notion of content-addressable networks [24] proposed for an Internet environment, which allows queries to be routed depending on the requested content rather than on the identity of the target machine. We adopt context labels; a form of attribute-based naming. In our architecture, however, context labels are *active* elements. Not only do they provide a mechanism for *addressing* nodes that sense specific environmental conditions, but also they can *host context-specific computation* that tracks a target in the environment.

Recent research on system software for sensor networks has seen the introduction of distributed virtual machines designed to provide convenient high-level abstractions to application programmers, while implementing low-level distributed protocols transparently in an efficient manner [26]. This approach is taken in MagnetOS [12], which exports the illusion of a single Java virtual machine on top of a distributed sensor network. The application programmer writes a single Java program. The run-time system is responsible for code partitioning, placement, and automatic migration such that total energy consumption is minimized. Maté [20] is another example of a virtual machine developed for sensor networks. It implements its own bytecode interpreter, built on top of TinyOS [16].

A somewhat different approach of providing high-level programming abstractions is to view the sensor network as a distributed database, in which sensors produce series of data values and signal processing functions generate abstract data types. The database management engine replaces the virtual machine in that it accepts a query language that allows applications to perform arbitrarily complex monitoring functions. This approach is implemented in the COUGAR sensor network database [8]. A middleware implementation of the same general abstraction is also found in SINA [25], a sensor information networking architecture that abstracts the sensor network into a collection of distributed objects.

Our system is different in that it is geared for real-time environmental tracking. To the authors' knowledge, we describe the first programming language for sensor networks that explicitly facilitates the coding of tracking applications, and the first sensor network communication protocols that conider real-time constraints. These novel abstractions and underlying mechanisms are well-suited for monitoring targets that move in the physical world. They can therefore have a major impact on application development for sensor networks.

5 Conclusions

This paper reviewed a new protocol suite and programming system for sensor network applications, that may considerably improve real-time behavior and reduce the development cost of deeply embedded systems. This reduction comes from offloading from the application developer the details of managing low-level abstractions. Future work of the authors will involve refinement of the real-time protocols and the environmental tracking problem such that more precise semantics and failure models are achieved. With such refinements we hope to build a predictable sensor network "virtual machine" that exports timely, reliable behavior and well-defined semantics, implemented on the unreliable, unpredictable, and resource constrained hardware and communication infrastructure typical of sensor networks. Such a virtual machine would hide the complexity of sensor network programming from the application developer, making a new more robust and more dynamic realm of sensor network applications attaintable to impact future defense, surveillance, habitat monitoring, and disaster management systems.

References

[1] I. Aad and C. Castelluccia. Differentiation mechanisms for ieee 802.11. In *IEEE Infocom*, Anchorage, Alaska, April 2001.

[2] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper. Implementing a sentient computing system. *IEEE Computer*, 34(8):50–56, August 2001.

[3] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The design and implementation of an intentional naming system. In *ACM Symposium on Operating Systems Principles*, Kiawah Island, SC, December 1999.

[4] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. A distance routing effect algorithm for mobility (dream). In *ACM MOBICOM*, pages 76–84, October 1998.

[5] K. Birman, A. Schiper, and P. Stephenson. Lightweight causal and atmoic group multicast. *ACM Transactions on Computer Systems*, 9(3):272–314, August 1991.

[6] A. Birrel and B. Nelson. Implementing remote procedure calls. *ACM Transactions on Computer Systems*, 2(1), February 1984.

[7] B. Blum, P. Nagaraddi, A. Wood, T. Abdelzaher, J. Stankovic, and S. Son. An entity maintenance and connection service for sensor networks. In *The First International Conference on Mobile Systems, Applications, and Services (MobiSys)*, San Francisco, CA, May 2003.

[8] P. Bonnet, J. Gehrke, and P. Seshardi. Towards sensor database systems. In *2nd International Conference on Mobile Data Management*, pages 3–14, Hong Kong, January 2001.

[9] J. Carter, J. Bennet, and W. Zwaenepoel. Implementation and performance of munin. In *ACM Symposium on Operating Systems Principles*, pages 151–164, October 1991.

[10] P. Debaty and D. Caswell. Uniform web presence architecture for people, places, and things. *IEEE Personal Communications*, 8(4):46–51, August 2001.

[11] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Mobile networking for smart dust. In *ACM MOBICOM*, Seattle, WA, August 1999.

[12] R. B. et al. On the need for system-level support for ad hoc and sensor networks. *Operating System Review*, 36(2):1–5, April 2002.

[13] T. He, J. Stankovic, C. Lu, and T. Abdelzaher. Speed: A stateless protocol for real-time communication in sensor networks. In *International Conference on Distributed Computing Systems*, Providence, Rhode Island, May 2003.

[14] J. Heideman, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan. Building efficient wireless sensor networks with low-level naming. *Operating Systems Review*, 35(5):146–159, December 2001.

[15] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8), August 2001.

[16] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for network sensors. In *ASPLOS*, Cambridge, MA, November 2000.

[17] T. Imielinski and S. Goel. Dataspace - querying and monitoring deeply networked collections in physical space. *IEEE Personal Communications*, 7(5):4–9, October 2000.

[18] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *ACM MOBICOM*, Boston, Massachusetts, August 2000.

[19] Y.-B. Ko and V. Nitin. Location-aided routing (lar) in mobile ad hoc networks. In *ACM MOBICOM*, pages 66–75, October 1998.

[20] P. Levis and D. Culler. Mate: A tiny virtual machine for sensor networks. In *ASPLOS*, San Jose, CA, October 2002.

[21] J. Li, J. Jannotti, D. D. Couto, D. Karger, and R. Morris. A scalable location service for goegraphic ad hoc routing. In *ACM MOBICOM*, Boston, Massachusetts, August 2000.

[22] C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, and T. He. Rap: A real-time communication architecture for large-scale wireless sensor networks. In *Real-Time Technology and Applications Symposium*, San Jose, CA, September 2002.

[23] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *ACM MOBICOM*, Boston, MA, August 2000.

[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Sigcomm*, San Diego, CA, August 2001.

[25] C.-C. Shen, C. Srisathapornphat, and C. Jaikeo. Sensor information networking architecture and applications. *IEEE Personal Communications*, 8(4):52–59, August 2001.

[26] E. Sirer, R. Grimm, A. Gregory, and B. Bershad. 'design and implementation of a distributed virtual machine for networked computers. In *ACM Symposium on Operating System Principles*, pages 202–216, Kiawah Island, SC, December 1999.

[27] S. Vinoski. Corba: Integrating diverse applications within distributed heterogeneous environments. *IEEE Communications Magazine*, 14(2), February 1997.

An Overview of the VigilNet Architecture

Tian He, Liqian Luo, Ting Yan, Lin Gu, Qing Cao, Gang Zhou, Radu Stoleru
Pascal Vicaire, Qiuhua Cao, John A. Stankovic, Sang H. Son and Tarek F. Abdelzaher
Department of Computer Science
University of Virginia, Charlottesville, VA 22903

Abstract

Battlefield surveillance often involves a high element of risk for military operators. Hence, it is very important for the military to execute unmanned surveillance by using large-scale wireless sensor systems. This invited paper summarizes the architecture of the VigilNet system – a long-term real-time networked sensor system for military surveillance. Specifically, we review the design of several major subsystems within VigilNet including sensing and classification, localization, tracking, networking, power management, reconfiguration, graphic user interface, and the debugging subsystem. High-level programming abstractions are also presented. This is a balanced design to achieve real-time response, high confidence detection, accurate tracking and energy efficiency simultaneously.

1. INTRODUCTION

Recently, many efforts have been undertaken to support new military applications by using large-scale wireless sensor networks. Unmanned real-time surveillance is one of the most promising applications. By combining the computation, sensing, actuation and wireless networking together, large-scale sensor networks have several advantages over many other distributed systems. First, sensor networks can be quickly deployed in an infrastructure-free environment, which is highly desired in military operations. Second, the redundancy introduced by a large-scale dense deployment makes a sensor system robust to node failures, which is critical in a hostile environment. Third, massively distributed in-network data processing allows exploration of new techniques for detection and classification, not posible with only a small number of devices. Along with these advantages, however, several challenges arise. The constrained resources in wireless sensor nodes, such as limited memory, power, processing, and communication bandwidth, impose problems previous research did not need to address. To realize the vision of wireless surveillance systems, many research efforts have been published that address challenging problems concerning networking [14], self-organization [4], energy-conservation [27, 30] and tracking [16] in this type of systems. However, most existing efforts address individual protocols or subproblems in the design space. Few systems actually provide a complete architecture and a running implementation tested in outdoor environments. The VigilNet project is, therefore, distinguished in this aspect. It is a large-scale sensor network system which has been successfully designed, built, demonstrated and delivered to the military for realistic deployment. To accomplish different mission objectives, the VigilNet system consists of 40,000 lines of code, supporting multiple existing mote platforms including MICA2DOT, MICA2, and XSM. To accommodate various mission requirements, VigilNet is dynamically reconfigurable and reprogrammable. For example, we can flexibly explore trade-offs between surveillance quality parameters and network lifetime by adjusting various system parameters online.

The remainder of this paper is organized as follows: Section 2 discusses the related work. Section 3 presents the overarching architecture of VigilNet and the details of individual subsystems. We list several lessons learned from our experience in Section 4 and conclude the paper in Section 5.

2. RELATED WORK

Middleware services form the basis of sensor network systems. Localization is a key service to identify the locations from where sensor readings are obtained. Two categories of localization have been proposed: range-based schemes [22, 24] and range-free schemes [2, 11]. The former category uses absolute point-to-point distance estimates (range) or angle estimates to localize nodes; The latter makes no assumptions about the availability of such information. Time synchronization is another critical middleware service. The reference broadcast scheme (RBS) proposed in [7] maintains information about the phase and frequency of each pair of clocks in the neighborhood of a node. While RBS achieves a precision of about 1 μs, the message overhead in maintaining the neighborhood information is high and may not be energy-efficient in large-scale systems. Maroti [21] synchronizes the network through limited flooding with timestamp values reassigned at intermediate nodes immediately prior to transmission. This scheme reduces the synchronization error introduced by uncertainty due to MAC contention. Our VigilNet system uses a variation of this approach. In addition, power management is employed to ensure network longevity. Other important protocols developed for modern sensor network hardware [5, 6] include medium access control [23, 31], sensing coverage [27, 30], energy aware routing [29], data aggregation [20, 10], topology management [4] and energy-aware applications [12, 26, 28].

With the assistance of various middleware services, several outdoor sensor network systems have recently been built. The Great Duck Island Project [26] explores long-term habitat monitoring in remote environments. ZebraNet [15] focuses on wildlife tracking in Africa. The Extreme Scaling project [6] investigates scalability issues in surveillance systems. The shooter localization system [8] combines precise time synchronization with accurate acoustic sensing to localize positions of snipers. The Wisden system [28] monitors the health of building structures by continuously retrieving structural response data. To complement the aforementioned efforts, VigilNet aims at building a practical military surveillance system, which can survive in harsh and hostile environments for a long period of time, and exhibit a high detection, classification and tracking performance. These requirements necessitate the design of unique solutions to various sensing, communication, and tracking problems [9, 17, 18, 19, 25, 32] and call for seamless integration of the resulting middleware components [12].

