Switch Fabrics

Switching Technology S38.3165
http://www.netlab.hut.fi/opetus/s383165

Switch fabrics

- Multipoint switching
- Self-routing networks
- Sorting networks
- **Fabric implementation technologies**
- Fault tolerance and reliability

Fabric implementation technologies

- Time division fabrics
  - Shared media
  - Shared memory
- Space division fabrics
  - Crossbar
  - Multi-stage constructions
- **Buffering techniques**

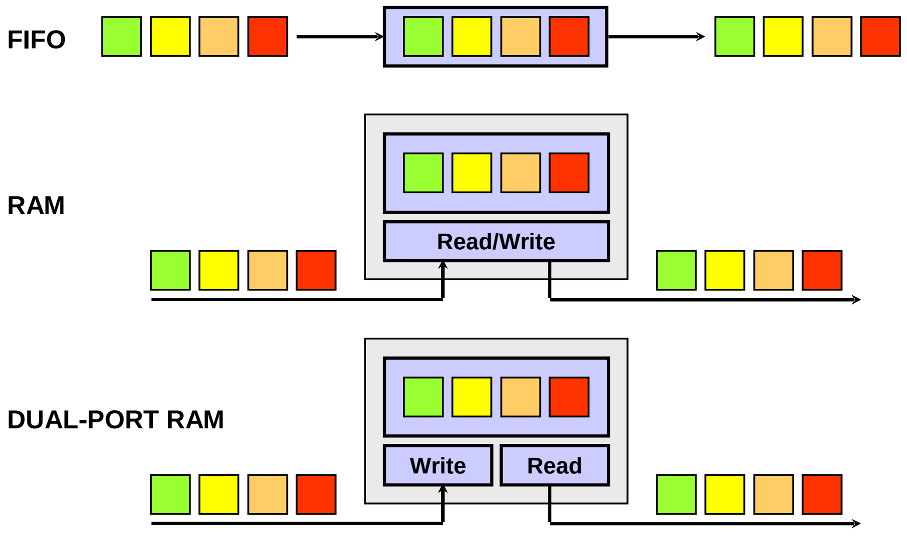
Buffering alternatives

- Input buffering
- Output buffering
- Central buffering
- Combinations
  – input-output buffering
  – central-output buffering

Basic memory types for buffering

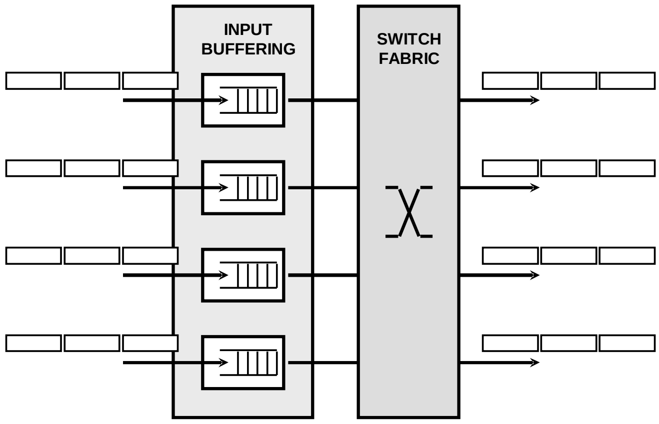
- FIFO (First-In-First-Out)
- RAM (Random Access Memory)
- Dual-port RAM

Basic memory types for buffering (cont.)



Input buffering

Buffer memories at the input interfaces



Input buffering (cont.)

- Pros
  - low required memory access speed
    - in FIFO and dual-port RAM solutions equal to incoming line rate
    - in one-port RAM solutions *twice* the incoming line rate
  - speed of switch fabric
    - multi-stages and crossbars operate at input wire speed
    - shared media fabrics operate at the aggregate speed of inputs
  - low cost solution (due to low memory speed)

Input buffering (cont.)

- Cons
  - FIFO type of buffering
    => HOL problem (limits throughput to 58.6 % for uniform traffic)
  - windowing technique can be used to increase throughput
    - multiple packets from each input are examined and considered for transmission to outputs
    - at most one packet per input/output is chosen in each time-slot
    - the number of examined packets per input determines the window size (WS)
    - WS = 2 yields 70 % throughput (WS>2 does not improve throughput significantly)
  - buffer size may be large (due to HOL)
  - HOL avoided by having a buffer for each output at each input, i.e., virtual output queuing

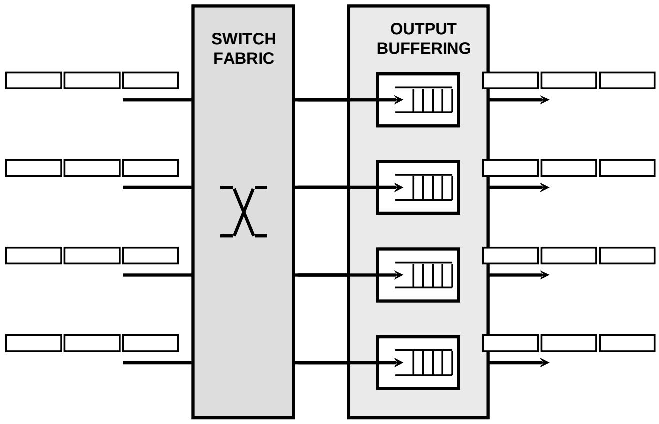
Virtual output queuing (VOQ)

Each input buffer divided into *N* logical queues, which share the same physical memory

- Pros
  - solves HOL problem
  - benefits of input queuing (low memory and switch fabric speed)
  - throughput increased (up to 100 %)
- Cons
  - HOL packets of all logical queues (= N^2 packets) need to be arbitrated in each time-slot
    => need for fast and intelligent arbitration mechanism

