HELSINKI UNIVERSITY OF TECHNOLOGY
Networking laboratory

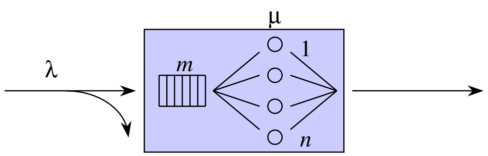
7. Loss systems

Contents

- Refresher: Simple teletraffic model
- Poisson model (∞ customers, ∞ servers)
- Erlang model (∞ customers, $n < \infty$ servers)
- Binomial model ($k < \infty$ customers, $n = k$ servers)
- Engset model ($k < \infty$ customers, $n < k$ servers)

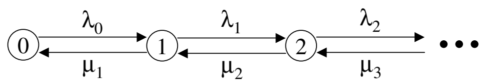
Simple teletraffic model

- **Customers arrive** at a rate λ (customers per time unit)
  - $1/\lambda$ = average inter-arrival time
- Customers are **served** by n parallel **servers**
- When busy, a server serves at a rate μ (customers per time unit)
  - $1/\mu$ = average service time of a customer
- There are m **waiting** places
- It is assumed that blocked customers (arriving to a full system) are lost



Birth-death process

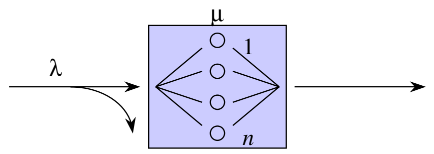
- State transition diagram of an infinite-state irreducible BD process :



- Intensities $\lambda_i > 0$ and $\mu_i > 0$ depend on the model.
- Here modeling is finding the appropriate parameters.

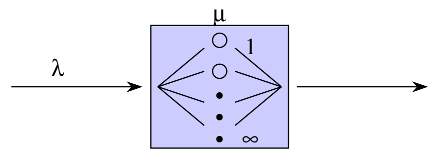
Pure loss system

- No waiting places ($m = 0$)
  - If the system is full (with all n servers occupied) when a customer arrives, she is not served at all but lost
  - Some customers are lost
- From the customer's point of view,
  - it is interesting to know e.g. the blocking probability
- Note: In addition to the case where the arrival rate λ is constant, we will consider the case where it, λ_i, depends on the state of the system i.



Infinite system

- Infinite number of servers ($n = \infty$)
  - No customers are lost nor do they even have to wait before getting served
- **Note**: Also here, in addition to the case where the arrival rate λ is constant, we will consider the case where it, λ_i, depends on the state of the system i.



Blocking

- In a loss system some calls are lost
  - a call is lost if all n channels are occupied when the call arrives
  - the term **blocking** refers to this event
- There are (at least) two different types of blocking quantities:
  - **Call blocking** B_c = probability that an arriving call finds all n channels occupied = the fraction of calls that are lost
  - **Time blocking** B_t = probability that all n channels are occupied at an arbitrary time = the fraction of time that all n channels are occupied
- The two blocking quantities are not necessarily equal
  - If calls arrive according to a Poisson process, then $B_c = B_t$
- Call blocking is a better measure for the quality of service experienced by the subscribers but, typically, time blocking is easier to calculate

Contents

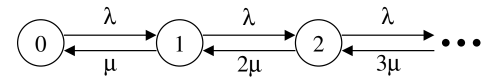
- Refresher: Simple teletraffic model
- Poisson model (∞ customers, ∞ servers)
- Erlang model (∞ customers, $n < \infty$ servers)
- Binomial model ($k < \infty$ customers, $n = k$ servers)
- Engset model ($k < \infty$ customers, $n < k$ servers)

Poisson model (M/M/∞)

- **Definition**: **Poisson model** is the following simple teletraffic model:
  - Infinite number of independent customers ($k = \infty$)
  - Interarrival times are IID and exponentially distributed with mean $1/\lambda$
    - so, customers arrive according to a Poisson process with intensity λ
  - Infinite number of servers ($n = \infty$)
  - Service times are IID and exponentially distributed with mean $1/\mu$
  - No waiting places ($m = 0$)
- Poisson model:
  - Using Kendall's notation, this is an M/M/∞ queue
  - Infinite system, and, thus, **lossless**
- Notation:
  - $a = \lambda/\mu$ = traffic intensity

State transition diagram

- Let $X(t)$ denote the number of customers in the system at time t
  - Assume that $X(t) = i$ at some time t, and consider what happens during a short time interval $(t, t+h]$:
    - with prob. $\lambda h + o(h)$, a new customer arrives (state transition $i \to i+1$)
    - if $i > 0$, then, with prob. $i\mu h + o(h)$, a customer leaves the system (state transition $i \to i-1$)
- Process $X(t)$ is clearly a Markov process with state transition diagram



- **Note** that process $X(t)$ is an irreducible birth-death process with an infinite state space $S = \{0,1,2,\ldots\}$, $\lambda_i = \lambda$, $\mu_i = i\mu$

Equilibrium distribution (1)

- Local balance equations (LBE):

$$\pi_i \lambda = \pi_{i+1}(i+1)\mu \qquad \text{(LBE)}$$

$$\Rightarrow \quad \pi_{i+1} = \frac{\lambda}{(i+1)\mu}\pi_i = \frac{a}{i+1}\pi_i$$

$$\Rightarrow \quad \pi_i = \frac{a^i}{i!}\pi_0, \quad i = 0,1,2,\ldots$$

- Normalizing condition (N):

$$\sum_{i=0}^{\infty}\pi_i = \pi_0 \sum_{i=0}^{\infty}\frac{a^i}{i!} = 1 \qquad \text{(N)}$$

$$\Rightarrow \quad \pi_0 = \left(\sum_{i=0}^{\infty}\frac{a^i}{i!}\right)^{-1} = \left(e^a\right)^{-1} = e^{-a}$$

Equilibrium distribution (2)

- Thus, the equilibrium distribution is a **Poisson distribution**:

$$X \sim \text{Poisson}(a)$$

$$P\{X = i\} = \pi_i = \frac{a^i}{i!}e^{-a}, \quad i = 0,1,2,\ldots$$

$$E[X] = a, \quad D^2[X] = a$$

- Remark (insensitivity):
  - The result is **insensitive** to the service time distribution, that is: it is valid for **any** service time distribution with mean $1/\mu$
  - So, instead of the M/M/∞ model, we can consider, as well, the more general M/G/∞ model

Contents

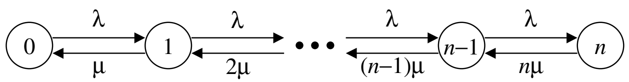
- Refresher: Simple teletraffic model
- Poisson model (∞ customers, ∞ servers)
- Erlang model (∞ customers, $n < \infty$ servers)
- Binomial model ($k < \infty$ customers, $n = k$ servers)
- Engset model ($k < \infty$ customers, $n < k$ servers)

Erlang model (M/M/n/n)

- **Definition**: **Erlang model** is the following simple teletraffic model:
  - Infinite number of independent customers ($k = \infty$)
  - Interarrival times are IID and exponentially distributed with mean $1/\lambda$
    - so, customers arrive according to a Poisson process with intensity λ
  - Finite number of servers ($n < \infty$)
  - Service times are IID and exponentially distributed with mean $1/\mu$
  - No waiting places ($m = 0$)
- Erlang model:
  - Using Kendall's notation, this is an M/M/n/n queue
  - Pure loss system, and, thus, **lossy**
- Notation:
  - $a = \lambda/\mu$ = traffic intensity

State transition diagram

- Let $X(t)$ denote the number of customers in the system at time t
  - Assume that $X(t) = i$ at some time t, and consider what happens during a short time interval $(t, t+h]$:
    - with prob. $\lambda h + o(h)$, a new customer arrives (state transition $i \to i+1$)
    - with prob. $i\mu h + o(h)$, a customer leaves the system (state transition $i \to i-1$)
- Process $X(t)$ is clearly a Markov process with state transition diagram



- **Note** that process $X(t)$ is an irreducible birth-death process with a finite state space $S = \{0,1,2,\ldots,n\}$, $\lambda_i = \lambda$, $\mu_i = i\mu$

Equilibrium distribution (1)

- Local balance equations (LBE):

$$\pi_i \lambda = \pi_{i+1}(i+1)\mu \qquad \text{(LBE)}$$

$$\Rightarrow \quad \pi_{i+1} = \frac{\lambda}{(i+1)\mu}\pi_i = \frac{a}{i+1}\pi_i$$

$$\Rightarrow \quad \pi_i = \frac{a^i}{i!}\pi_0, \quad i = 0,1,\ldots,n$$

- Normalizing condition (N):

$$\sum_{i=0}^{n} \pi_i = \pi_0 \sum_{i=0}^{n} \frac{a^i}{i!} = 1 \qquad \text{(N)}$$

$$\Rightarrow \quad \pi_0 = \left(\sum_{i=0}^{n} \frac{a^i}{i!} \right)^{-1}$$

Equilibrium distribution (2)

- Thus, the equilibrium distribution is a **truncated Poisson distribution**:

$$P\{X=i\}=\pi_i=\frac{\frac{a^i}{i!}}{\sum_{j=0}^{n}\frac{a^j}{j!}},\quad i=0,1,\ldots,n$$

- Remark (insensitivity):
  - The result is **insensitive** to the service time distribution, that is: it is valid for **any** service time distribution with mean $1/\mu$
  - So, instead of the M/M/n/n model, we can consider, as well, the more general M/G/n/n model

Time blocking

- **Time blocking** B_t = probability that all n servers are occupied at an arbitrary time = the fraction of time that all n servers are occupied
- For a stationary Markov process, this equals the probability π_n of the equilibrium distribution π. Thus,

$$B_t := P\{X=n\}=\pi_n=\frac{\frac{a^n}{n!}}{\sum_{j=0}^{n}\frac{a^j}{j!}}$$

Call blocking

- **Call blocking** B_c = probability that an arriving customer finds all n servers occupied = the fraction of arriving customers that are lost
- However, due to Poisson arrivals and PASTA property, the probability that an arriving customer finds all n servers occupied equals the probability that all n servers are occupied at an arbitrary time,
- In other words, call blocking B_c equals time blocking B_t:

$$B_c = B_t = \frac{\frac{a^n}{n!}}{\sum_{j=0}^{n}\frac{a^j}{j!}}$$

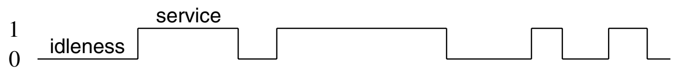
- This is **Erlang's blocking formula**, introduced earlier

Contents

- Refresher: Simple teletraffic model
- Poisson model (∞ customers, ∞ servers)
- Erlang model (∞ customers, $n < \infty$ servers)
- Binomial model ($k < \infty$ customers, $n = k$ servers)
- Engset model ($k < \infty$ customers, $n < k$ servers)

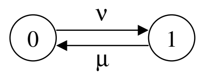
Binomial model (M/M/k/k/k)

- **Definition**: **Binomial model** is the following (simple) teletraffic model:
  - Finite number of independent customers ($k < \infty$)
    - **on-off type** customers (alternating between idleness and activity)
  - Idle times are IID and exponentially distributed with mean $1/\nu$
  - As many servers as customers ($n = k$)
  - Service times are IID and exponentially distributed with mean $1/\mu$
  - No waiting places ($m = 0$)
- Binomial model:
  - Using Kendall's notation, this is an M/M/k/k/k queue
  - Although a finite system, this is clearly **lossless**
- On-off type customer (cf. lecture 4 slide 17):



On-off type customer (1)

- Let $X_j(t)$ denote the state of customer j ($j = 1,2,\ldots,k$) at time t
  - State 0 = idle, state 1 = active = in service
  - Consider what happens during a short time interval $(t, t+h]$:
    - if $X_j(t) = 0$, then, with prob. $\nu h + o(h)$, the customer becomes active (state transition $0 \to 1$)
    - if $X_j(t) = 1$, then, with prob. $\mu h + o(h)$, the customer becomes idle (state transition $1 \to 0$)
- Process $X_j(t)$ is clearly a Markov process with state transition diagram



- **Note** that process $X_j(t)$ is an irreducible birth-death process with a finite state space $S = \{0,1\}$

On-off type customer (2)

- Local balance equations (LBE):

$$\pi_0^{(j)}\nu = \pi_1^{(j)}\mu \;\Rightarrow\; \pi_1^{(j)} = \frac{\nu}{\mu}\pi_0^{(j)}$$

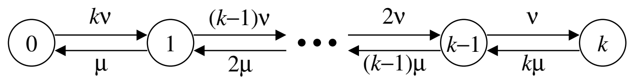
- Normalizing condition (N):

$$\pi_0^{(j)} + \pi_1^{(j)} = \pi_0^{(j)}(1+\frac{\nu}{\mu}) = 1 \;\Rightarrow\; \pi_0^{(j)} = \frac{\mu}{\nu+\mu},\;\; \pi_1^{(j)} = \frac{\nu}{\nu+\mu}$$

- So, the equilibrium distribution of a single customer is the **Bernoulli distribution** with success probability $\nu/(\nu+\mu)$
- From this, we could deduce that the equilibrium distribution of the state of the whole system (that is: the number of active customers) is the binomial distribution $\mathrm{Bin}(k, \nu/(\nu+\mu))$

State transition diagram

- Let $X(t)$ denote the number of active customers
  - Assume that $X(t) = i$ at some time t, and consider what happens during a short time interval $(t, t+h]$:
    - if $i < k$, then, with prob. $(k-i)\nu h + o(h)$, an idle customer becomes active (state transition $i \to i+1$)
    - if $i > 0$, then, with prob. $i\mu h + o(h)$, an active customer becomes idle (state transition $i \to i-1$)
- Process $X(t)$ is clearly a Markov process with state transition diagram



- **Note** that process $X(t)$ is an irreducible birth-death process with a finite state space $S = \{0,1,\ldots,k\}$, $\lambda_i = (k - i)\lambda$, $\mu_i = i\mu$

Equilibrium distribution (1)

- Local balance equations (LBE):

$$\pi_i(k-i)\nu = \pi_{i+1}(i+1)\mu \qquad \text{(LBE)}$$

$$\Rightarrow \quad \pi_{i+1} = \frac{(k-i)\nu}{(i+1)\mu}\pi_i$$

$$\Rightarrow \quad \pi_i = \frac{k!}{i!(k-i)!}\left(\frac{\nu}{\mu}\right)^i \pi_0 = \binom{k}{i}\left(\frac{\nu}{\mu}\right)^i \pi_0, \quad i = 0,1,\ldots,k$$

- Normalizing condition (N):

$$\sum_{i=0}^{k}\pi_i = \pi_0 \sum_{i=0}^{k}\binom{k}{i}\left(\frac{\nu}{\mu}\right)^i = 1 \qquad \text{(N)}$$

$$\Rightarrow \quad \pi_0 = \left(\sum_{i=0}^{k}\binom{k}{i}\left(\frac{\nu}{\mu}\right)^i\right)^{-1} = \left(1+\frac{\nu}{\mu}\right)^{-k} = \left(\frac{\mu}{\nu+\mu}\right)^k$$

Equilibrium distribution (2)

- Thus, the equilibrium distribution is a **binomial distribution**:

$$X \sim \mathrm{Bin}\left(k, \frac{\nu}{\nu+\mu}\right)$$

$$P\{X=i\} = \pi_i = \binom{k}{i}\left(\frac{\nu}{\nu+\mu}\right)^i\left(\frac{\mu}{\nu+\mu}\right)^{k-i}, \quad i = 0,1,\ldots,k$$

$$E[X] = \frac{k\nu}{\nu+\mu}, \quad D^2[X] = k\cdot\frac{\nu}{\nu+\mu}\cdot\frac{\mu}{\nu+\mu} = \frac{k\nu\mu}{(\nu+\mu)^2}$$

- Remark (insensitivity):
  - The result is **insensitive both** to the service **and** the idle time distribution, that is: it is valid for **any** service time distribution with mean $1/\mu$ and **any** idle time distribution with mean $1/\nu$
  - So, instead of the M/M/k/k/k model, we can consider, as well, the more general G/G/k/k/k model

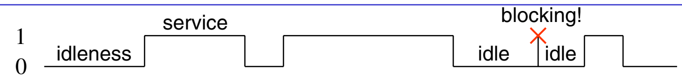
Contents

Engset model (M/M/n/n/k)

- **Definition**: **Engset model** is the following (simple) teletraffic model:
  - Finite number of independent customers ($k < \infty$)
    - **on-off type** customers (alternating between idleness and activity)
  - Idle times are IID and exponentially distributed with mean $1/\nu$
  - Less servers than customers ($n < k$)
  - Service times are IID and exponentially distributed with mean $1/\mu$
  - No waiting places ($m = 0$)
- Engset model:
  - Using Kendall's notation, this is an M/M/n/n/k queue
  - This is a pure loss system, and, thus, **lossy**

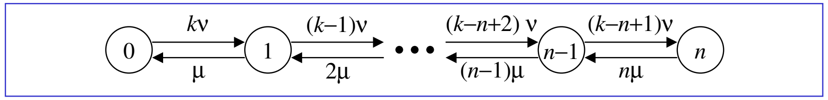
Note: If the system is full when an idle cust. tries to become an active cust., a new idle period starts.

- On-off type customer:



State transition diagram

- Let $X(t)$ denote the number of active customers
  - Assume that $X(t) = i$ at some time t, and consider what happens during a short time interval $(t, t+h]$:
    - if $i < n$, then, with prob. $(k-i)\nu h + o(h)$, an idle customer becomes active (state transition $i \to i+1$)
    - if $i > 0$, then, with prob. $i\mu h + o(h)$, an active customer becomes idle (state transition $i \to i-1$)
- Process $X(t)$ is clearly a Markov process with state transition diagram



- **Note** that process $X(t)$ is an irreducible birth-death process with a finite state space $S = \{0,1,\ldots,n\}$, $\lambda_i = (k - i)\lambda$, $\mu_i = i\mu$

Equilibrium distribution (1)

- Local balance equations (LBE):

$$\pi_i (k-i)\nu = \pi_{i+1}(i+1)\mu \qquad \text{(LBE)}$$

$$\Rightarrow \quad \pi_{i+1} = \frac{(k-i)\nu}{(i+1)\mu}\pi_i$$

$$\Rightarrow \quad \pi_i = \frac{k!}{i!(k-i)!}\left(\frac{\nu}{\mu}\right)^i \pi_0 = \binom{k}{i}\left(\frac{\nu}{\mu}\right)^i \pi_0, \quad i = 0,1,\ldots,n$$

- Normalizing condition (N):

$$\sum_{i=0}^{n} \pi_i = \pi_0 \sum_{i=0}^{n} \binom{k}{i}\left(\frac{\nu}{\mu}\right)^i = 1 \qquad \text{(N)}$$

$$\Rightarrow \quad \pi_0 = \left(\sum_{i=0}^{n} \binom{k}{i}\left(\frac{\nu}{\mu}\right)^i\right)^{-1}$$

Equilibrium distribution (2)

- Thus, the equilibrium distribution is a **truncated binomial distribution**:

$$P\{X = i\} = \pi_i = \frac{\binom{k}{i}\left(\frac{\nu}{\mu}\right)^i}{\sum_{j=0}^{n}\binom{k}{j}\left(\frac{\nu}{\mu}\right)^j} = \frac{\binom{k}{i}\left(\frac{\nu}{\nu+\mu}\right)^i\left(\frac{\mu}{\nu+\mu}\right)^{k-i}}{\sum_{j=0}^{n}\binom{k}{j}\left(\frac{\nu}{\nu+\mu}\right)^j\left(\frac{\mu}{\nu+\mu}\right)^{k-j}}, i = 0,1,\ldots,n$$

- **Remark** (insensitivity):
  - The result is **insensitive both** to the service **and** the idle time distribution, that is: it is valid for **any** service time distribution with mean $1/\mu$ and **any** idle time distribution with mean $1/\nu$
  - So, instead of the M/M/n/n/k model, we can consider, as well, the more general G/G/n/n/k model

Time blocking

- **Time blocking** B_t = probability that all n servers are occupied at an arbitrary time = the fraction of time that all n servers are occupied
- For a stationary Markov process, this equals the probability π_n of the equilibrium distribution π. Thus,

$$B_t := P\{X = n\} = \pi_n = \frac{\binom{k}{n}\left(\frac{\nu}{\mu}\right)^n}{\sum_{j=0}^{n}\binom{k}{j}\left(\frac{\nu}{\mu}\right)^j}$$

Call blocking (1)

- **Call blocking** B_c = probability that an arriving customer finds all n servers occupied = the fraction of arriving customers that are lost
- In the Engset model, however, the "arrivals" do **not** follow a Poisson process. Thus, we cannot utilize the PASTA property any more.
- In fact, the distribution of the state that an "arriving" customer sees differs from the equilibrium distribution.
- Thus, call blocking B_c does **not** equal time blocking B_t in the Engset model.