3. VIGILNET ARCHITECTURE

The VigilNet system has a layered architecture as shown in Figure 1. This architecture provides an end-to-end solution for supporting military surveillance applications. As an overview, this paper focuses on the design of individual components of this system, but omits the details of system implementation and performance evaluation. Such details can be found in other publications by the authors [9, 12, 17, 18, 19, 25, 32]

3.1 Sensing Subsystem

Sensing is the basis for any surveillance system. The VigilNet sensing subsystem implements detection and classification of targets using continuous online sensor calibration (to a changing environment) and frequency filters to determine critical target features. These filters extract the target signatures from a specific spectrum band, eliminating the burden of applying a computation-intensive Fast Fourier Transform. The sensing subsystem contains three detection algorithms for the magnetic sensor, acoustic sensor and passive infrared sensor (PIR), respectively.

- The magnetic sensor detection algorithm computes two moving averages of most recent magnetic readings. The slower moving average, with more weight on previous readings, establishes a baseline to follow the thermal drift noise caused by the changing temperature during the day. The faster moving average, with more weights on the current reading, detects the swift change in magnetic filed caused by ferrous targets. To make a detection decision, the difference between the two moving average values is compared to a dynamic threshold, which is established during the calibration phase.
- The acoustic sensor detection algorithm uses a lightweight power-based approach. It first computes a moving average of multiple recent acoustic readings, then establishes an auto-adapting acoustic threshold by calculating a moving standard deviation of readings over a certain time window. If an acoustic reading is larger than the sum of the moving average and its corresponding moving standard deviation, we consider it is a crossover. If the number of crossovers exceeds a certain threshold during a unit of time, this algorithm signals a detection to the upper layer components.
- The passive infrared sensor is designed to sense changes in thermal radiation that are indicative of motion. When there is no movement, the thermal reading is stable and does not trigger detections. If an object is moving in front of a PIR sensor, this object causes a thermal disturbance, triggering the PIR. Most moving objects, such as shaking leaves, rain drops, and vehicles, can trigger the PIR sensor. However, different thermal signatures generate trigger events with different frequencies. Low frequency detections ($< 2Hz$) are normally triggered by wind-induced motion and other slow moving objects. On the other hand, fast-moving targets such as vehicles generate signals with a much higher frequency. Therefore, it is sufficient to design a high pass ARMA filter to filter out the frequency components lower than 2Hz. Similar to other detection algorithms, this threshold is dynamically adaptated to accommodate the changing environment. For example, thermal noise is much higher in a hot and humid environment than that in a dry and cool one. In fact, thermal and humidity variations over the course of a day can significantly change threshold values.

Due to the space constraints, we do not detail the specific implementation of each sensing algorithm. More information on this subsystem can be found in [9].

3.2 Context-Awareness Subsystem

Sensed data is meaningful only when it is interpreted along with the context in which it is obtained. For example, a temperature reading is useless, if we don't know where and when such value is measured. The Context-Awareness subsystem comprises lower-level context detection components such as time synchronization and localization. These components form the basis for implementing other subsystems, such as the tracking subsystem. Localization ensures that each node is aware of its location, so that we can determine the location of detected targets. Time synchronization is responsible for synchronizing the local clocks of nodes with the clock of the base station, so that every node in the network has a consistent global view of time. Combining time synchronization and localization, we are able to estimate the velocity of targets.

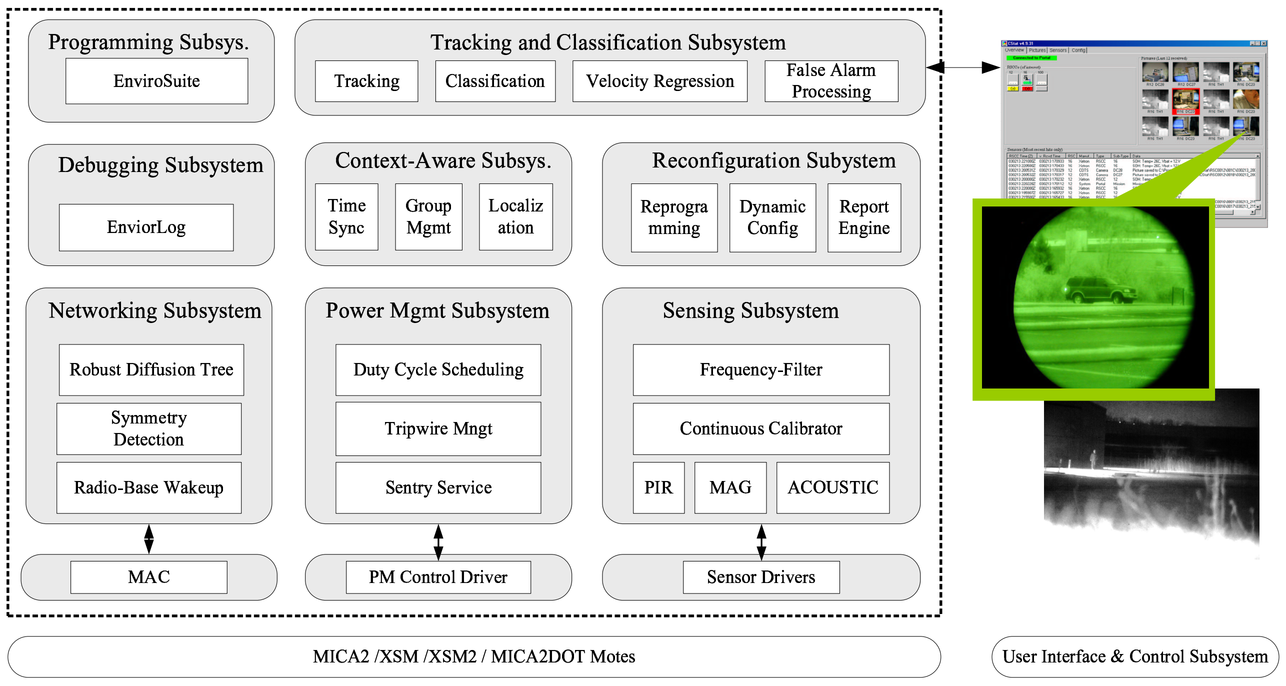


Figure 1: The VigilNet System Architecture

In the VigilNet system, we adopt a variation of the time synchronzation protocol developed by Maroti [21] and we design and implement a walking GPS solution [25], which assigns nodes their location at the time they are deployed. We are currently investigating realistic implementations of more dynamic localization schemes such as those described in [11].

3.3 Tracking and Classification Subsystem

When a target is detected by a set of nearby nodes, the tracking component creates a group. All nodes that detect the same event join the same group. The main contribution of the tracking component is to ensure the uniqueness (one-to-one mapping of external events to logical groups) and consistent identification (immutability of the mapping function) of targets, as long as targets are far enough apart from each other or have different signatures. When targets are very near each other and process an identical signature, we provide a disambiguation mechanism based on their path-histories. More information on this subsystem can be found in [1] and [18].

3.4 Networking Subsystem

After VigilNet collects detection information about incoming targets through the tracking and classification subsystem, it needs to deliver detection reports back to the control center through a multi-hop network. The networking subsystem consists of three major components: a link symmetry detection service, a robust diffusion service and a radio-based wakeup service. Low power radio components, such as Chipcon CC1000 used by MICA2 [5], exhibit very irregular communication patterns. To address this problem, we design a Link Symmetry Detection (LSD) module to reduce the impact of radio irregularity on upper layer protocols. The main idea of the LSD module is to build a symmetric overlay on top of the anisotropic radio layer, so that those protocols whose correctness depends on the link symmetry can be used without modification. More details on this solution can be found in [32]. The robust diffusion service utilizes a well-known path-reversal technique [14]. Basically, a base node disseminates tree construction requests to the rest of the network with a running hop-count initialized to zero. Requests are flooded outwards with hop-count incremented at every intermediate hop. After receiving tree construction requests, nodes establish multiple reverse paths towards the sending node. As a result, a multi-parent diffusion tree is constructed with the base node residing at the root. The Radio-based Wakeup service is designed to ensure end-to-end data delivery even intermediate nodes are in the dormant state (due to power management). To support the illusion of on-demand wakeup, a dormant node wakes up and checks radio activity periodically (e.g., for five milliseconds every several hundred milliseconds). If no radio activity is detected, this node goes back to sleep. Otherwise, it remains active to receive and relay messages. If an active node wants to wake up all neighboring nodes, it only needs to send out a message with a long enough preamble to last longer than the checking period of the dormant nodes.

3.5 Graphic User Interface and Control Subsystem

The networking subsystem delivers the reports to one or more command and control centers, where the Graphic User Interface and Control subsystem is located. This subsystem provides three major functionalities. First, it accepts the reports from the sensor field and displays such information graphically to the mission operators. Second, it allows the mission operators to disseminate the system configurations through the reconfiguration subsystem. Third, based on the initial detections from the sensor field, it makes final decisions on whether to wake up more advanced sensors. These advanced sensors not only classify the type of targets but also differentiate the model of the targets. Since they are extremely power consuming, they are normally turned off and only used when awakened by initial detections coming from the sensor field.

3.6 The Power Management Subsystem

One of the key design objectives of the VigilNet system is to increase the system lifetime to $3 \sim 6$ months in realistic deployment. Due to the small form factor and low-cost requirements, sensor devices such as XSM motes [6] are equipped with limited power sources (e.g., two AA batteries). The normal lifetime for such a sensor node is about 4 days if it remains active all the time. To bridge such a gap, we add a power management subsystem. Among all the middleware services, the tripwire service, sentry selection, duty cycle scheduling and wakeup service form the basis for the power management subsystem. We organize them into a multi-dimensional architecture. At the top level, we use the tripwire service to divide the sensor field into multiple sections, called tripwire sections. A tripwire section can be scheduled either into an active or a dormant state at a given point of time. When a tripwire section is dormant, all nodes within this section are in a deep-sleep state to conserve energy. When a tripwire section is active, we apply a second-level sentry service within this section. The basic idea of the sentry service is to select only a subset of nodes, which we define as *sentries*, to be in charge of surveillance. Other nodes, defined as *non-sentries*, can be put into a deep-sleep state to conserve energy. Rotation is periodically done among all nodes, selecting the nodes with more remaining energy as sentries. At a third-level, since a target can normally be sensed for a non-negligible period of time, it is not necessary to turn sentry nodes on all the time. We can schedule a sentry node in and out of sleep state to conserve energy. The sleep/awake schedule of a sentry node can be either independent of other nodes or coordinated with that of others in order to further reduce the detection delay and increase the detection probability. More information on nodes' duty cycle scheduling can be found at [3].

3.7 The Reconfiguration Subsystem

The VigilNet system is designed to accommodate different node densities, network topologies, sensing and communication capabilities and different mission objectives. Therefore, it is important to design an architecture that is flexible enough to accommodate various system scenarios. The reconfiguration subsystem addresses this issue through two major components: a multi-hop reconfiguration module and a multi-hop reprogramming module. The reconfiguration module allows fast parameter tuning through a data dissemination service, which supports limited flooding. Data fragmentation and defragmentation are supported in the reconfiguration subsystem to allow various sizes of the system parameters. The reprogramming module provides a high level of flexibility by reprogramming the nodes. More information on reprogramming can be found at [13].