Output buffering

Buffer memories at the output interfaces



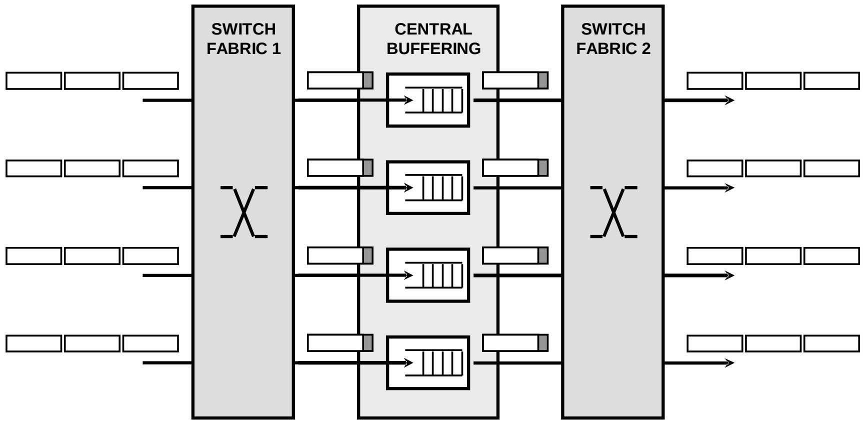
Output buffering (cont.)

- Pros
  - throughput/delay performance better than in input buffered systems
  - no HOL problem
  - capable of achieving 100 % throughput
- Cons
  - access speed of buffer memory
    - in FIFO and dual-port RAM solutions *N* times the incoming line rate
    - in one-port RAM solutions *N*+1 times the incoming line rate

    => switch size limited by memory speed
  - high cost due to high memory speed requirement
  - switch fabric operates at the aggregate speed of inputs
  - concentrator used for alleviating memory speed requirement => solution leads to packet loss

Central buffering

Buffer memory located between two switch fabrics
- shared by all inputs/outputs
- virtual buffer for each input or output



Central buffering (cont.)

- Pros
  - smaller buffer size requirement and lower average delay than in input or output buffering system
  - HOL problem can be avoided
  - optimal throughput (100 %)
- Cons
  - speed of buffer memory
    - in dual-port RAM solutions larger than *N* times the incoming line rate
    - in one-port RAM solutions larger than 2x*N* times the incoming line rate

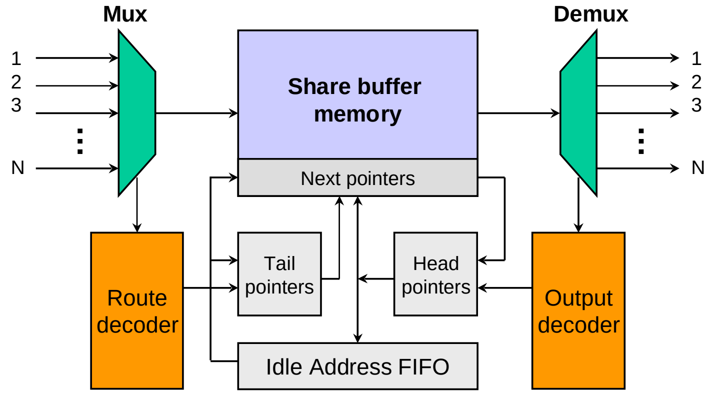
    => switch size limited by memory speed
  - speed of switch fabric is *N* x wire speed
  - complicated buffer control
  - high cost due to high memory speed requirement and control complexity

Shared memory based central buffering

- RAM based solution
  - memory organized into separate logical (FIFO) queues, one for each output
  - incoming packets time-division multiplexed to two synchronous streams: data packets to memory and corresponding packet headers to route decoder for maintaining queues
  - packets destined for the same output are linked together in the same logical queue
  - output stream of packets formed by retrieving HOL packets from the queues sequentially, one per queue, and packets are time-division demultiplexed and transmitted on output lines
  - each logical queue is controlled by two pointers (head and tail pointer)
- CAM based solution eliminates need to maintain logical queues
  - packets uniquely identified by tags
  - a tag is composed of packet's output port and sequence number

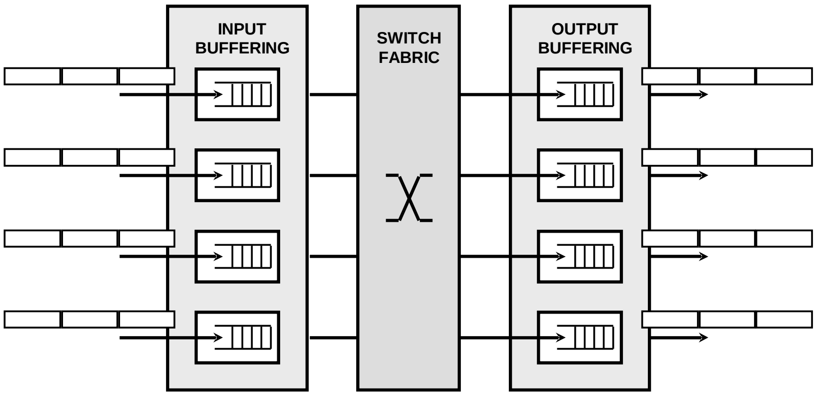
Shared memory based central buffering (cont.)

RAM based solution



Input-output buffering

Input-output buffering common in QoS aware switches/routers
- inputs implement output specific buffers to avoid HOL
- outputs implement dedicated buffers for different traffic classes
- combined buffering distributes buffering complexity between inputs and outputs



Input-output buffering (cont.)

