Digital Networks

Dr. Vladimir Sidorenko
Prof. Dr. -Ing. Martin Bossert

University of Ulm
Institute of Telecommunications and Applied Information Theory

February 22, 2008

Contents

1.Introduction

1.1 This lecture is about...

...the basic principles and theory of data networks.

Evaluation of network performance is needed for

- Development of communication systems, i.e. system/protocol standardization
- Network planning
- Operation and maintenance of networks

Tools:

- Mathematical analysis
- Simulation

Note:

- A purely mathematical analysis of communication systems is usually not possible due to the complexity of real-world systems (Alternative: Simulation).
- But understanding the fundamentals of network theory helps
  - To analyze and understand dependencies in a system
  - To set up meaningful simulations, and
  - To interpret the simulation results

... about network protocols.

Protocols: Description, how multiple parties (peer entities) interact when they are jointly working on a single job.

1.2 Overview

- Hierarchical structure of network functionality- OSI (Open System interconnections) model
  - Physical Layer
  - Data Link Control Layer (wit Medium access control)
  - Network, Transport, Session, Presentation, Application Layers
- Protocols for point-to-point connections and End-to-End connections
  - Reliable data transmission, ARQ protocols
  - Framing
- Multiple access control
  - ALOHA protocols
  - Splitting algorithms
  - Token rings, Token passing, Polling
- Routing
  - Basics from Graph theory
  - Dijkstra's algorithm
  - Bellmann-Ford algorithm
- Queuing Theory
  - Little's theorem
  - Exponential queuing systems

1.3 References

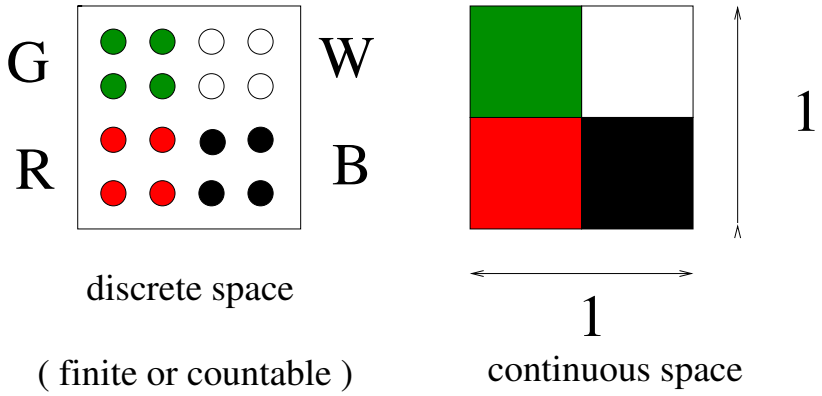
- D.Bertsekas and R.Gallager: Data Networks, Prentice-Hall,1992
- A.Tanenbaum: Computer Networks, Prentice-Hall, 1996
- M.Bossert and M.Breitbach: Digitale Netze, Teubner,1999

1.4 Introduction to Probability theory

1.5 Review of Markov chain theory

Chapter 1

Introduction to probability theory



Ω is a sample space sample (point) $\omega \in \Omega$ (is an elementary event)

Definition: any subset $A \subseteq \Omega$ is called an event.

Example: Events G, R, W, B

Special events:

- the elementary event $A = \{\omega\}$
- the impossible event $A = \{\} = \phi$
- the certain event $A = \Omega$

1.1 Probability of event

Discrete space: Let $\Omega = \{\omega_1, \omega_2, \ldots\}$, and $P(\omega_i) \geq 0$, $\sum_i P(\omega_i) = 1$

Definition: Probability $P(A)$ of event A is sum of probabilities $P(\omega_i)$ of samples ω_i that form A.

Example: Let $\Omega = \{\omega_1, \ldots, \omega_{16}\}$ and $P(\omega_i) = 1/16$. Then $P(G) = 4 \cdot \frac{1}{16} = 1/4$.

Continuous space Ω:

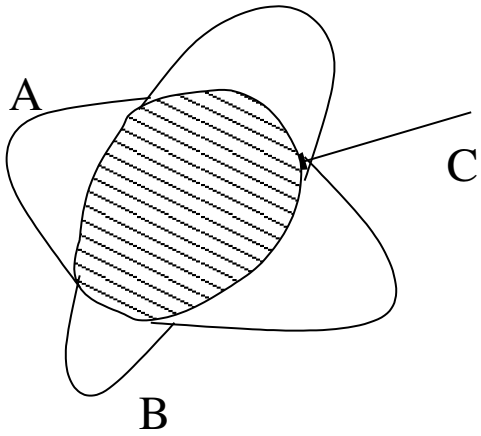
Problem: $P(\omega) = 0$!

Example: Define $P(G) = \frac{|G|}{|\Omega|} = \frac{1/4}{1} = 1/4$.

1.2 Combining events

Product of events $C = AB \quad (A \cap B; A \text{ and } B; A\&B)$

Event C takes place $\Leftrightarrow$ A and B happens.

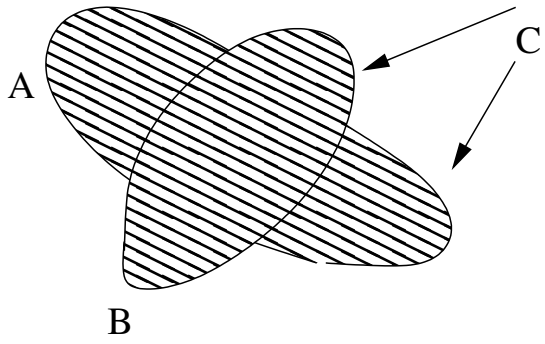


$C = AB = A \cap B; \quad P(AB) = P(A \cap B).$

If $A \cap B = \emptyset$ then events A, B are called mutually exclusive.

Sum of events $C = A + B$ $(A \cup B;\ A\ or B)$

Event C happens $\Leftrightarrow$ A or B happens.



$C = A + B = A \cup B$

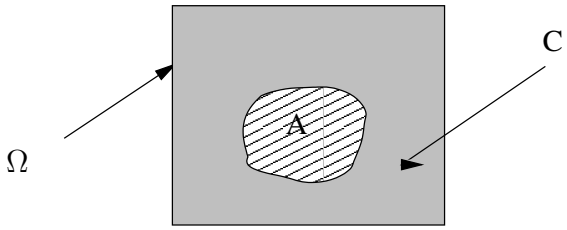
$P(A + B) = P(A) + P(B) - P(A \cap B)$

$P(A + B) \leq P(A) + P(B)$ union bound;

equality only if $P(A \cap B) = 0$.

The complement $C = A^c$ $(\overline{A})$

Event C happens $\Leftrightarrow$ A does not happen.



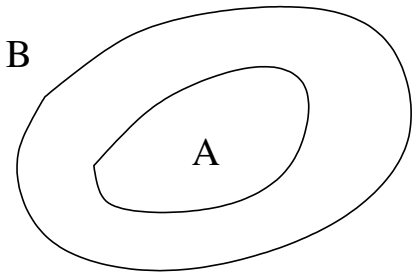
$A^c = \Omega \setminus A$

$P(A^c) = 1 - P(A)$

Inclusion $A \subset B$

B includes A if $A \to B$, i.e.,

if A happens then B happens too.



$P(A) \leq P(B)$

If $A \cap B = \emptyset$: $A \to B^c$ $\quad A \subseteq B^c$ *or* $B \to A^c$ $\quad B \subseteq A^c$

1.3 Kolmogorov's axioms

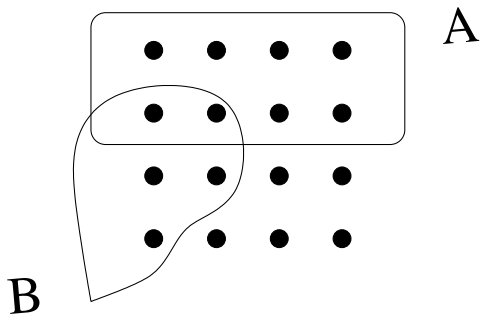
Let $A, B \subseteq \Omega$. Probability $P(\cdot)$ satisfies:

1. $0 \leq P(A) \leq 1$
2. $P(\Omega) = 1$
3. $A \cap B = \emptyset \Rightarrow P(A \cup B) = P(A) + P(B)$

1.4 Conditional probabilities

Assume that $A, B \subseteq \Omega$ and we know that event A happens, then probability of B is called conditional probability and is denoted by $P(B|A)$.