3.8 The Debugging Subsystem

Debugging and tuning event-driven sensor network applications such as VigilNet are of great difficulty for the following reasons: (i) big discrepancies exist between simulations and empirical results due to various practical issues (e.g., radio and sensing irregularity) not captured in simulators, which makes them less accurate; (ii) In-field tests of the system require walking or driving through the field to generate events of interest actively, which makes in-field tests extremely costly. To address this issue, we add a debugging subsystem called EnviroLog [19] into VigilNet. EnviroLog logs environmental events into non-volatile storage on the motes (e.g., the 512 KB external flash memory) with timestamps. These events can then be replayed in their original time sequence on demand. EnviroLog reduces experimental overhead by eliminating the need to physically re-generate events of interest hundreds of times for debugging or parameter tuning purposes. It also facilitates comparisons between different evaluated protocols.

3.9 The Programming Interface

The programming interface is an extension of our prior work on EnviroSuite [18]. It adopts an object-based programming model that combines logical objects and physical elements in the external environment into the same object space. EnviroSuite differs from traditional object-oriented languages in that its objects may be representatives of physical environmental elements. EnviroSuite makes such objects the basic computation, communication and actuation unit, as opposed to individual nodes. Thus, it hides implementation details such as individual node activities and interactions among nodes from developers. Using language primitives provided by EnviroSuite, developers of tracking or monitoring applications simply can specify object creation conditions (sensory signatures of targets), object attributes (monitored aggregate properties of targets), and object methods (desired computation, communication or actuation in the vicinity of targets). Such specifications can be translated by an EnviroSuite compiler into real applications that are directly executable on motes. When defined object conditions are met, dynamic object instances are automatically created by the run-time system of EnviroSuite to collect object attributes and execute object methods. Such instances float across the network following the targets they represent, and are destroyed when the targets disappear or move out of the network.

3.10 System Work Flow

To avoid interference among different operations, VigilNet employs a multiple-phase work flow. The transition between phases is time-driven, as shown in Figure 2. Phases I through VII comprise the initialization process which normally takes about several minutes. In Phase I, the reconfiguration subsystem initializes the whole network with a set of parameters. In Phase II, the context-awareness subsystem synchronizes all nodes in the field with the master clock at the base, followed by the localization process in Phase III. In Phase IV and V, the networking subsystem establishes a robust diffusion tree for end-to-end data delivery. Phase VI invokes the power management subsystem to activate tripwire sections and select a subset of the nodes as sentries. The system layout, sentry distribution and network topology are reported to the graphic user interface and control subsystem in Phase VII. After that, the nodes enter into the main phase VIII– the surveillance phase. In this phase, nodes enable the power management subsystem in absence of significant events, and activate the tracking subsystem once a target enters into the area of interest.

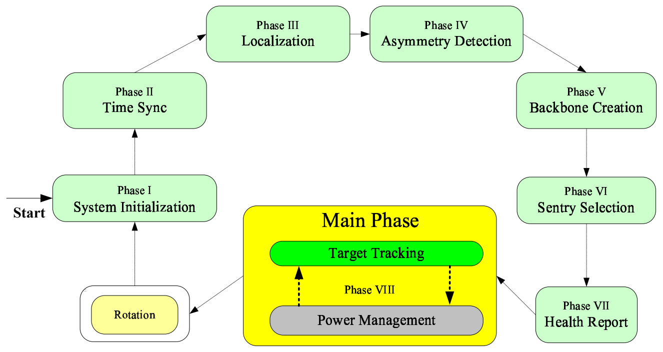


Figure 2: Phase Transition and Rotation

4. LESSONS LEARNED

We obtained valuable lessons during the process of building the VigilNet system. We share them here to assist similar efforts in other application domains.

1. **False Alarm Reduction:** False alarms introduce unnecessary energy consumption and inappropriate reactions. To deal with transient false alarms caused by distortion of sensing readings, one can use a simple exponential weighted moving average (EWMA). To address false alarms caused by slow-changing environments, one needs to use adaptive detection thresholds. To address persistent false alarms due to errors in a single sensor device, one can utilize in-network aggregation of inputs from a group of nodes to detect such an anomaly. In the worst case, when multiple persistent false alarms are generated simultaneously, one can filter out such false alarms by analyzing spatial and temporal correlations among the consecutive reports at the base station. More information on false alarm reduction can be found at [9].

2. **Communication Reliability:** Communication reliability is affected by link quality. Poor link quality can be addressed by retransmissions, however with a very high overhead. With high link redundancy in a dense sensor network, it is beneficial to carefully select high quality links for data delivery than to use FEC/ARQ techniques to improve transmission reliability over poor radio links. To select high quality links, one can use the link symmetry detection service we developed [32] for the VigilNet system. Moreover, it is beneficial to provide reliability selectively according to the semantics of the payload. Application-level mechanisms are appropriate to achieve such differentiation.

3. **Energy Bottleneck:** Although the radio power draw is very high, the amortized power draw from communication is actually very low in a surveillance system, due to its very low duty cycle. In contrast, we need to monitor the environment continuously to ensure detection. Accordingly, the amortized power draw for sensing is much higher than that of communication. Therefore, the most effective way to conserve energy is to reduce sensing redundancy (turn off a subset of nodes) in the absence of significant events and to activate nodes on-demand. In addition, a promising direction is to utilize hardware-driven wakeup functions to significantly reduce energy consumption during surveillance.

4. **Other Lessons:** Debugging and performance tuning in distributed sensor networks are extremely time consuming, especially during field tests. It is critical to have appropriate built-in system support for these functions, such as the reconfiguration subsystem and the debugging subsystem [19]. Second, in addition to the hardware and software, the mechanical design is very important to ensure good system performance. For example, enclosure design can significantly affect the sensitivity of senor nodes. Third, since the sensor nodes fail at a much higher rate in hostile outdoor environments, self-healing should be supported by every protocol integrated into the system.

5. CONCLUSIONS

This paper presents the design, implementation and evaluation of VigilNet – an integrated sensor system for long-term surveillance. We describe the functionalities of different subsystems within VigilNet. From our experience, we believe that surveillance using wireless sensor networks is a very promising direction. It has a lot of advantages such as fast ad hoc deployment, fine-grained robust sensing and tracking, low-power consumption, and low cost. VigilNet presents a proof that viable surveillance systems can be implemented and successfully deployed on current motes hardware.

6 ACKNOWLEDGEMENT

This work was supported in part by the DAPRPA IXO offices under the NEST project (grant number F336615-01-C-1905), the MURI award N00014-01-1-0576 from ONR and NSF grant CCR-0098269.

References

[1] B. M. Blum, P. Nagaraddi, A. Wood, T. F. Abdelzaher, S. Son, and J. A. Stankovic. An Entity Maintenance and Connection Service for Sensor Networks. In *The First Intl. Conference on Mobile Systems, Applications, and Services (MobiSys)*, May 2003.

[2] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less Low Cost Outdoor Localization for Very Small Devices. *IEEE Personal Communications Magazine, Special Issue on Networking the Physical World*, 7(4):28–34, August 2000.

[3] Q. Cao, T. Abdelzaher, T. He, and J. Stankovic. Towards Optimal Sleep Scheduling in Sensor Networks for Rare Event Detection . In *The Fourth International Conference on Information Processing in Sensor Networks*, 2005.

[4] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. In *6th ACM MOBICOM Conference*, 2001.

[5] CrossBow. *Mica2 data sheet*. Available at http://www.xbow.com.

[6] P. Dutta, M. Grimmer, A. Arora, S. Biby, and D. Culler. Design of a Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events. In *IPSN'05*, 2005.

[7] J. Elson and K. Romer. Wireless Sensor Networks: A New Regime for Time Synchronization. In *Proc. of the Workshop on Hot Topics in Networks (HotNets)*, October 2002.

[8] G.Simon and et. al. Sensor Network-Based Countersniper System. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.

[9] L. Gu, D. Jia, P. Vicaire, T. Yan, L. Luo, T. He, A. Tirumala, Q. Cao, J. A. Stankovic, T. Abdelzaher, and B.H.Krogh. Lightweight Detection and Classification for Wireless Sensor Networks in Realistic Environments. In *In submission*, 2004.

[10] T. He, B. M. Blum, J. A. Stankovic, and T. F. Abdelzaher. AIDA: Adaptive Application Independent Data Aggregation in Wireless Sensor Networks. *ACM Transactions on Embedded Computing System, Special issue on Dynamically Adaptable Embedded Systems*, 2004.

[11] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-Free Localization Schemes in Large-Scale Sensor Networks. In *Proc. of the Intl. Conference on Mobile Computing and Networking (MOBICOM)*, September 2003.

[12] T. He, S. Krishnamurthy, J. A. Stankovic, and T. Abdelzaher. An Energy-Efficient Surveillance System Using Wireless Sensor Networks. In *The Second International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2004.

[13] J. Hui and D. Culler. The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.

[14] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *the Sixth Annual International Conference on Mobile Computing and Networks*, 2000.

[15] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet. In *Proc. of ASPLOS-X*, October 2002.

[16] J. Liu, J. Reich, and F. Zhao. Collaborative In-Network Processing for Target Tracking. *J. on Applied Signal Processing*, March 2003.

[17] L. Luo, T. Abdelzaher, T. He, and J. Stankovic. Design and Comparison of Lightweight Group Management Strategies in EnviroSuites. In *International Conference on Distributed Computing in Sensor Networks*, 2005.

[18] L. Luo, T. Abdelzaher, T. He, and J. A. Stankovic. EnviroSuite: An Environmentally Immersive Programming Framework for Sensor Networks. *ACM Transactions on Embedded Computing System, Special issue on Dynamically Adaptable Embedded Systems*, accepted.

[19] L. Luo and et. al. EnviroLog: Asynchronous Event Record and Replay Service for Wireless Sensor Network. In *in submission*.

[20] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks. In *Operating Systems Design and Implementation*, December 2002.

[21] M.Maroti, B. Kusy, G. Simon, and A. Ledeczi. The Flooding Time Synchronization Protocol. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.

[22] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust Distributed Network Localization with Noise Range Measurements. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.

[23] J. Polastre and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.

[24] A. Savvides, C. C. Han, and M. B. Srivastava. Dynamic Fine-grained Localization in Ad-Hoc Networks of Sensors. In *Proc. of Mobile Computing and Networking (MOBICOM)*, 2001.

[25] R. Stoleru, T. He, and J. A. Stankovic. Walking GPS: A Practical Solution for Localization in Manually Deployed Wireless Sensor Networks. In *1st IEEE Workshop on Embedded Networked Sensors EmNetS-I*, October 2004.

[26] R. Szewczyk, A. Mainwaring, J. Anderson, and D. Culler. An Analysis of a Large Scale Habit Monitoring Application. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.

[27] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks. In *First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, November 2003.

[28] N. Xu, S. Rangwala, K. K. Chintalapudi, D. Ganesan, A. Broad, R. Govindan, and D. Estrin. A Wireless Sensor Network for Structural Monitoring. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.

[29] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *MobiCom*, 2001.