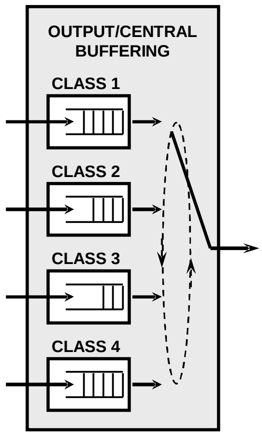
- Pros
  - combines advantages of input and output buffering
    - low speed requirement of input buffers
    - high throughput of output buffering (up to 100 %)
  - HOL problem can be avoided at inputs by implementing output specific buffers
  - speedup factor *L* (1< *L* < *N*) can be fixed for output switch and memory allowing max *L* packets to be switched to an output in a time-slot
  - when more than *L* packets destined to an output, excess packets stored at inputs
- Cons
  - complicated arbitration (control) mechanism to determine, which of the *L* packets of *N* HOL packets go to outputs

Summary of buffering techniques

Buffering principle	Memory space	Memory speed	Memory control	Queueing delay	Multi-casting capabilities
Input buffering	high	slow (~input rate)	simple	longest (due to HOL)	extra logic needed
Output buffering	medium	fast (~N x input rate)	simple	medium	supported
Central buffering	low	fast (~2N x input rate)	complicated	shortest	supported but complex

Priorities and buffering

- Separate buffer for each traffic class
- A scheduler needed to control transmission of data
  - highest priority served first
  - longest queue served first
  - minimization of lost packets/cells
- Priority given to high quality traffic
  - low delay and delay variation traffic
  - low loss rate traffic
  - best customer traffic
- Scheduling principles
  - round robin
  - weighted round robin
  - fair queuing
  - weighted fair queuing
  - etc.



Switch fabrics

- Multipoint switching
- Self-routing networks
- Sorting networks
- Fabric implementation technologies
- **Fault tolerance and reliability**

Fault tolerance and reliability

- Definitions
- Fault tolerance of switching systems
- Modeling of reliability

Definitions

- **Failure, malfunction -** is deviation from the intended/specified performance of a system
- **Fault -** is such a state of a device or a program which can lead to a failure
- **Error -** is an incorrect response of a program or module. An error is an indication that the module in question may be faulty, the module has received wrong input or it has been misused. An error can lead to a failure if the system is not tolerant to this sort of an error. A fault can exist without any error taking place.

Fault tolerance

- **Fault tolerance** is the ability of a system to continue its intended performance in spite of a fault or faults
- **A switching system** is an example of a fault tolerant system
- Fault tolerance always requires some sort of redundancy

Categorization of faults

- **Duration based**
  - **permanent** or stuck-at (stuck at zero or stuck at one)
  - **intermittent** - fault requires repair actions, but its impact is not always observable
  - **transient** - fault can be observed for a short period of time and disappears without repair
- **Observable** or **latent** (hidden)
- Based on the **scope** of the impact (serious - less serious)

Graceful degradation

- **Capability of a system to continue its functions under one or more faults, but on a reduced level of performance**
- **For example**
  - in some RAID (Redundant Array Inexpensive Disks) configurations, write speed drops in case of a disk fault, but continues on a lower level of performance even while the fault has not been repaired

Reliability and availability

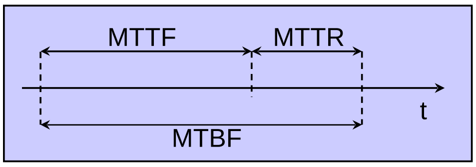
- **Reliability *R(t)* -** probability that a system does not fail within time ***t*** under the condition that it was functioning correctly at ***t*** = 0
  - for all known man-made systems ***R(t)*** $\rightarrow$ **0** when ***t*** $\rightarrow \infty$
- **Availability *A(t)* -** probability that a system will function correctly at time *t*
  - for a system that can be repaired ***A(t)*** approaches some value asymptotically during the useful lifetime of the system

Repairable system

- **Maintainability *M(t)* -** probability that a system is returned to its correct functioning state during time *t* under the condition that it was faulty at time ***t*** = 0

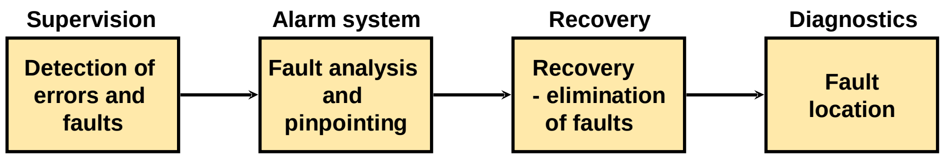
MTTF, MTTR and MTBF

- **MTTF (Mean-Time-To-Failure) -** expected value of the time duration from the present to the next failure
- **MTTR (Mean-Time-To-Repair) -** expected value of the time duration from a fault until the system has been restored into a correct functioning state
- **MTBF (Mean-Time-Between-Failures) -** expected value of the time duration from occurrence of a fault until the next occurrence of a fault
- **MTBF = MTTF + MTTR**



High availability of a switching system

High availability of a switching system is obtained by maintenance software



Supervision
- **In a unit under normal working load**
- **HW implementation => fast**
- **SW implementation => detection delay**

Alarm system
- **Often a rule based system**

Recovery

Utilizes
- **redundancy**
- **switch-overs**
  - **active <=> standby**
- **restarts**
  - **a single program**
  - **a preprocessor**
  - **a single main processor**
  - **whole system**
  - **fall back to previous SW package**

Diagnostics
- **In a unit temporarily without normal load**

Maintenance software is one of the most important software sub-systems in a switching system in parallel with call/connection control and charging

Main types of redundancy

- **Hardware redundancy**
  - duplication (1+1) - need for “self-checking”-recovery blocks that detect their own faults
  - *n+r* -principle (*n* active units and *r* standby units)
  - *n*:*r* -principle (*n* active units and r of them used to back up the other *n-r* units)
- **Software redundancy**
  - required always in telecom systems
- **Information redundancy**
  - parity bits, block codes, etc.
- **Time redundancy**
  - delayed re-execution of transactions

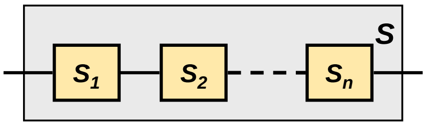
Modeling of reliability

- Combinatorial models
- Markov analysis
- Other modeling techniques (not covered here)
  - Fault tree analysis
  - Reliability block diagrams
  - Monte Carlo simulation