Example:



$P(B) = 5/16$; $P(A) = 12/16$; $|\Omega|$=16

$$P(B|A) = \frac{4}{12} = \frac{|A \cap B|}{|A|} = \frac{P(AB)|\Omega|}{P(A)|\Omega|} = \frac{P(AB)}{P(A)}$$

Definition: $P(B|A) = \frac{P(AB)}{P(A)}$

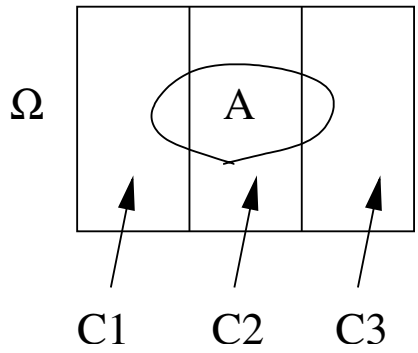
All theorems about probabilities can be applied to conditional probabilities too.

Example: $P(A \cup B|C) = P(A|C) + P(B|C) - P(AB|C)$

We have: $P(AC) = P(A|C)P(C)$

Recurrently: $P(ABC) = P(A|BC)P(B|C)P(C)$

1.5 Decomposition



Let $C_1, \ldots, C_n$ be mutually exclusive events and $\cup C_i = \Omega$. Then $A = AC_1 \cup AC_2 \cup \ldots \cup AC_n$

Since AC_i are also mutually exclusive:

$$P(A) = \sum_i P(A|C_i)P(C_i)$$

1.6 Independency

Events A and B are independent if $P(A|B) = P(A)$.

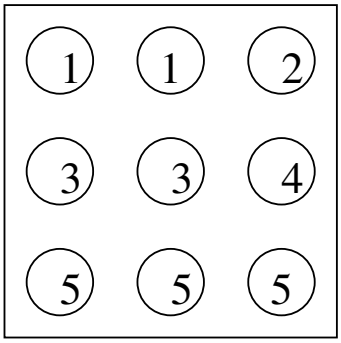
Recall $P(A|B) = \frac{P(AB)}{P(B)}$ should be $= P(A)$.

Definition: Events A and B are called independent if $P(AB) = P(A)P(B)$.

1.7 Random variable

Definition: A (real) random variable is a mapping $X : \Omega \to R$ [, for which $\forall r \in R \; \{\omega \in \Omega : X(\omega) \leq r\}$ is an event]

Example:



Ω

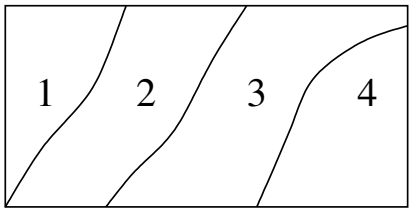
$\Omega = \{\omega_1, \ldots, \omega_9\}, \omega_1 \to 1, \omega_2 \to 1, \omega_3 \to 2, \ldots, \omega_9 \to 5$

X	1	2	3	4	5
$P(X)$	2/9	1/9	2/9	1/9	3/9

probability distribution

If $X(\Omega)$ is discrete, the random variable X is called discrete.

Example:



Ω

$X(\Omega) = \{1, 2, 3, 4\}$, Ω is continuous, X is discrete.

Definition: The function $f_x : X(\Omega) \rightarrow R$ with

$$f_X(x) = P(X = x)$$

is called the probability distribution function (for a discrete variable) or probability mass function.

Definition: The function $F_X : R \rightarrow R$ with

$$F_X(x) = P(X \leq x)$$

is called the cumulative distribution function (CDF).

Definition: If $\exists f_X(x) \geq 0$ such that

$$F_X(x) = \int_{-\infty}^{x} f_X(y)dy$$

then $F_X(x)$ is called probability density function (PDF).

Mean The expected value (mean, mathematical expectation) of the random variable X is $\mu_X = E(X)$, where

$$E(X) = \sum_x xP(X = x)$$

for discrete variable X.

$$E(X) = \int_{-\infty}^{\infty} xf(x)dx$$

for continuous variable.

Property: $E(aX + bY) = aE(X) + bE(y), \quad a, b \in R$

Variance $var(X) = E((X - \mu_X)^2) = \sigma_X^2$

Properties:

$$var(aX + b) = a^2 var(X)$$

$$var(X) = E(X^2) - (E(X))^2$$

If X, Y are independent, then

$$var(X + Y) = var(X) + var(Y)$$

1.8 Stochastic process

A discrete time stochastic (random) process can be considered as a series of random variables

$$X_1, X_2, X_3, \ldots,$$

where $X_i : \Omega_i \to R$. Here $I = \{1, 2, 3, \ldots\}$ is a discrete index set.

Each point in the sample space

$$\Omega = \Omega_1 \times \Omega_2 \times \ldots$$

corresponds to a particular value for each of the random variables and is known as a realization of the process.

The process is determined by the joint probability distribution of the random variables.

Continuous stochastic process

In a continuous stochastic process the index set is continuous (usually space or time), resulting in an uncountably infinite number of random variables

$$X_t, \quad t \geq 0.$$

1.9 Bernoulli process and Bernoulli distribution

A Bernoulli process is a discrete-time stochastic process consisting of a finite or infinite sequence of independent random variables $X_1, X_2, \ldots$, with distribution: $P(X = 1) = p$; $P(X = 0) = 1 - p$.

Example: Unfair coin, we call $X = 1$ the successful event. Consider n throwings of the coin, get Bernoulli process of length n.

Binomial distribution

Define a new discrete random variable I as the number of successes in n attempts. The probability function for I is called Binomial distribution:

$$P(I = i) = \begin{pmatrix} n \\ i \end{pmatrix} p^i (1-p)^{n-i},$$

where $\begin{pmatrix} n \\ i \end{pmatrix} = \frac{n!}{i!(n-i)!}$ is binomial coefficients.

Mean value $E(I) = np$

Variance $\sigma_I^2 = np(1-p)$

Waiting time in a Bernoulli process

Define a new discrete random variable N as the number of throwings, the coin until we get i successes.

The probability function is

$$P(N = n) = \begin{pmatrix} n-1 \\ i-1 \end{pmatrix} p^i (1-p)^{n-i},$$

Mean: $E(N) = \frac{i}{p}$

Variance: $\sigma_N^2 = i\frac{1-p}{p^2}$

1.10 From Bernoulli to Poisson

Bernoulli: $B(i) = P(I = i) = \begin{pmatrix} n \\ i \end{pmatrix} p^i (1-p)^{n-i}$

$n \to \infty \quad p \to 0 \quad np = \lambda = const$, then

$$B(i) \approx \frac{(np)^i}{i!} e^{-np} = \frac{\lambda^i}{i!} e^{-\lambda}$$

Poisson approximation for Bernoulli distribution, where $\lambda = np$ is average number of "successes" during n trials.

Continuous time t:

λ - an average number of successes (arrivals) per unit time

Probability to have i arrivals during unit time:

$$P(i) = \frac{\lambda^i}{i!} e^{-\lambda}$$

(Poisson distribution with parameter λ)

(n trials correspond to unit time, $np = \lambda$)

Probability to have i arrivals during time interval τ is

$$P(i) = \frac{(\lambda\tau)^i}{i!} e^{-\lambda\tau}$$

(Poisson distribution with parameter $\lambda\tau$)

(n trials correspond to the interval τ, $\tau\lambda = np$)

1.11 Poisson process

A stochastic process $A(t),\ t \geq 0$ taking nonnegative integer values is a Poisson process with rate λ if

1. $A(t)$ is the number of arrivals in the time interval (0,t].
2. The number of arrivals that occur in disjoint time intervals are independent.
3. The number of arrivals in any interval of length τ (denote this number by random variable I) is Poisson distributed with parameter $\lambda\tau$:
   $P_0(i;\lambda\tau) = P\{A(t+\tau) - A(t) = i\} = P(I = i) = \frac{(\lambda\tau)^i}{i!}e^{-\lambda\tau}, i = 0, 1, \cdots$

Properties

1. mean of $I \quad E(I) = \lambda\tau$
2. variance of $I \quad \sigma_I^2 = \lambda\tau$
3. Interarrival times (random variables J_i)

   $\tau_i = t_{i+1} - t_i$ are independent and exponentially distributed:

   $$P(J_i \leq s) = 1 - e^{-\lambda s}$$

   with probability density function

   $$f_J(\tau) = \lambda e^{-\lambda\tau}$$

   mean $E(\tau) = 1/\lambda \quad$ variance $\sigma_\tau^2 = 1/\lambda^2$

4.