Call blocking (2)

- Let π_i^* denote the probability that there are i active customers when an idle customer becomes active (which is called an "arrival")
- Consider a long time interval $(0,T)$:
  - During this interval, the average time spent in state i is $\pi_i T$
  - During this time, the average number of "arriving" customers (who all see the system to be in state i) is $(k-i)\nu\cdot\pi_i T$
  - During the whole interval, the average number of "arriving" customers is $\Sigma_j (k-j)\nu\cdot\pi_j T$
- Thus,

$$\pi_i^* = \frac{(k-i)\nu\cdot\pi_i T}{\sum_{j=0}^{n}(k-j)\nu\cdot\pi_j T} = \frac{(k-i)\nu\cdot\pi_i}{\sum_{j=0}^{n}(k-j)\nu\cdot\pi_j}, \quad i=0,1,\ldots,n$$

Call blocking (3)

- It can be shown (exercise!) that

$$\pi_i^* = \frac{\binom{k-1}{i}(\frac{\nu}{\mu})^i}{\sum_{j=0}^{n}\binom{k-1}{j}(\frac{\nu}{\mu})^j}, \quad i=0,1,\ldots,n$$

- If we write explicitly the dependence of these probabilities on the total number of customers, we get the following result:

$$\pi_i^*(k) = \pi_i(k-1), \quad i=0,1,\ldots,n$$

- In other words, an "arriving" customer sees such a system where there is one customer less (himself!) in equilibrium

Call blocking (4)

- By choosing $i = n$, we get the following formula for the call blocking probability:

$$B_c(k) = \pi_n^*(k) = \pi_n(k-1) = B_t(k-1)$$

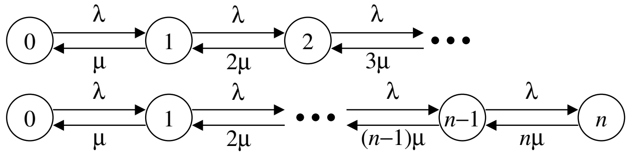
- Thus, for the Engset model, the call blocking in a system with k customers equals the time blocking in a system with $k-1$ customers:

$$B_c(k) = B_t(k-1) = \frac{\binom{k-1}{n}(\frac{\nu}{\mu})^n}{\sum_{j=0}^{n}\binom{k-1}{j}(\frac{\nu}{\mu})^j}$$

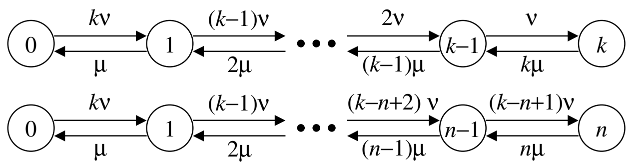
- This is **Engset's blocking formula**

Summary

- Transition diagram if infinite number of customers
  - eq. dist of X is a **Poisson-process** or **truncated Poisson-process**



- Transition diagram if finite number of customers, $k > n$
  - eq. dist of X is a **binomial distribution** or **truncated binomial distribution**



HELSINKI UNIVERSITY OF TECHNOLOGY
Networking laboratory

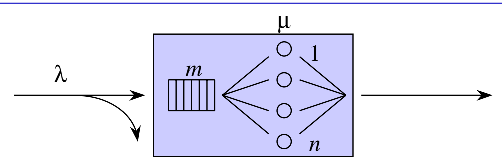
8. Queueing systems

Contents

- Refresher: Simple teletraffic model
- M/M/1 (1 server, ∞ waiting places)
- M/M/n (n servers, ∞ waiting places)

Simple teletraffic model

- **Customers arrive** at rate λ (customers per time unit)
  - $1/\lambda$ = average inter-arrival time
- Customers are **served** by n parallel **servers**
- When busy, a server serves at rate μ (customers per time unit)
  - $1/\mu$ = average service time of a customer
- There are m **waiting** places

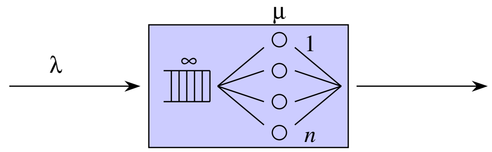


Refresher

- In loss systems there are no waiting places ($m = 0$)
  - We have considered $n < \infty$ and $n = \infty$
  - Also the number of customers was finite or infinite
- In queueing systems ($m = \infty$)
  - We consider $n < \infty$
  - We assume an infinite number of customers i.e. Poisson arrivals

Pure waiting system

- Infinite number of waiting places ($m = \infty$)
  - If all n servers are occupied when a customer arrives, she occupies one of the waiting places
  - No customers are lost but some of them have to wait before getting served
- From the customer's point of view, it is interesting to know e.g.
  - what is the probability that she has to wait "too long"?



Queueing discipline

- Consider a single server ($n = 1$) queueing system
- **Queueing discipline** determines the way the server serves the customers
  - Are the customers served one-by-one or simultaneously
  - If the customers are served one-by-one,
    - in which order are they taken into the service
  - And if the customers are served simultaneously,
    - how is the service capacity shared among them
- A queueing discipline is called **work-conserving** if customers are served with full service rate μ whenever the system is non-empty

Various work-conserving queueing disciplines

- First In First Out (FIFO) = First Come First Served (FCFS)
  - the most ordinary queueing discipline ("queue")
  - customers served one-by-one (with full service rate μ)
  - always serve the customer that has been waiting for the longest time
- Last In First Out (LIFO) = Last Come First Served (LCFS)
  - "stack"
  - customers served one-by-one (with full service rate μ)
  - always serve the customer that has been waiting for the shortest time
- Processor Sharing (PS)
  - "fair queueing"
  - customers served simultaneously
  - when i customers in the system, each of them served with equal rate μ/i

Contents

- Refresher: Simple teletraffic model
- M/M/1 (1 server, ∞ waiting places)
- M/M/n (n servers, ∞ waiting places)

M/M/1 queue

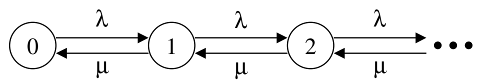
- Consider the following simple teletraffic model:
  - Infinite number of independent customers ($k = \infty$)
  - Interarrival times are IID and exponentially distributed with mean $1/\lambda$
    - so, customers arrive according to a Poisson process with intensity λ
  - One server ($n = 1$)
  - Service times are IID and exponentially distributed with mean $1/\mu$
  - Infinite number of waiting places ($m = \infty$)
  - Default queueing discipline: FIFO
- Using Kendall's notation, this is an **M/M/1 queue**
  - more precisely: M/M/1-FIFO queue
- Notation:
  - $\rho = \lambda/\mu$ = traffic load

Interesting random variables

- X = number of customers in the system at an arbitrary time
  = queue length in equilibrium
- X^* = number of customers in the system at an (typical) arrival time
  = queue length seen by an arriving customer
- W = waiting time of a (typical) customer
- S = service time of a (typical) customer
- $D = W + S$ = total time in the system of a (typical) customer = delay

State transition diagram

- Let $X(t)$ denote the number of customers in the system at time t
  - Assume that $X(t) = i$ at some time t, and
    consider what happens during a short time interval $(t, t+h]$:
    - with prob. $\lambda h + o(h)$,
      a new customer arrives (state transition $i \rightarrow i+1$)
    - if $i > 0$, then, with prob. $\mu h + o(h)$,
      a customer leaves the system (state transition $i \rightarrow i-1$)
- Process $X(t)$ is clearly a Markov process with state transition diagram



- Note that process $X(t)$ is an irreducible birth-death process
  with an infinite state space $S = \{0,1,2,...\}$, $\lambda_i = \lambda$, $\mu_i = \mu$ (only 1 server)

Equilibrium distribution (1)

- Local balance equations (LBE):

$$\pi_i \lambda = \pi_{i+1}\mu \qquad \text{(LBE)}$$
$$\Rightarrow \quad \pi_{i+1} = \frac{\lambda}{\mu}\pi_i = \rho\pi_i$$
$$\Rightarrow \quad \pi_i = \rho^i \pi_0, \quad i = 0,1,2,\ldots$$

- Normalizing condition (N):

$$\sum_{i=0}^{\infty} \pi_i = \pi_0 \sum_{i=0}^{\infty} \rho^i = 1 \qquad \text{(N)}$$
$$\Rightarrow \quad \pi_0 = \left(\sum_{i=0}^{\infty} \rho^i\right)^{-1} = \left(\frac{1}{1-\rho}\right)^{-1} = 1-\rho, \quad \text{if } \rho < 1$$

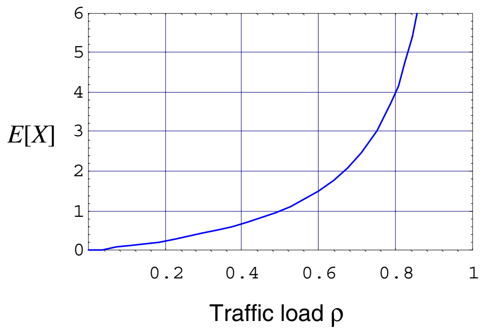
Equilibrium distribution (2)

- Thus, for a **stable** system ($\rho < 1$), the equilibrium distribution exists and is a **geometric distribution**:

$$\rho < 1 \quad \Rightarrow \quad X \sim \mathrm{Geom}(\rho)$$
$$P\{X = i\} = \pi_i = (1-\rho)\rho^i, \quad i = 0,1,2,\ldots$$
$$E[X] = \frac{\rho}{1-\rho}, \quad D^2[X] = \frac{\rho}{(1-\rho)^2}$$

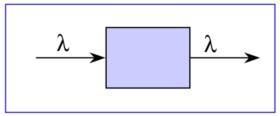
- Remarks:
  - This result is valid for any **work-conserving** queueing discipline
    - FIFO, LIFO, PS, ...
  - This result is **not** insensitive to the service time distribution as far as the FIFO queueing discipline is concerned
  - However, for any **symmetric** queueing discipline (such as LIFO or PS) the result is, indeed, insensitive to the service time distribution

Mean queue length $E[X]$ vs. traffic load ρ



Recall Little's formula

- Consider a system where
  - new customers arrive at a rate λ
- Assume **stability**:
  - Every now and then, the system is empty
  - customers thus leave the system at a rate λ



- **Little's formula**:

$$\bar{N} = \text{average nr of customers in the system}$$
$$\bar{T} = \text{average time a customer spends in the system}$$

$$\bar{N} = \lambda \bar{T}$$

Very useful formula: does not require PASTA property, works for all STABLE systems

Mean delay

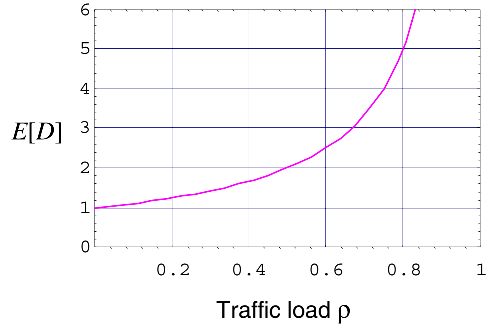
- Let D denote the total time (delay) in the system of a (typical) customer
  - including both the waiting time W and the service time S: $D = W + S$
- Little's formula: $E[X] = \lambda \cdot E[D]$. Thus,

$$E[D] = \frac{E[X]}{\lambda} = \frac{1}{\lambda} \cdot \frac{\rho}{1-\rho} = \frac{1}{\mu} \cdot \frac{1}{1-\rho} = \frac{1}{\mu - \lambda}$$

- Remarks:
  - The mean delay is the same for all **work-conserving** queueing disciplines
    - FIFO, LIFO, PS, ...
  - But the variance and other moments are different!

Mean delay $E[D]$ vs. traffic load ρ

- Note. Delay in units of mean service time



Mean waiting time

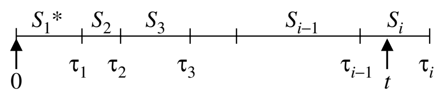
- Let W denote the waiting time of a (typical) customer
- Since $W = D - S$, we have

$$E[W] = E[D] - E[S] = \frac{1}{\mu} \cdot \frac{1}{1-\rho} - \frac{1}{\mu} = \frac{1}{\mu} \cdot \frac{\rho}{1-\rho}$$

- Remarks:
  - The mean waiting time is the same for all **work-conserving** queueing disciplines
    - FIFO, LIFO, PS, ...
  - But the variance and other moments are different!

Waiting time distribution (1)

- Let W denote the waiting time of a (typical) customer
- Let X^* denote the number of customers in the system at the arrival time
- PASTA: $P\{X^* = i\} = P\{X = i\} = \pi_i$.
- Assume now, for a while, that $X^* = i$
  - Service times $S_2, \ldots, S_i$ of the waiting customers are IID and $\sim \mathrm{Exp}(\mu)$
  - Due to the memoryless property of the exponential distribution, the **remaining** service time S_1^* of the customer in service also follows $\mathrm{Exp}(\mu)$-distribution (and is independent of everything else)
  - Due to the FIFO queueing discipline, $W = S_1^* + S_2 + \ldots + S_i$
  - Construct a Poisson (point) process τ_n by defining $\tau_1 = S_1^*$ and $\tau_n = S_1^* + S_2 + \ldots + S_n$, $n \geq 2$. Now (since $X^* = i$): $W > t \Leftrightarrow \tau_i > t$



Waiting time distribution (2)

- Since $W = 0 \Leftrightarrow X^* = 0$, we have

$$P\{W = 0\} = P\{X^* = 0\} = \pi_0 = 1 - \rho$$

$$P\{W > t\} = \sum_{i=1}^{\infty} P\{W > t \mid X^* = i\} P\{X^* = i\}$$

$$= \sum_{i=1}^{\infty} P\{\tau_i > t\} \pi_i = \sum_{i=1}^{\infty} P\{\tau_i > t\}(1-\rho)\rho^i$$

- Denote by $A(t)$ the Poisson (counter) process corresponding to τ_n
  - It follows that: $\tau_i > t \Leftrightarrow A(t) \le i\text{-}1$
  - On the other hand, we know that $A(t) \sim \text{Poisson}(\mu t)$. Thus,

$$P\{\tau_i > t\} = P\{A(t) \le i-1\} = \sum_{j=0}^{i-1} \frac{(\mu t)^j}{j!} e^{-\mu t}$$

Waiting time distribution (3)

- By combining the previous formulas, we get

$$P\{W > t\} = \sum_{i=1}^{\infty} P\{\tau_i > t\}(1-\rho)\rho^i$$

$$= \sum_{i=1}^{\infty} \sum_{j=0}^{i-1} \frac{(\mu t)^j}{j!} e^{-\mu t} (1-\rho)\rho^i$$

$$= \rho \sum_{j=0}^{\infty} \frac{(\mu t \rho)^j}{j!} e^{-\mu t} (1-\rho) \sum_{i=j+1}^{\infty} \rho^{i-(j+1)}$$

$$= \rho \sum_{j=0}^{\infty} \frac{(\mu t \rho)^j}{j!} e^{-\mu t} = \rho\, e^{\mu t \rho} e^{-\mu t} = \rho\, e^{-\mu(1-\rho)t}$$

Waiting time distribution (4)

- Waiting time W can thus be presented as a product $W = JD$ of two independent random variables $J \sim \text{Bernoulli}(\rho)$ and $D \sim \text{Exp}(\mu(1-\rho))$:

$$P\{W = 0\} = P\{J = 0\} = 1 - \rho$$

$$P\{W > t\} = P\{J = 1, D > t\} = \rho \cdot e^{-\mu(1-\rho)t}, \quad t > 0$$

$$E[W] = E[J]E[D] = \rho \cdot \frac{1}{\mu(1-\rho)} = \frac{1}{\mu} \cdot \frac{\rho}{1-\rho}$$

$$E[W^2] = P\{J = 1\}E[D^2] = \rho \cdot \frac{2}{\mu^2(1-\rho)^2} = \frac{1}{\mu^2} \cdot \frac{2\rho}{(1-\rho)^2}$$

$$D^2[W] = E[W^2] - E[W]^2 = \frac{1}{\mu^2} \cdot \frac{\rho(2-\rho)}{(1-\rho)^2}$$

Contents

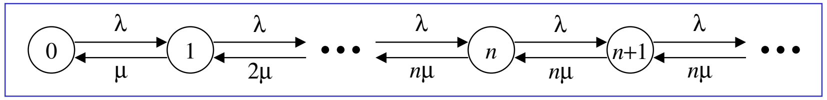
- Refresher: Simple teletraffic model
- M/M/1 (1 server, ∞ waiting places)
- M/M/n (n servers, ∞ waiting places)

M/M/*n* queue

- Consider the following simple teletraffic model:
  - Infinite number of independent customers ($k = \infty$)
  - Interarrival times are IID and exponentially distributed with mean $1/\lambda$
    - so, customers arrive according to a Poisson process with intensity λ
  - Finite number of servers ($n < \infty$)
  - Service times are IID and exponentially distributed with mean $1/\mu$
  - Infinite number of waiting places ($m = \infty$)
  - Default queueing discipline: FIFO
- Using Kendall's notation, this is an **M/M/*n* queue**
  - more precisely: M/M/n-FIFO queue
- Notation:
  - $\rho = \lambda/(n\mu)$ = traffic load

State transition diagram

- Let $X(t)$ denote the number of customers in the system at time t
  - Assume that $X(t) = i$ at some time t, and consider what happens during a short time interval $(t, t+h]$:
    - with prob. $\lambda h + o(h)$, a new customer arrives (state transition $i \to i+1$)
    - if $i > 0$, then, with prob. $\min\{i,n\}\cdot\mu h + o(h)$, a customer leaves the system (state transition $i \to i-1$)
- Process $X(t)$ is clearly a Markov process with state transition diagram



- Note that process $X(t)$ is an irreducible birth-death process with an infinite state space $S = \{0,1,2,\ldots\}$, $\lambda_i = \lambda$, $\mu_i = min(i,n)\cdot\mu$

Equilibrium distribution (1)