Combinatorial reliability

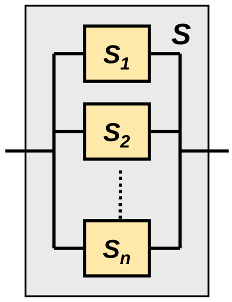
- A **serial system *S*** functions if and only if all its parts S_i ($1 \le i \le n$) function

  => $R_s = \prod_{i=1}^{n} R_i$ and $F_s = (1 - R_s)$



- Failures in sub-systems are supposed to be independent
- A **parallel** (replicated) **system** fails if all its sub-systems fail

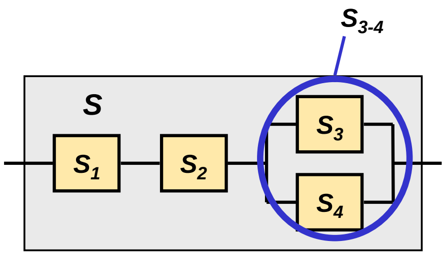
  => $F_s = \prod_{i=1}^{n} (1 - R_i)$ and $R_s = 1 - F_s = 1 - \prod_{i=1}^{n} (1 - R_i)$



- Reliability of a duplicated system ($R_i = R$) is $R_s = 1 - (1 - R)^2$

Combinatorial reliability example 1

- Calculate reliability R_s and failure probability F_s of system S given that failures in sub-systems S_i are independent and for some time interval it holds that $R_1 = 0.90$, $R_2 = 0.95$ and $R_3 = R_4 = 0.80$



=> $R_s = \prod R_i = R_1 \times R_2 \times R_{3\text{-}4}$

=> $R_{3\text{-}4} = 1 - \prod (1 - R_i) = 1 - (1 - R_3)(1 - R_4)$

=> $R_s = R_1 \times R_2 \times [1 - (1 - R_3)(1 - R_4)]$

=> $F_s = 1 - R_s = 1 - R_1 \times R_2 \times [1 - (1 - R_3)(1 - R_4)]$

=> $R_s = 0.82$ and $F_s = 0.18$

Combinatorial reliability (cont.)

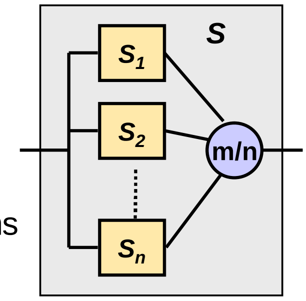
- A load sharing system functions if m of the total of n sub-systems function
- If failures in sub-systems S_i are independent then probability that the system fails is

  P(fails) = P($k<m$)

  and probability that it functions is

  P(functions) = P($k \geq m$) = 1- P($k<m$)

  where k is the number of functioning sub-systems



$$P(k \geq m) = \sum_{i=m}^{n} P(k=i) \quad \text{and} \quad P(k<m) = \sum_{i=0}^{m-1} P(k=i)$$

Combinatorial reliability example 2

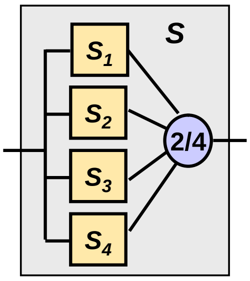
- As an example, suppose we have a system, which has $m = 2$ and $n = 4$ and each of the four sub-systems have a different R, i.e. R_1, R_2, R_3 and R_4, and failures in the sub-systems are independent
- Probability that the system fails is

  $$P(\text{fails}) = P(k<2) = \sum_{i=0}^{1} P(k=i) = P(k=0) + P(k=1)$$

- P(k=0) and P(k=1) can be derived to be

  $$P(k=0) = (1-R_1)(1-R_2)(1-R_3)(1-R_4)$$

  $$P(k=1) = R_1(1-R_2)(1-R_3)(1-R_4) + (1-R_1)R_2(1-R_3)(1-R_4) + (1-R_1)(1-R_2)R_3(1-R_4) + (1-R_1)(1-R_2)(1-R_3)R_4$$



- If R_1=0.9 ,R_2,=0.95 ,R_3 =0.85 and R_4 =0.8 then R_s = 0.994 and F_s = 0.0058

Combinatorial reliability (cont.)

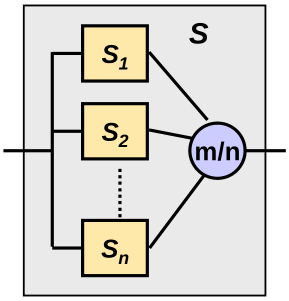
- If failures in sub-systems S_i of an *m*/*n* system are independent and $R_i = R$ for all $i \in [1,n]$ then the system is a Bernoulli system and binomial distribution applies

$$\Rightarrow R_s = \sum_{k=m}^{n} \binom{n}{k} R^k (1-R)^{n-k}$$

- For a system of *m*/*n* = 2/3

$$\Rightarrow R_{2/3} = \sum_{k=2}^{3} \frac{3!}{k!(3-k)!} R^k (1-R)^{3-k} = 3R^2 - 2R^3$$

If for example $R = 0.9 \Rightarrow R_{2/3} = 0.972$



Computing MTTF

- $\mathrm{MTTF} = \int_0^{\infty} R(t)\,dt$ is valid for any reliability distribution
- Exponential distribution widely used in reliability calculations
- Probability density function (PDF) for a single component with a constant failure rate (CFR) λ is
  $r(t) = \lambda e^{-\lambda t}$ and corresponding reliability function is