$P(I=0) = 1 - \lambda\tau + o(\tau)$	$e^{-\lambda\tau} = 1 - \lambda\tau + \frac{(\lambda\tau)^2}{2} - \ldots$
$P(I=1) = \lambda\tau + o(\tau)$	$\lambda\tau e^{-\lambda\tau} = \lambda\tau - (\lambda\tau)^2 + \ldots$
$P(I>1) = o(\tau)$	$1 - P(I=0) - P(I=1)$

5. Let $A = A_1 + \ldots + A_k$, where A_i are independent Poisson processes, then A is a Poisson process with $\lambda = \lambda_1 + \ldots + \lambda_k$.

6. Waiting time for i arrivals (random variable J). The probability density function for J is (Erlang-i distribution)

$$f_J(\tau) = \lambda \frac{(\lambda\tau)^{i-1}}{(i-1)!} e^{-\lambda\tau}$$

with mean $E(J) = i/\lambda$, variance $\sigma_J^2 = i/\lambda^2$

For $i = 1$ we get an exponential distribution.

Agner Krarup Erlang (1878-1929)

Danish mathematician, invented queueing theory.

1.12 Time-discrete Markov Chains

A Markov chain describes a process with memory: The outcome of an experiment depends on the previous one.

$S = \{0, 1, 2, \ldots\}$ is a set of positive integers called states. S can either be finite or infinite (depends on type of Markov chain).

Definition: Markov Chain

A discrete stochastical process $\{X(k) \mid k = 0, 1, 2, \ldots\}$ is a Markov Chain if

- $X(k) \in S$ for all $k = 0, 1, 2, \ldots$
- the probability $P_{ij} = P\{X(k) = j \mid X(k-1) = i\}$ is independent from the history of the process before time instant $k - 1$:

$$P_{ij} = P\{X(k) = j \mid X(k-1) = i,\ X(k-2) = i_{k-2}, \ldots\} = P\{X(k) = j \mid X(k-1) = i\}$$

At time instant $k - 1$ the Markov chain is in state i, and at time k in state j. P_{ij} is the transition probability from state i to state j.

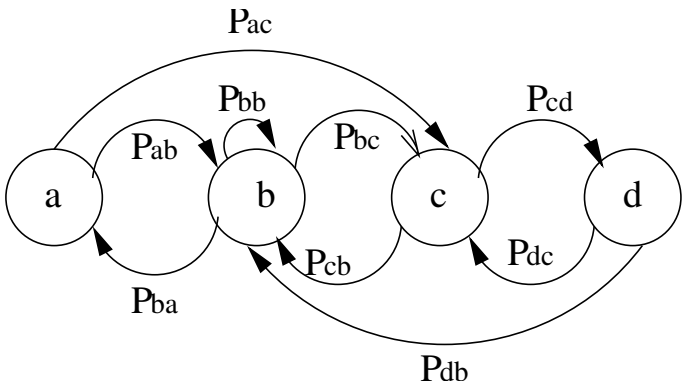
The sum over all transition probabilities leaving state i must equal 1:

$$\sum_{j \in S} P_{ij} = 1 \qquad for\ all\ i \in S$$

The transition probabilities can be written into a transition matrix:

$$P = \begin{Bmatrix} P_{00} & P_{01} & P_{02} & \cdots \\ P_{10} & P_{11} & P_{12} & \cdots \\ & & \ddots & \\ P_{i0} & P_{i1} & P_{i2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{Bmatrix}$$

Example: Markov chain with 4 states and the corresponding transition probability matrix.



$$P = \begin{Bmatrix} & a: & b: & c: & d: \\ a: & 0 & P_{ab} & P_{ac} & 0 \\ b: & P_{ba} & P_{bb} & P_{bc} & 0 \\ c: & 0 & P_{cb} & 0 & P_{cd} \\ d: & 0 & P_{db} & P_{dc} & 0 \end{Bmatrix}$$

The probability of moving from state i to state j in $m+n$ steps is described by the Chapman-Kolmogorov equation:

$$P_{ij}^{m+n} = P\{X(k+m+n) = j \mid X(k) = i\} = \sum_{l \in S} P_{il}^{(m)} P_{lj}^{(n)}$$

$$for\ n, m \geq 0;\ i, j \in S$$

A Markov chain is irreducible if each state can be reached from any other state, otherwise it is reducible.

If a state i can be reentered only after d ($d > 1$) steps or a multiple of d steps, it is called periodic. Markov chain is aperiodic if none of its states is periodic.

For a periodic state i with period d

$$P_{ii}^{(m)} = \begin{cases} > 0 \ if \ d \mid m \\ = 0 \ if \ d \nmid m \end{cases}$$

A probability distribution $\{P_j \mid j \in S\}$ is a stationary distribution if

$$p_j = \sum_{i=0}^{\infty} p_i P_{ij}, \ \forall j \in S$$

For $\overline{p} = (p_0, p_1, \ldots)$:

$$\overline{p} = \overline{p}P$$

For irreducible and aperiodic Markov chains:

$$p_j = \lim_{n \to \infty} P(X_n = j \mid X_0 = i) \quad \forall i, j \in S$$

and with probability 1:

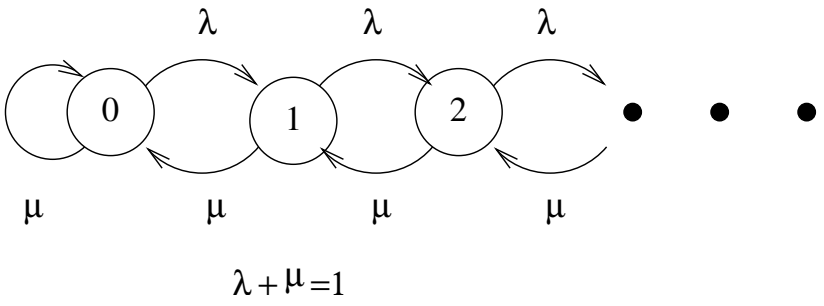
$$p_j = \lim_{k \to \infty} \frac{number \ of \ visits \ to \ state \ j \ up \ to \ time \ k}{k}$$
$$\forall j \in S$$

Theorem: In an irreducible, aperiodic Markov chain, there are two possibilities for p_j

1. $p_j = 0 \quad \forall j$; ($\nexists$ stationary distribution)
2. $p_j > 0 \quad \forall j$; $\overline{p}$ is the unique stationary distribution.

Remark: If $\mid S \mid < \infty$, case 1 never happens.

Example:



If $\mid S \mid = \infty$ and $\lambda > \mu$ then we have case 1.

1.13 Global balance equations

$$p_j \sum_i P_{ji} = \sum_i p_i P_{ij} \quad \forall j \quad (G1)$$

Proof: follows from

$$p_j = \sum_i p_i P_{ij} \quad and \quad \sum_i P_{ji} = 1$$

Another form of (G1)

$$p_j \sum_{i \neq j} P_{ji} = \sum_{i \neq j} p_i P_{ij} \quad \forall j \quad (G2)$$

Remark:

$p_i P_{ij}$ - frequency of transitions $i \to j$.

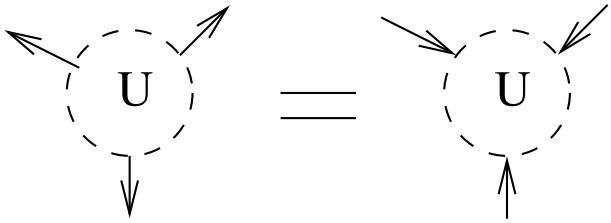
Equation (G2) means that frequency of transitions

$j \to$ equals to that of $\to j$

1.14 Generalized global balance equations

Let $U \subset S$, then

$$\sum_{j \in U} p_j \sum_{i ! \in U} P_{ji} = \sum_{i ! \in U} p_i \sum_{j \in U} P_{ij} \quad (G3)$$

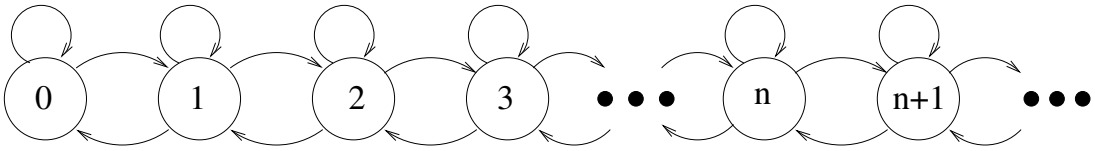


Proof: Follows from (G1) and

$$\sum_{i} = \sum_{i \in U} + \sum_{i ! \in U}$$

1.15 Detailed balance equations

Birth-death systems:



$P_{ij} = 0 \quad \text{if} \quad |\, i - j \,| > 1$

$P_{i,i+1} > 0 \quad \text{and} \quad P_{i+1,i} > 0 \quad \forall i \in S$

This is necessary and sufficient condition for the chain to be irreducible.