- Local balance equations (LBE) for $i < n$:

$$\pi_i\lambda = \pi_{i+1}(i+1)\mu \qquad \text{(LBE)}$$

$$\Rightarrow \quad \pi_{i+1} = \frac{\lambda}{(i+1)\mu}\pi_i = \frac{n\rho}{i+1}\pi_i$$

$$\Rightarrow \quad \pi_i = \frac{(n\rho)^i}{i!}\pi_0, \quad i = 0,1,\ldots,n$$

- Local balance equations (LBE) for $i \geq n$:

$$\pi_i\lambda = \pi_{i+1}n\mu \qquad \text{(LBE)}$$

$$\Rightarrow \quad \pi_{i+1} = \frac{\lambda}{n\mu}\pi_i = \rho\pi_i$$

$$\Rightarrow \quad \pi_i = \rho^{i-n}\pi_n = \rho^{i-n}\frac{(n\rho)^n}{n!}\pi_0 = \frac{n^n\rho^i}{n!}\pi_0, \quad i = n, n+1, \ldots$$

Equilibrium distribution (2)

- Normalizing condition (N):

$$\sum_{i=0}^{\infty}\pi_i = \pi_0\left(\sum_{i=0}^{n-1}\frac{(n\rho)^i}{i!} + \sum_{i=n}^{\infty}\frac{n^n\rho^i}{n!}\right) = 1 \qquad \text{(N)}$$

$$\Rightarrow \quad \pi_0 = \left(\sum_{i=0}^{n-1}\frac{(n\rho)^i}{i!} + \frac{(n\rho)^n}{n!}\sum_{i=n}^{\infty}\rho^{i-n}\right)^{-1}$$

$$= \left(\sum_{i=0}^{n-1}\frac{(n\rho)^i}{i!} + \frac{(n\rho)^n}{n!(1-\rho)}\right)^{-1} = \frac{1}{\alpha+\beta}, \quad \text{if } \rho < 1$$

$$\text{Notation}: \quad \alpha = \sum_{i=0}^{n-1}\frac{(n\rho)^i}{i!}, \quad \beta = \frac{(n\rho)^n}{n!(1-\rho)}$$

Equilibrium distribution (3)

- Thus, for a **stable** system ($\rho < 1$, that is: $\lambda < n\mu$), the equilibrium distribution exists and is as follows:

$$\rho < 1 \;\Rightarrow$$

$$P\{X = i\} = \pi_i = \begin{cases} \frac{(n\rho)^i}{i!} \cdot \frac{1}{\alpha+\beta}, & i = 0,1,\ldots,n \\ \frac{n^n \rho^i}{n!} \cdot \frac{1}{\alpha+\beta}, & i = n, n+1, \ldots \end{cases}$$

$$n = 1: \;\; \alpha = 1, \;\; \beta = \frac{\rho}{1-\rho}, \;\; \pi_0 = \frac{1}{\alpha+\beta} = 1-\rho$$

$$n = 2: \;\; \alpha = 1+2\rho, \;\; \beta = \frac{2\rho^2}{1-\rho}, \;\; \pi_0 = \frac{1}{\alpha+\beta} = \frac{1-\rho}{1+\rho}$$

Probability of waiting

- Let p_W denote the probability that an arriving customer has to wait
- Let X^* denote the number of customers in the system at an arrival time
- An arriving customer has to wait whenever all the servers are occupied at her arrival time. Thus,

$$p_W = P\{X^* \geq n\}$$

- PASTA: $P\{X^* = i\} = P\{X = i\} = \pi_i$. Thus,

$$p_W = P\{X^* \geq n\} = \sum_{i=n}^{\infty} \pi_i = \sum_{i=n}^{\infty} \pi_0 \cdot \frac{n^n \rho^i}{n!} = \pi_0 \cdot \frac{(n\rho)^n}{n!(1-\rho)} = \frac{\beta}{\alpha+\beta}$$

$$n = 1: \;\; p_W = \rho$$

$$n = 2: \;\; p_W = \frac{2\rho^2}{1+\rho}$$

Mean number of waiting customers

- Let X_W denote the number of waiting customers in equilibrium
- Then

$$E[X_W] = \sum_{i=n}^{\infty} (i-n)\pi_i = \pi_0 \frac{(n\rho)^n}{n!(1-\rho)} \sum_{i=n}^{\infty} (i-n) \cdot (1-\rho)\rho^{i-n}$$

$$= p_W \cdot \frac{\rho}{1-\rho}$$

$$n = 1: \;\; E[X_W] = p_W \cdot \frac{\rho}{1-\rho} = \frac{\rho^2}{1-\rho}$$

$$n = 2: \;\; E[X_W] = p_W \cdot \frac{\rho}{1-\rho} = \frac{2\rho^2}{1+\rho} \cdot \frac{\rho}{1-\rho} = \frac{2\rho^3}{1-\rho^2}$$

Mean waiting time

- Let W denote the waiting time of a (typical) customer
- Little's formula: $E[X_W] = \lambda \cdot E[W]$. Thus,

$$E[W] = \frac{E[X_W]}{\lambda} = \frac{1}{\lambda} \cdot p_W \cdot \frac{\rho}{1-\rho} = \frac{1}{\mu} \cdot \frac{p_W}{n(1-\rho)} = p_W \cdot \frac{1}{n\mu - \lambda}$$

$$n = 1: \;\; E[W] = \frac{1}{\mu} \cdot \frac{p_W}{1-\rho} = \frac{1}{\mu} \cdot \frac{\rho}{1-\rho}$$

$$n = 2: \;\; E[W] = \frac{1}{\mu} \cdot \frac{p_W}{2(1-\rho)} = \frac{1}{\mu} \cdot \frac{\rho^2}{1-\rho^2}$$

Mean delay

- Let D denote the total time (delay) in the system of a (typical) customer
  - including both the waiting time W and the service time S: $D = W + S$
- Then,

$$E[D]=E[W]+E[S]=\frac{1}{\mu}\cdot\left(\frac{p_W}{n(1-\rho)}+1\right)=p_W\cdot\frac{1}{n\mu-\lambda}+\frac{1}{\mu}$$

$$n=1:\quad E[D]=\frac{1}{\mu}\cdot\left(\frac{p_W}{1-\rho}+1\right)=\frac{1}{\mu}\cdot\left(\frac{\rho}{1-\rho}+1\right)=\frac{1}{\mu}\cdot\frac{1}{1-\rho}$$

$$n=2:\quad E[D]=\frac{1}{\mu}\cdot\frac{p_W}{2(1-\rho)}=\frac{1}{\mu}\cdot\left(\frac{\rho^2}{1-\rho^2}+1\right)=\frac{1}{\mu}\cdot\frac{1}{1-\rho^2}$$

Mean queue length

- Let X denote the number of customers in the system (queue length) in equilibrium
- Little's formula: $E[X] = \lambda \cdot E[D]$. Thus,

$$E[X]=\lambda\cdot E[D]=p_W\cdot\frac{\lambda}{n\mu-\lambda}+\frac{\lambda}{\mu}=p_W\cdot\frac{\rho}{1-\rho}+n\rho$$

$$n=1:\quad E[X]=p_W\cdot\frac{\rho}{1-\rho}+\rho=\rho\cdot\frac{\rho}{1-\rho}+\rho=\frac{\rho}{1-\rho}$$

$$n=2:\quad E[X]=p_W\cdot\frac{\rho}{1-\rho}+2\rho=\frac{2\rho^2}{1+\rho}\cdot\frac{\rho}{1-\rho}+2\rho=\frac{2\rho}{1-\rho^2}$$

Waiting time distribution (1)

- Let W denote the waiting time of a (typical) customer
- Let X^* denote the number of customers in the system at the arrival time
- The customer has to wait only if $X^* \geq n$. This happens with prob. p_W.
- Under the assumption that $X^* = i \geq n$, the system, however, looks like an ordinary M/M/1 queue with arrival rate λ and service rate $n\mu$.
  - Let W' denote the waiting time of a (typical) customer in this M/M/1 queue
  - Let $X^{*'}$ denote the number of customers in the system at the arrival time
- It follows that

$$P\{W=0\}=1-p_W$$

$$P\{W>t\}=P\{X^*\geq n\}P\{W>t\mid X^*\geq n\}$$

$$=p_W\cdot P\{W'>t\mid X^{*'}\geq 1\}=p_W\cdot e^{-n\mu(1-\rho)t},\quad t>0$$

Waiting time distribution (2)

- Waiting time W can thus be presented as a product $W = JD'$ of two indep. random variables $J \sim \mathrm{Bernoulli}(p_W)$ and $D' \sim \mathrm{Exp}(n\mu(1-\rho))$:

$$P\{W=0\}=P\{J=0\}=1-p_W$$

$$P\{W>t\}=P\{J=1,D'>t\}=p_W\cdot e^{-n\mu(1-\rho)t},\quad t>0$$

$$E[W]=E[J]E[D']=p_W\cdot\frac{1}{n\mu(1-\rho)}=\frac{1}{\mu}\cdot\frac{p_W}{n(1-\rho)}$$

$$E[W^2]=P\{J=1\}E[D'^2]=p_W\cdot\frac{2}{n^2\mu^2(1-\rho)^2}=\frac{1}{\mu^2}\cdot\frac{2p_W}{n^2(1-\rho)^2}$$

$$D^2[W]=E[W^2]-E[W]^2=\frac{1}{\mu^2}\cdot\frac{p_W(2-p_W)}{n^2(1-\rho)^2}$$

Example (1)

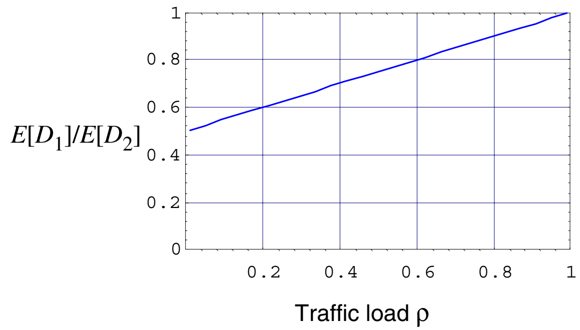
- Printer problem
  - Consider the following two different configurations:
    - One rapid printer (IID printing times ~ Exp(2μ))
    - Two slower parallel printers (IID printing times ~ Exp(μ))
  - Criterion: minimize mean delay $E[D]$
    - One rapid printer (M/M/1 model with ρ = λ/(2μ)):

$$E[D_1] = \frac{1}{2\mu} \cdot \frac{1}{1-\rho}$$

- Two slower printers (M/M/2 model with ρ = λ/(2μ)):

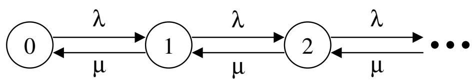
$$E[D_2] = \frac{1}{\mu} \cdot \frac{1}{1-\rho^2} = \frac{1}{2\mu} \cdot \frac{2}{(1-\rho)(1+\rho)} = E[D_1] \cdot \frac{2}{1+\rho} > E[D_1]$$

Example (2)

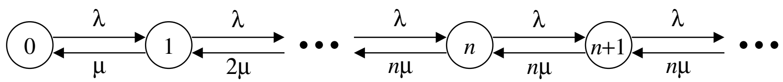


Summary

- State transitions when one server, infinite number of customers and infinite number of waiting places



- State transitions when many servers, infinite number of customers and infinite number of waiting places



- **All results presented in this lecture can be deduced from these state transition diagrams!**

HELSINKI UNIVERSITY OF TECHNOLOGY
Networking laboratory

9. Simulation

Contents

- Introduction
- Generation of traffic process realizations
- Generation of random variable realizations
- Collection of data
- Statistical analysis

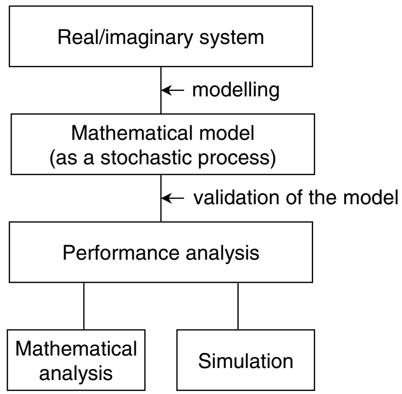
What is simulation?

- **Simulation** is (at least from the teletraffic point of view) a statistical method to estimate the performance (or some other important characteristic) of the system under consideration.
- It typically consists of the following four phases:
  - Modelling of the system (real or imaginary) as a dynamic stochastic process
  - Generation of the realizations of this stochastic process (“observations”)
    - Such realizations are called **simulation runs**
  - Collection of data (“measurements”)
  - Statistical analysis of the gathered data, and drawing conclusions

Alternative to what?

- In previous lectures, we have looked at an alternative way to determine the performance: **mathematical analysis**
- We considered the following two phases:
  - Modelling of the system as a stochastic process. (In this course, we have restricted ourselves to birth-death processes.)
  - Solving of the model by means of mathematical analysis
- The modelling phase is common to both of them
- However, the accuracy (faithfulness) of the model that these methods allow can be very different
  - unlike simulation, mathematical analysis typically requires (heavily) restrictive assumptions to be made

Performance analysis of a teletraffic system



Analysis vs. simulation (1)

- **Pros** of analysis
  - Results produced rapidly (after the analysis is made)
  - Exact (accurate) results (for the model)
  - Gives insight
  - Optimization possible (but typically hard)
- **Cons** of analysis
  - Requires restrictive assumptions
    ⇒ the resulting model is typically too simple
    (e.g. only stationary behavior)
    ⇒ performance analysis of complicated systems impossible
  - Even under these assumptions, the analysis itself may be (extremely) hard

Analysis vs. simulation (2)

- **Pros** of simulation
  - No restrictive assumptions needed (in principle)
    ⇒ performance analysis of complicated systems possible
  - Modelling straightforward
- **Cons** of simulation
  - Production of results time-consuming
    (simulation programs being typically processor intensive)
  - Results inaccurate (however, they can be made as accurate as required by increasing the number of simulation runs, but this takes even more time)
  - Does not necessarily offer a general insight
  - Optimization possible only between very few alternatives (parameter combinations or controls)

Steps in simulating a stochastic process

- Modelling of the system as a stochastic process
  - This has already been discussed in this course.
  - In the sequel, we will take the model (that is: the stochastic process) for granted.
  - In addition, we will restrict ourselves to simple teletraffic models.
- Generation of the realizations of this stochastic process
  - Generation of random numbers
  - Construction of the realization of the process from event to event (discrete event simulation)
  - Often this step is understood as THE simulation, however this is not generally the case
- Collection of data
  - Transient phase vs. steady state (stationarity, equilibrium)
- Statistical analysis and conclusions
  - Point estimators
  - Confidence intervals

Implementation

- Simulation is typically implemented as a computer program
- Simulation program generally comprises the following phases (excluding modelling and conclusions)
  - Generation of the realizations of the stochastic process
  - Collection of data
  - Statistical analysis of the gathered data
- Simulation program can be implemented by
  - a **general-purpose programming language**
    - e.g. C or C++
    - most flexible, but tedious and prone to programming errors
  - utilizing **simulation-specific program libraries**
    - e.g. CNCL
  - utilizing **simulation-specific software**
    - e.g. OPNET, BONeS, NS (in part based on p-libraries)
    - most rapid and reliable (depending on the s/w), but rigid

Other simulation types

- What we have described above, is a **discrete event simulation**
  - the simulation is **discrete** (event-based), **dynamic** (evolving in time) and **stochastic** (including random components)
  - i.e. how to simulate the time evolvement of the mathematical model of the system under consideration, when the aim is to gather information on the system behavior
  - We consider only this type of simulation in this lecture
- Other types:
  - **continuous** simulation: state and parameter spaces of the process are continuous; description of the system typically by differential equations, e.g. simulation of the trajectory of an aircraft
  - **static** simulation: time plays no role as there is no process that produces the events, e.g. numerical integration of a multi-dimensional integral by Monte Carlo method
  - **deterministic** simulation: no random components, e.g. the first example above

Contents

Generation of traffic process realizations

- Assume that we have modelled as a stochastic process the evolution of the system
- Next step is to generate realizations of this process.
  - For this, we have to:
    - Generate a realization (value) for all the random variables affecting the evolution of the process (taking properly into account all the (statistical) dependencies between these variables)
    - Construct a realization of the process (using the generated values)
  - These two parts are **overlapping**, they are not dome separately
  - Realizations for random variables are generated by utilizing (**pseudo**) **random number generators**
  - The realization of the process is constructed from event to event (**discrete event simulation**)

Discrete event simulation (1)

- Idea: simulation evolves **from event to event**
  - If nothing happens during an interval, we can just skip it!
- **basic events** modify (somehow) the state of the system
  - e.g. arrivals and departures of customers in a simple teletraffic model
- **extra events** related to the data collection
  - including the event for stopping the simulation run or collecting data
- Event identification:
  - occurrence time (when event is handled) and
  - event type (what and how event is handled)

Discrete event simulation (2)

- Events are organized as an **event list**
  - Events in this list are ordered (ascendingly) by the occurrence time
    - first: the event occurring next
  - Events are handled one-by-one (in this order) while at the same time generating new events to occur later
  - When the event has been processed, it is removed from the list
- **Simulation clock** tells the occurrence time of the next event
  - progressing by jumps
- **System state** tells the current state of the system

Discrete event simulation (3)

- General algorithm for a single **simulation run**:
  1 Initialization
    - simulation clock = 0
    - system state = given initial value
    - for each event type, generate next event (whenever possible)
    - construct the event list from these events
  2 Event handling
    - simulation clock = occurrence time of the next event
    - handle the event including
      - generation of new events and their addition to the event list
      - updating of the system state
    - delete the event from the event list
  3 Stopping test
    - if positive, then stop the simulation run; otherwise return to 2

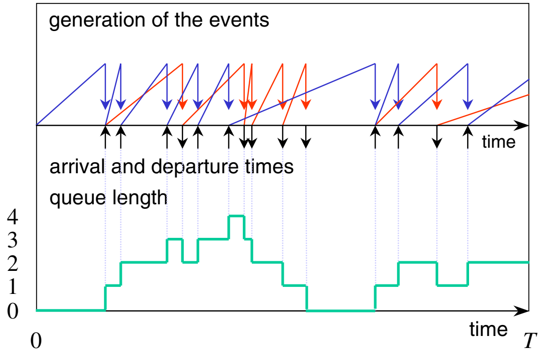
Example (1)

- **Task**: Simulate the M/M/1 queue (more precisely: the evolution of the queue length process) from time 0 to time T assuming that the queue is empty at time 0 and omitting any data collection
  - System state (at time t) = queue length X_t
    - initial value: $X_0 = 0$
  - Basic events:
    - customer arrivals
    - customer departures
  - Extra event:
    - stopping of the simulation run at time T
  - **Note**. No collection of data in this example

Example (2)

- Initialization:
  - initialize the system state: $X_0 = 0$
  - generate the time till the first arrival from the $\mathrm{Exp}(\lambda)$ distribution
- Handling of an arrival event (occurring at some time t):
  - update the system state: $X_t = X_t + 1$
  - if $X_t = 0$, then generate the time $(t + S)$ till the next departure, where S is from the $\mathrm{Exp}(\mu)$ distribution
  - generate the time $(t + I)$ till the next arrival, where I is from the $\mathrm{Exp}(\lambda)$ distribution
- Handling of a departure event (occuring at some time t):
  - update the system state: $X_t = X_t - 1$
  - if $X_t > 0$, then generate the time $(t + S)$ till the next departure, where S is from the $\mathrm{Exp}(\mu)$ distribution
- Stopping test: $t > T$

Example (3)



Contents

- Introduction
- Generation of traffic process realizations
- Generation of random variable realizations
- Collection of data
- Statistical analysis

Generation of random variable realizations

- Based on **(pseudo) random number generators**
- First step:
  - generation of independent uniformly distributed between 0 and 1, i.e. $\mathrm{U}(0,1)$ distributed, random variables using random number generators
- Step from the $\mathrm{U}(0,1)$ distribution to the desired distribution:
  - **rescaling** ($\Rightarrow \mathrm{U}(a,b)$)
  - **discretization** ($\Rightarrow$ Bernoulli(p), Bin(n,p), Poisson(a), Geom(p))
  - **inverse transform** ($\Rightarrow \mathrm{Exp}(\lambda)$)
  - **other transforms** ($\Rightarrow \mathrm{N}(0,1) \Rightarrow \mathrm{N}(\mu,\sigma^2)$)
  - **acceptance-rejection method** (for any continuous random variable defined in a finite interval whose density function is bounded)
    - two independent $\mathrm{U}(0,1)$ distributed random variables needed

Random number generator

- **Random number generator** is an algorithm generating (pseudo) random integers Z_i in some interval $0,1,\ldots, m-1$
  - The sequence generated is **always** periodic (goal: this period should be as long as possible)
  - Strictly speaking, the numbers generated are not random at all, but totally predictable (thus: pseudo)
  - In practice, however, if the generator is well designed, the numbers **“appear” to be IID with uniform distribution** inside the set $\{0,1,\ldots,m-1\}$
- Validition of a random number generator can be based on empirical (statistical) and theoretical tests:
  - uniformity of the generated empirical distribution
  - independence of the generated random numbers (no correlation)

Random number generator types

- **Linear congruential generator**
  - most simple
  - next random number is based on just the current one: $Z_{i+1} = f(Z_i)$ $\Rightarrow$ period at most m
- **Multiplicative congruential generator**
  - a special case of the first type
- Other:
  - Additive congruential generators
  - Shuffling, etc.