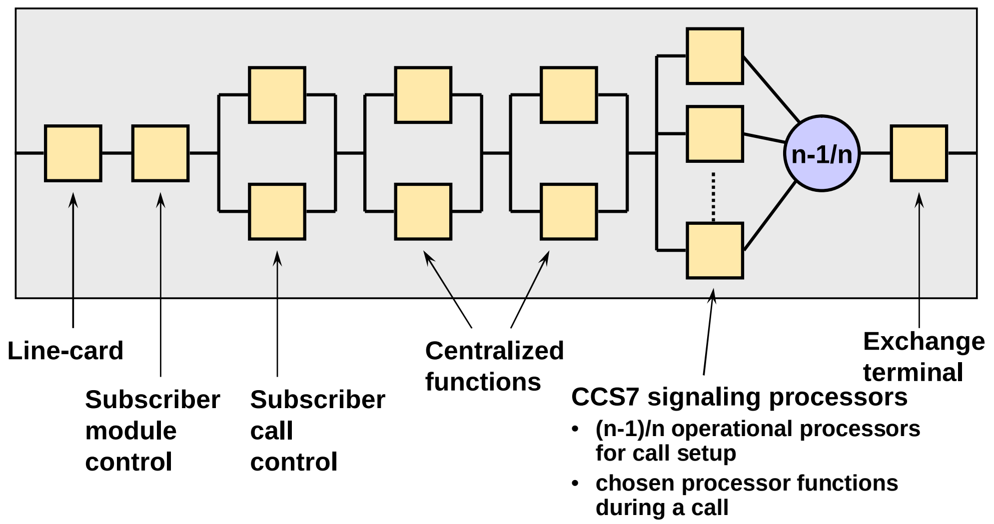
$$R(t) = \int_t^{\infty} r(t)\,dt = e^{-\lambda t}$$

=> MTTF = 1/λ

Computing MTTF (cont.)

- Serial systems with *n* CFR components
  - $R_s(t) = R_1(t) \times R_2(t) \times \ldots \times R_n(t) = e^{-(\lambda_1 + \lambda_2 + \ldots + \lambda_n)t} = e^{-\lambda_s t}$
  - $\lambda_s = \lambda_1 + \lambda_2 + \ldots + \lambda_n$
- $MTTF_s = 1/\lambda_s$
- $1/MTTF_s = 1/MTTF_1 + 1/MTTF_2 + \ldots + 1/MTTF_n$

Telecom exchange reliability from subscriber's point of view



Premature release requirement **P** ≤ **2x10**$^{-5}$ applied

Failure intensity

- Unit of failure intensity λ is defined to be **$[\lambda]$ = fit = number of faults /10^9 h**
- Failure intensities for replaceable plug-in-units varies in the range 0.1 - 10 kfit

- Example:
  - if failure intensity of a line-card in an exchange is 2 kfit, what is its MTTF ?

$$\textbf{MTTF} = \mathbf{1/\lambda} = \frac{10^9\ \mathbf{h}}{2000} = \frac{1\ 000\ 000\ \mathbf{h}}{2x24x365} = 57 \text{ years}$$

Example of exponential distribution

Reliability of a switching equipment is given by $R(t) = e^{-\lambda t}$ and its failure rate is λ = 20 kfit. What is the probability that the device survives one year (in continuous operation) ? What is its MTTF ?

Since λ = 20 kfit = $20000x10^{-9}$ = $2x10^{-5}$ and one year is 365x24 hours = 8760 hours, we get

– $R(t) = e^{-\lambda t} = e^{-2x10^{-5}x8760} = 0.84$ and
– MTTF = $1/\lambda$ = 1/ $2x10^{-5}$ = 50 000 hours = 5.7 years

Example of exponential distribution (cont.)

Suppose the device has been functioning without failures 2 years. What is the probability that the device will fail during the next year?

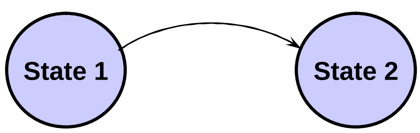
Let's write t_1= 2 years and t_2 = 1 year. Since we have a time independent process and the device is functioning at t_1, we can write

$$\begin{aligned} Pr(T \le (t_1+t_2) | T > t_1) &= Pr(T \le (t_2)) \\ &= 1 - Pr(T > t_2) \\ &= 1 - R(t_2) \\ &= 1 - e^{-2x10^{-5}x8760} \\ &= 0.16 \end{aligned}$$

Reliability modeling using Markov chains

Markov chains

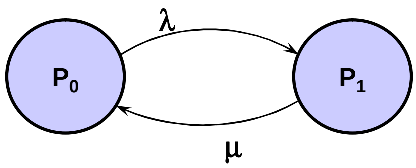
- A system is modeled as a set of states of transitions
- Each state corresponds to fulfillment of a set of conditions and each transition corresponds to an event in a system that changes from one state to another



- By using this method it is possible to find reliability behavior of a complex system having a number of states and non-independent failure modes

Markov chain modeling

- A set of states of transitions leads to a group of linear differential equations => Chapman-Golmogorow equation used to solve equations
- For a given modeling goal it is essential to choose a minimal set of states for equations to be easily solved
- By setting the derivatives of the probabilities to zero an asymptotic state is obtained if such exists



λ = failure intensity (=failure rate)

μ = repair intensity (repair time is exponentially distributed)

P_i = probability of state i, e.g. $P_0 = R(t)$ and $P_1 = F(t)$

Markov chain modeling (cont.)

- Probabilities (π_i) of the states and transition rates (λ_{ij}) between the states are tied together with the following formula

$$\pi\Lambda = 0$$

where

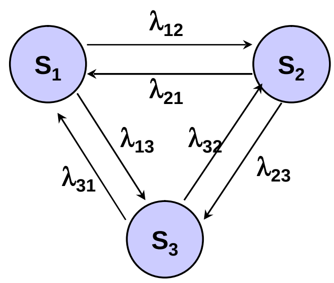
$$\pi = [\pi_1 \quad \pi_2 \quad \ldots \quad \pi_n]$$

$$\Lambda = \begin{bmatrix} -(\lambda_{12}+\lambda_{13}+\cdots) & \lambda_{12} & \lambda_{13} & \cdots \\ \lambda_{21} & -(\lambda_{21}+\lambda_{23}+\cdots) & \lambda_{23} & \cdots \\ \lambda_{31} & \lambda_{32} & -(\lambda_{31}+\lambda_{32}+\cdots) & \cdots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Markov chain modeling (cont.)