[30] T. Yan, T. He, and J. Stankovic. Differentiated Surveillance Service for Sensor Networks. In *First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, November 2003.

[31] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient mac protocol for wireless sensor networks. In *INFOCOM*, 2002.

[32] G. Zhou, T. He, and J. A. Stankovic. Impact of Radio Irregularity on Wireless Sensor Networks. In *The Second International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2004.

Sensor Localization with Ring Overlapping Based on Comparison of Received Signal Strength Indicator

Chong Liu
Computer Science Dept.
University of Victoria
BC, Canada V8W 3P6
chongliu@cs.uvic.ca

Kui Wu
Computer Science Dept.
University of Victoria
BC, Canada V8W 3P6
wkui@cs.uvic.ca

Tian He
Computer Science Dept.
University of Virginia
Charlottesville, Virginia 22904-4740
tianhe@cs.virginia.edu

Abstract—
Sensor localization has become an essential requirement for realistic applications over Wireless Sensor Networks (WSNs). Radio propagation irregularity and the stringent constraint on hardware cost, however, make localization in WSNs very challenging. Range-free localizations are more appealing for range-based ones, since it does not depend on received signal strength to estimate distance and thus needs simple and cheap hardware only. In this paper, we propose a ring-overlapping, range-free approach using Ring Overlapping based on Comparison of Received Signal Strength Indicator (ROCRSSI). Simulation results have verified the high estimation accuracy achieved with ROCRSSI.

Key Words—
Range-Free Localization, Wireless Sensor Networks

I. Introduction and Background

Wireless Sensor Networks(WSNs) become the current hot spot of networking area and have been used for various applications, such as habitant monitoring, environment monitoring, and target tracking. Location information plays a crucial role in understanding the application context in WSNs [5] [6], and many localization algorithms for WSNs have been proposed to provide per-node location information. They can be divided into two categories: ranged-based methods [1] [3] and ranged-free methods [2] [7] [4]. Range-based localization depends on the assumption that the absolute distance between a sender and a receiver can be estimated by received signal strength or by the time-of-flight of communication signal from the sender to the receiver. The accuracy of such estimation, however, is subject to the transmission medium and surrounding environment and usually relies on complex hardware [8]. In contrast, range-free localization never tries to estimate the absolute point-to-point distance based on received signal strength. As such, the design of hardware can be greatly simplified, making rang-free localization very appealing for WSNs.

In this paper, we propose another range-free localization approach, Ring Overlapping based on Comparison of Received Signal Strength Indicator (ROCRSSI), that uses ring-overlapping to estimate nodes' location. This approach has the following prominent features. First, it does not require sensor nodes to send out control messages, and the communication cost is light and on anchor nodes only. Second, ring-overlapping, compared to triangle-overlapping in APIT [4] that is demonstrated to perform best for randomly deployed WSNs among existing range-free localization approaches, generates small intersection area and results in more accurate location estimation. Finally, the proposed ring-overlapping method is robust under irregular radio propagation patterns.

II. Ring Overlapping based on Comparison of Received Signal Strength Indicator (ROCRSSI)

A. Introduction of ROCRSSI

The motivation of ROCRSSI is to get accurate estimation and reduce communication overhead with small number of anchors. Anchors, which is generally required to deploy with sensor nodes by most of range-free localization methods [2] [7] [4], are those nodes who know their locations and usually have larger transmission power than normal sensor nodes. The general idea of ROCRSSI is that each sensor node uses a series of overlapping rings to narrow down the possible area in which it resides. As the example shown in Fig. 1(b), if S can determine that its distance to A is larger than the distance between A and B, but less than the distance between A and C, it can conclude that it falls within the ring center at A with the inner radius equal to the distance between A and B and the outer radius equal to the distance between A and C. Similarly, S can figure out another ring centered at anchor B, and a circle centered at anchor C. Then it calculates the intersection area of these rings (or circles) and takes the gravity of this area as its estimated location.

The rings can be generated by comparison of the signal strength a sensor node receives from a specific anchor and the signal strength other anchors receive from the same anchor. For example, in Fig. 1(a), assume that A, B, and C are three anchor nodes and S is a sensor node. Assume that anchor A sends out beacon messages and the signal strength received by anchor B, anchor C, and sensor S is $RSSI_{AB}$, $RSSI_{AC}$, and $RSSI_{AS}$ respectively. If $RSSI_{AB} > RSSI_{AS} > RSSI_{AC}$, then S is likely to fall within the shadowed ring area if anchor B, anchor C, and node S are roughly in the same direction from anchor A. Since ROCRSSI does not try to map the received signal strength to absolute point-to-point distance, it belongs to range-free localization approaches. Notably, ROCRSSI only compares the relative strength of RSSI and does not depend on absolute RSSI values.

The correctness of ROCRSSI is based on the assumption that in a certain range of direction, with the increase of distance between a sender and a receiver, the signal strength at the receiver decreases monotonically. This assumption is usually true for

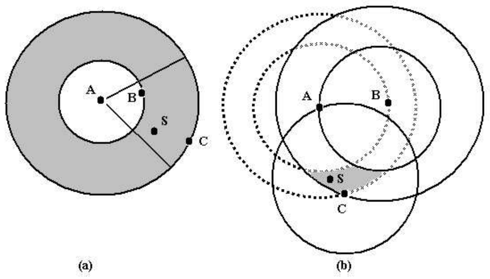


Fig. 1. Examples of ROCRSSI

realistic sensor networks [4]. If omni-directional antennas are used in a homogeneous environment, we can assume isotropic radio attenuation model and ROCRSSI can get the best estimation accuracy. Note that although ROCRSSI depends on the homogeneity of radio transmission in a large range of direction, the estimated location errors by ROCRSSI are generally small even under circumstances with very irregular radio propagation. Briefly, this is because we introduce extra constraints on usable RSSI information. The detailed reasons will be illustrated in Section II-D.

ROCRSSI works in a purely distributed fashion. In ROCRSSI, the received signal strength from a certain anchor, say anchor A, can be measured by neighbouring nodes (anchors and sensors) that fall within the radio transmission range of anchor A. All neighbouring *anchors* will broadcast the measured signal strength from anchor A. In this way, each sensor node will be able to collect enough information to generate a series of overlapping rings that it believes itself falls within. Note that *only* anchor nodes are required to send out control messages.

The ROCRSSI algorithm can be broken into two phases: RSSI propagation and location estimation, which are described in detail in the sequel.

B. RSSI Propagation

At initial period, each anchor broadcasts a specific number of beacon messages. During this period, each of its neighbouring anchors and sensors will constantly sample received signal strength. At the end of this period (for example, the number of broadcast beacon messages in each anchor have reached a predefined number), all neighbouring anchors and sensors will calculate the mean of the measured signal strength. For a sensor node, it will store the mean value for later use. For an anchor node, it will broadcast a RSSI message, including the mean value of the measured signal strength, its own ID, the ID of the reference anchor (the one who has been measured), and its own location information. Any sensor node that receives the RSSI message will record the related information.

C. Location Estimation

When sensor node S obtains enough information after the initial RSSI propagation stage, it can make use of the information to calculate its own location. The pseudo-code of the algorithm is shown in Fig. 2. The basic idea is to generate a series of rings, each of which S believes itself falls within. S calculates the intersection area of these rings, and takes the gravity of the final intersection area as its estimated location.

```
// Assume that sensor node $S$ wants to calculate its location
// Input:
//        $S_n$ denotes the set of neighbouring anchors of sensor $S$.
//        $S_A$ denotes the set of neighbouring anchors of anchor $A$,
//        where $A$ is one of neighbouring anchors of sensor $S$.
//        $RRSI_{AB}$ denotes the signal strength recorded from node $A$ to node $B$.
Process LocationEstimation( )
{
   RingSet $R$ = {};
   While($S_n$ has more elements){
         step 1: Anchor $A$= $S_n$.nextElements();
         step 2: Split $S_A$ into two parts: $S_{A1}$ and $S_{A2}$, such that
                 each element $I$ in $S_{A1}$ has larger $RRSI_{AI}$ than $RRSI_{AS}$ and
                 each element $J$ in $S_{A2}$ has smaller $RRSI_{AJ}$ than $RRSI_{AS}$.
         step 3: if (($S_A$ == null) or ($S_{A2}$==null)) goto step 1.
                 $J$ = the element with the largest value in $S_{A2}$;
                 $d_2$ = distance between $J$ and $A$;
                 if ($S_{A1}$ ==null) { $d_1 = d_2$;}
                   else {
                         $I$ = the element with the smallest value in $S_{A1}$;
                         $d_1$ = distance between $I$ and $A$;
                   }
                 Generate a ring $r$ centered at $A$ with inner radius $d_1$ and outer radius $d_2$;
                 $R = R + \{r\}$; // insert $r$ into $R$.
   }
   step 4: Calculate the intersection area of all rings in R;
   step 5: return {the gravity of this intersection area};
}
```

Fig. 2. Algorithm for Location Estimation

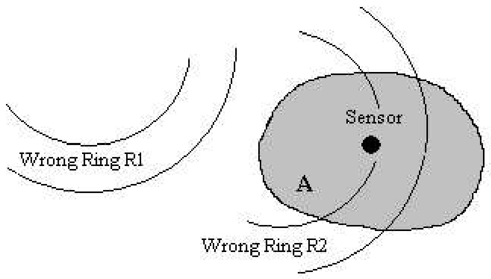


Fig. 3. Grid scan algorithm alleviates the influence of wrong rings

Note that in the algorithm, if the set S_A or the set S_{A2} is empty, then no ring will be generated. If the set S_{A1} is empty but the set S_{A2} is not, only one circle is generated. If both sets S_{A1} and S_{A2} are not empty, S will generate a ring.

In step 4, a grid-scan algorithm [4] is used to calculate the gravity of the intersection area. In this algorithm, the whole terrain is divided into small pieces of grids. Each grid maintains a counter which is initialized to 0. Every time a ring is generated by comparison of signal strength, the counter values associated with the grids within the ring are increased. Once all possible rings are calculated, we scan the whole grid array to find the area with the maximum counter value, then take this area as the final intersection area and calculate its gravity. It is easy to see that the size of the grids determines the possible smallest granularity of location error. Small grids are thus preferred but small grids need more calculation time.

D. Handling Radio Irregularity

According to the measurement results over real sensor devices, radio propagation is usually not homogenous in all directions [9], that is, different directions have different radio attenuation rates. So a good localization algorithm should accommodate radio irregularity by not assuming isotropic path losses. ROCRSSI does not exclude generating wrong rings and thus makes incorrect estimation because of the irregularity of radio propagation. Nevertheless, the grid-scan algorithm that sensor nodes use to calculate the gravity of intersection area helps reduce the influence of wrong rings. As mentioned before, grid-scan algorithm takes the area consisting of grids with maximum counter as the final intersection area. In Fig. 3, suppose *more than half* of the rings generated by RSSI comparison are correct, and the intersection area of all correct rings is the gray area labeled as A. The gravity of area A will be taken finally as the estimated location of the sensor, because even if all wrong rings have no intersections with area A and even if all wrong rings happen to overlap at one place, the counter value associated with the grids at that (wrong) place must be smaller than that of grids in area A. As a result, wrong rings will not be taken into consideration in the location calculation.