Linear congruential generator (LCG)

- **Linear congruential generator** (LCG) uses the following algorithm to generate random numbers belonging to $\{0,1,\ldots, m-1\}$:

$$Z_{i+1} = (aZ_i + c) \bmod m$$

  - Here a, c and m are fixed non-negative integers ($a < m$, $c < m$)
  - In addition, the starting value (**seed**) $Z_0 < m$ should be specified
- Remarks:
  - Parameters a, c and m should be chosen with care, otherwise the result can be very poor
  - By a right choice of parameters, it is possible to achieve the full period m
    - e.g. $m = 2^b$, c odd, $a = 4k + 1$ (b often 48)

Multiplicative congruential generator (MCG)

- **Multiplicative congruential generator** (MCG) uses the following algorithm to generate random numbers belonging to $\{0,1,\ldots, m-1\}$:

$$Z_{i+1} = (aZ_i) \bmod m$$

  - Here a and m are fixed non-negative integers ($a < m$)
  - In addition, the starting value (seed) $Z_0 < m$ should be specified
- Remarks:
  - MCG is clearly a special case of LCG: $c = 0$
  - Parameters a and m should (still) be chosen with care
  - In this case, it is not possible to achieve the full period m
    - e.g. if $m = 2^b$, then the maximum period is 2^{b-2}
  - However, for m **prime**, period $m-1$ is possible (by a proper choice of a)
    - PMMLCG = prime modulus multiplicative LCG
    - e.g. $m = 2^{31}-1$ and $a = 16{,}807$ (or $630{,}360{,}016$)

U(0,1) distribution

- Let Z denote a (pseudo) random number belonging to $\{0,1,\ldots, m-1\}$
- Then (approximately)

$$U = \frac{Z}{m} \approx \mathrm{U}(0,1)$$

U(a,b) distribution

- Let $U \sim \mathrm{U}(0,1)$
- Then

$$X = a + (b-a)U \sim \mathrm{U}(a,b)$$

- This is called the **rescaling** method

Discretization method

- Let $U \sim \mathrm{U}(0,1)$
- Assume that Y is a **discrete** random variable
  - with value set $S = \{0,1,\ldots,n\}$ or $S = \{0,1,2,\ldots\}$
- Denote: $F(x) = P\{Y \le x\}$, then

$$X = \min\{x \in S \mid F(x) \ge U\} \sim Y$$

- This is called the **discretization** method
  - a special case of the inverse transform method
- **Example**: $\mathrm{Bernoulli}(p)$ distribution

$$X = \begin{cases} 0, & \text{if } U \le 1-p \\ 1, & \text{if } U > 1-p \end{cases} \sim \mathrm{Bernoulli}(p)$$

Inverse transform method

- Let $U \sim \mathrm{U}(0,1)$
- Assume that Y is a **continuous** random variable
- Assume further that $F(x) = P\{Y \le x\}$ is strictly increasing
- Let $F^{-1}(y)$ denote the inverse of the function $F(x)$, then

$$X = F^{-1}(U) \sim Y$$

- This is called the **inverse transform** method
- Proof: Since $P\{U \le u\} = u$ for all $u \in (0,1)$, we have

$$P\{X \le x\} = P\{F^{-1}(U) \le x\} = P\{U \le F(x)\} = F(x)$$

Exp(λ) distribution

- Let $U \sim \mathrm{U}(0,1)$
  - Then also $1-U \sim \mathrm{U}(0,1)$
- Let $Y \sim \mathrm{Exp}(\lambda)$
  - $F(x) = P\{Y \le x\} = 1-e^{-\lambda x}$ is strictly increasing
  - The inverse transform is $F^{-1}(y) = -(1/\lambda)\log(1-y)$
- Thus, by the inverse transform method,

$$X = F^{-1}(1-U) = -\tfrac{1}{\lambda}\log(U) \sim \mathrm{Exp}(\lambda)$$

N(0,1) distribution

- Let $U_1 \sim \mathrm{U}(0,1)$ and $U_2 \sim \mathrm{U}(0,1)$ be **independent**
- Then, by so called Box-Müller method,
  the following two (transformed) random variables are **independent** and identically distributed obeying the $\mathrm{N}(0,1)$ distribution:

$$X_1 = \sqrt{-2\log(U_1)}\,\sin(2\pi U_2) \sim N(0,1)$$
$$X_2 = \sqrt{-2\log(U_1)}\,\cos(2\pi U_2) \sim N(0,1)$$

N(μ,σ²) distribution

- Let $X \sim \mathrm{N}(0,1)$
- Then, by the rescaling method,

$$Y = \mu + \sigma X \sim N(\mu, \sigma^2)$$

Contents

- Introduction
- Generation of traffic process realizations
- Generation of random variable realizations
- Collection of data
- Statistical analysis

Collection of data

- Our starting point was that simulation is needed to estimate the value, say α, of some performance parameter
  - This parameter may be related to the **transient** or the **steady-state** behaviour of the system.
  - Examples 1 & 2 (transient phase characteristics)
    - average waiting time of the first k customers in an M/M/1 queue assuming that the system is empty in the beginning
    - average queue length in an M/M/1 queue during the interval $[0,T]$ assuming that the system is empty in the beginning
  - Example 3 (steady-state characteristics)
    - the average waiting time in an M/M/1 queue in equilibrium
- Each simulation run yields one sample, say X, describing somehow the parameter under consideration
- For drawing statistically reliable conclusions, multiple samples, $X_1,\ldots,X_n$, are needed (preferably IID)

Transient phase characteristics (1)

- Example 1:
  - Consider e.g. the average waiting time of the first k customers in an M/M/1 queue assuming that the system is empty in the beginning
  - Each simulation run can be stopped when the kth customer enters the service
  - The sample X based on a single simulation run is in this case:

$$X = \frac{1}{k} \sum_{i=1}^{k} W_i$$

    - Here W_i = waiting time of the ith customer in this simulation run
- Multiple IID samples, $X_1,\ldots,X_n$, can be generated by the method of **independent replications:**
  - multiple independent simulation runs (using independent random numbers)

Transient phase characteristics (2)

- Example 2:
  - Consider e.g. the average queue length in an M/M/1 queue during the interval $[0,T]$ assuming that the system is empty in the beginning
  - Each simulation run can be stopped at time T (that is: simulation clock = T)
  - The sample X based on a single simulation run is in this case:

$$X = \frac{1}{T} \int_0^T Q(t)dt$$

    - Here $Q(t)$ = queue length at time t in this simulation run
    - Note that this integral is easy to calculate, since $Q(t)$ is piecewise constant
- Multiple IID samples, $X_1,\ldots,X_n$, can again be generated by the method of independent replications

Steady-state characteristics (1)

- Collection of data in a single simulation run is in principle similar to that of transient phase simulations
- Collection of data in a single simulation run can **typically** (but not always) be done only after a **warm-up** phase (hiding the transient characteristics) resulting in
  - overhead =“extra simulation”
  - bias in estimation
  - need for determination of a **sufficiently long** warm-up phase
- Multiple samples, $X_1,\ldots,X_n$, may be generated by the following three methods:
  - independent replications
  - batch means

Steady-state characteristics (2)

- Method of **independent replications**:
  - multiple independent simulation runs of the same system (using independent random numbers)
  - each simulation run includes the warm-up phase $\Rightarrow$ inefficiency
  - samples IID $\Rightarrow$ accuracy
- Method of **batch means**:
  - one (very) long simulation run divided (artificially) into one warm-up phase and n equal length periods (each of which represents a single simulation run)
  - only one warm-up phase $\Rightarrow$ efficiency
  - samples only approximately IID $\Rightarrow$ inaccuracy,
    - choice of n, the larger the better

Contents

- Introduction
- Generation of traffic process realizations
- Generation of random variable realizations
- Collection of data
- Statistical analysis

Parameter estimation

- As mentioned, our starting point was that simulation is needed to estimate the value, say α, of some performance parameter
- Each simulation run yields a (random) sample, say X_i, describing somehow the parameter under consideration
  - Sample X_i is called **unbiased** if $E[X_i] = \alpha$
- Assuming that the samples X_i are IID with mean α and variance σ^2
  - Then the **sample average**

$$\bar{X}_n := \tfrac{1}{n}\sum_{i=1}^{n} X_i$$

  - is **unbiased** and **consistent** estimator of α, since

$$E[\bar{X}_n] = \tfrac{1}{n}\sum_{i=1}^{n} E[X_i] = \alpha$$

$$D^2[\bar{X}_n] = \tfrac{1}{n^2}\sum_{i=1}^{n} D^2[X_i] = \tfrac{1}{n}\sigma^2 \to 0 \quad (\text{as } n \to \infty)$$

Example

- Consider the average waiting time of the first 25 customers in an M/M/1 queue with load $\rho = 0.9$ assuming that the system is empty in the beginning
  - Theoretical value: $\alpha = 2.12$ (non-trivial)
  - Samples X_i from ten simulation runs ($n = 10$):
    - 1.05, 6.44, 2.65, 0.80, 1.51, 0.55, 2.28, 2.82, 0.41, 1.31
  - Sample average (point estimate for α):

$$\bar{X}_n = \tfrac{1}{n}\sum_{i=1}^{n} X_i = \tfrac{1}{10}(1.05 + 6.44 + \ldots + 1.31) = 1.98$$

Confidence interval (1)

- **Definition**: Interval $(\bar{X}_n - y, \bar{X}_n + y)$ is called the **confidence interval** for the sample average at **confidence level** $1 - \beta$ if

$$P\{|\bar{X}_n - \alpha| \le y\} = 1 - \beta$$

  - Idea: "with probability $1 - \beta$, the parameter α belongs to this interval"
- Assume then that samples X_i, $i = 1,\ldots,n$, are IID with unknown mean α but **known** variance σ^2
- By the Central Limit Theorem (see Lecture 5, Slide 48), for large n,

$$Z := \frac{\bar{X}_n - \alpha}{\sigma / \sqrt{n}} \approx \mathrm{N}(0,1)$$

Confidence interval(2)

- Let z_p denote the p-**fractile** of the $\mathrm{N}(0,1)$ distribution
  - That is: $P\{Z \le z_p\} = p$, where $Z \sim \mathrm{N}(0,1)$
  - Example: for $\beta = 5\%$ $(1-\beta = 95\%) \Rightarrow z_{1-(\beta/2)} = z_{0.975} \approx 1.96 \approx 2.0$
- **Proposition**: The confidence interval for the sample average at confidence level $1-\beta$ is

$$\overline{X}_n \pm z_{1-\frac{\beta}{2}} \cdot \frac{\sigma}{\sqrt{n}}$$

- **Proof**: By definition, we have to show that

$$P\{|\overline{X}_n - \alpha| \le z_{1-\frac{\beta}{2}} \cdot \frac{\sigma}{\sqrt{n}}\} = 1-\beta$$

$$P\{|\overline{X}_n - \alpha| \le y\} = 1-\beta$$

$$\Leftrightarrow P\{\frac{|\overline{X}_n - \alpha|}{\sigma/\sqrt{n}} \le \frac{y}{\sigma/\sqrt{n}}\} = 1-\beta$$

$$\Leftrightarrow P\{\frac{-y}{\sigma/\sqrt{n}} \le \frac{\overline{X}_n - \alpha}{\sigma/\sqrt{n}} \le \frac{y}{\sigma/\sqrt{n}}\} = 1-\beta$$

$$\Leftrightarrow \Phi(\frac{y}{\sigma/\sqrt{n}}) - \Phi(\frac{-y}{\sigma/\sqrt{n}}) = 1-\beta \qquad [\Phi(x) := P\{Z \le x\}]$$

$$\Leftrightarrow \Phi(\frac{y}{\sigma/\sqrt{n}}) - (1-\Phi(\frac{y}{\sigma/\sqrt{n}})) = 1-\beta \qquad [\Phi(-x) = 1-\Phi(x)]$$

$$\Leftrightarrow \Phi(\frac{y}{\sigma/\sqrt{n}}) = 1-\frac{\beta}{2}$$

$$\Leftrightarrow \frac{y}{\sigma/\sqrt{n}} = z_{1-\frac{\beta}{2}}$$

$$\Leftrightarrow y = z_{1-\frac{\beta}{2}} \cdot \frac{\sigma}{\sqrt{n}}$$

Confidence interval (3)

- In general, however, the variance σ^2 is unknown (in addition to the mean α)
- It can be estimated by the **sample variance**:

$$S_n^2 := \frac{1}{n-1}\sum_{i=1}^n (X_i - \overline{X}_n)^2 = \frac{1}{n-1}(\sum_{i=1}^n X_i^2 - n\overline{X}_n^2)$$

- It is possible to prove that the sample variance is an unbiased and consistent estimator of σ^2:

$$E[S_n^2] = \sigma^2$$

$$D^2[S_n^2] \to 0 \quad (n \to \infty)$$

Confidence interval (4)

- Assume that samples X_i are IID obeying the $\mathrm{N}(\alpha,\sigma^2)$ distribution with unknown mean α and **unknown** variance σ^2
- Then it is possible to show that

$$T := \frac{\overline{X}_n - \alpha}{S_n/\sqrt{n}} \sim \mathrm{Student}(n-1)$$

- Let $t_{n-1,p}$ denote the p-fractile of the $\mathrm{Student}(n-1)$ distribution
  - That is: $P\{T \le t_{n-1,p}\} = p$, where $T \sim \mathrm{Student}(n-1)$
  - Example 1: $n = 10$ and $\beta = 5\%$, $t_{n-1,1-(\beta/2)} = t_{9,0.975} \approx 2.26 \approx 2.3$
  - Example 2: $n = 100$ and $\beta = 5\%$, $t_{n-1,1-(\beta/2)} = t_{99,0.975} \approx 1.98 \approx 2.0$
- Thus, the conf. interval for the sample average at conf. level $1-\beta$ is

$$\overline{X}_n \pm t_{n-1,1-\frac{\beta}{2}} \cdot \frac{S_n}{\sqrt{n}}$$

Example (continued)

- Consider the average waiting time of the first 25 customers in an M/M/1 queue with load $\rho = 0.9$ assuming that the system is empty in the beginning
  - Theoretical value: $\alpha = 2.12$
  - Samples X_i from ten simulation runs ($n = 10$):
    - 1.05, 6.44, 2.65, 0.80, 1.51, 0.55, 2.28, 2.82, 0.41, 1.31
  - Sample average = 1.98 and the square root of the sample variance:

$$S_n = \sqrt{\tfrac{1}{9}((1.05-1.98)^2 + \ldots + (1.31-1.98)^2)} = 1.78$$

  - So, the confidence interval (that is: interval estimate for α) at confidence level 95% is

$$\overline{X}_n \pm t_{n-1,1-\frac{\beta}{2}} \cdot \frac{S_n}{\sqrt{n}} = 1.98 \pm 2.26 \cdot \frac{1.78}{\sqrt{10}} = 1.98 \pm 1.27 = (0.71, 3.25)$$

Observations

- Simulation results become more accurate (that is: the interval estimate for α becomes narrower) when
  - the number n of simulation runs is increased, or
  - the variance σ^2 of each sample is reduced
    - by running longer individual simulataion runs
    - variance reduction methods
- Given the desired accuracy for the simulation results, the number of required simulation runs can be determined dynamically

Literature

- I. Mitrani (1982)
  - “Simulation techniques for discrete event systems”
  - Cambridge University Press, Cambridge
- A.M. Law and W. D. Kelton (1982, 1991)
  - “Simulation modeling and analysis”
  - McGraw-Hill, New York

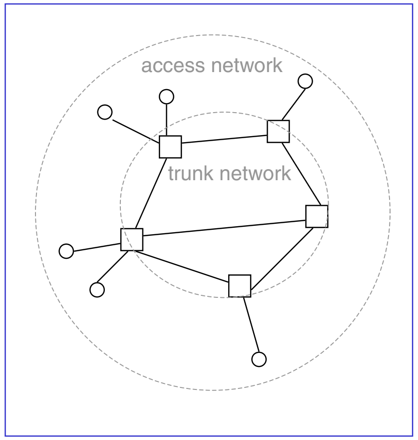
- **Note**. In fall 2004 the subject is the topic of the course
  - S-38.148 Simulation of Data Networks (2 cr)
  - http://keskus.hut.fi/opetus/s38148/

10. Network planning and dimensioning

Contents

Telecommunication network

- A simple model of a telecommunication network consists of
  - **nodes**
    - terminals ○
    - network nodes □
  - **links** between nodes
- **Access network**
  - connects the terminals to the network nodes
- **Trunk network**
  - connects the network nodes to each other



Why network planning and dimensioning?

- "The purpose of dimensioning of a telecommunications network is to ensure that

  the expected needs will be met in an economical way

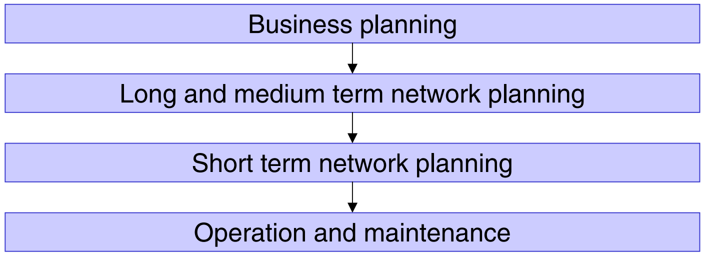
  both for subscribers and operators."