Example

$$\Lambda = \begin{bmatrix} -(\lambda_{12}+\lambda_{13}) & \lambda_{12} & \lambda_{13} \\ \lambda_{21} & -(\lambda_{21}+\lambda_{23}) & \lambda_{23} \\ \lambda_{31} & \lambda_{32} & -(\lambda_{31}+\lambda_{32}) \end{bmatrix}$$

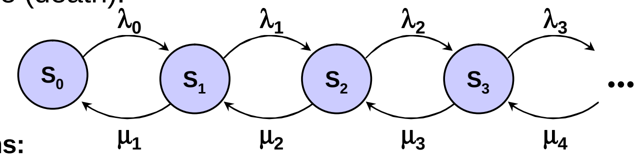


$\pi\Lambda = \mathbf{0}$ and $\pi = [\pi_1 \quad \pi_2 \quad \text{K} \quad \pi_n]$

$$\begin{cases} -(\lambda_{12}+\lambda_{13})\pi_1 + \lambda_{21}\pi_2 + \lambda_{31}\pi_3 = \mathbf{0} \\ \lambda_{12}\pi_1 - (\lambda_{21}+\lambda_{23})\pi_2 + \lambda_{32}\pi_3 = \mathbf{0} \\ \lambda_{13}\pi_1 + \lambda_{23}\pi_2 - (\lambda_{31}+\lambda_{32})\pi_3 = \mathbf{0} \end{cases}$$

Birth-death process

Birth-death process is a special case of continuous-time Markov chain, which models the size of population that increases by 1 (birth) or decreases by one (death).



Balance equations:

- State S_0 $\lambda_0\pi_0 = \mu_1\pi_1$ => $\pi_1 = \frac{\lambda_0}{\mu_1}\pi_0$

- State S_1 $(\lambda_1 + \mu_1)\pi_1 = \lambda_0\pi_0 + \mu_2\pi_2$ => $\pi_2 = \frac{\lambda_1\lambda_0}{\mu_2\mu_1}\pi_0$

- State S_k $(\lambda_{k-1} + \mu_{k-1})\pi_{k-1} = \lambda_{k-2}\pi_{k-2} + \mu_k\pi_k$ => $\pi_k = \frac{\lambda_{k-1} \text{L} \lambda_1\lambda_0}{\mu_k \text{L} \mu_2\mu_1}\pi_0$

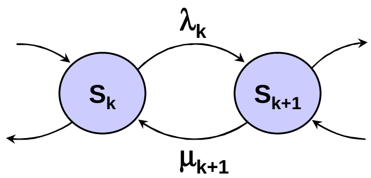
Birth-death process (cont.)

$$\pi_k = \left(\frac{\lambda_{k-1}}{\mu_k}\right)\cdots\left(\frac{\lambda_1}{\mu_2}\right)\left(\frac{\lambda_0}{\mu_1}\right)\pi_0 = \rho_{k-1}\cdots\rho_2\rho_1\pi_0 \qquad \text{where} \quad \rho_k = \frac{\lambda_k}{\mu_{k+1}} \quad (k=1, 2, 3, \ldots)$$

Substituting these expressions for π_k into $\sum_{k=0}^{\infty}\pi_k = 1$ yields

$$\pi_0 + \sum_{k=1}^{\infty}\frac{\lambda_{k-1}\cdots\lambda_1\lambda_0}{\mu_k\cdots\mu_2\mu_1}\pi_0 = 1 \quad \Rightarrow \quad \pi_0\left[1+\sum_{k=1}^{\infty}\frac{\lambda_{k-1}\cdots\lambda_1\lambda_0}{\mu_k\cdots\mu_2\mu_1}\right] = 1$$

$$\Rightarrow \quad \frac{1}{\pi_0} = \left[1+\sum_{k=1}^{\infty}\frac{\lambda_{k-1}\cdots\lambda_1\lambda_0}{\mu_k\cdots\mu_2\mu_1}\right]$$

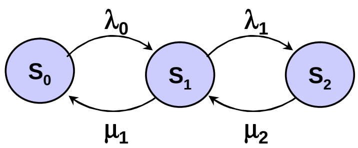


$$\Rightarrow \quad \pi_k = \frac{\lambda_{k-1}\cdots\lambda_1\lambda_0}{\mu_k\cdots\mu_2\mu_1}\pi_0 \quad (k=1, 2, 3, \ldots)$$

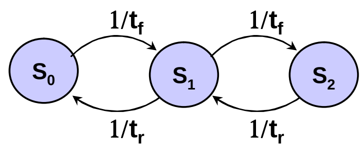
Example of birth-death process

A switching system has two control computers, one on-line and one standby. The time interval between computer failures is exponentially distributed with mean t_f . In case of a failure, the standby computer replaces the failed one.
A single repair facility exist and repair times are exponentially distributed with mean t_r .
What fraction of time the system is out of use, i.e., both computers having failed ?

The problem can be solved by using a three state birth-death model.



=>



Example of birth-death process (cont.)