Let $U = \{0, 1, \ldots, n\}$. From (G3) we get

$$p_n P_{n,n+1} = p_{n+1} P_{n+1,n} \quad \forall n \in S$$

Detailed balance equations:

$$p_j P_{ji} = p_i P_{ij} \quad \forall i, j \in S \quad (DBE)$$

A solution $\overline{p}$ of (DEB) is the stationary distribution.

A common approach:

Try to solve (DBE) with $\sum_j p_j = 1$

1. The system of equations if inconsistent (DB hypothesis is wrong)
2. The found solution is the stationary distribution

1.16 Continuous-time Markov chain

is a process $X(t),\ t \geq 0$, taking values from the set of states $S = \{0, 1, \ldots\}$. Each time it enters state i:

1. The time it spends in state i is exponentially distributed with parameter ν_i, where ν_i is the rate (trans./sec) at which the process makes a transition from state i.

2. When the process leaves state i, it will enter state j with probability P_{ij}, $\sum_j P_{ij} = 1$. We call
$$q_{ij} = \nu_i P_{ij}$$
the transition rate (trans./sec) from i to j.

Discrete - time Markov chain with transition probabilities P_{ij} is called the embedded chain.

We require that the embedded chain is irreducible.

We also require that the number of transitions in any finite interval of time is finite with probability 1 (regular chains).

The limit

$$p_j = \lim_{t\to\infty} P\{X(t) = j \mid X(0) = i\}$$

exists, is independent of i. p_j is steady-state occupancy probability of state j.

Denote $T_j(t)$ the time spent in state j up to time t, then

$$p_j = \lim_{t\to\infty} \frac{T_j(t)}{t}$$

There are two cases:

$$p_j = 0 \quad \forall j; \quad or\ p_j > 0 \quad \forall j$$

Global balance equations

$$p_j \sum_i q_{ji} = \sum_i p_i q_{ij} \quad \forall j \quad (GC)$$

If $\overline{p}$ satisfies (GC) then $\overline{p}$ is steady-state distribution.

$p_i q_{ij}$ is the frequency of transitions (per sec) $i \to j$.

Detailed balance equations

$$p_j q_{ji} = p_i q_{ij} \quad i, j \in S \quad (DBEC)$$

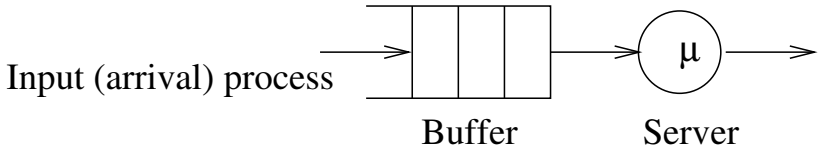
For birth-death systems $q_{ij} = 0$ for $\mid i - j \mid > 1$ and (DBEC) holds.

Again one can try to solve (DBEC) with $\sum_j p_j = 1$.

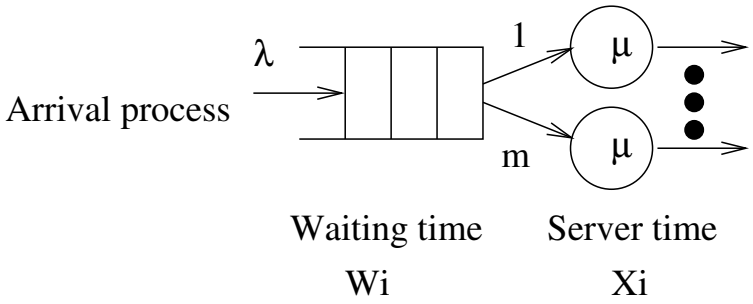
Chapter 2

Queueing Theory

Single server system:



Multiple server system:



Applications:

- Supermarkets. Priority for quick customers.
- Traffic. Control of traffic lights.
- Economy. Design of plants.
- Digital networks. Design and operate the networks to achive maximum performance..

Typical questions in Queueing theory:

1. The average number of customers in the system.

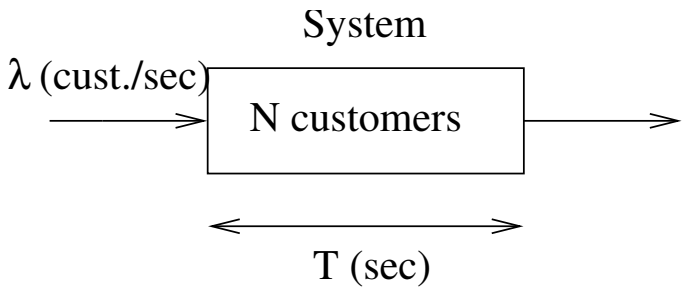
2. The average delay per customer $E(W_i + X_i)$.

What is known:

1. The customer arrival rate λ.

2. The customer service rate (when busy) $m\mu$.

2.1 Little's Theorem



$N(t)$ - number of customers in the system at time t.

N_t - time-average number of customers up to time t.

$$N_t = \frac{1}{t}\int_0^t N(\tau)d\tau.$$

N - (steady-state) time average

$$N = \lim_{t\to\infty} N_t$$

$\alpha(t)$ number of customers who arrived in $[0, t]$

$\lambda_t = \frac{1}{t}\alpha_t$ time-average arrival rate in $[0, t]$

$\lambda = \lim_{t\to\infty} \lambda_t$ (steady-state) arrival rate

T_i time spent in the system by customer i

$T_t = \frac{1}{\alpha(t)} \sum_{i=0}^{\alpha(t)} T_i$ - average time spent in system per customer in $[0, t]$

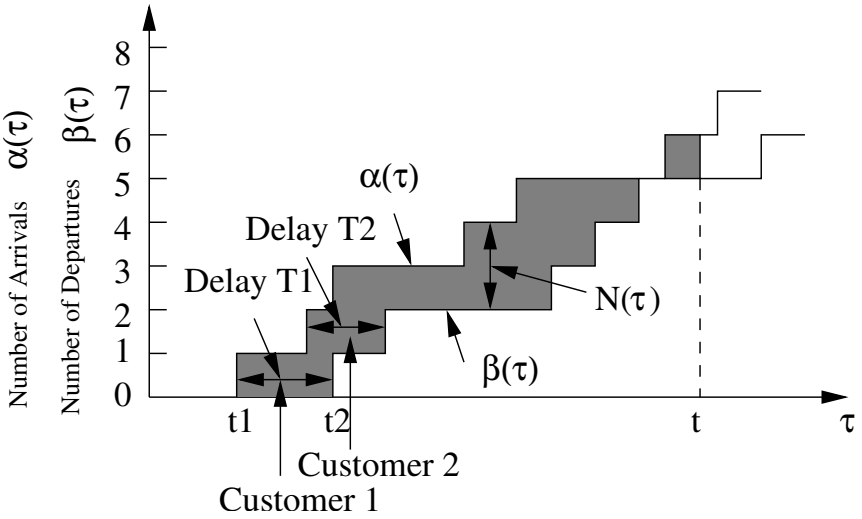
$T = \lim_{t\to\infty} T_t$ - (steady-state) customer delay

Theorem [Little, 1961]

$$N = \lambda T$$

2.1.1 Idea of proof

τ



Assumptions:

- $N(0) = 0$
- The customers depart in the same order they arrive

$\beta(t)$ the number of departures up to time t

$N(t) = \alpha(t) - \beta(t)$

The shaded area $= \int_0^t N(\tau)d\tau$

• If $N(t) = 0$ the shaded area $= \sum_{i=1}^{\alpha(t)} T_i$

$$\frac{1}{t}\int_0^t N(\tau)d\tau = \frac{1}{t}\sum_{i=1}^{\alpha(t)} T_i = \frac{\alpha(t)}{t} \cdot \frac{1}{\alpha(t)} \sum_{i=1}^{\alpha(t)} T_i$$

$$N_t = \lambda_t T_t$$

$\lim_{t\to\infty}$ gives the Little's theorem.