Source: [1]

Contents

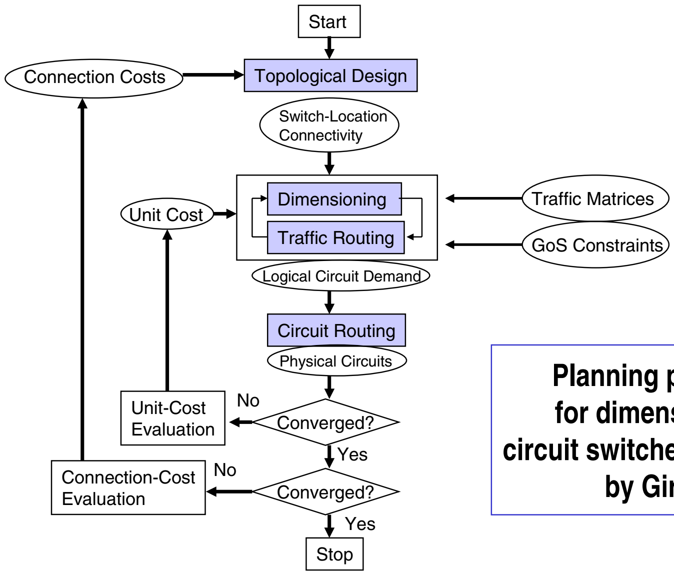
Network planning in a stable environment (1)

- Traditional planning model:



Network planning in a stable environment (2)

- Traffic aspects
  - Data collection (current status)
    - traffic measurements
    - subscriber amounts and distribution
  - Forecasting
    - service scenarios
    - traffic volumes and profiles
- Economical aspects
- Technical aspects
- Network optimisation and dimensioning
  - hierarchical structure and topology
  - traffic routing and dimensioning
  - circuit routing



Planning process for dimensioning circuit switched networks by Girard

Network planning in a turbulent environment (1)

- Additional decision data are needed from the following areas:
  - The market, with regard to a specific business concept
    - due to competition!
    - operator's future role (niche): dominance/co-operation
  - Customer demands:
    - new services: Internet & mobility (first of all)
    - new business opportunities
  - Technology:
    - new technology: ATM, xDSL, GSM, CDMA, WDM
  - Standards:
    - new standards issued continuously
  - Operations and network planning support:
    - new computer-aided means
  - Costs:
    - trends: equipment costs going down, staff costs going up

Network planning in a turbulent environment (2)

- Safeguards for the operator:
  - Change the network architecture so that it will be more **open**, with generic **platforms**, if possible
  - Build the network with a certain prognosticated overcapacity (**redundancy**) in generic parts where the marginal costs are low
- New planning situation (shift of focus to a strategic-tactical approach):

Business planning; Strategic-tactical planning of network resources for **flexible use**

↓

Business-driven, dynamic network management for **optimal use** of network resources

Source: [1]

Contents

- Introduction
- Network planning
- **Traffic forecasts**
- Traffic dimensioning

Need for traffic measurements and forecasts

- To properly dimension the network we need to

estimate the traffic offered

- If the network is already operating,
  - the current traffic is most precisely estimated by making traffic measurements
- Otherwise, the estimation should be based on other information, e.g.
  - estimations on characteristic traffic generated by a subscriber
  - estimations on the number of subscribers
- Long time-span of network investments $\Rightarrow$
  - it is not enough to estimate only the current traffic
  - forecasts of future traffic are also needed

Traffic forecasting

- Information about future demands for telecommunications
  - an estimation of future tendency or direction
- Purpose
  - provide a basis for decisions on investments in network
- Forecast periods
  - time aspect important (reliability)
  - need for forecast periods of different lengths

Source: [1]

Traffic forecast

- Traffic forecast defines
  - the estimated traffic growth in the network over the planning period
- Starting point:
  - current traffic volume during busy hour (measured/estimated)
- Other affecting factors:
  - changes in the number of subscribers
  - change in traffic per subscriber (characteristic traffic)
- Final result (that is, the forecast):
  - **traffic matrix** describing the **traffic interest** between exchanges (traffic areas)

Traffic matrix

- Traffic matrix $T = (T(i,j))$
  - describes traffic interest between exchanges
  - N^2 elements (N = nr of exchanges)
  - element $T(i,i)$ tells the estimated traffic within exchange i (in Erlangs)
  - element $T(i,j)$ tells the estimated traffic from exchange i to exchange j (in Erlangs)
- Problem:
  - easily grows too big: 600 exchanges $\Rightarrow$ 360,000 elements!
- Solution: hierarchical representation
  - higher level: traffic between traffic areas
  - lower level: traffic between exchanges within one traffic area

Example (1): one local exchange

- **Data**:
  - There are 1000 private subscribers and 10 companies with their own PBX's in the area of a local exchange.
  - The characteristic traffic generated by a private subscriber and a company are estimated to be 0.025 erlang and 0.200 erlang, respectively.
- **Questions**:
  - What is the total traffic intensity a generated by all these subscribers?
  - What is the call arrival rate λ assumed that the mean holding time is 3 minutes?
- **Answers**:
  - a = 1000 * 0.025 + 10 * 0.200 = 25 + 2 = **27 erlang**
  - h = 3 min
  - $\lambda = a/h$ = 27/3 calls/min = **9 calls/min**

Example (2): one local exchange

- **Data**:
  - In a 5-year forecasting period the number of new subscribers is estimated to grow linearly with rate 100 subscribers/year.
  - The characteristic traffic generated by a private subscriber is assumed to grow to value 0.040 erlang.
  - The total nr of companies with their own PBX is estimated to be 20 at the end of the forecasting period.
- **Question**:
  - What is the estimated total traffic intensity a at the end of the forecasting period?
- **Answer**:
  - a = (1000 + 5*100)* 0.040 + 20 * 0.200 = 60 + 4 = **64 erlang**

Example (3): many local exchanges

- **Data**:
  - Assume that there are three similar local exhanges.
  - Assume further that one half of the traffic generated by a local exchange is local traffic and the other half is directed uniformly to the two other exchanges.
- **Question**:
  - Construct the traffic matrix T describing the traffic interest between the exchanges at the end of the forecasting period.

•**Answer:**

– $T(i,i)$ = 64/2 = 32 erlang

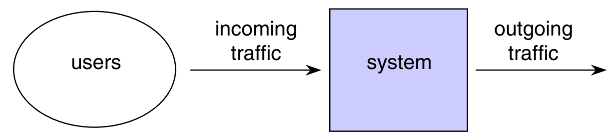
– $T(i,j)$ = 64/4 = 16 erlang

area	**1**	**2**	**3**	**sum**
1	32	16	16	64
2	16	32	16	64
3	16	16	32	64
sum	64	64	64	192

Contents

Traffic dimensioning (1)

- Telecommunications system from the traffic point of view:



- Basic task in **traffic dimensioning**:

Determine the minimum **system capacity** needed
in order that the incoming **traffic** meet
the specified **grade of service**

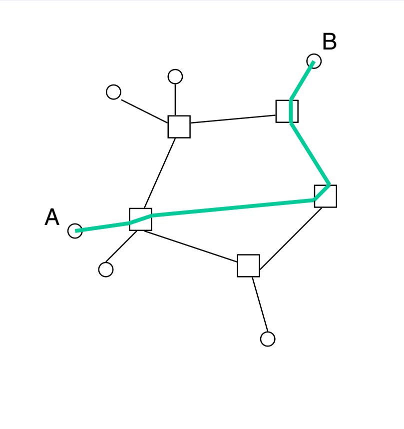
Traffic dimensioning (2)

- Observation:
  - Traffic is varying in time
- General rule:
  - Dimensioning should be based on peak traffic not on average traffic
- However,
  - Revenues are based on average traffic
- For dimensioning (of telephone networks), peak traffic is defined via the concept of busy hour:

Busy hour ≈ the continuous 1-hour period
for which the traffic volume is greatest

Telephone network model

- Simple model of a telephone network consists of
  - network nodes (exchanges)
  - links between nodes
- Traffic consists of **calls**
- Each call has two phases
  - first, the connection has to set up through the network (**call establishment** phase)
  - only after that, the information transfer is possible (**information transfer** phase)

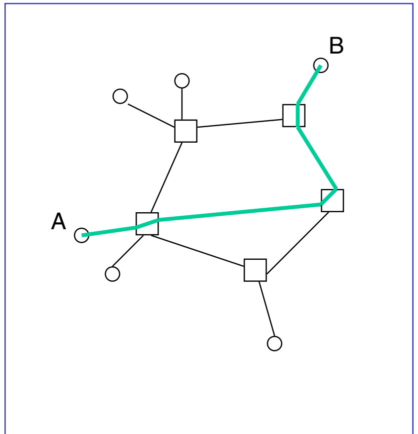


Two kinds of traffic processes

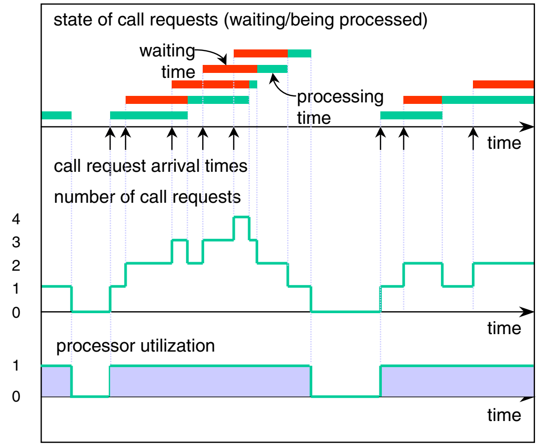
- Traffic process in each **network node**
  - due to **call establishments**
  - during the call establishment phase
    - each call needs (and competes for) processing resources in each network node (switch) along its route
  - it typically takes **some seconds** (during which the call is processed in the switches, say, **some milliseconds**)
- Traffic process in each **link**
  - due to **information transfer**
  - during the information transfer phase
    - each call occupies one channel on each link along its route
  - information transfer lasts as long as one of the participants disconnects
    - ordinary telephone calls typically hold **some minutes**
- **Note:** totally **different time scales** of the two processes

Simplified traffic dimensioning in a telephone network

- Assume
  - fixed topology and routing
  - given traffic matrix
  - given GoS requirements
- Dimensioning of network nodes: Determine the required **call handling capacity**
  - max number of call establishments the node can handle in a time unit
- Dimensioning of links: Determine the required **number of channels**
  - max number of ongoing calls on the link



Traffic process during call establishment (1)



Traffic process during call establishment (2)

- Call (request) arrival process is modelled as
  - a Poisson process with intensity λ
- Further we assume that call processing times are
  - IID and exponentially distributed with mean s
    - typically s is in the range of **milliseconds** (not minutes as h)
    - s is more a **system parameter** than a traffic parameter
- Finally we assume that the call requests are processed by
  - a single processor with an infinite buffer
    - $1/s$ tells the processing rate of call requests
- The resulting traffic process model is
  - the **M/M/1 queueing model** with traffic load $\rho = \lambda s$

Traffic process during call establishment (3)

- Pure delay system $\Rightarrow$

Grade of Service measure = Mean waiting time $E[W]$

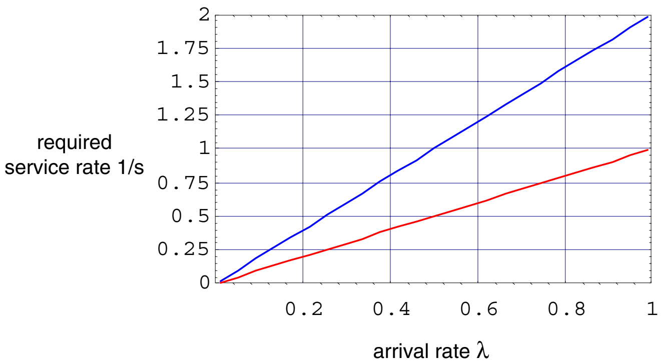
- Formula for the mean waiting time $E[W]$ (assuming that $\rho < 1$):

$$E[W] = s \cdot \frac{\rho}{1-\rho}$$

  - $\rho = \lambda s$
  - **Note**: $E[W]$ grows to infinity as ρ tends to 1

Dimensioning curve

- Grade of Service requirement: $E[W] \leq s$
  $\Rightarrow$ Allowed load $\rho \leq 0.5 = 50\% \Rightarrow \lambda s \leq 0.5$
  $\Rightarrow$ Required service rate $1/s \geq 2\lambda$ (blue line)



Dimensioning rule

- To get the required Grade of Service (the average time a customer waits before service should be less than the average service time) …

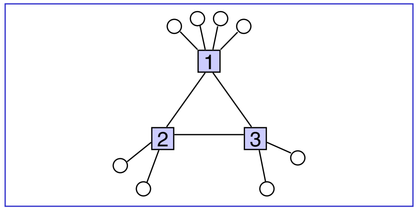
… Keep the traffic load less than 50%

- If you want a less stringent requirement, still remember the **safety margin** …

Don't let the total traffic load approach to 100%

- Otherwise you'll see an explosion!

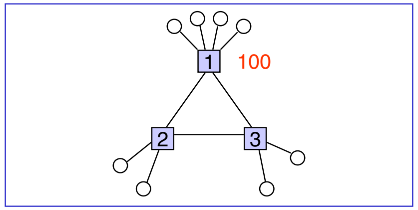
Example: dimensioning the nodes (1)



area	1	2	3	sum
1	60	15	15	90
2	30	30	15	75
3	30	15	30	75
sum	120	60	60	240

- **Assumptions**:
  - 3 local exchanges completely connected to each other
  - Traffic matrix T describing the busy hour traffic interest (in erlangs) given below
  - Fixed (direct) routing: calls are routed along shortest paths.
  - Mean holding time h = 3 min.
- **Task**:
  - Determine the call handling capacity needed in different network nodes according to the GoS requirement $\rho < 50\%$

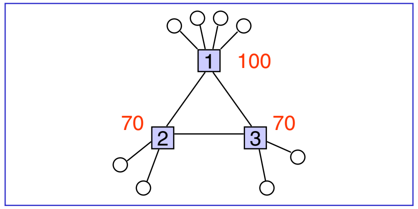
Example: dimensioning the nodes (2)



area	1	2	3	sum
1	60	15	15	90
2	30	30	15	75
3	30	15	30	75
sum	120	60	60	240

- **Node 1**:
  - call requests from own area: $[T(1,1) + T(1,2) + T(1,3)]/h = 90/3 = 30$ calls/min
  - call requests from area 2: $T(2,1)/h = 30/3 = 10$ calls/min
  - call requests from area 3: $T(3,1)/h = 30/3 = 10$ calls/min
  - total call request arrival rate: $\lambda(1) = 30+10+10 = 50$ calls/min
  - required call handling capacity: $\rho(1) = \lambda(1)/\mu(1) = 0.5 \Rightarrow$ $\mathbf{\mu(1)} \geq 2*\lambda(1) = \mathbf{100}$ **calls/min**

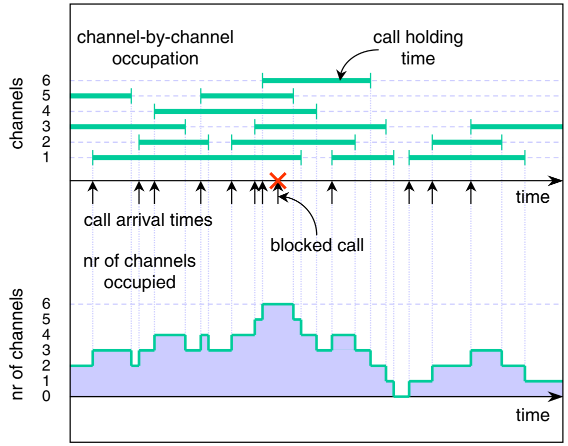
Example: dimensioning the nodes (3)



area	1	2	3	sum
1	60	15	15	90
2	30	30	15	75
3	30	15	30	75
sum	120	60	60	240

- **Node 2**:
  - total call request arrival rate: $\lambda(2) = [T(2,1) + T(2,2) + T(2,3) + T(1,2) + T(3,2)]/h = (75+15+15)/3 = 35$ calls/min
  - required call handling capacity: $\mathbf{\mu(2)} \geq 2*\lambda(2) = \mathbf{70}$ **calls/min**
- **Node 3**:
  - total call request arrival rate : $\lambda(3) = [T(3,1) + T(3,2) + T(3,3) + T(1,3) + T(2,3)]/h = (75+15+15)/3 = 35$ calls/min
  - required call handling capacity: $\mathbf{\mu(3)} \geq 2*\lambda(3) = \mathbf{70}$ **calls/min**

Traffic process during information transfer (1)



Traffic process during information transfer (2)

- Call arrival process has already been modelled as
  - a Poisson process with intensity λ
- Further we assume that call holding times are
  - IID and generally distributed with mean h
    - typically h is in the range of **minutes** (not milliseconds as s)
    - h is more a **traffic parameter** than a system parameter
- The resulting traffic process model is
  - the **M/G/n/n loss model** with (offered) traffic intensity $a = \lambda h$

Traffic process during information transfer (3)

- Pure loss system $\Rightarrow$

Grade of Service measure = Call blocking probability B

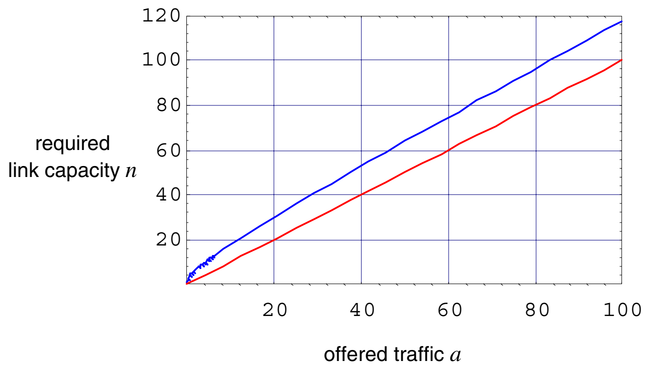
- **Erlang's blocking formula**:

$$B = \mathrm{Erl}(n,a) = \frac{\frac{a^n}{n!}}{\sum_{i=0}^{n} \frac{a^i}{i!}}$$

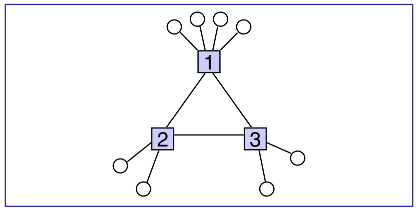
  - $a = \lambda h$
  - $n! = n(n-1)(n-2)\ldots 1$

Dimensioning curve

- Grade of Service requirement: $B \leq 1\%$
  $\Rightarrow$ Required link capacity: $n = \min\{i = 1,2,\ldots \mid \mathrm{Erl}(i,a) \leq B\}$



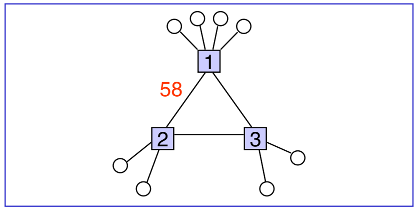
Example: dimensioning the links (1)



area	1	2	3	sum
1	60	15	15	90
2	30	30	15	75
3	30	15	30	75
sum	120	60	60	240

- **Assumptions**:
  - 3 local exchanges completely connected to each other with two-way links
  - Traffic matrix T describing the busy hour traffic interest (in erlangs) given below
  - Fixed (direct) routing: calls are routed along shortest paths.
  - Mean holding time h = 3 min.
- **Task**:
  - Dimension trunk network links according to the GoS requirement $B \leq 1\%$

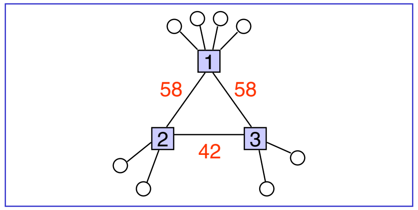
Example: dimensioning the links (2)



area	1	2	3	sum
1	60	15	15	90
2	30	30	15	75
3	30	15	30	75
sum	120	60	60	240

- **Link 1-2** (betw. nodes 1 and 2):
  - offered traffic 1 → 2: $a(1,2) = T(1,2) = 15$ erlang
  - offered traffic 2 → 1: $a(2,1) = T(2,1) = 30$ erlang
  - total offered traffic: $a(1\text{–}2) = T(1,2) + T(2,1)$ $= 15+30 = 45$ erlang
  - required capacity: $n(1\text{-}2) \geq \min\{i \mid \mathrm{Erl}(i,45) \leq 1\%\}$ $\Rightarrow$ ***n*(1-2) ≥ 58 channels**