It holds for a birth-death process that

$$\frac{1}{\pi_0} = \left[1 + \sum_{k=1}^{\infty} \frac{\lambda_{k-1} \cdots \lambda_1 \lambda_0}{\mu_k \cdots \mu_2 \mu_1}\right] \quad \text{and} \quad \pi_k = \frac{\lambda_{k-1} \cdots \lambda_1 \lambda_0}{\mu_k \cdots \mu_2 \mu_1} \pi_0 \qquad (k=1, 2, 3, \ldots)$$

Applying these equations, we get

$$\Rightarrow \frac{1}{\pi_0} = \left[1 + \frac{\lambda_0}{\mu_1} + \frac{\lambda_1 \lambda_0}{\mu_2 \mu_1}\right] \quad \Rightarrow \quad \pi_0 = \left[\frac{\mu_2 \mu_1}{\lambda_1 \lambda_0 + \lambda_0 \mu_2 + \mu_2 \mu_1}\right]$$

$$\Rightarrow \pi_2 = \frac{\lambda_1 \lambda_0}{\mu_2 \mu_1} \pi_0 = \frac{\lambda_1 \lambda_0}{\mu_2 \mu_1} \left[\frac{\mu_2 \mu_1}{\lambda_1 \lambda_0 + \lambda_0 \mu_2 + \mu_2 \mu_1}\right] = \frac{\lambda_1 \lambda_0}{\lambda_1 \lambda_0 + \lambda_0 \mu_2 + \mu_2 \mu_1}$$

Example of birth-death process (cont.)

Since $\lambda_1 = \lambda_2 = 1/(t_f)$ and $\mu_1 = \mu_2 = 1/(t_r)$, the probability that both computers have failed gets the form

$$\pi_2 = \frac{\left(\frac{1}{t_f}\right)^2}{\left(\frac{1}{t_f}\right)^2 + \left(\frac{1}{t_f}\right)\left(\frac{1}{t_r}\right) + \left(\frac{1}{t_r}\right)^2} = \frac{t_r^{\,2}}{t_r^{\,2} + t_r t_f + t_f^{\,2}}$$

Note the meanings of the three states:

- S_0 - both computers operable
- S_1 - one computer failed
- S_2 - both computers failed

Example of birth-death process (cont.)

Numerical examples:

- Suppose that t_f = 450 days and t_r = 15 days then the probability that both computers fail is π_2 = 0.0107.
- If in general t_f/t_r = 10, i.e. the average repair time is 10 % of the average time between failures, then π_2 =0.009009 and both computers will simultaneously be out of service 0.9 % of the time.

Additional reading of Markov chain modeling

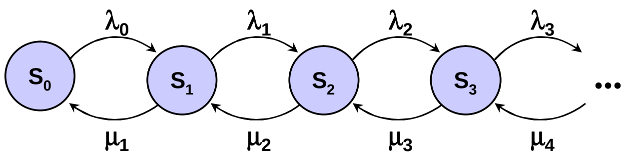
Switching Technology S38.3165

http://www.netlab.hut.fi/opetus/s383165

Markov chain modeling

A continuous-time Markov Chain is a stochastic process $\{X(t): t \geq 0\}$

- $X(t)$ can have values is S={0,1,2,3,...}
- Each time the process enters a state i, the amount of time it spends in that state before making a transition to another state has an exponential distribution with mean $1/\lambda_i$
- When leaving state i, the process moves to a state j with probability p_{ij} where $p_{ii}=0$
- The next state to be visited after i is independent of the length of time spend in state i



Markov chain modeling (cont.)

Transition probabilities

$$p_{ij}(t) = P\{X(t+s) = j \mid X(s) = i\}$$

Continuous at t=0, with

$$\lim_{t\to 0} p_{ij}(t) = \begin{cases} 1 & if \quad i = j \\ 0 & if \quad i \neq j \end{cases}$$

Transition matrix is a function of time

$$P(t) = \begin{bmatrix} p_{11}(t) & p_{12}(t) & \cdots \\ p_{21}(t) & \vdots & \\ \vdots & & \ddots \end{bmatrix}$$

Markov chain modeling (cont.)

Transition intensity:

$$\lambda_j(t) = -\frac{d}{dt} p_{jj}(0)$$ (rate at which the process leaves state *j* when it is in state *j*)

$$\lambda_{ij}(t) = \frac{d}{dt} p_{ij}(0) = \lambda_i p_{ij}$$ (transition rate into state *j* when the process in is state *i*)

The process, starting in state *i*, spends an amount of time in that state having exponential distribution with rate λ_i. It then moves to state *j* with probability

$$p_{ij} = \frac{\lambda_{ij}}{\lambda_i} \quad \forall i, j \qquad \sum_{j=1}^{n} p_{ij} = \sum_{j=1}^{n} \frac{\lambda_{ij}}{\lambda_i} = \frac{\sum_{j=1}^{n} \lambda_{ij}}{\lambda_i} = 1 \quad \Rightarrow \quad \lambda_i = \sum_{j=1}^{n} \lambda_{ij}$$

Markov chain modeling (cont.)

Chapman-Kolmogorov equations:

$$p_{ij}(t+s) = \sum_{k \in S} p_{ik}(t) p_{kj}(s) \qquad \forall i, j \in S \quad \forall s, t \geq 0$$

Since *p(t)* is a continuous function

$$p_{ij}(\Delta t) = p_{ij}(0) + \frac{d}{dt} p_{ij}(0) \Delta t + o(\Delta t^2)$$

We have defined => $$\lambda_{ij}(t) = \frac{d}{dt} p_{ij}(0)$$

For *i≠j*: $$p_{ij}(\Delta t) = p_{ij}(0) + \lambda_{ij}\Delta t + o(\Delta t^2) \approx \lambda_{ij}\Delta t$$ (for small Δt)

For *i=j*: $$p_{ii}(\Delta t) = p_{ii}(0) + \lambda_{ii}\Delta t + o(\Delta t^2) \approx 1 + \lambda_{ii}\Delta t$$ (for small Δt)

Markov chain modeling (cont.)