2.1.2 Probabilistic form of Little's theorem

$p_n(t)$ - probability of n customers in the system at t

$\overline{N}(t) = \Sigma_{n=0}^{\infty} n p_n(t)$ the average number in the system at time t

Assume we have (steady-state) limit

$$\lim_{t\to\infty} p_n(t) = p_n, \quad n = 0, 1, \ldots$$

$\overline{N} = \Sigma_{n=0}^{\infty} n p_n$ and typically $\overline{N} = \lim_{t\to\infty} \overline{N}(t)$

We consider ergodic processes only:

the time average N with probability 1 equals to the statistical average $\overline{N}$

$$N = \lim_{t\to\infty} N_t = \lim_{t\to\infty} \overline{N}(t) = \overline{N}$$

$\overline{T_k}$ average delay of customer k

$\overline{T} = \lim_{k\to\infty} \overline{T_k}$ (steady-state) delay

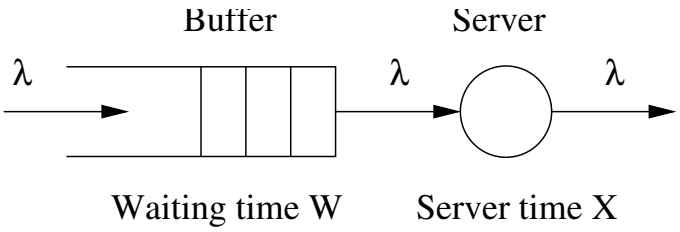
Little's Theorem

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

where

$$\lambda = \lim_{t\to\infty} \frac{1}{t} \, (expected\ number\ of\ arrivals\ in\ [0, t])$$

2.1.3 Applications of Little's theorem



1. System: Buffer + Server

   $T = W + X$ average time in the system

   N average number of packets in the system

   $$N = \lambda T$$

2. System: Server

   ρ - the average number of packets under transmission

   $$\rho = \lambda X$$

   Since 1 or 0 packets can be under in server transmission, ρ means server or line's utilization factor, part of time the line is busy.

3. System: Buffer

   N_Q - average number of packets in queue

   $$N_Q = \lambda W$$

4. Compare ordinary restaurant and fast-food restaurant serving λ customers per hour.

   $$T_f \ll T_o \Rightarrow N_f = \lambda T_f \ll N_o = \lambda T_o$$

Kendall notation:

Arrival process/ service process/ ♯servers/ ♯places in system

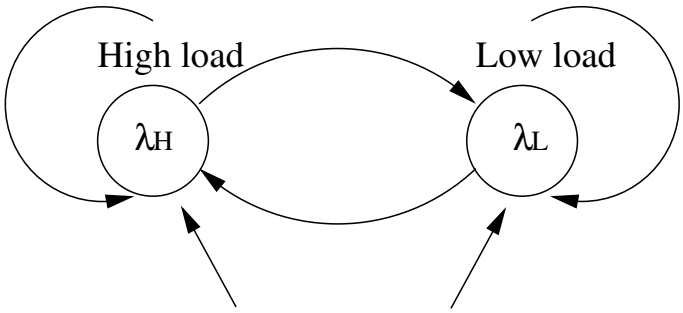
Arrival process and service process denote the statistical properties of time between two successive input/service events;

- M: memoryless process, Poisson process
- D: deterministic: constant time (e.g. constant service time)
- E_k: Erlang-k distributed time
- $MMPP$: Markov-modulated Poisson process
- G: Process with general distribution function

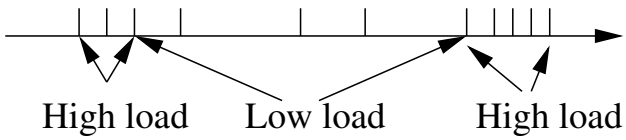
Example Kendall notation: $M/D/2/5$

- M: Poisson arrival process
- D: constant service time
- 2: 2 servers
- 5: 5-2=3 buffer places

Example Markov-modulated Poisson process:



Rates of Poisson process



2.2 $M/M/1$ Queueing system

Customers arrive according to a Poisson process with rate λ. Interarrival times

$$\tau_n = t_{n+1} - t_n$$

are independent and

$$P\{\tau_n \leq x\} = 1 - e^{-\lambda x}$$

Probability distribution of the service time is exponential with mean $E(X_i) = 1/\mu$ seconds. Service rate μ customers per second.

Customer service times s_n are mutually independent and independent of all interarrival times and

$$P\{s_n \leq x\} = 1 - e^{-\mu x}$$

2.2.1 Memoryless character of exponential distribution

$$P\{\tau_n > r + t | \tau_n > t\} = P\{\tau_n > r\}$$
$$P\{s_n > r + t | s_n > t\} = P\{s_n > r\}$$

Proof: $P\{\tau_n > r + t | \tau_n > t\} = \frac{P\{\tau_n > r+t\}}{P\{\tau_n > t\}} = \frac{e^{-\lambda(r+t)}}{e^{-\lambda t}} = e^{-\lambda r} = P\{\tau_n > r\}$

Little's Theorem:

$$N = \lambda T, \quad N = \sum_{n=0}^{\infty} n p_n, \quad N_Q = \lambda W, \quad T = \frac{N}{\lambda}$$

2.2.2 Markov chain formulation

From memoryless property $\rightarrow$ $N(t)$ is a continuous-time Markov chain.

Consider discrete-time Markov chain at times $0,\ \delta,\ 2\delta, \ldots, k\delta, \ldots$

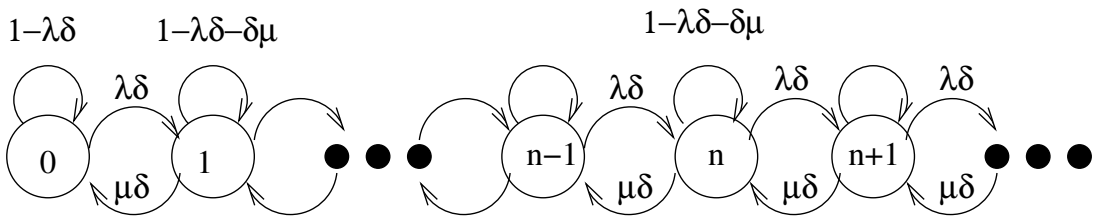
$N_k = N(t = k\delta)$ - number of customers

$P_{ij} = P\{N_{k+1} = j | N_k = i\}$

1. $P_{00} = 1 - \lambda\delta + o(\delta)$
2. $P_{ii} = 1 - \lambda\delta - \mu\delta + o(\delta) \quad i \geq 1$
3. $P_{i,i-1} = \mu\delta + o(\delta)$
4. $P_{i,i+1} = \lambda\delta + o(\delta)$
5. $P_{ij} = o(\delta) \quad j \neq i, i+1, i-1$

Proof:

1. Probability of 0 arrivals in Poisson process
2. Probability of 0 arrivals and 0 departures in δ interval is $[1 - \lambda\delta + o(\delta)][1 - \mu\delta + o(\delta)]$
3. Probability of 0 arrivals and 1 departure $[1 - \lambda\delta + o(\delta)][\mu\delta + o(\delta)]$
4. Probability of 1 arrival and 0 departures $[\lambda\delta + o(\delta)][1 - \mu\delta + o(\delta)]$
5. Probability of > 1 arrivals or > 1 departures in δ interval is $o(\delta)$.