Example: dimensioning the links (3)



area	1	2	3	sum
1	60	15	15	90
2	30	30	15	75
3	30	15	30	75
sum	120	60	60	240

- **Link 1-3**:
  - total offered traffic: $a(1\text{-}3) = T(1,3) + T(3,1)$ $= 15+30 = 45$ erlang
  - required capacity: $n(1\text{-}3) \geq \min\{i \mid \mathrm{Erl}(i,45) \leq 1\%\}$ $\Rightarrow$ ***n*(1-3) ≥ 58 channels**
- **Link 2-3**:
  - total offered traffic: $a(2\text{-}3) = T(2,3) + T(3,2)$ $= 15+15 = 30$ erlang
  - required capacity: $n(2\text{-}3) \geq \min\{i \mid \mathrm{Erl}(i,30) \leq 1\%\}$ $\Rightarrow$ ***n*(2-3) ≥ 42 channels**

Table: $B = \mathrm{Erl}(n,a)$

- ***B* = 1%**
  - ***n***: ***a***:
  - 35 channels 24.64 erlang
  - 36 channels 25.51 erlang
  - 37 channels 26.38 erlang
  - 38 channels 27.26 erlang
  - 39 channels 28.13 erlang
  - 40 channels 29.01 erlang
  - 41 channels 29.89 erlang
  - 42 channels 30.78 erlang
  - 43 channels 31.66 erlang
  - 44 channels 32.55 erlang
  - 45 channels 33.44 erlang

***B* = 1%**

n:	*a*:
50 channels	37.91 erlang
51 channels	38.81 erlang
52 channels	39.71 erlang
53 channels	40.61 erlang
54 channels	41.51 erlang
55 channels	42.41 erlang
56 channels	43.32 erlang
57 channels	44.23 erlang
58 channels	45.13 erlang
59 channels	46.04 erlang
60 channels	46.95 erlang

End-to-end blocking probability

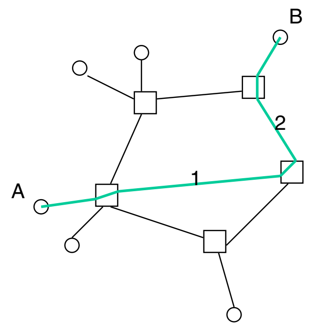
- Thus far we have concentrated on the single link case, when calculating the call blocking probability B_c
- However, there can be many (trunk network) links along the route of a (long distance) call. In this case it is more interesting to calculate the total **end-to-end blocking probability** B_e experienced by the call. A method (called **Product Bound**) to calculate B_e is given below.
- Consider a call traversing through links $j = 1, 2, \ldots, J$. Denote by $B_c(j)$ the blocking probability experienced by the call in each single link j. Then

$$B_e = 1 - (1 - B_c(1))*(1 - B_c(2))*\ldots*(1 - B_c(J))$$

$$B_c(j)\text{'s small} \Rightarrow B_e \approx B_c(1) + B_c(2) + \ldots + B_c(J)$$

Example

- The call from A to B is traversing through trunk network links 1 and 2
- Let $B_c(1)$ and $B_c(2)$ denote the call blocking probability in these links
- Product Bound (PB):
  $B_e = 1 - (1 - B_c(1))(1 - B_c(2))$
  $= B_c(1) + B_c(2) - B_c(1)\,B_c(2)$
- Approximately:
  $B_e \approx B_c(1) + B_c(2)$



Literature

1 A. Olsson, ed. (1997)
- "Understanding Telecommunications 1"
- Studentlitteratur, Lund, Sweden

2 A. Girard (1990)
- "Routing and Dimensioning in Circuit-Switched Networks"
- Addison-Wesley, Reading, MA

TEKNILLINEN KORKEAKOULU
Tietoverkkolaboratorio

11. Traffic management in ATM

Contents

Traffic management

- Problems:
  - Traffic is **random** in nature (varying unpredictably)
    - every now and then, **congestion** occurs (unavoidably)
  - Traffic sources may behave “badly”, i.e. try to use more than their fair share
- **Traffic management** is needed in order to
  - achieve the required QoS and performance under these circumstances
  - protect the network and other users against badly behaving traffic sources
- Two approaches to “manage” congestion
  - **predictive** methods to **avoid** congestion (before it occurs)
  - **reactive** methods to **alleviate** and **remove** congestion (after it has occurred)

Examples on traffic management

- Telephone network (circuit switching):
  - only predictive methods:
    - call admission control
    - resource reservation (bandwidth)
- X.25 (connection oriented packet switching):
  - only predictive methods:
    - call admission control (CAC)
    - resource reservation (buffers)
    - window-based open-loop flow control
- IP (connectionless packet switching):
  - only reactive methods:
    - window-based closed-loop flow control

RTT * BW

- The product of round-trip time (RTT) and bandwidth (BW) determines
  - the amount of information transmitted before the first feedback signal to adjust the transmission rate is received from the network and the destination
- In broadband wide-area-networks (WAN) this product can be very large $\Rightarrow$ reactive methods alone are insufficient
- Example:
  - Assume that
    - distance between two users is 1500 km
    - Bandwidth BW = 100 Mbps
  - The two-way propagation delay will be
    - 2*1500/300,000 s = 0.01 s
  - Thus, the product of RTT and BW is at least
    - 0.01*100,000,000 bits = 1,000,000 bits = 1 Mbit

Contents

History

- Traditionally: dedicated networks for different services
  - For example: telephone, telex, data, broadcast networks
  - Optimized for the corresponding service
- Need for integration of all services into a single ubiquitous network
  - “One policy, one system, universal service” (T. Vail, AT&T’s first president)
  - Early 80’s: Research on Fast Packet Switching started
- Answer from the “Telecom World” : **B-ISDN**
  - 1985: B-ISDN specification started by Study Group SGXVIII of CCITT
  - 1988: Approval of the first B-ISDN recommendation (I.121) by CCITT
- Chosen implementation method: **ATM**
  - 1990: ATM chosen for the final transfer mode for B-ISDN by CCITT
  - 1991: ATM Forum founded
    - to accelerate the development of ATM standards
    - to take into account the needs of the “Computer World”

Technical choices

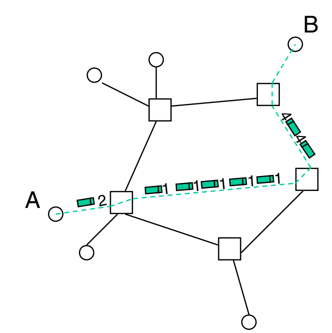
- Connection oriented
  ⇒ resource reservation possible
  ⇒ guaranteed Quality of Service (QoS) possible

- Packet switching
  ⇒ statistical multiplexing
  ⇒ flexibility (any bit rate possible) and efficiency (high utilization)

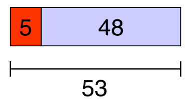
- Packets small and fixed-length (called cells)
  ⇒ cell switching
  ⇒ fast switching

ATM

- Connection oriented:
  - virtual connections set up end-to-end before information transfer
  - resource reservation possible but not mandatory
- All information carried in short, fixed-length packets i.e. **cells**
  - along the route chosen for that virtual connection
  - statistical multiplexing at nodes
  - identifier label (local address) at cell header (VPI/VCI)
  - no error detection/recovery for the information field



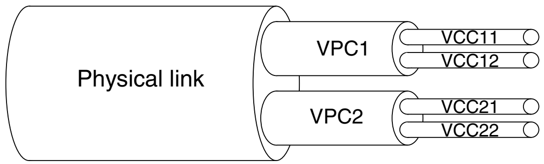
Cell



- **Cell = short, fixed-length packet**
  - Total length = **53 bytes** (octets) = 424 bits
  - **Header**: 5 bytes
    - **GFC**, generic flow control (4 [0] bits at UNI [NNI])
    - **VPI**, virtual path identifier (8 [12] bits ⇒ 256 [4096] values)
    - **VCI**, virtual channel identifier (16 bits ⇒ 65,536 values)
    - **PT**, payload type (3 bits)
    - **CLP**, cell loss priority (1 bit)
    - **HEC**, header error control (8 bits)
  - **Information field**: 48 bytes
    - compromise (Europe: 32 bytes; USA: 64 bytes; (64 + 32)/2 = 48)
    - carried transparently, without error detection/recovery

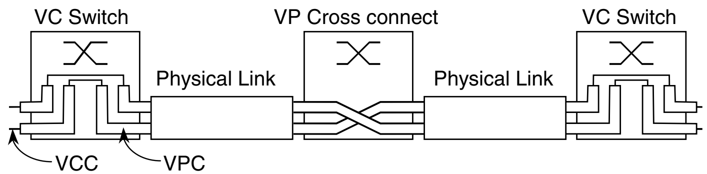
Virtual connections

- Basic connection: **Virtual Channel Connection** (**VCC**)
  - identified by the **VPI/VCI** pair (24 [28] bits) in **each** cell header
    - max. 16,777,216 [268,435,456] VCCs per physical link
  - VPI and VCI fields are **local** labels
    ⇒ reuse possible in different physical links ⇒ scalability
- Aggregated connection: **Virtual Path Connection** (**VPC**)
  - identified by the **VPI** field (8 [12] bits) in **each** cell header
    - max. 256 [4096] VPCs per physical link
  - consists of the VCCs with the same VPI (can be switched together!)



Virtual paths

- Pros
  - faster connection establishment
  - easier network management
  - differentiated QoS possible
  - virtual networks possible
- Cons
  - reduced statistical multiplexing gain (statistical multiplexing only inside paths not between them)



Contents

Traffic sources

- **Traffic source** =
  end-system application generating the cells of an ATM connection
- **Real-time** traffic sources
  - e.g. interactive voice and video
  - stream (rate-oriented) traffic
    - require tight delay and delay variation constraints at cell level
    - May require a dedicated bandwidth (e.g. 64 kbps for voice)
    - constant-bit-rate (CBR) vs. variable-bit-rate (VBR) traffic sources
      - voice and video can be either CBR-coded or VBR-coded
- **Non-real-time** traffic sources
  - e.g. data messaging (such as E-mail) and data or image retrieval (such as WWW) applications
  - elastic (unit-oriented) traffic
    - does not require tight delay and delay variation constraints (at cell level)
      but typically benefits from shorter delays at “session” level
    - without any inherit rate requirement (the more the better)

Quality of service

- **Connection level** quality called **Grade of Service (GoS)**:
  - low connection blocking probability required by all connection types
  - Operators dimensioning choices affect this
- **Cell level** quality called **Quality of Service (QoS)**:
  - Depends on the connection type
  - For real-time traffic sources:
    - End-to-end delay and especially delay variation are important
    - Certain transmission rate should be guaranteed
    - Instead, a certain degree of information corruption or loss is tolerable
  - For non-real-time traffic sources:
    - most important: uncorrupted and lossless information transfer
    - benefits from (but no strict constraints on) shorter delay
    - benefits from (but no strict constraints on) higher transmission rate

Service categories (1)

- Due to differences in the nature of traffic sources and also in the required QoS they impose
  - connections are divided into different **service categories**
    - ATM-Forum terminology [2]: 'Service category'
    - ITU-T terminology [3]: 'ATM transfer capability'
  - Functions, such as routing, connection admission control, and resource allocation are, in general, structured differently for different service categories
- During the connection establishment phase
  - A service category is chosen
  - more detailed description of the traffic characteristics and the QoS requirements is given by a set of traffic and QoS parameters defined in the **traffic contract**

Service categories (2)

- **Service categories** (according to ATM Forum [2]):
  - **CBR** = constant bit rate
    - real-time, guaranteed QoS
  - **VBR-rt** = variable bit rate, real-time
    - real-time, guaranteed QoS
  - **VBR-nrt** = variable bit rate, non-real-time
    - non-real-time, guaranteed QoS
  - **ABR** = available bit rate
    - non-real-time, no absolute QoS guarantees
    - Category specific flow control, where ABR sources must adjust their rate according to the feedback from the network
  - **UBR** = unspecified bit rate
    - non-real-time, no QoS guarantees at all

Source: [2]

Service categories (3)

- **CBR** (Constant Bit Rate)
  - mainly for real-time constant-bit-rate traffic sources requiring tightly constrained cell transfer delay and delay variation
    - e.g. cbr-coded interactive voice and video, circuit emulation
  - traffic characterized and policed in terms of Peak Cell Rate (PCR)
  - guaranteed QoS in terms of cell transfer delay (CTD), cell delay variation (CDV) and cell loss ratio (CLR)

Service categories (4)

- **VBR-rt** (Variable Bit Rate, real-time)
  - for real-time variable-bit-rate traffic sources requiring tightly constrained cell transfer delay and delay variation
    - e.g. vbr-coded interactive voice and video
  - traffic characterized and policed in terms of Peak Cell Rate (PCR), Sustainable Cell Rate (SCR) and Maximum Burst Size (MBS)
  - guaranteed QoS in terms of CTD, CDV and CLR
- **VBR-nrt** (Variable Bit Rate, non-real-time)
  - for non-real-time variable-bit-rate traffic sources expecting a low cell loss ratio
    - e.g. vbr-coded video retrieval
  - traffic characterized and policed in terms of PCR, SCR and MBS
  - guaranteed QoS in terms of CLR

Service categories (5)

- **ABR** (Available Bit Rate)
  - for non-real-time elastic traffic sources expecting a low cell loss ratio
    - e.g. transfer of large bulks of data (large files, images, video clips, etc.)
  - traffic initially specified by the maximum required bandwidth (PCR) and the minimum usable bandwidth (MCR)
  - however, traffic sources should be able to adjust their rate according to a specific feed-back control mechanism (ABR flow control)
  - guaranteed Minimum Cell Rate (MCR)
  - no absolute QoS guarantees ("low" cell loss ratio promised, fairness as a target)
- **UBR** (Unspecified Bit Rate)
  - for non-real-time elastic traffic sources without any QoS requirements
    - e.g. TCP/IP data traffic
  - no rate or QoS guarantees at all ("best effort" service)
  - relies totally on upper level traffic control (such as TCP)

Traffic Contract

- During the connection establishment phase, the user and the network negotiate a **Traffic Contract**
  - concerns traffic (i.e. transmitted cells) at the interface between the user and the network (**UNI**)
- Traffic Contract specifies
  - the service category of the connection
  - the traffic characteristics of the connection
    - given by a set of **traffic parameters** and **delay tolerances** (PCR, SCR, MBS, MCR, CDVT, BT)
  - Quality of Service (QoS) guaranteed for the connection by the network
    - given by a set of **QoS parameters** (maxCTD, peak-to-peak CDV, CLR)

Traffic parameters

- **PCR** = $1/T$ = peak cell rate (cells per second)
  - max. instantaneous transmission rate
  - defined for all connections
- **SCR** = $1/T_s$ = sustainable cell rate
  - max. rate at which the traffic source can transmit cells continuously
  - defined for VBR connections
- **MBS** = maximum burst size
  - max. number of cells that can be sent consecutively at the PCR
  - defined for VBR connections
- **MCR** = minimum cell rate
  - max. rate at which cells are guaranteed to be carried by the network
  - defined for ABR connections

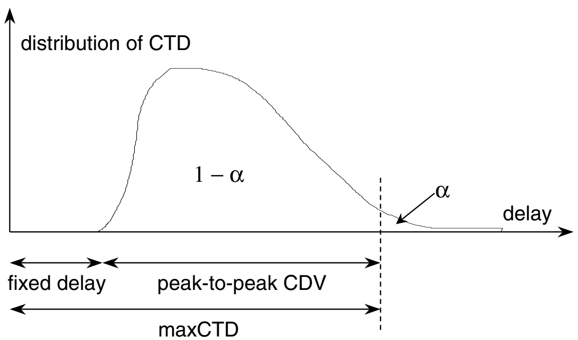
Delay tolerances

- **CDVT** = τ = cell delay variation tolerance
  - an upper bound for cell "clumping" due to ATM layer functions (independent of the traffic source), e.g.
    - cell multiplexing (slotted system!)
    - insertion of physical layer overhead or OAM cells
  - defined for all connections
- **BT** = τ_s = burst tolerance
  - $\mathrm{BT} = (\mathrm{MBS} - 1)(T_s - T)$ where
    - $T_s = 1/\mathrm{SCR}$
    - $T = 1/\mathrm{PCR}$
  - defined for VBR connections

QoS parameters (negotiated)

- **maxCTD** = maximum cell transfer delay
  - maximum end-to-end cell transfer delay (more precisely: $1-\alpha$ fractile)
  - end-to-end cell transfer delay (CTD) includes
    - fixed part: propagation, transmission and pure processing delays
    - randomly varying part: synchronization and queueing delays
  - defined for CBR and VBR-rt connections (that is: for real-time connections)
- **peak-to-peak CDV** = cell delay variation
  - difference between maxCTD and the fixed part of CTD
  - defined for CBR and VBR-rt connections (that is: for real-time connections)
- **CLR** = cell loss ratio
  - CLR = nr of lost cells / total nr of transmitted cells
  - defined for CBR and VBR connections

maxCTD and peak-to-peak CDV



Source: [2]

ATM Service Category Parameters

	ATM Layer Service Category				
	CBR	VBR-rt	VBR-nrt	UBR	ABR
Traffic parameters:					
PCR, CDVT	specified				
SCR, MBS	n/a	specified		n/a	
MCR	n/a				specified
QoS parameters:					
Peak-to-peak CDV	specified		unspecified		
MaxCTD	specified		unspecified		
CLR	specified			unspecified	

Source: [2]

Contents

Traffic and congestion control in different time scales

- In ATM:

Traffic control = predictive traffic handling methods
Congestion control = reactive traffic handling methods

Response time:	Traffic control functions:	Congestion control functions:
Long term (hours - days)	• Resource Management using Virtual Paths	
Connection duration (secs - mins)	• Connection Admission Control (CAC)	
Round trip propagation time (ms)	• Fast Resource Management	• Explicit Forward Congestion Indication (EFCI) • ABR Flow Control
Cell insertion time (µs)	• Usage Parameter Control (UPC) • Priority Control • Traffic Shaping	• Selective Cell Discard • Frame Discard

Traffic control functions (1)

- **Resource management using virtual paths (VP)**
  - grouping of similar VCCs, e.g. according to traffic type
  - differentiated QoS for different VCC groups
  - VPC performance objectives determined by the most demanding VCC QoS requirement
  - Predictive, static bandwidth allocation
- **Connection admission control (CAC)**
  - to protect the network from excessive loads
  - during the connection establishment phase, the network decides whether the requested connection can be established or should be rejected
  - when established, the traffic characteristics and the QoS requirements of the connection are specified in a traffic contract
  - if the connection behaves as specified in the traffic contract (compliant connection), the network should guarantee the required QoS

Source: [2]

Traffic control functions (2)

- **Usage parameter control (UPC)**
  - to protect the other connections against badly behaving connections
  - during the information transfer phase, the network monitors the connection whether the traffic conforms to the traffic contract (traffic policing)
  - the non-conforming cells are either discarded or tagged as low-priority cells (CLP = 1)
- **Traffic shaping**
  - to smooth out a traffic flow to conform with the contract and reduce cell clumping
  - users can ensure that the traffic conforms to the traffic contract
  - network may also shape the traffic (within the limits of the traffic contract)

Source: [2]

Congestion control functions (1)

- **Explicit forward congestion indication (EFCI)**
  - the EFCI bit in the cell header is used to warn the end systems (first the destination, and then the source) about (impending) congestion
  - a network element in a congested state (or if it is impending) may set the EFCI bit (EFCI = 1) --- but cannot never reset it (EFCI = 0)
  - reasonable if end-systems cooperate reducing their cell rates
- **ABR flow control**
  - in the ABR service, the source adapts its rate to changing network conditions
  - ABR flow control is based on specific resource management (RM) cells
    - each RM cell include the **explicit cell rate** (ER) field
  - instead of one bit EFCI information (included in every cell), a congested network element can express, by the ER field, an upper bound for the rate at which the source is permitted to transmit cells