If there are some wrong rings overlap with area A, such as the wrong ring R_2 in Fig. 3, then the final intersection area may not contain the sensor node. But the final intersection area is a subset of area A and is thus within the area A. Since the size of area A is usually small if the number of audible anchors is large enough, the gravity of the final intersection area will be very close to the real location of the sensor. As such, even if ROCRSSI may generate some wrong rings, it can still yield fairly accurate location estimation.

III. Simulation results and analysis

In our simulation, we assume all the sensor nodes have the same maximal sensor radio transmission range R. Notably, R is used for normalization *only*. All distances including error estimation are normalized by R to ensure generally applicable results. Sensor nodes and anchor nodes are randomly deployed within a square area with each edge of length 10R. For each simulation scenario, ten runs with different random seeds were executed and the results were averaged. We define the location estimation error as the Euclidian distance between the real location of a node and its estimated location. We use average location error as the metric to evaluate the accuracy of location estimation. It is defined as the mean of location estimation errors collected over all determined sensor nodes in ten runs.

We implement three localization methods, ROCRSSI, APIT, an improved version of APIT, denoted as APIT+. APIT+ differs from APIT in that when a triangle is added, only those grid points within maximum range of all heard anchors are added. This method will improve the performance of APIT in the face of radio irregularity with extra checking for each grid point. In order to compare the performance of different methods under radio irregularity, we extend the DOI model in [4] so that it can calculate the possible received signal strength at any specific point within the maximal radio range of a sender. DOI value is used to adjust the degree of radio irregularity and large DOI values represent large variation of radio irregularity.

Fig. 4 demonstrates that ROCRSSI always outperforms APIT and APIT+ in terms of average estimation error, no matter whether the radio propagation is regular or irregular. This is because the intersection of rings usually has smaller size than the intersection of triangles. A simple example can be found in 1(b), where it is easy to see that the size of the shadowed ring intersection area is smaller than the size of $\triangle$ABC (if we assume that S falls within $\triangle$ABC).

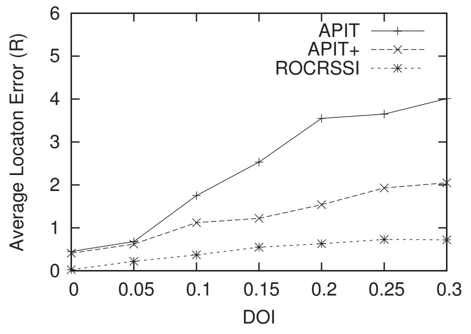


Fig. 4. The approximate comparison of APIT and ROCRSSI

IV. Conclusion and Future Work

Range-free localization presents a promising solution for the localization problem in WSNs. This paper proposes a ROCRSSI method which achieves more accurate location estimation than existing high performance APIT method. The ROCRSSI method has the following nice features:

1) It does not require sensor nodes to send out control messages and thus poses very little overhead on sensor nodes.
2) It generates small intersection area and results in accurate location estimation.
3) It is robust under irregular radio propagation patterns.

Acknowledgement

This work is supported partially by the Natural Sciences and Engineering Research Council of Canada (NSERC). Also, the authors would like to express their gratitude to Dr. I. Stojmenovic for providing some background materials as well as feedback about the first draft of this paper.

References

[1] P. Bahl and V. N. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System," *Proceedings of the IEEE INFOCOM 2000*, Telaviv, Israel, March 2000.

[2] N. Bulusu, J. Heidemann and D. Estrin, "GPS-less Low Cost Outdoor Localization for Very Small Devices," *IEEE Personal Communications Magazine*, newblock Vol. 7, No. 5, October 2000, pp. 28-34.

[3] L.Girod and D.Estrin, "Range Estimation using Acoustic and Multimodal Sensing," *Procedings of IROS 2001,* Maui, Hawii, October 2001.

[4] T. He, C. Huang, B.M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-Free Localization Schemes for Large Scale Sensor Networks," *Proceedings of the ninth annual international conference on Mobile computing and networking (MobiCom 2003)*, San Diego, California, September 2003, pp. 81-95.

[5] Y.B. Ko and N. H. Vaidya, "Location-Aided Routing (LAR) Mobile Ad Hoc Networks," *Proceedings of the fourth annual international conference on Mobile computing and networking (MobiCom 98)*, Dallas, Texas, October 1998, pp. 66-75.

[6] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M.B. Srivastava, "Coverage Problems in Wireless Ad-hoc Sensor Neworks," Proceedings of IEEE Infocom 2001, Ankorange, Alaska, April 2001.

[7] D. Niculescu and B. Nath, "DV Based Positioning in Ad hoc Networks," *Journal of Telecommunication Systems*, Vol. 1 2003.

[8] A. Savvides, C. Han, and M. B. Strivastava, "Dynamic Fine-Grained Localization in Ad-hoc Networks of Sensors," *Proceedings of the seventh annual international conference on Mobile computing and networking (MobiCom 01)*, Rome, Italy, July 2001.

[9] G. Zhou, T. He, J. Stankovic, "Impact of Radio Asymmetry on Wireless Sensor Networks," *Proceedings of MobiSys 2004,* Boston,Massachusetts, June, 2004. To appear.

RESTORE: A Real-Time Event Correlation and Storage Service for Sensor Networks

Sudha Krishnamurthy
Deutsche Telekom Laboratories
Berlin, Germany
sudha.krishnamurthy@telekom.de

Tian He
Department of Computer Science
University of Minnesota
Minneapolis, MN
tianhe@cs.umn.edu

Gang Zhou, John A. Stankovic, Sang H. Son
Department of Computer Science
University of Virginia
Charlottesville, VA
{gzhou,stankovic,son}@cs.virginia.edu

***Abstract*— This paper describes the design of RESTORE, which is a framework for providing in-network event correlation and storage service for sensor environments. RESTORE uses a data-centric approach in which it partitions a sensor network into zones and maps every event to a zone. The sensor nodes in a zone make use of their cooperative storage resources and redundancy to improve information availability and energy efficiency. RESTORE is useful for temporarily buffering information in sensor environments that have intermittent connectivity to a base station. RESTORE may also be used as the underlying framework for an event notification service that publishes events directly from a sensor network to the subscribers. The contributions of this paper include a) an event taxonomy for correlating sensor events within the network, and b) mechanisms to organize information using the collaborative resources, in order to enable real-time event correlation within a sensor network. We also present preliminary simulation results that examine how the collaboration within a zone in RESTORE impacts the energy consumption, information availability, and message overhead.**

I. Introduction

Wireless sensor networks are primarily large-scale, decentralized information systems. An information system for sensors needs to have the ability to store the stream of observations generated locally by the sensors and more importantly, correlate the raw observations from different parts of the network to generate events that are meaningful to the end users. There have been several efforts to correlate the information gathered by the deployed sensors and generate higher-level inferences in the context of different applications, such as surveillance [1], environmental monitoring [2], [3], and structural health monitoring [4]. However, in a majority of these efforts, the storage and correlation of sensor information is performed at the edge of the network in more powerful devices that serve as base stations. This approach is certainly the most viable option when the devices in the network are primitive and the infrastructure allows easy and uninterrupted access to the powerful edge devices. However, recent technological advances are making it possible to integrate multi-modal sensing capabilities, resulting in network devices that are smarter and capable of fine-grained classification. Should this technological trend continue, performing event correlation and storage within the sensor network should be an increasingly viable option and benefit several applications. In this paper, we motivate the need for an in-network service that buffers the observations from the sensors for a limited period of time and uses that as a basis for correlating events within the network. We also present an architecture that makes use of the inherent redundancy and collaborative nature of sensor networks to realize such a service.

A. The Case for In-Network Correlation and Storage

The use of external base stations for storage and correlation has some drawbacks. Unlike the tiny sensor devices that can be left unattended in remote environments, the base stations are often more power-consuming and less unobtrusive. Leaving these base stations unattended in remote sites may not be a feasible option, especially in harsh and unfriendly environments where stealthiness is important, such as in battlefield surveillance applications. This problem can be addressed by using long-range relays to transmit the sensor observations to remote base stations. However, this long-range communication may not always be reliable. End-to-end connectivity may be intermittent or periodic and may be disrupted due to several factors, such as the weather. Moreover, such long-range communication may be more vulnerable to interception.

A better alternative in such environments, where communication may be disrupted, is to buffer the observations of the sensors within the sensor network for limited periods of time and correlate the observations within the network. Event notifications generated as a result of the correlation can then be directly delivered to the end subscribers. The data stored in the network may also be uploaded in batches, whenever the base stations are connected. Such an approach reduces the dependence on external base stations and minimizes the use of long-range communication. Recent efforts, such as the Zigbee gateway working group [5], delay-tolerant architectures [6], and TinyREST [7] are addressing the interoperability issues involved in integrating the sensor networks more closely with the Internet and traditional networks. We think that an in-network event correlation and storage service would vastly improve the utility of sensor networks and provide impetus and motivation for such efforts that attempt to bridge the gap between sensor networks and traditional networks.

Sensor networks are typically perceived as challenged environments [6]. So a valid question is whether the deployed sensor devices are capable of realizing an in-network storage and correlation service. Recent technological trends have been encouraging in this regard. If the sensor devices are low-end devices, like motes, each of which has a storage capacity of only 512 KB [8], building an in-network storage scheme will require cooperative caching schemes. However, if the motes are deployed along with devices that have larger storage capacity, such as PDAs and Stargate processors [9], then such a heterogeneous collection of devices would make it increasingly feasible to provide long-term, in-network storage.

In order to implement an in-network event correlation service, the nodes in the network should at the minimum be able to order events in space and time. Mote-like devices possess this ability and this allows them to perform simple correlations pertaining to a single object, such as tracking the motion of a single car with the help of magnetic sensors. However, if multiple cars of identical size appear in the network concurrently and their trajectories intersect, this basic ability is insufficient to distinguish between the different objects. To perform event correlation within the network in such scenarios, we would need nodes with the ability to perform finer-grained classification. Recent advances in nodes with multi-modal sensing is promising in this regard. For example, the integrated RFID-sensor systems from SkyeTek [10] combine a MicaDot node [11] with an RFID reader. This combination of sensors and RFID provides access to a richer source of data and allows different instances of the same kind of event to be distinguished, making it possible to perform more fine-grained event correlation within the network.

B. Overview of RESTORE

In this paper, we describe an in-network event-correlation and storage service for sensor networks, called RESTORE. The design principle behind RESTORE is that no single node in a zone has enough resources to store and correlate information about all the events in the network. So RESTORE makes use of the inherent redundancy and cooperative resources of a sensor network to store and correlate events within the network. RESTORE organizes the nodes in the sensor network into zones. Each zone is responsible for storing information about one or more events. A zone functions like a cooperative cache and is the smallest unit of collaborative storage in the sensor network. Each zone has a storage manager that is responsible for coordinating the storage and event correlation within its zone. Zone members collaborate to achieve energy efficiency and fault tolerance. The two main goals of RESTORE are the following:

Real-time event correlation: RESTORE enables a sensor network to publish event notifications directly to the subscribers, without any intermediate processing at base stations. In many applications, the subscribers may be interested in being notified about composite events instead of every individual event. The cooperative resources of the members within a zone can be used to record observations from sensors across the network in a decentralized manner to form a more composite view of the events in the network. As new observations arrive, they can be correlated with relevant portions of the stored data in real-time. While the correlation happens as new data arrives, the event notifications are generated according to the frequency and specifications of the subscribers.