2.2.3 Stationary distribution p_n

Detailed balance equations:

$$p_n\lambda\delta + o(\delta) = p_{n+1}\mu\delta + o(\delta)$$

$$\delta \to 0;\ p_n\lambda\delta = p_{n+1}\mu\delta$$

$$p_{n+1} = \rho p_n,\ n = 0, 1, \ldots,$$

where $\rho = \frac{\lambda}{\mu}$.

$$p_n = \rho^n p_0,\ n = 0, 1, \ldots$$

Normalization:

$$1 = \sum_{n=0}^{\infty} p_n = \sum_{n=0}^{\infty} \rho^n p_0 = \frac{p_0}{1-\rho}$$

$$p_0 = 1 - \rho,\ \underline{\rho < 1},$$

$$\underline{\rho = 1 - p_0 \quad utilization\ factor}$$

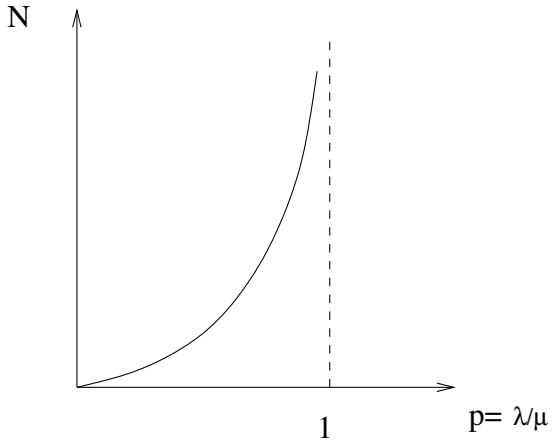
$$\underline{p_n = \rho^n(1-\rho)}\ \ stationary\ distribution$$

Average number of customers:

$$N = \sum_{n=0}^{\infty} np_n = \sum_{n=0}^{\infty} n\rho^n(1-\rho) = \rho(1-\rho)\sum_{n=0}^{\infty} n\rho^{n-1}$$

$$= \rho(1-\rho)\frac{d}{d\rho}(\sum_{n=0}^{\infty}\rho^n) = \rho(1-\rho)\frac{d}{d\rho}(\frac{1}{1-\rho})$$

$$= \rho(1-\rho)\frac{1}{(1-\rho)^2} = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu-\lambda} = N$$



The average delay:

$$T = \frac{N}{\lambda} = \frac{\rho}{\lambda(1-\rho)} = \frac{1}{\mu - \lambda}$$

The average waiting time in queue:

$$W = T - X = \frac{1}{\mu - \lambda} - \frac{1}{\mu} = \frac{\rho}{\mu - \lambda}$$

The average number of customers in queue:

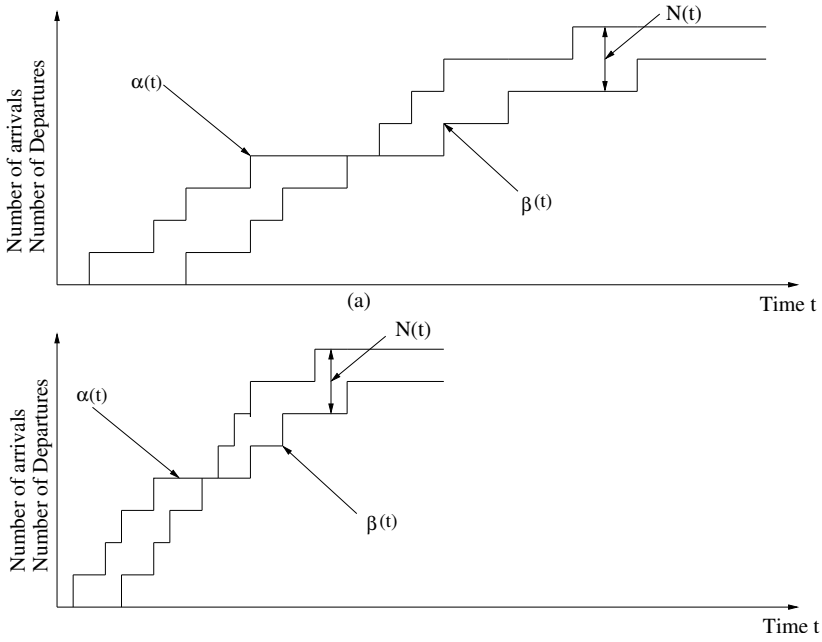
$$N_Q = \lambda W = \frac{\rho^2}{1-\rho}$$

Increasing λ and μ by the same factor

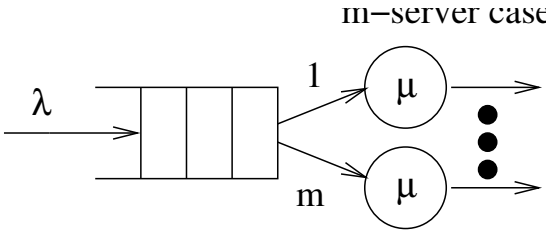
Let $\lambda \leftarrow K\lambda,\ \mu \leftarrow K\mu$. Then

$N = \frac{K\lambda}{K\mu - K\lambda} = \frac{\lambda}{\mu - \lambda}$ stays the same.

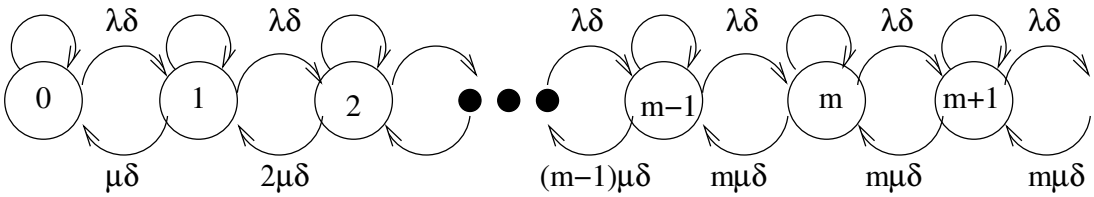
$T = \frac{1}{K\mu - K\lambda} = \frac{1}{K(\mu - \lambda)}$ decreased K times.



2.3 $M/M/m$ Queueing system



Markov chain:



Detailed balance equations: $\delta \to 0$

$$p_n = \frac{1}{n}\frac{\lambda}{\mu}p_{n-1} = \frac{m\rho}{n}p_{n-1}, \ \ n \le m$$

$$p_n = \frac{1}{m}\frac{\lambda}{\mu}p_{n-1} = \rho p_{n-1}, \ \ n \ge m$$

$$p_m = p_0\frac{m^m\rho^m}{m!}$$

$$p_n = \begin{cases} p_0\frac{(m\rho)^n}{n!}, & n \le m \\ \\ p_0\frac{m^m\rho^n}{m!}, & n \ge m \end{cases}$$

where

$$\rho = \frac{\lambda}{m\mu} < 1$$

Normalization:

$$p_0 = [\sum_{n=0}^{m-1} \frac{(m\rho)^n}{n!} + \sum_{n=m}^{\infty} \frac{(m\rho)^m}{m!} \frac{1}{\rho^{m-n}}]^{-1}$$

and finally

$$p_0 = [\sum_{n=0}^{m-1} \frac{(m\rho)^n}{n!} + \frac{(m\rho)^m}{m!(1-\rho)}]^{-1}$$

2.3.1 Erlang C formula

$$P(Queueing) = \sum_{n=m}^{\infty} p_n = \sum_{n=m}^{\infty} \frac{p_0 m^m \rho^n}{m!} = \frac{p_0 (m\rho)^m}{m!} \sum_{n=m}^{\infty} \rho^{n-m} =$$

$$\frac{p_0 (m\rho)^m}{m!(1-\rho)} = P_Q$$

In $M/M/m$ model the customers remain in queue (alternative is $M/M/m/m$ system)

The expected number of customers waiting in queue

$$N_Q = \sum_{n=0}^{\infty} n p_{n+m}$$

From Erlang C formula

$$N_Q = \sum_{n=0}^{\infty} n p_0 \frac{m^m \rho^{m+n}}{m!} = \frac{p_0 (m\rho)^m}{m!} \sum_{n=0}^{\infty} n\rho^n = \frac{p_0 (m\rho)^m}{m!} \frac{\rho}{(1-\rho)^2} = P_Q \frac{\rho}{1-\rho}$$

or

$$\frac{N_Q}{P_Q} = \frac{\rho}{1-\rho}$$

$$\frac{N_Q}{P_Q} = \frac{\rho}{1-\rho},\ \rho = \frac{\lambda}{m\mu}$$

This is average number of customers in queue conditioned that all servers are busy.

Compare with $M/M/1$ system, where

$$\frac{N_Q}{P_Q} = \frac{N_Q}{\rho} = \frac{\rho}{1-\rho},\ \rho = \frac{\lambda}{\mu}.$$

So, as long as queue is not empty, the queue size of $M/M/m$ system is as for $M/M/1$ system with service rate $m\mu$.