Source: [2]

Congestion control functions (2)

- **Selective cell discard**
  - traffic sources may tag less important cells as low-priority cells (CLP = 1)
  - UPC can tag non-conforming cells as low-priority cells
  - a congested network element (beyond UPC) may selectively discard low-priority cells or cells belonging to a non-compliant connection
- **Frame discard**
  - If a network element needs to discard cells, it is in many cases more efficient to discard at the frame (packet) level rather than at the cell level
  - Frame discard may be used whenever it is possible to delineate frame boundaries

Source: [2]

Contents

Connection admission control (CAC)

- During the connection establishment phase, CAC determines in each network node
  - whether the requested connection can be accepted or not
  - the traffic parameters needed by UPC
  - allocation of network resources and routing decisions
- Resource allocation is due to the operator itself
  - no method is standardized (but is open for competition)
  - Per quality of service class
  - should be based on mathematical traffic models
- A new connection can be accepted only if
  - the network has resources to support the new connection while at the same maintaining the agreed QoS level of existing connections
- Rejections mean a higher connection blocking probability
  - the problem (of lacking resources) is pushed up to a higher time scale (cell to call)
  - the only (permanent) solution is to increase the network resources

Resource allocation

- CBR
  - bandwidth allocated according to PCR
- VBR
  - bandwidth allocated according to so called **Effective Cell Rate** (ECR)
  - ECR is determined by the operator
    - SCR ≤ ECR ≤ PCR
    - ECR depends not only on the traffic characteristics and the QoS requirements of the traffic source but also on the network resources (e.g. link capacities)
- ABR
  - bandwidth allocated according to MCR
  - The actual rate controlled during the connection
- UBR
  - no bandwidth allocation at all nor guaranties

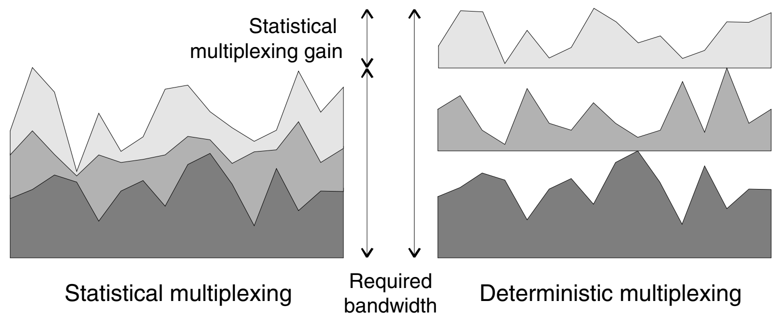
Example

- Consider a VPC dedicated to a single service category
  - C = VPC's PCR (capacity of logical link)
  - Existing connections: $i = 1,\ldots n$
  - New connection request of the same type: $n + 1$
- The new connection is accepted if …
  - CBR: … $\text{PCR}_1 + \ldots + \text{PCR}_{n+1} \leq C$
  - VBR: … $\text{ECR}_1 + \ldots + \text{ECR}_{n+1} \leq C$
  - ABR: … $\text{MCR}_1 + \ldots + \text{MCR}_{n+1} \leq C$
  - UBR: … always!

Effective cell rate

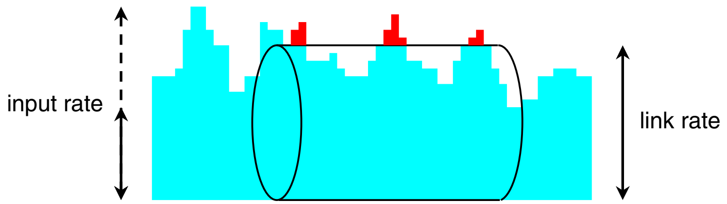
- Useful especially for VBR service category
- Often called **effective bandwidth**
  - then units are bits per second (not cells per second)
- Take into account the gains of statistical multiplexing
  - The bandwidth need can be compensated by buffering (the more buffer space the smaller the ECR)
- Assume that the traffic sources are independent
  - Then the probability that the source peak rates coincide is small
- Note.
  - In the worst case VBR traffic sources are totally synchronized, and the bandwidth allocation has to be according to peak cell rate (though buffering may alleviate this a little)

Statistical multiplexing of independent VBR sources



Effective bandwidth (1)

- Consider n **independent** and identical VBR traffic sources
  - Let R_i denote the instantaneous cell rate of source i
  - Assume that the corresponding VBR connections are carried by a VPC of capacity C
- Quality requirement: $\mathrm{CLR} \le \varepsilon$
- In the so called **bufferless** model: $\mathrm{CLR} \approx P\{\Sigma_i R_i > C\}$
- Now, given ε, we try to determine $e = \mathrm{ECR}_i$ so that $P\{\Sigma_i R_i > ne\} = \varepsilon$



Effective bandwidth (2)

- Denote
  - $m = E[R_i]$, $v = \mathrm{Var}[R_i]$
- Due to independency,
  - $M := E[\Sigma_i R_i] = nm$, $V := \mathrm{Var}[\Sigma_i R_i] = nv$
- By the Central Limit Theorem,
  - $\Sigma_i R_i \approx \mathrm{N}(M,V) = \mathrm{N}(nm,nv)$
- Thus,

$$P\{\textstyle\sum_i R_i > ne\} = \varepsilon \Leftrightarrow ne = nm + z_{1-\varepsilon}\sqrt{nv} \Leftrightarrow e = m + z_{1-\varepsilon}\sqrt{\frac{v}{n}}$$

  - Here z_p refers to the p-fractile of the N(0,1) distribution
- This simple model describes the most essential characteristics of effective cell rate e (also called: effective bandwidth)
  - e decreases as the number of sources, n, increases
  - e tends to the mean rate m as n increases without limits

Example

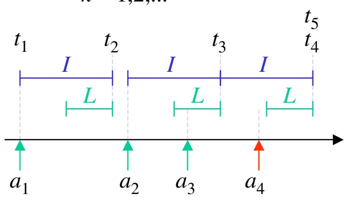
- Consider such VBR traffic sources for which
  - $m = E[R_i] = 32$ kbps, $v = \mathrm{Var}[R_i] = (32 \text{ kbps })^2$
- Assume that the corresponding connections are carried by a VPC with
  - $C = 2$ Mbps
- **Question**: What is the maximum number of connections that CAC can accept, if the QoS requirement is
  - $\mathrm{CLR} \le \varepsilon = 10^{-4}$
- **Answer**:
  - Since $z_{1-\varepsilon} = z_{0.9999} = 3.719$, we are looking for such n^* that
    - $n^* = \max\{n \mid nm + z_{1-\varepsilon}(nv)^{1/2} \le C\} = \max\{n \mid 32n + 119(n)^{1/2} \le 2000\}$
  - Equation $32x + 119(x)^{1/2} = 2000$ is solved by $x = 39.2$. Thus
    - $n^* = 39$
    - $e = C/n^* = 51$ kpbs

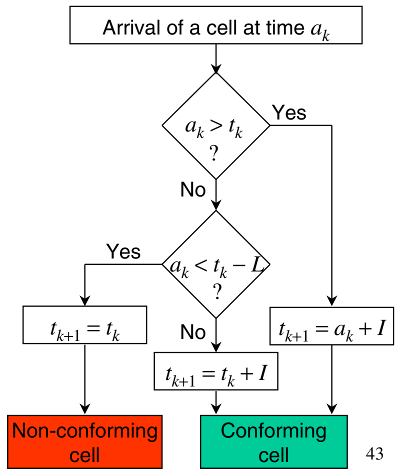
Usage Parameter Control (UPC)

- During the information transfer phase, UPC controls the cell flow of the connection (at the UNI interface) by
  - passing the conforming cells
  - tagging (CLP = 0 → CLP = 1) or discarding the non-conforming cells
- The conformance of cells is determined by the **Generic Cell Rate Algorithm** (GCRA)
  - $\mathrm{GCRA}(I,L)$ is specified by the following two time parameters:
    - I = Increment = $1/R$, where R is the target rate (under policing)
    - L = Limit = the maximum negative deviation from the "scheduled" time (delay tolerance)
- Peak cell rate policing:
  - $\mathrm{GCRA}(1/\mathrm{PCR},\mathrm{CDVT}) = \mathrm{GCRA}(T,\tau)$
- Sustainable cell rate policing:
  - $\mathrm{GCRA}(1/\mathrm{SCR},\mathrm{BT}) = \mathrm{GCRA}(T_s,\tau_s)$

GCRA(I,L)

- Notation:
  - t_k = theoretical arrival time of the kth cell
  - a_k = arrival time of the kth cell
- Initialization at time a_1 :
  - $t_1 = a_1$
- Algorithm run at times a_k
  - $k = 1,2,...$

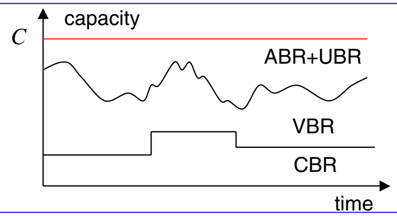




Contents

“Best Effort” services

- Guaranteed QoS for CBR and VBR connections ⇒ dedicated resources ⇒ low utilization
- Available free capacity can be offered to the (lower priority) ABR and UBR connections (so called “best effort” services)
- Because of no explicit resource reservation for ABR and UBR connections,
  - essentially larger buffers are needed
  - fairness is an important aspect

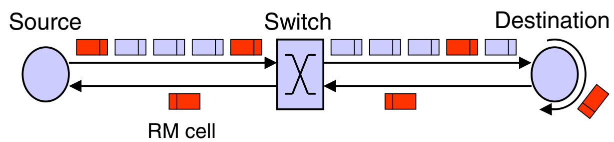


ABR flow control

- Target:
  - to achieve a high link utilization and a low cell loss ratio at the same time
- Traffic of an ABR connection initially specified by
  - the maximum required bandwidth (PCR) and
  - the minimum usable bandwidth (MCR)
- To adjust the total rate of ABR connections to fit with the available rate (left over by CBR and VBR connections), the transmission rate of each ABR connection is controlled continuously by the network (closed-loop control)
  - the allowed (instantaneous) transmission rate of a single connection can have any value between MCR and PCR during the connection holding time
  - implementation by means of specific Resource Management (RM) cells
- This control method is clearly reactive
  - Reaction according to buffer occupancy or change in occupancy

Rate based ABR flow control

- Source transmits RM cells after each n ordinary cells
  - telling its current rate (CCR) and the rate at which it wishes to transmit (ER)
- Switches along the (fwd) route of the connection read this information
  - they compute the fair share of the link capacity
  - they can reduce the value of the ER field
- Destination returns the RM cell to the source
  - it can also reduce the value of the ER field
- Switches along the (bwd) route of the connection read this information
  - they can still reduce the value of the ER field
- Source adjusts its current rate according to the feedback information



Literature

1 W. Stallings (1998)
- "High-Speed Networks: TCP/IP and ATM design principles"
- Prentice Hall, New Jersey

2 ATM-Forum, Technical Committee
- "Traffic Management Specification, Version 4.0"
- April 1996

3 ITU-T, Study Group 13
- "Recommendation I.371: Traffic Control and Congestion Control in B-ISDN"
- July 1995

TEKNILLINEN KORKEAKOULU
Tietoverkkolaboratorio

12. Traffic management in the Internet

Contents

- Introduction
- IP-networks
- Traffic and congestion control in the Internet
- QoS architectures in the Internet

Traffic management

- Problems:
  - Traffic is **random** in nature (varying unpredictably)
    - every now and then, **congestion** occurs (unavoidably)
  - Traffic sources may behave “badly”, i.e. try to use more than their fair share
- Traffic management is needed in order to
  - achieve the required QoS and performance under these circumstances
  - protect the network and other users against badly behaving traffic sources
- Two approaches to “manage” congestion
  - **predictive** methods to **avoid** congestion (before it occurs)
  - **reactive** methods to **alleviate** and **remove** congestion (after it has occurred)

Why not ATM? (1)

- ATM chosen as the implementation method for connection oriented packet (=cell) switching network
- As it is able to offer
  - traffic management
  - voice/data integration
    - Using traffic classes: CBR, VBR, ABR, UBR
  - signaling and connection set up for QoS
- But in order to be a plausible end-to-end solution
  - New interfaces (new HW and SW) at the end-systems!
    - ATM signaling (UNI) considered complex, thus HW + SW for it has been considered expensive for end-systems
- Other shortcomings
  - Difficult to use the QoS capabilities of ATM
    - standards do not consider, how applications should choose all the traffic contract parameters
  - Design based on short cells favors voice
  - Too much, too soon ...

Why not ATM? (2)

- More importantly, while the HW+SW for ATM devices were being developed, the Internet exploded (WWW)!
  - the web browser created an easy and inexpensive access to the Internet
- Exponential growth in the number of users and traffic amounts Internet thus won the race to the desktop!
  - Internet is now ubiquitous (everywhere)
  - But still not able to service all different types of traffic
- The current approach is to try to enhance Internet’s QoS capabilities
  - rather than build a new network (based e.g. on ATM)
  - ATM still used in the core network

Current trends driving the evolution

- Decreasing HW costs (CPU, memory)
  - Increasing computing power and more powerful machines
    - more bandwidth hogging applications
- The link speeds increase dramatically
  - 1993: 100 Mbit/s (FDDI), 2000: 1 Tbit/s (dense WDM technologies)
- More traffic or more capacity?
  - If traffic is much greater than capacity
    - engineering and traffic control mechanisms are required
    - If the opposite holds
      - just add more capacity ...
  - Capacity likely to be a problem in WANs
- 1999 the amount of data traffic exceeded voice traffic

Source: [4]

Contents

- Introduction
- IP-networks
- Traffic and congestion control in the Internet
- QoS architectures in the Internet

Internet

- Wide spread network technology
  - reaches the desktop
    - The required protocol handling capacities are built into the computers and their applications
  - The web browser has enabled the exponential yearly growth of user and traffic amounts
    - The browser has made possible integration of new types of applications
      - Streaming media, interactive chats, games etc.
- IP-network is based on best-effort service
  - Traditionally no QoS guarantees not even a notion of quality classes
  - Roots in the technology for computer communication based on connectionless packet networks and routers (ARPANET)
    - Network nodes do not store state information on the users/connections
  - Originally designed for delivery of non-real time messages
    - File transfer, remote use of computers, e-mail
- New service architectures (DiffServ, IntServ) try to bring quality differentiation

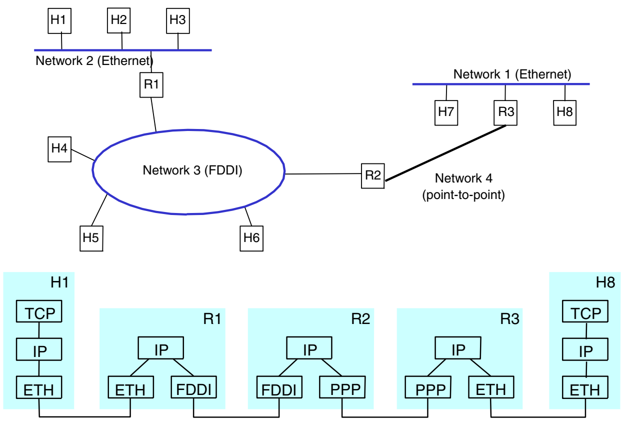
Flows

- Flow is an approximation of the traffic of higher layers as seen by the network layer (i.e. routers)
- A **flow** is a sequence of IP packets related to each other through a mutual source and destination
  - In a coarse grained classification
    - Only based on source and destination address
  - In a more specific classification
    - Can be based on e.g. the protocol (e.g. TCP, UDP, HTTP, FTP,…)
  - Sequential flows separated from each other using timers
    - Sequential packets belong to the same flow, only if their inter arrival separation is small enough
  - Note. The definition of a flow is very flexible. How to classify packets to flows is a separate research task
    - Choosing the granularity
    - Choosing the timeout

Quality of Service (QoS)

- Quality of service can be studied both at the flow and the packet level
- Two approaches on the **Flow level**
  - **All flows**
    - Total network throughput is an important measure of efficiency
    - Fairness of the division of bandwidth between flows has to be considered
  - **Per flow**
    - For elastic traffic the most important quality measure is the throughput, or more specifically the goodput
- **Packet level**
  - Streaming traffic requires
    - small end-to end delays and delay variations for its packets
    - but is able to sustain some packet losses or corruptions
  - Elastic traffic requires
    - very small packet losses

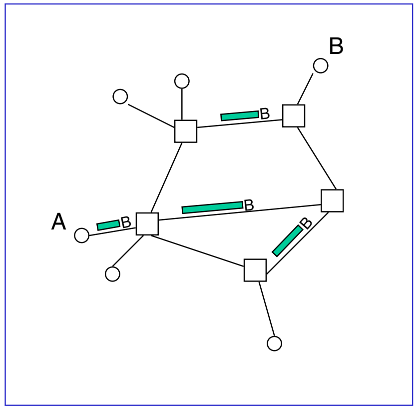
IP-protocol stack



Lähde: [2]

IP

- **IP** = Internet Protocol
- **Connectionless**:
  - No connection establishment
  - No resource reservations
- Information transfer as discrete packets
  - Variable length
  - Includes a header with the global address of the destination
- **Best Effort** –service paradigm
  - Network nodes, i.e. routers, forward packets “as well as possible”
  - Packets may be lost, delayed or their order may change



IP-packet

- IP-packet = datagram
  - Variable length
  - **Header** (min 20 bytes) [RFC791]:
    - **Version**: Current version 4
    - **IHL**: length of the header
    - **TOS**: packet prioritization
    - **TotalLength**:
    - **Ident**, **Flags**, **Offset**: fragmentation
    - **TTL**: time to live (in hops)
    - **Protocol**: What upper layer protocol e.g. TCP (6), UDP (17)
    - **Checksum**: for header error detection
    - **SourceAddress**, **DestinationAddress**: Internet addresses

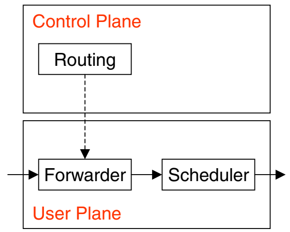
0 4 8 15 16 31

Version	IHL	TOS	TotalLength	
Ident			Flags	Offset
TTL		Protocol	Checksum	
SourceAddress				
DestinationAddress				
Options (variable)			Padding (variable)	
Data				

IP header	Data

Handling of IP packets in BE-routers

- Packets are **forwarded:**
  - A packet from the incoming buffer is chosen
  - Destination address is studied
  - The corresponding outgoing port is looked up from the routing table
  - And the packet is delivered to this port
- Packet **scheduling**
  - Packet is chosen from the outgoing buffer
  - And its sending is triggered on the outgoing link
  - Most common scheduling algorithm: **FIFO**
- Compare to **routing**:
  - A distributed process, where the routing tables are formed