Short-term, reliable storage: RESTORE not only minimizes the need for long-range communication, it also provides reliability. In environments where connectivity to an external persistent storage is intermittent, if the observations are directly transmitted to the base station and are lost in the process due to unreliable channels, the base station may not be able to generate some event notifications. In such an environment, RESTORE enables a sensor network to store information and correlate events within the network. If the event notifications to the subscribers are lost, they can be retransmitted, because the observations used to generate the notifications are stored in the network for a period of time. Whenever a base station is available, the individual zone managers can upload the stored data from their zones and reclaim the storage resources. RESTORE also allows the stored data to be uploaded to mobile base stations, which is useful for applications, such as, wildlife tracking[12].

The RESTORE architecture described in this paper assumes a network consisting of homogeneous low-end devices, like the motes. However, if nodes with more powerful storage capabilities, such as PDAs, are also present, RESTORE would take advantage of such nodes by favoring them as storage managers. RESTORE also takes advantage of RFID-enabled objects, if they are present, and uses the tags to distinguish between different instances of an object and perform fine-grained event correlation.

The rest of the paper is organized as follows. In Section II, we describe the use of an in-network correlation and storage service for a few application scenarios. In Section III, we discuss the architecture of RESTORE and describe how it partitions the network into zones, in order to provide energy-efficient storage. In Section IV, we present an event taxonomy and describe how RESTORE uses the zones to store and correlate information for events characterized by this taxonomy. In Section V, we present preliminary performance results. In Section VI, we compare RESTORE with other sensor network storage schemes. In Section VII, we present our conclusions.

II. Applications

We now illustrate the use of an in-network event correlation and storage service, like RESTORE, in some typical sensor applications.

- Consider an application that makes use of integrated RFID-sensor nodes in parking meters to monitor parking violations. Most vehicles these days have RFID-based toll tags or license plates. The sensors keep track of the parking duration and the RFID reader identifies the vehicle. The sensor nodes are grouped into regions and collectively store the violations in that region. The stored data can be logged to an external persistent storage either periodically or when the number of violations in a region exceeds a certain threshold. Alternatively, a cop can drive around with a PDA and retrieve the stored data from

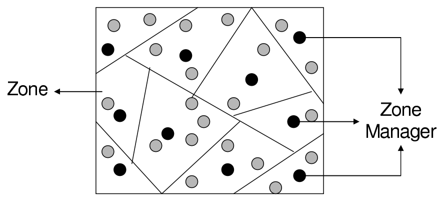


Fig. 1. Zone partitioning in RESTORE

the wireless sensors. Reliable storage is important here, because loss of data results in improper law enforcement and loss of revenue. The in-network event correlation triggers an event whenever the number of violations in a region exceeds a certain threshold.

- Consider an application that makes use of sensors to monitor the behavior of autistic patients. These patients usually follow a very regular pattern and caregivers need to be notified only when the behavior on a particular day does not match the pattern. An in-network store can be used to record the behavior of the patients daily. As new sensor observations arrive each day, they can be correlated with the stored data and if there is a significant deviation in the behavior of a patient on any day, the in-network correlation service triggers an event notification.
- Whenever new sensor network protocols are developed, they often need to be debugged and evaluated experimentally through an actual deployment. This usually involves collecting statistics from the sensor network over a period of time and performing an offline analysis of the observations. During this phase, the sensors are typically deployed in isolated environments where it may not be possible to provide continuous access to base stations for the entire experimental period. In such a scenario, it would be convenient to eliminate the use of base stations by using an in-network store to collect the statistics at runtime. The data collected can then be uploaded to a base station at the conclusion of the experimental period.

III. Description of RESTORE Architecture

The RESTORE architecture needs to support in-network storage and correlation of events in a timely manner. One way to achieve this is to allow the nodes that have observed an event to store the information locally and then retrieve the information from all the nodes that have observed the same event during the correlation process. However, such a dispersed storage of information makes the retrieval process more complex and increases the time for correlation. Instead, our goal in designing RESTORE is to enable local decision making based on a global view of an event. In order to achieve this, RESTORE logically partitions a sensor network into storage zones as shown in Figure 1. RESTORE follows a data-centric approach in which it maps every network event to a zone. The zone to which an event is mapped becomes the primary storage zone (PSZ) for that event. Members of that zone cooperatively use their resources to store and correlate information gathered by sensors across the network about that event. One of the nodes in each zone serves as the *zone manager*. The zone managers serve as the primary storage managers (PSM) in their zone. They are responsible for deciding how to store and correlate information within their zone by making use of the cooperative resources of their zone members to achieve energy efficiency and increased availability.

Thus, the design of RESTORE must address two main issues: a) a mechanism for partitioning a sensor network into zones, and b) mechanisms for utilizing the collective resources of a zone for in-network storage and event correlation. In order to address these issues, RESTORE makes the following assumptions about the nodes in the network. RESTORE assumes that all of the sensor nodes know their location relative to each other with some degree of accuracy. In this paper, we assume that all sensing nodes have the same sensing range, although this can be relaxed when multiple types of sensors are involved. RESTORE assumes that the communication range of a node is at least twice its sensing range. This ensures that if managers are chosen in such a way that there is at least one manager in every sensing radius, then adjacent managers will be within communication range of each other. RESTORE also assumes that the sensor network is reasonably dense and has a fairly uniform distribution of nodes. We now describe how RESTORE partitions a sensor network into storage zones. In the next section, we describe how these zones are used for in-network event correlation and storage.

The goals of the partitioning algorithm in RESTORE are as follows. The basic goal is to split the sensor network into disjoint zones in which each zone member is within one-hop communication radius from its manager. There is a zone manager within every sensing radius. Since zone managers are always awake, this helps to provide sensing coverage for the entire network. Every zone must be reachable from every other zone in the network, so that reports about an event detected by a zone can be routed to the PSZ for that event. This is done by allowing all of the zone managers in RESTORE to form a communication backbone for the entire network. A zone in RESTORE is identified by its manager and whenever the members of a zone detect an event, the zone manager uses the identities (locations) of the managers to find the mapping between an event and its PSZ. Hence, every zone manager needs to know the identities of all the other zone managers in the network.

Several clustering and partitioning algorithms have been proposed for sensor networks and some of them follow the minimum dominating set approach to minimize the number of partitions by maximizing the size of each partition (for e.g. [13]). Such an approach primarily attempts to minimize energy consumption and maximize coverage. However, RESTORE uses the partitions to store information about a certain number of distinct events (N) and provide a minimum degree of information availability (A) for each event. The zone size, which we define as the number of nodes in a zone, is a tradeoff between the parameters N and A. Maximizing the

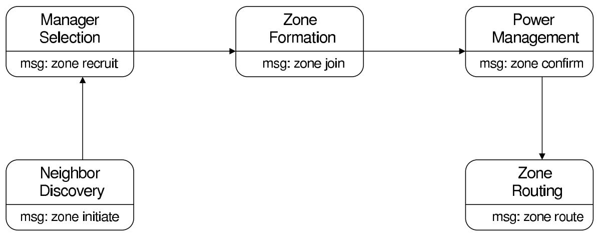


Fig. 2. Phase transition during zone partitioning in RESTORE

size of each zone helps to store more information about an event and improves the information availability for the event. However, it reduces the number of distinct events that can be stored in the network. RESTORE regards both N and A as application-specific inputs. The parameter, A determines the minimum number of nodes that need to be present in each zone. We now describe a way to partition the network into storage zones, in order to meet the goals stated above. Figure 2 shows the phases involved in the partitioning process and the messages transmitted in each phase.

A. Neighbor Discovery

The sensor nodes in RESTORE organize themselves into zones based on their locality. In order to do this, the sensor nodes first discover their neighbors by propagating discovery beacons that contain the sender's location, residual energy, storage capacity, and local clock value. In addition to neighbor discovery, these beacons help the nodes to synchronize their local clocks. Each node broadcasts its discovery beacon locally within its sensing range. We omit the details of how the discovery beacons are propagated through the network, because there are well-known mechanisms to do this (for e.g., [1]).

At the end of the discovery process, every sensor node conceptually has its own zone. Its potential zone members are the nodes within its sensing range that are recorded in its local neighborhood table. In a high density network, some of the nodes may be included in multiple neighboring zones. Hence, the next step is to select a subset of the nodes as zone managers and ensure with a high probability that every node in the network joins the zone of exactly one of the selected managers [1].

B. Manager Selection

The decision to become a manager is taken locally by each node by first computing the eligibility value (EV) for each of the nodes in its neighborhood table. In RESTORE, we use the information that a node receives from its neighbors during the neighbor discovery phase to compute the eligibility value. The EV of a node m in a temporary zone z defined by its neighborhood table is denoted as $EV_m(z)$ and is defined by Equation 1. $P_m(t)$ is the probability that a node m that remains awake during the entire interval t will not fail at the end of the interval, where t is the time interval before the managers are rotated again. Here, we primarily consider failure due to power drain. So $P_m(t)$ is determined by the residual energy of the node m. S_m is the storage capacity of node m and d_{mi} is the Euclidean distance between nodes m and i, both of which are members of the temporary zone z.

$$EV_m(z) = \frac{P_m(t) * S_m}{\sum_{i \in z} d_{mi}} \tag{1}$$

A node nominates itself as a zone manager if a) its temporary zone z has at least the minimum number of nodes required to meet the availability A, defined earlier, and b) it has the highest eligibility value among its potential zone members. The first criterion favors the creation of zones that have enough members for storing information. The second criterion favors nodes that have higher residual energy, storage capacity, and that are closer to their zone members as zone managers. Since a zone manager remains awake all the time and consumes energy, choosing a manager node that has higher residual energy is preferable. A zone manager needs to communicate with its zone members in order to store information and the zone members, in turn, need to communicate the information they have stored to their zone manager during event correlation. In order to reduce the energy spent in this two-way communication, it is preferable to choose zone managers that are closer to most of their zone members. Finally, in the case of heterogeneous networks, choosing nodes that have higher storage capacity as zone managers helps reduce the frequency of communication with the zone members.

A node that nominates itself as a zone manager advertises its nomination by locally broadcasting a zone recruit message. The recruit message lists the nodes in the manager's neighborhood table as potential zone members. A node listens to these recruitments for a certain period of time and records a) all the nodes that have nominated themselves as managers in its neighborhood, and b) all those managers that have recruited it as their zone member. The recruitment message serves two purposes. First, it allows the non-manager nodes to decide which zone they should join. Second, a manager node can snoop on the recruit messages broadcast in its neighborhood and build its list of neighboring managers. The next step in zone partitioning is to ensure that the zones are disjoint by allowing every non-managerial node to join only one of the zones in which it has been recruited.

C. Zone Formation

After listening to the recruitment messages for a certain period of time, a node is in one of four states: it may have been recruited as a zone member in a single zone, multiple zones, none of the zones, or it may have nominated itself as a manager.