The average waiting time W in a queue (from Little's theorem)

$$W = \frac{N_Q}{\lambda} = \frac{\rho P_Q}{\lambda(1-\rho)}$$

The average delay per customer ($\rho = \frac{\lambda}{m\mu}$)

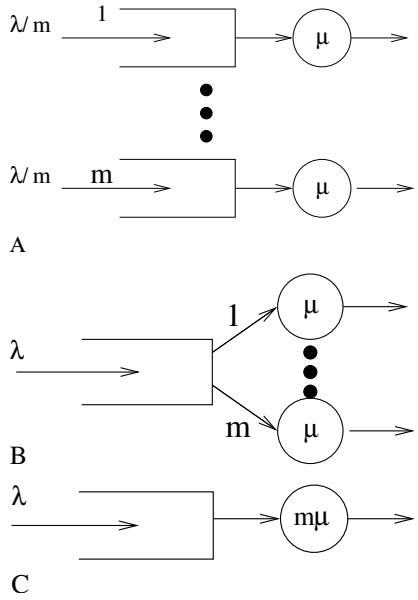
$$T = \frac{1}{\mu} + W = \frac{1}{\mu} + \frac{\rho P_Q}{\lambda(1-\rho)} = \frac{1}{\mu} + \frac{P_Q}{m\mu - \lambda}$$

The average number of customers in the system (from Little's)

$$N = \lambda T = \frac{\lambda}{\mu} + \frac{\lambda P_Q}{m\mu - \lambda} = m\rho + \frac{\rho P_Q}{1-\rho}$$

2.3.2 Statistical multiplexing

For $M/M/m$ system: $T = \frac{1}{\mu} + \frac{P_Q}{m\mu - \lambda}$



A

$$T = \frac{1}{\mu} + \frac{mP_Q}{m\mu - \lambda}$$

B

$$\widehat{T} = \frac{1}{\mu} + \frac{\widehat{P_Q}}{m\mu - \lambda}$$

C

$$\widetilde{T} = \frac{1}{m\mu} + \frac{\widetilde{P_Q}}{m\mu - \lambda}$$

1. $\rho \ll 1$ lightly loaded system

$$(P_Q, \widehat{P_Q}, \widetilde{P_Q}) \approx 0$$

$$T \approx \tfrac{1}{\mu} \quad \widehat{T} \approx \tfrac{1}{\mu} \quad \widetilde{T} \approx \tfrac{1}{m\mu}$$

2. $\rho \approx 1$ heavy loaded

$$(P_Q, \widehat{P_Q}, \widetilde{P_Q}) \approx 1$$

$$T \approx \tfrac{m}{m\mu - \lambda} \quad \widehat{T} \approx \tfrac{1}{m\mu - \lambda} \quad \widetilde{T} \approx \tfrac{1}{m\mu - \lambda}$$

2.4 $M/M/\infty$: The infinite-server case

Consider limit of $M/M/m$ system, m$\rightarrow \infty$

Detailed balance equation:

$$\lambda p_{n-1} = n\mu p_n, \ n = 1, 2, \ldots$$

$$p_n = \frac{\lambda}{\mu n} p_{n-1} = p_0 (\frac{\lambda}{\mu})^n \frac{1}{n!}, \ n = 0, 1, \ldots$$

Normalization:

$$p_0 = [\sum_{n=0}^{\infty} (\frac{\lambda}{\mu})^n \frac{1}{n!}]^{-1} = e^{-\lambda/\mu}$$

$$p_n = (\frac{\lambda}{\mu})^n \frac{e^{-\lambda/\mu}}{n!}, \ n = 0, 1, \ldots$$

In steady-state, the number in the system is Poisson distributed with parameter $\frac{\lambda}{\mu}$.

The average number (of customers) is the system

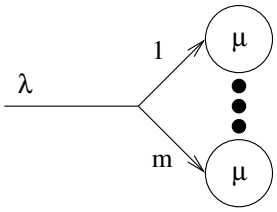
$$N = \frac{\lambda}{\mu}$$

The average delay (by Little's Theorem)

$$T = \frac{1}{\mu}$$

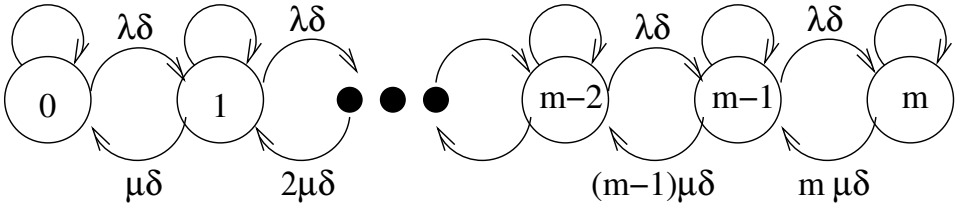
(no queue in $M/M/\infty$ system)

2.5 $M/M/m/m$: The m-server loss system



Telephony, circuit-switched networks; $1/\mu$ is average duration of telephone conversation.

Blocking: When all servers are busy, arriving customer is refused.



$$\lambda p_{n-1} = n\mu p_n, \;\; n = 1, 2, \ldots, m$$
$$p_n = p_0 (\frac{\lambda}{\mu})^n \frac{1}{n!}, \;\; n = 0, 1, \ldots, m$$

Normalization:

$$p_0 = [\sum_{n=0}^{m} (\frac{\lambda}{\mu})^n \frac{1}{n!}]^{-1}$$

Blocking probability Erlang B formula:

$$p_m = \frac{(\frac{\lambda}{\mu})^m / m!}{\sum_{n=0}^{m} (\frac{\lambda}{\mu})^n / n!}$$

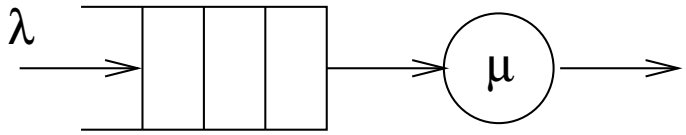
The average delay

$$T = \frac{1}{\mu}(1 - p_m) + 0 \cdot p_m$$

The average number in the system

$$N = \lambda T = \frac{\lambda}{\mu}(1 - p_m)$$

2.6 $M/G/1$ system



• Customers arrive according to Poisson process with rate λ

• and are served in the order they arrive;

• Customer service time have a general distribution

• X_i is service duration of $i-th$ arrival

• X_i are identically distributed, mutually independent, independent of the inter-arrival time.

$\overline{X} = E(X) = \frac{1}{\mu}$ average service time,

$\overline{X^2} = E(X^2)$ second moment

2.7 Pollaczek-Khinchin (P-K) formula

Average waiting time in queue (Pollaczek-Khinchin):

$$W = \frac{\lambda \overline{X^2}}{2(1-\rho)}$$

where

$$\rho = \frac{\lambda}{\mu} = \lambda \overline{X}$$

Average delay (queue + service):

$$T = \overline{X} + \frac{\lambda \overline{X^2}}{2(1-\rho)}$$

Average number of customers it the queue:

$$N_Q = \lambda W = \frac{\lambda^2 \overline{X^2}}{2(1-\rho)}$$

Average number in the system:

$$N = \lambda T = \rho + \frac{\lambda^2 \overline{X^2}}{2(1-\rho)}$$

$\rho = \lambda/\mu$

$M/G/1$	$M/M/1$	$M/D/1$
$\overline{X^2}$	$\overline{X^2} = \frac{2}{\mu^2}$	$\overline{X^2} = \frac{1}{\mu^2}$
$W = \frac{\lambda \overline{X^2}}{2(1-\rho)}$	$W = \frac{\rho}{\mu(1-\rho)}$	$W = \frac{\rho}{2\mu(1-\rho)}$

For given $\overline{X}$, variance $\sigma_X^2 = \overline{X^2} - (\overline{X})^2$ is minimum $\Leftrightarrow$ $\overline{X^2}$ is minimum

σ_X^2 is minimum for deterministic (D) case.

Hence, W, T, N_Q, N for an $M/D/1$ system are lower bounds to the $M/G/1$ system for the same λ and μ.