Routing

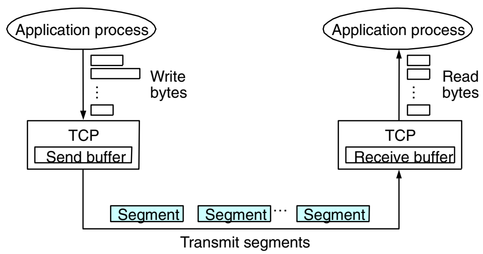
- Internet is divided into routing domains
- The routing task can be divided hierarchically
  - **Intra domain routing** tries to find the optimal routes
    - Find the optimal route between any two nodes inside the domain, when the topology and link costs are known
    - Different algorithms to solve this optimization problem
      - In the **Bellman-Ford**-algorithm nodes inform their neighbors their distances to all other nodes inside the domain e.g. RIP
      - In the **Dijkstra**-algorithm nodes inform all other nodes inside the domain of their distance to their neighbors e.g. **OSPF**
    - Usually the link cost (i.e. the distance between neighbor nodes) is one
      - Then the optimal choice minimizes the number of hops
  - **Inter domain routing** tries to assure reachability
    - Find some route between two domains, when the topology between the domains is known, e.g. **BGP**

End-to-end protocols in the transport layer

- On top of the network layer there is a transport layer
  - Takes care of handling the IP-packets in the terminals
- **TCP** = Transmission Control Protocol
  - **Connection oriented**
  - Meant for elastic, non-real time traffic
  - No absolute QoS guarantees, but **reliable byte stream** data transfer
  - Protocol specific flow and congestion control mechanisms for traffic control
    - Based on the use of an adaptive sliding window
- **UDP** = User Datagram Protocol
  - **connectionless**
  - Applicable for transactions (interactive traffic, with short transfers)
  - Used (when no other choice) for streaming, real time traffic
    - But then on top of UDP, other protocols needed, e.g. RTP
  - No QoS guarantees, **unreliable**

TCP

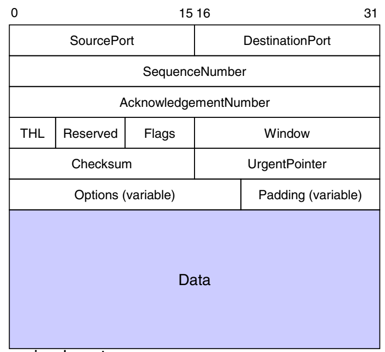
- **TCP** = Transmission Control Protocol
  - **Connection oriented end-to-end transmission layer protocol**
  - On top of IP for a reliable byte stream transfer
    - The delivery of packets in the right order is checked using acknowledgements and retransmissions
    - **flow control**: prevents over flooding the receiver
    - **congestion control**: prevents over flooding the network

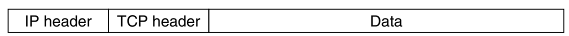


Lähde: [2]

TCP-packet

- TCP-packet = **segment**
  - Variable length
  - Transmitted through the network inside an IP-packet
  - **Header** (min. 20bytes) [RFC793]:
    - **SourcePort**, **DestinationPort**
    - **SequenceNumber**: the first byte of the first segment
    - **AcknowledgementNumber**: the next byte to be received
    - **THL**: length of header
    - **Flags**: used for indication of opening and closing of a connection
    - **Window**: the amount of bytes that can be received next
    - **Cheksum**: mandatory





UDP

- **UDP** = User Datagram Protocol
  - **Connectionless end-to-end transmission layer protocol**
  - On top of IP, but only for multiplexing
    - No guarantees of packet transfer (unreliable)
    - No flow control: can over flood the receiver
    - No congestion control: can over flood the network
- UDP-packet = **datagram**
  - Variable length
  - Transmitted through the network inside an IP-packet
  - **Header** (8 byte) [RFC768]:
    - **SourcePort**, **DestinationPort**:
    - **Length**:
    - **Checksum**: optional

0 – 15	16 – 31
SourcePort	DestinationPort
Length	Checksum
Data	

IP header	UDP header	Data

Contents

- Introduction
- IP-networks
- Traffic and congestion control in the Internet
- QoS architectures in the Internet

Traffic and congestion control mechanisms in different time scales

Response time:	**Proactive** methods:	**Reactive** methods:
Long term (hours - days)	- Routing algorithm - MPLS Traffic Engineering	
Connection duration (secs - mins)		
RTT (ms - secs)	- TCP Vegas congestion avoidance	- TCP flow control - TCP congestion control
Packet handling time (µs - ms)	- Scheduling - Active queue management	- Queue management

source based methods

router based methods

Source based mechanisms

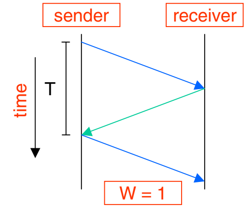
- **TCP flow control**
  - Prevents the source from over flooding the receiver
  - Implemented using the adaptive sliding window mechanism
    - The receiver indicates how many bytes can be received
- **TCP congestion control**
  - Prevents the source from over flooding the network
  - Implemented using the (same!) adaptive sliding window mechanism
    - The network however is not able to indicate how many bytes can be sent
    - The source has to determine, when the network is congested
      - Indicated by a packet loss
    - When a packet is lost, the window is decreased
    - Otherwise the window is increased to find out the state of the network
      - but never higher than the receiver advertised window
- **TCP Vegas congestion avoidance**
  - Aim is to constrain the number of packets of a flow in a bottleneck link
    - And thus avoid congestion
  - Window size is controlled by comparing expected and shortest round trip times

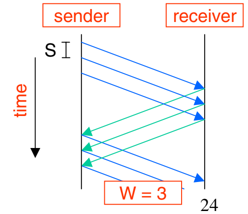
Router based mechanisms

- **Packet scheduling**
  - What packet to forward next
  - **FIFO** (First In First Out): packets sent in the order they arrived
  - Per flow based mechanisms
    - For each active flow its own queue/buffer
    - **Fair Queuing** (FQ): packets of different flows chosen fairly and equally
    - **Weighted Fair Queuing**
  - Priority based mechanisms
- **Queue management**
  - What packet to discard from the buffer, when it is filling up
  - **Tail Drop**: packet arriving to a full buffer is discarded
  - **Random Drop**
- **Active queue management**
  - Discarding packets before the buffer is full
  - Prevents congestion build up and synchronization of TCP sources
  - e.g. **RED** (Random Early Detection)

Sliding Window

- For reliable data transfer, packets must be acknowledged
- If packets are lost or corrupted, they need to be detected using timeouts and retransmitted
- A simple method: **stop-and-wait**
  - A new packet is sent, when the previous one is acknowledged
  - Problem: link is under utilized
- More effective: **sliding window**
  - The size of the window W, indicates how many unacknowledged packets can be in transit
    - stop-and-wait: W = 1
  - The window size W, round trip time T and packet processing time S determine the sending rate R:
    - R = min{1/S, W/T}
  - The window thus has an effective upper bound:
    - W ≤ T/S (the bandwidth delay product!)



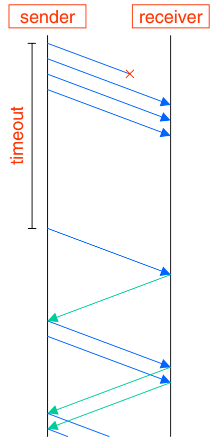


Adaptive sliding window for flow control

- Sliding window can be used for flow control
  - If the receiver processes its buffer with a constant rate C
    - The maximum window size is W ≤ CT,
    - This prevents the source from over flooding the receivers buffer
  - If the receiver processes its buffer randomly
    - It has to explicitly advertise how many packets it can receive
    - i.e. how many packets it can receive into its buffer next
- In TCP flow control the information of the advertised window is transmitted (in bytes) in the header of the acknowledging TCP packet in the Window-field

Adaptive sliding window for congestion control

- Sliding window can also be used for congestion control
  - Aim is to adjust the window size W according to the bandwidth C that is at the moment available for the flow
    - W = CT, T is the roundtrip time
  - This is not trivial, as both C and T change randomly and the user has no explicit information of C, T nor W
  - The source can implicitly infer the level of congestion from the packet losses
    - e.g. indicated by a timeout of the retransmission timer
    - When a packet is lost, the window must be reduced
      - The transmission rate decreases
    - When packets are successfully transmitted, the window can be increased
      - The transmission rate increases



TCP congestion control

- Original TCP [RFC793] (1981) used the adaptive sliding window mechanism only for flow control
  - But due to congestion, retransmissions were frequent, and the Internet experienced congestion collapses quite often
- Jacobson (1988) proposed the use of the adaptive sliding window for congestion control, named **TCP Tahoe**, with the following mechanisms:
  - **slow start**
  - **congestion avoidance**
  - **fast retransmit**
- Later Jacobson (1990) proposed **TCP Reno** with the addition of
  - **fast recovery**
- The TCP-congestion control mechanisms can be found in RFC2581(1999)

Contents

QoS architectures

- **Integrated services** (IntServ)
  - defined in IETF in 1995-97
  - fine grained
    - Provides QoS on a per connection (flow) basis
    - Absolute end-to-end quality guarantees
      - admission control and resource reservations
  - scalability problems due to per flow service and state space
- **Differentiated services** (DiffServ)
  - Defined in IETF 1998-2000
  - Tries to solve the scalability problems of IntServ by aggregating flows
  - coarse grained
    - Static Service Level Agreements (SLA)
    - QoS provided for large aggregated traffic streams (nothing is guaranteed for individual flows)
    - Only relative per hop quality guarantees
- **Compare to ATM**
  - Fine grained: QoS per each VC
  - Coarse grained: QoS per VPs i.e. aggregation of VCs

Integrated Services

- **Service classes** (= Service Category in ATM)
  - **Guaranteed service** (cf. CBR)
    - For real time sources, with an intolerant bound on the delay
    - Requires admission control, resource reservations and packet prioritization
  - **Controlled load service** (cf. VBR)
    - Tolerant sources that require a lightly loaded network
    - Requires admission control and isolation of service classes inside routers
  - **Best-effort** (cf. ABR and UBR)
    - on the IP level is based on TCP
- **Flow level mechanisms**
  - **Flow specification** (Flowspec)
    - Traffic specification (Tspec) and Request specification (Rspec)
  - **Admission control**
  - **Flow reservation** using signaling (Resource reservation protocol, RSVP)
- **Packet level mechanisms**
  - Packet **classification** and **prioritizing**
  - Packet **scheduling** per flow using WFQ

IntServ Flow specification

- Similar to the Traffic Contract of ATM
- Consists of 2 parts
  – Rspec (cf. ATM QoS parameters)
  – Tspec (cf. ATM traffic parameters)
- RSpec (Request Specification):
  – Only for guaranteed service: consists of the (queuing)delay target
  – Controlled load service has already a promise of a small load
- TSpec (traffic specification):
  – Source mean rate
  – Source peak rate
  – Burstiness
  – Other parameters to deal with the variable length packet unit

Token bucket

- The source is policed and characterized using the token bucket mechanism
  – source mean rate ↔ token generation rate (r)
  – burstiness ↔ depth (capacity) of the bucket (b)
  – other parameters to relate size of the token to the variable size of a packet
- Token bucket description:
  – The bucket is initially full of (b) tokens
  – Traffic that passes through the bucket uses up tokens according to its size
  – Tokens accumulate at rate r
  – Parameter m, is the smallest packet size, i.e. the token unit
- Token bucket determines if a packet is conformant (in-profile)
- Cf. ATM UPC and the GCRA-algorithm
  – Largest difference is that the arriving packets are of variable length

IntServ Flow level mechanisms in routers

- **Connection admission control**
  – Based on the flow description (RSpec and Tspec)
  – Preceded by some parameter control (token bucket and TSpec)
  – The router determines if the connection can be admitted without degrading the QoS of other already admitted connections
    - Depends heavily on the used scheduling mechanisms implemented by the router
- **Reservation protocol** (RSVP)
  – In connectionless networks user state is not stored by the network
  – RSVP tries to achieve robustness by using soft state information
    - Connection lifetimes are periodically refreshed (every 30 s)
    - RSVP tries to support multicast as effectively as unicast
  – Receiver oriented approach:
    - Receiver decides (opposite to connection oriented protocols) how much resources he needs
    - Receiver can change resource requirements dynamically with the refresh messages

RSVP example

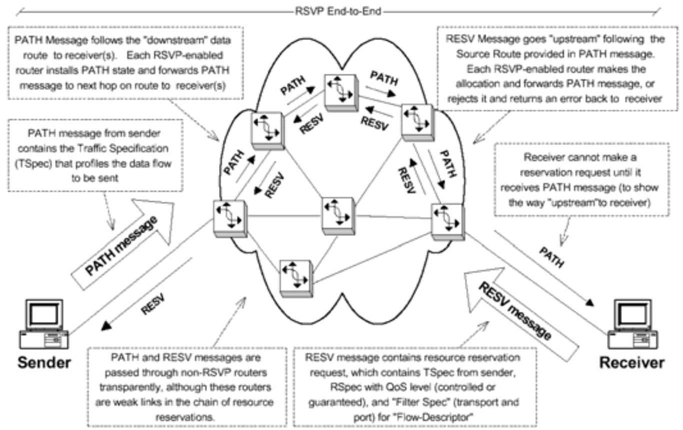


Figure 1: RSVP "PATH" and "RESV" messages are used to establish a resource reservation between a sender and receiver. There is an explicit tear-down of reservations also (not shown).

Source: [5]

IntServ Packet level mechanisms in routers

- **Packet classification**
  - Per flow
    - same source/destination address, protocol number, source/destination port
  - Associate each packet with the appropriate per flow resource reservation
- **Packet scheduling**
  - Traffic classes differentiated using scheduling algorithms
    - Per flow scheduling
    - Priority queuing
    - Weighted Fair Queuing (WFQ), capacity divided per flow based on weights
  - With scheduling the promised quality guarantees can be achieved
    - Vendors can try to create as efficient implementations as possible

Differentiated Services - Basic philosophy

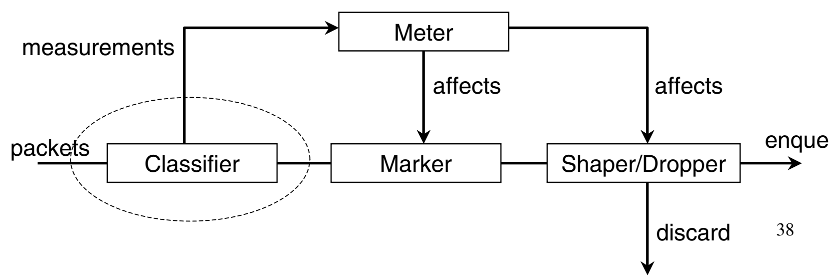
- Problem of **IntServ** is that everything is **per flow** and thus **not scalable**
  - Per flow resource reservation, scheduling, per flow state refreshing (RSVP)
- **DiffServ**
  - No signaling, but static service level agreements (SLA)
  - Network nodes implement defined per hop behaviors (PHBs)
    - not individual services, where parameter choices create infinite number of classes
    - Per hop behaviors for packets not flows!
- **QoS is differential**
  - No strict and absolute Qos guarantees
  - "Higher priority traffic gets better QoS than low priority traffic"
- Complex (**per flow**) traffic management functions **only at the edge** of the network
  - Packet **classification** and **marking**
  - Flow monitoring: **metering** and **shaping**
- Simple (**aggregate**) traffic management functions **in the core** of the network
  - Packet **scheduling** per PHB group using per class WFQ
  - Class and priority level based **queue management** mechanisms

Packet Per Hop Behavior (PHB)

- **Per hop behavior** (PHB)
  - The traffic class of DiffServ
  - Different PHBs provide different QoS
  - However, PHB definitions do not specify how the PHB should be realized
- **Expedited Forwarding** (EF) PHB
  - Aim is to offer small delays, delay variations and packet losses
  - the packet service rate equals or exceeds a specified peak rate
  - Implemented by giving EF class packets highest priority
- **Assured Forwarding** (AF) PHB Group
  - Each (max. 4) class has a guaranteed minimum rate
  - Within a class, there are 3 drop precedence priorities
  - In Total 12 different packet classes
- Traffic class of a packet marked in the packet header
  - 6 bits reserved, each bit combination is called a Differentiated Service Code Point (DSCP)
  - Type of Service field in IPv4, Traffic Class in IPv6

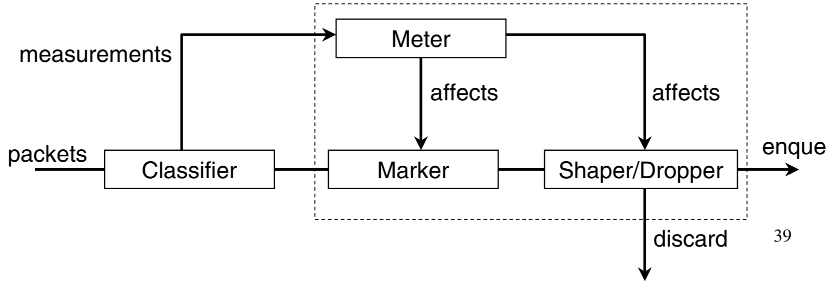
DiffServ - Edge Router Functions

- **Packet classification** determining the flow and the per hop class
  - BA (Behavior Aggregate) classifier
    - classifies packets based on DSCP only
  - MF (Multi-Field) classifier
    - selects packets based on the value of a combination of several header fields (DSCP, addresses, protocol id, ...)



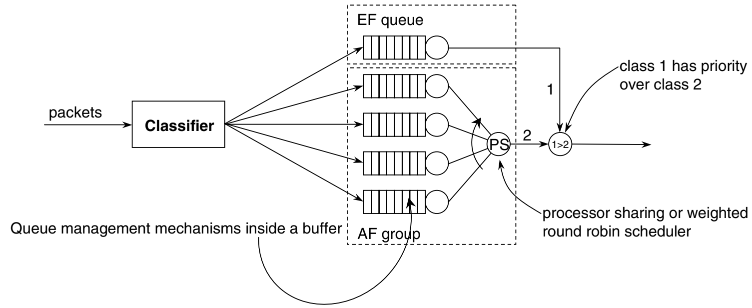
DiffServ - Edge Router Functions

- **Per flow mechanisms on packets** of a PHB
  - **Meter**: measure the traffic stream against a traffic profile, e.g. EF sending rate
  - **Marker**: are packets in-profile or out-of-profile
    - Can be used to mark packets into priorities
      - Lower priority, if traffic exceeds the PHB traffic profile
      - Higher priority, if clearly less traffic than reference PHB traffic profile
  - **Shaper/Dropper**: may delay out-profile packets (shaping) or drop them



DiffServ - Core Router Functions

- Performs only packet forwarding
  - Inspects the DSCP of each packet
  - Forwards the packet to the appropriate queue
- Efficient implementation of the queuing disciplines required by different PHBs not trivial!



Problems with Internet traffic management

- **DiffServ**
  - End-to-end QoS difficult to realize just based on PHB classes
    - Traffic monitoring and policing only at the edge, minimal control inside the network
    - Long lasting and high bandwidth (e.g. video) flows need per flow guarantees
  - SLAs are static, but network conditions and traffic needs change dynamically
  - Network dimensioning inside the DiffServ network is difficult, as there are no specific resource requirements
- **IntServ**
  - Per flow QoS impossible to implement in a scalable fashion
- It is hard to combine the flexibility of IP-networks and packet switching to the guarantees given by connection oriented networks so that all parties are satisfied!

Literature

1 W. Stallings (1998)
  - "High-Speed Networks: TCP/IP and ATM design principles"
  - Prentice-Hall, New Jersey

2 L.L. Peterson and B.S. Davie (2000)
  - "Computer Networks – A Systems Approach", 2nd edition
  - Morgan Kaufmann, San Francisco

3 V. Jacobson (1988)
  - "Congestion Avoidance and Control"
  - ACM SIGCOMM '88, Stanford, CA, August 1988

4 Professor Raj Jain's homepage
  - http://www.cis.ohio-state.edu/~jain/
  - e.g. course material from "Recent Advances in Networking (1999)"

5 Quality of Service Forum Homepage
  - http://www.qosforum.com/tech_resources.htm
  - e.g. white papers on QoS and links to related sites