From Chapman-Kolmogorov equations:

$$p_{ij}(t+\Delta t)=\sum_k p_{ik}(t)p_{kj}(\Delta t)\;=p_{ij}(t)p_{jj}(\Delta t)+\sum_{k\neq j} p_{ik}(t)p_{kj}(\Delta t)$$

$$=p_{ij}(t)\left[1+\lambda_{jj}\Delta t+o(\Delta t^2)\right]+\sum_{k\neq j} p_{ik}(t)\left[\lambda_{kj}\Delta t+o(\Delta t^2)\right]$$

$$p_{ij}(t+\Delta t)=p_{ij}(t)+\left[\sum_k p_{ik}(t)\lambda_{kj}\right]\Delta t+\left[\sum_k p_{ik}(t)\right]o(\Delta t^2)$$

$$\frac{p_{ij}(t+\Delta t)-p_{ij}(t)}{\Delta t}=\sum_k p_{ik}(t)\lambda_{kj}+\left[\sum_k p_{ik}(t)\right]\frac{o(\Delta t^2)}{\Delta t}$$

Taking the limit as $\Delta t \to 0$

$$\frac{d}{dt}p_{ij}(t)=\sum_k p_{ik}(t)\lambda_{kj}\qquad \forall i,j$$

Markov chain modeling (cont.)

The process is described by the system of differential equations:

$$\frac{d}{dt}p_{ij}(t)=\sum_k p_{ik}(t)\lambda_{kj}\qquad \forall i,j$$

which can be given in the form

$$\frac{d}{dt}P(t)=P(t)\Lambda\qquad \forall i,j$$

$$\sum_j p_{ij}(t)=1\qquad \forall i,t$$

$$\frac{d}{dt}\sum_j p_{ij}(t)=\frac{d}{dt}(1)=0$$

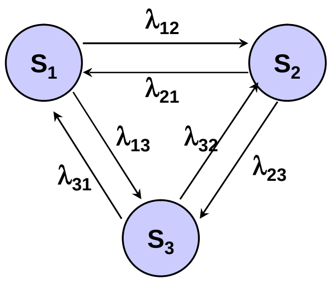
$$\frac{d}{dt}\sum_j p_{ij}(t)=0$$

$$\sum_j \lambda_{ij}=0$$

The sum of each row of Λ is zero !

Markov chain modeling (cont.)

Example



$$\Lambda = \begin{bmatrix} -(\lambda_{12}+\lambda_{13}) & \lambda_{12} & \lambda_{13} \\ \lambda_{21} & -(\lambda_{21}+\lambda_{23}) & \lambda_{23} \\ \lambda_{31} & \lambda_{32} & -(\lambda_{31}+\lambda_{32}) \end{bmatrix}$$

The sum of each row of Λ must be zero !

Markov chain modeling (cont.)

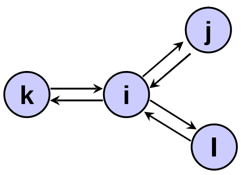
Steady state probabilities

$\lim_{t\to\infty} p_{ij}(t) = \pi_j$ **(independent of initial state *i*)**

must be non-negative and must satisfy $\sum_{i=1}^{n} \pi_i = 1$

In case of continuous-time Markov chains, balance equation is used to determine π.
For each state *i*, the rate at which the system leaves the state must equal to the rate at which the system enters the state

=> $\lambda_i \pi_i = \lambda_{ji}\pi_j + \lambda_{ki}\pi_k + \lambda_{li}\pi_l$



Markov chain modeling (cont.)

Balance equation

$$\left(\sum_{j\neq i}\lambda_{ij}\right)\pi_i = \sum_{k\neq i}\lambda_{ki}\pi_k \qquad \forall i$$

Steady state distribution is computed by solving this system of equations

$$\left(\sum_{j\neq i}\lambda_{ij}\right)\pi_i = \sum_{k\neq i}\lambda_{ki}\pi_k \qquad \forall i$$

$$\sum_{i=1}^{n}\pi_i = 1$$

Markov chain modeling (cont.)

An alternative derivation of the steady-state conditions begins with the differential equation describing the process:

$$\frac{d}{dt}p_{ij}(t) = \sum_k p_{ik}(t)\lambda_{kj} \qquad \forall i,j$$

Suppose that we take the limit of each side as $t \to \infty$

=> $$\lim_{t\to\infty}\frac{d}{dt}p_{ij}(t) = \lim_{t\to\infty}\sum_k p_{ik}(t)\lambda_{kj}$$

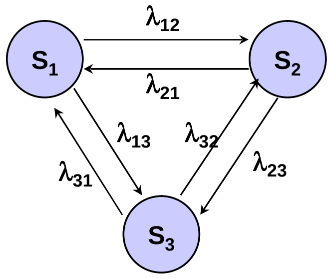
=> $$\frac{d}{dt}\lim_{t\to\infty}p_{ij}(t) = \sum_k \lim_{t\to\infty}p_{ik}(t)\lambda_{kj}$$

=> $$\sum_k \pi_k\lambda_{kj} = 0$$ **i.e. $\pi\Lambda=0$**

Markov chain modeling (cont.)

Example

$$\Lambda = \begin{bmatrix} -(\lambda_{12}+\lambda_{13}) & \lambda_{12} & \lambda_{13} \\ \lambda_{21} & -(\lambda_{21}+\lambda_{23}) & \lambda_{23} \\ \lambda_{31} & \lambda_{32} & -(\lambda_{31}+\lambda_{32}) \end{bmatrix}$$



$\pi\Lambda = \mathbf{0}$ and $\boldsymbol{\pi} = [\pi_1 \quad \pi_2 \quad \pi_3]$

$$\begin{cases} -(\lambda_{12}+\lambda_{13})\pi_1 + \lambda_{21}\pi_2 + \lambda_{31}\pi_3 = 0 \\ \lambda_{12}\pi_1 - (\lambda_{21}+\lambda_{23})\pi_2 + \lambda_{32}\pi_3 = 0 \\ \lambda_{13}\pi_1 + \lambda_{23}\pi_2 - (\lambda_{31}+\lambda_{32})\pi_3 = 0 \end{cases}$$