[1]On account of the unreliability of the underlying communication channels, we can only provide probabilistic guarantees for the zone membership.

If a non-manager has been recruited by only a single manager, then it simply acknowledges by sending that manager a zone join message. A non-manager node that has heard recruitment messages from multiple managers, joins the zone of the manager that has recruited the least number of nodes and whose recruitment message had good signal strength. The former criterion tries to distribute the availability across the zones, while the latter tries to ensure good connectivity between a manager and its zone members. The manager nodes snoop on the zone join messages broadcast in their neighborhood and eliminate their potential zone members that have joined other zones. Nodes that have been recruited in multiple zones can serve as bridge nodes between two zones. If a manager does not receive enough join responses to meet the minimum availability requirement mentioned earlier in Section III, then the zone members can regroup and join an appropriate adjacent zone. Alternatively, the manager can meet the required degree of availability by soliciting nodes from nearby zones that have more zone members than required.

At the end of the above two-phase message exchange in which the recruitment messages are acknowledged by join messages, every node in the network is either a zone manager or is a non-managerial member of a single zone. This two-phase exchange also ensures that the communication between a zone manager and its members does not suffer from asymmetric effects [14]. A manager uses the knowledge of its zone members and their residual energies to provide energy-efficient storage, as we now describe.

D. Power Management

Every manager assigns a rank to each of its zone members based on their residual energy. The rank of a member in a zone determines its sleep schedule. Nodes with lower residual energy are assigned higher ranks and sleep for a longer duration of time. Thus the zone members form a hierarchy. A manager broadcasts the ranks to its zone members in a zone confirmation message. This message confirms the membership of a node in a zone and allows the zone members to begin their power management schedule according to their ranks. In addition, since a manager broadcasts this message in its communication range, it allows all the managers to finalize their list of neighboring managers.

Finally, in order to route messages from zones that detect an event (source) to the primary storage zone corresponding to that event (sink), every zone needs to be reachable from every other zone. This can be done by creating a common communication backbone that connects all the zone managers (for e.g., as done in [1]).

In order to balance the energy load, the zone managers need to be rotated. This can be done whenever the data stored in the zones is flushed or when the data is uploaded to a base station that becomes accessible to the sensor network. During each rotation, the phase transition process depicted in Figure 2 is repeated. This process also helps the nodes to resynchronize their local clocks. If the duration between successive rotations is too long, the zone members can synchronize their local clocks with their zone manager's clock periodically, in order to ensure that the clock skew is bounded.

IV. In-Network Event Correlation and Storage

In this section, we describe how the resources within a zone can be collectively used to store and correlate events that occur across the network. When the nodes in a zone detect an event, they send the information they collect to their zone manager. The manager uses a hash function to map the event to its primary storage zone (PSZ) and routes the information it has collected to the PSZ for that event. If the event is mobile, this process is repeated by each zone that detects the event, as the event propagates through the network. The primary storage manager (PSM) receives reports about the event from different zones and uses the resources of its zone members to store and correlate information pertaining to the event in real-time. The information stored in the zones can be retrieved by connecting a base station to some point in the network. The zone managers form a spanning tree to connect to the base station on demand and upload the information from their respective zones. We now describe the mapping process, in-network event correlation, and organization of information within a storage zone.

A. Mapping Events to Zones

RESTORE maps every event to a storage zone using a hash function. Since a storage zone is represented by its zone manager, the hash function effectively maps an event to one of the zone managers. RESTORE can be used with any suitable hash function. For example, if nodes are identified by their geographic coordinates, then a geographic hashing scheme [15] may be used. In order to map an event or object to a zone, RESTORE makes use of a unique identity for each object. This identification may be obtained from RFID tags, wherever tagging is possible, or from application-specific signatures gathered from other types of sensors. While RESTORE does not focus on how the identification is obtained, the type of identification influences the granularity of the mapping. For example, if all the vehicles have RFID tags that allow different brands of the same type of vehicle to be distinguished, then it is possible for RESTORE to assign a different zone to keep track of the traffic statistics of each brand of car along a highway. However, if such fine-grained distinction is not possible, then RESTORE has to resort to a coarser level of mapping, such as that induced by the vehicle size. In such a case, all car-related statistics would be stored in one zone, truck-related statistics would be in another zone, and SUV-related statistics would be in a third zone.

B. Real-time Event Correlation in a Zone

Many of the sensor-based information can be correlated and notifications generated using simple operators, such as max, min, average, and sum. Such simple operations are well within the capability of low-end sensor nodes, such as motes, which makes it possible to carry out event correlation within the network. Table I presents a taxonomy of some of the events

TABLE I

EVENT TAXONOMY

	Static	**Mobile**
Single scope	temperature monitoring, WSN debugging, parking violations	tracking small vehicles, patient behavior
Adjacent scope	energy monitoring in a building floor	tracking large trucks
Diffuse scope		fire, pipeline cracks, chemical spills

that we can correlate using RESTORE. Each row in the table classifies events based on their scope during their lifetime with respect to a zone in RESTORE, while the columns classify events based on their mobility with respect to the zones. In RESTORE, static events are observed by the same zone or set of zones for their entire lifetime, while mobile events are observed by different zones at different points in time. We now explain this taxonomy with examples.

Single Scope: Each instance of a single-scoped event occurs only in a single zone at any given time. As a result, a PSZ for a single-scoped event receives reports about each individual instance of the event from at most one zone at any given time.

A single-scoped event may be static, in which case the scope of the event is confined to the same zone for the lifetime of the event. An example of a static, single zonal event is a group of sensors monitoring the temperature in their vicinity. Another example is the sensor network debugging application listed in Section II, in which groups of sensors collect statistics locally for later introspection. Monitoring the parking violations in different zones, which was also described in Section II, is also a static, single-scoped event. Each occurrence of a parking violation is reported by only one zone at any instant of time. Different instances of parking violations may be detected at the same time or at different times in multiple zones. However, the violations that are detected within a zone neither propagate to other zones, nor are they typically related to the violations reported by other zones. Hence, the events in this case are static and have single scope. In the case of static, single-scoped events, the PSM can sequentially order the information it receives from the zones based on time.

Single-scoped events may also be mobile, in which case we assume that they typically follow continuous trajectories. So the event is tracked by adjacent zones over a continuous period of time. Tracking the movement of a normal-sized car or tracking human motion are examples of mobile, single-scoped events. As the event moves across the zones, only one of the zones detects and reports the event to the PSZ at any given instant of time. In this case, the PSM can sequentially order the information it receives from the zones based on space and time, because the reports from adjacent zones have different timestamps.

Adjacent Scope: Events with adjacent scope always span multiple adjacent zones at the same time. As a result, the PSZ for an adjacent-scoped event receives concurrent reports about each individual instance of the event from different, but adjacent zones. In the case of adjacent, multi-zonal events, the PSM has to correlate the reports from adjacent zones to infer that they relate to the same instance of the event and not to distinct, single-scoped events. This can be enabled by ordering the reports from the zones based on space, time, or a fine-grained identity of the object, if that is available.

Adjacent, multi-zonal events may be static, in which case the same set of adjacent zones report the event throughout the duration of the event. Monitoring the energy consumption in a building floor with multiple rooms, wherein every room provides the zonal perspective is an example of a static, adjacent-scoped event.

Adjacent, multi-zonal events may be mobile, in which case the PSM receives reports from different sets of adjacent zones at different intervals of time. Tracking the movement of a large truck that spans multiple zones is an example of a mobile, adjacent-scoped event.

Diffuse Scope: Diffuse events differ from single and adjacent-scoped events in that their scope varies with time. An instance of a diffuse event may initially be reported by a single zone, thereby making it a single-scoped event. Alternatively, different instances of the event may be reported by different single zones. However, the event may propagate in time to other adjacent zones and extend its scope from being a single-scoped event to becoming an adjacent, multi-zonal event. Similarly, an event that begins as an adjacent-scoped multi-zonal event may fragment over time to multiple, single-scoped events. Chemical spills, fire, cracks along a pipeline, and seismic activity along the fault lines in an earthquake prone area are examples of diffuse events. In this case, the PSM receives reports from different zones that may or may not be adjacent as the event progresses. Moreover, these reports may be concurrent or distributed in time. Due to the time-varying pattern of information, the PSM needs to store and correlate the information to determine if the reports are causally related.

C. *Information Organization in a Zone*

When a primary storage zone for an event receives the observations from the zones across the network or from its own zone members (in the case of static, single zonal events), the primary storage manager has to decide how to use the cooperative resources within its zone to store the information so as to enable real-time correlation. We now present some ways in which information can be organized within a zone for the event taxonomy presented in Table I. Each primary storage manager in the network chooses the organization depending on the event that is mapped to its zone.

Temporal Ordering: A PSM can use the resources of its zone members to store information related to an event in temporal order of the occurrence of the event. This scheme assumes that the clock skew in the sensor observations is bounded within an acceptable threshold. In this scheme, every zone member stores information related to an event over a specific time interval. A zone member needs to be awakened only when its corresponding time interval is active. RESTORE uses this organization for most static events. For instance, in the temperature monitoring application, the PSM uses a different zone member to store the temperature observations reported by the sensors during each 24-hour period. Similarly, images of driving violations captured at different intervals by camera sensors at a traffic intersection are temporally partitioned across the nodes in a zone. In each case, temporal ordering allows new observations to be easily correlated, as and when they arrive, with past events that have been stored. This can then be used to publish notifications about time-based events, as illustrated by the following examples:

- the average temperature in the boiler room for the past 1 hour has been greater than 90 degrees. (single scope, static)
- the building occupancy reported by different offices on the 5th floor of an office building has been less than 10% for the past 1 hour. (adjacent scope, static)

Spatial Ordering: Information in a PSZ can be ordered spatially, based on the zone that reported the event. RESTORE uses this approach for some mobile events. For example, when a PSM receives reports about the presence of a vehicle either from a single zone or concurrently from multiple zones, it uses the resources of different zone members to store the reports received from different zones. The information within the same zone member is ordered temporally. The PSM needs to awaken a zone member only when the incoming report is from a spatial region associated with that zone member. This organization may be used to correlate events within the same zone temporally or to causally relate events across adjacent zones and publish events of interest, as illustrated by the following examples:

- in a surveillance scenario, notify an abnormal activity when more than 10 enemy tanks are detected to be advancing towards a target. (adjacent scope, mobile)
- publish an alert when at least 5 different snipers have been detected in a region in the last hour and they are at most 10 meters from each other. (single scope, mobile)

Identity-based Ordering: Whenever it is possible to distinguish between different instances of the same event, for example using RFID or other means, the storage managers in RESTORE store information related to different instances in different zone members. Information stored in the same zone member may be ordered spatially or temporally. When new reports arrive, only the zone member that is associated with that particular instance of the event needs to be awakened. For example, in the application for monitoring the behavior of autistic patients in a facility, the storage manager stores the sensor observations related to each patient in a different zone member. Within the same zone member, the information is temporally ordered. When new observations about a patient arrive, the storage manager awakens the appropriate zone member and correlates the new readings with the stored data. This correlation can be used to generate a notification, if there is a deviation in the behavioral pattern or if there is an absence of an event for a patient at the expected interval of time.

Multi-resolution Storage: The storage managers can use the hierarchical organization of their zone members to store information at different resolutions. For example, in the WSN debugging application, some parameters are measured more frequently than others. The more frequently reported observations may be stored in zone members that are lower ranked and hence, wake up more frequently, while higher-ranked nodes that wake up less often may be used to store the parameters that are measured less frequently.

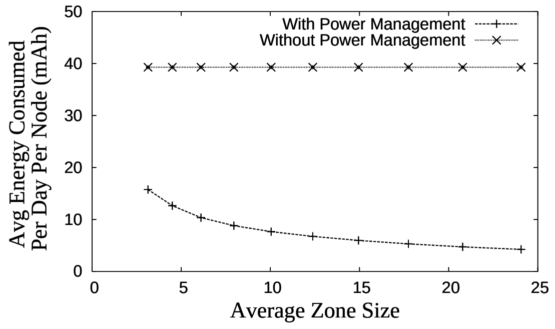


Fig. 3. Variation of energy consumption with zone size

Replication: When information needs to be stored reliably, the PSM can replicate information across the nodes in its zone. Replication can be combined with any of the above schemes to provide reliability. For example, replication is useful in the case of the parking violation application, where loss of information results in loss of revenue. It is also useful in some surveillance applications, where it is hard to reconstruct the events when some critical piece of information is lost. The PSM can either fully replicate the information across all of its zone members or choose the degree of replication depending on the required reliability. The PSM can replicate information in an energy-efficient manner by choosing the zone members having the same ranks as replicas. Since these members have the same sleep schedules, the PSM can download information to all of them simultaneously when they wake up. Thus, nodes that have the same ranks store consistent information.

While replication within a zone provides fault tolerance for independent failures occurring within a zone, it does not handle the case of spatially correlated failures in which a subset of the zone members may fail simultaneously (for e.g. when a truck runs over a section of motes). One way to handle such correlated failures is to use a buddy zone approach in which each event is mapped to a pair of spatially distributed zones. Both of the buddies store the same information. The buddy zone approach trades energy for increased reliability. We have currently not implemented this scheme in RESTORE, because we do not consider spatially correlated failures.

V. Performance Evaluation

The collaborative schemes in RESTORE impact the energy consumption, information availability, and message overhead. We have evaluated these parameters for the replication scheme described in Section IV-C, using a simulator program that we have written. In our experiments, we randomly distributed 10,000 nodes within a 100,000 m^2 rectangular area. We repeated each experiment 30 times with different random

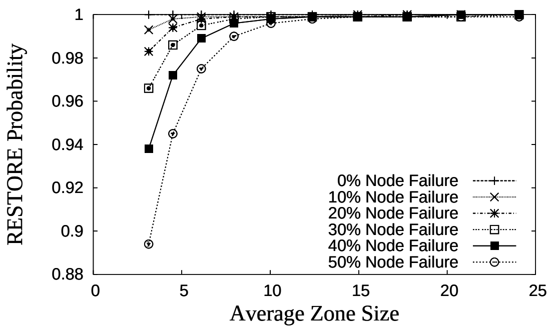


Fig. 4. Variation of information availability with zone size

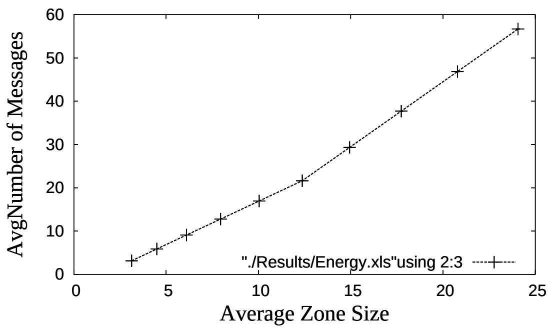


Fig. 5. Variation of message overhead with zone size

number seeds. In this section, we present the results of our experiments.

A. Energy Consumption

In this section, we study the impact of the zone size on the average energy consumption within the network. The zone size is defined as the number of nodes within a zone. To provide sufficient coverage, a zone manager is always active until the next rotation. To ensure reliability, we use the power management scheme in RESTORE to ensure that at least one of the member nodes, in addition to the zone manager, is active at any instant of time. Figure 3 shows that with a larger zone, more energy can be saved by turning off more zone members. For example, when the zone size increases from 5 to 25, we see that the energy consumption in RESTORE reduces by nearly 60%. In addition, Figure 3 shows that the power management scheme in RESTORE achieves significant energy efficiency and reduces the energy consumption by as much as 90% when the zone size is 25, compared to a scheme without power management.

B. Information Availability

In addition to providing energy efficiency, one of the design goals of RESTORE is to improve information availability within the sensor network. Figure 4 shows how the availability varies for different zone sizes and node failure percentages, when information is stored among the zone members using the full replication scheme. We assume that failure of a sensor node occurs due to power loss or an independent hardware fault. The y-axis plots the RESTORE-probability, which is the probability to restore information within a zone in the presence of failures, and is therefore a measure of information availability. It is trivial to conclude that if the percentage of node failures is zero, then RESTORE guarantees that information is always available, regardless of the zone size. However, when the percentage of node failures increases, a relatively large zone size is needed to restore the events with a very high probability. A larger zone size provides greater redundancy and thereby, increases the chances of complete recovery of the stored information. For example, Figure 4 shows that to achieve over 99.9% availability for a failure percentage of 10% and 50%, RESTORE would need a zone size of 7 and 15, respectively.

C. Message Overhead

In our final experiment, we study the impact of the zone size on the message overhead within the network. We specifically consider the message overhead involved in updating the zone member replicas about an event. Figure 5 shows that the overhead increases approximately linearly with the zone size. The reason for the increase is that as the size of a zone increases, there is more variation in the ranks of the zone members. This in turn results in more differentiation among their sleep schedules, as described in Section III-D. Such a differentiated schedule provides higher reliability by increasing the likelihood that at least one zone member, in addition to the manager will be awake at any given time. However, since different replicas awake at different times, the cost of updating the replicas also increases with the zone size. Thus, Figure 4 and Figure 5 reveal a tradeoff between information availability and message overhead. A higher availability is achieved at the cost of incurring higher message overhead.

VI. Related Work

We now compare and contrast RESTORE with some of the related efforts that have addressed the issue of data storage within sensor networks. The cluster-based collaborative storage mechanism presented in [16] organizes the network into clusters in order to store data. It primarily targets applications that do not need to access the in-network data in real-time. However, there is no description of how data is stored within each cluster. In contrast, RESTORE uses different mechanisms to organize data within each zone and in addition, uses this data to correlate events within the network in real-time. The data-centric storage (DCS) scheme [15] stores data by mapping the sensor data to a node in the network using geographic hashing. It focuses more on optimizing the routing of queries to the appropriate storage node in the network, by taking advantage of the routing features of GPSR. RESTORE is complimentary to this scheme in that it focuses more on using the cooperative resources within a sensor network to provide an effective in-network storage mechanism. Geographic hashing is one of many ways in which an event can be mapped to a storage zone in RESTORE. However, if nodes are identified by means other than their locations, then other hashing techniques may be employed in RESTORE. Moreover, RESTORE does not depend on a specific routing scheme, such as GPSR. In the multi-resolution storage mechanism [17], nodes at different hierarchical levels store information at different levels of resolution. Every node has detailed information about the local events, but has only a compressed view of the events witnessed by the nodes that are below it in the hierarchy. This hierarchical organization can be used to optimize query routing. In addition to multi-resolution storage scheme, RESTORE uses other schemes, such as temporal ordering, spatial ordering, and replication to organize information using the collaborative resources of its zone members.

VII. Conclusions

RESTORE is an overall framework that takes advantage of the collaboration among the sensor nodes to provide in-network event correlation and storage service for sensor networks. RESTORE uses a divide and conquer approach in which it partitions the network into zones, in order to store and correlate information related to multiple events. RESTORE can be used to buffer data over a limited period of time in sensor environments with intermittent connectivity to persistent storage. It can also be used as a basis for a publish-subscribe system involving sensor networks and subscribers on traditional networks. The zone partitioning and collaboration mechanism in RESTORE is a tradeoff between different parameters, such as energy consumption, information availability, message overhead, and the number of distinct types of events that can be stored in the network. We have presented some initial results that show how the zone partitioning scheme influences those parameters when information is replicated across a zone. As part of future work, we plan to implement RESTORE using sensor nodes and conduct a more detailed study of how the above parameters influence the performance of RESTORE for the different storage schemes presented in Section IV-C.

References

[1] T. He *et al.*, “An Energy-Efficient Surveillance System for Wireless Sensor Networks,” in *Proc. of MobiSys*, June 2004.

[2] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, “Habitat Monitoring: Application Driver for Wireless Communications Technology,” in *Proc. of the 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, April 2001.

[3] A. Mainwaring, J. Polastre, R. Szewczyk, D. E. Culler, and J. Anderson, “Wireless Sensor Networks for Habitat Monitoring,” in *Proc. of the ACM Workshop on Sensor Networks and Application (WSNA)*, September 2002.

[4] V. Kottapalli, A. Kiremidjian, J. Lynch, E. Carryer, T. Kenny, K. Law, and Y. Lei, “Two-tiered Wireless Sensor Network Architecture for Structural Health Monitoring,” in *Proc. of the Intl. Symp. on Smart Structures and Materials*, March 2003.

[5] Zigbee Alliance, available at http://www.zigbee.org/en/index.asp.

[6] K. Fall, “A Delay-Tolerant Network Architecture for Challenged Internets,” in *Proc. of SIGCOMM*, 2003.

[7] T. Luckenbach, P. Gober, S. Arbanowski, A. Kotsopoulos, and K. Kim, “TinyREST: A Protocol for Integrating Sensor Networks into the Internet,” in *Proc. of REALWSN*, 2005.

[8] *MICA2 Wireless Measurement System*, Crossbow Technologies, http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0042-04_A_MICA2.pdf.

[9] *STARGATE: X-Scale Processor Platform*, Crossbow Technologies, http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/6020-0049-01_A_STARGATE.pdf.

[10] *SkyeRead Mini*, SkyeTek Inc., available at skyetek.com/readers_Mini.html.

[11] *Mica2Dot Series*, Crossbow Inc., available at www.xbow.com/Products/Productsdetails.aspx?sid=73.

[12] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein, “Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet,” in *Proc. of ASPLOS-X*, October 2002.

[13] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, “Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks,” *ACM Wireless Networks*, vol. 8, no. 5, September 2002.

[14] G. Zhou, T. He, S. Krishnamurthy, and J. Stankovic, “Impact of Radio Irregularity on Wireless Sensor Networks,” in *Proc. of MOBISYS*, 2004.

[15] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, “Data-Centric Storage in Sensornets with GHT, A Geographic Hash Table,” in *Proc. of Mobile Networks and Applications (MONET)*, March 2003.

[16] S. Tilak, N. Abu-Ghazaleh, and W. Heinzelman, “Collaborative Storage Management for Sensor Networks,” *Intl. Journal of Ad-Hoc and Ubiquitous Computing*, vol. 1, pp. 47–58, 2005.

[17] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, R. Govindan, and J. Heidemann, “An Evaluation of Multi-Resolution Storage for Sensor Networks,” in *Proc. of SenSys*, November 2003.