2.7.1 Proof of Pollaczek-Khinchin formula

W_i waiting time in queue for $i-th$ customer

R_i residual service time seen by the $i-th$ customer

X_i service time of the $i-th$ customer

N_i number of customers waiting in queue when $i-th$ customer arrives

$$W_i = R_i + \sum_{j=i-N_i}^{i-1} X_j$$

$$E(W_i) = E(R_i) + \overline{X}E(N_i)$$

(Since N_i, X_j are statistically independent)

take $lim_{i\to\infty}$

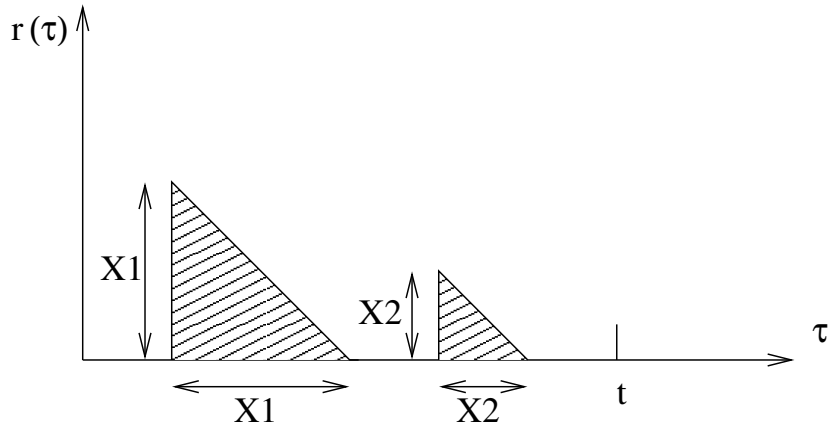
$$W = R + \frac{1}{\mu}N_Q$$

where $R = \lim_{i\to\infty} E(R_i)$ is mean residual time

By Little's theorem:

$$N_Q = \lambda W,\ W = R + \rho W,\ \rho = \frac{\lambda}{\mu},\ and\ W = \frac{R}{1-\rho}\ (*)$$

$r(\tau)$ residual service-time at time τ



$$\frac{1}{t}\int_0^t r(\tau)d\tau = \frac{1}{t}\sum_{i=1}^{M(t)}\frac{1}{2}X_i^2$$

$M(t)$-number of service completions within $[0,t]$

$$\frac{1}{t}\int_0^t r(\tau)d\tau = \frac{1}{2}\frac{M(t)}{t}\frac{1}{M(t)}\sum_{i=1}^{M(t)}X_i^2$$

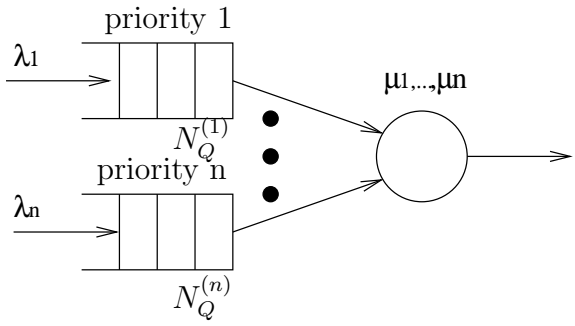
$$\lim_{t\to\infty}: \; R = \frac{1}{2}\lambda\overline{X^2}$$

and from (*)

$$W = \frac{\lambda\overline{X^2}}{2(1-\rho)}$$

2.8 Priority Queueing

$M/G/1$ nonpreemptive priority



$N_Q^{(k)}$ average number in queue for priority k

W_k average queueing time for priority k

$\rho_k = \lambda_k/\mu_k$ system utilization

R mean residual service time

First priority class:

$$W_1 = R + \frac{1}{\mu_1} N_Q^{(1)}$$

Little's Theorem: for queue 1:

$$N_Q^{(1)} = \lambda_1 W_1,\ W_1 = R + \rho_1 W_1,\ W_1 = \frac{R}{1-\rho_1}$$

Second priority class:

$$W_2 = R + \frac{1}{\mu_1} N_Q^{(1)} + \frac{1}{\mu_2} N_Q^{(2)} + \frac{1}{\mu_1} \lambda_1 W_2$$

The last item due to customers of higher priority that arrive while a customer is waiting in a queue.

From Little's Theory

$$N_Q^{(K)} = \lambda_k W_K$$

we get

$$W_2 = R + p_1 W_1 + p_2 W_2 + p_1 W_2$$

and

$$W_2 = \frac{R + p_1 W_1}{1 - p_1 - p_2}$$

Using

$$W_1 = R/(1 - p_1)$$

we get

$$W_2 = \frac{R}{(1 - p_1)(1 - p_1 - p_2)}$$

General case: priority class i:

$$W_i = R + \sum_{j=1}^{i} \frac{1}{\mu_j} N_Q^{(j)} + \sum_{j=1}^{i-1} \frac{1}{\mu_j} \lambda_j W_i$$

From Little's Theory:

$$\left.\begin{aligned} W_i &= R + \textstyle\sum_{j=1}^{i} p_j W_j + W_i \sum_{j=1}^{i-1} p_j \\ W_{i-1} &= R + \textstyle\sum_{j=1}^{i-1} p_j W_j + W_{i-1} \sum_{j=1}^{i-2} p_j \end{aligned}\right\} (*)$$

Denote

$$\sigma_i \doteq 1 - \sum_{j=1}^{i} p_j = 1 - p_1 - \ldots - p_i$$

From $(*)$ get

$$W_i \sigma_i = W_{i-1} \sigma_{i-2}$$

or

$$W_i = W_{i-1} \frac{\sigma_{i-2}}{\sigma_i}$$

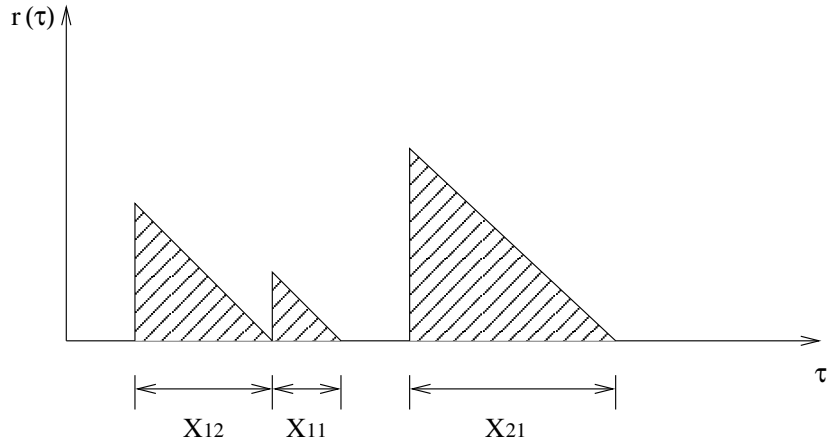
We have

$$W_2 = \frac{R}{\sigma_1 \sigma_2};$$

$$W_3 = W_2 \frac{\sigma_1}{\sigma_3} = \frac{R}{\sigma_1 \sigma_2} \frac{\sigma_1}{\sigma_3} = \frac{R}{\sigma_2 \sigma_3};$$

$$\ldots$$

$$W_k = \frac{R}{\sigma_{k-1} \sigma_k}$$



$r(\tau)$ residual service-time at time τ

X_{ij} service-time of $i-th$ customer of priority j

$M_j(t)$ number of service completions with in $[0,t]$

$$\frac{1}{t}\int_0^t r(\tau)d\tau = \frac{1}{t}\sum_{j=1}^{n}\sum_{i=1}^{M_j(t)}\frac{1}{2}X_{ij}^2 = \frac{1}{2}\sum_{j=1}^{n}\frac{M_j(t)}{t}\frac{1}{M_j(t)}\sum_{i=1}^{M_j(t)}X_{ij}^2$$

$$\lim_{t\to\infty}: \qquad R = \frac{1}{2}\sum_{j=1}^{n}\lambda_j\overline{X_j^2}$$

Average waiting time in queue for priority k

$$W_k = \frac{R}{\sigma_{k-1}\sigma_k} = \frac{\sum_{j=1}^{n}\lambda_j\overline{X_j^2}}{2(1-p_1-\ldots-p_{k-1})(1-p_1-\ldots-p_k)}$$

Average delay per customer:

$$T_k = \frac{1}{\mu_k} + W_k$$

2.9 $M/G/1$ Preemptive resume priority

Service of a customer is interrupted when a higher-priority customer arrives and is resumed from the point of interruption when all customers of higher priority have been served.

The delay T_k for a customer of priority k consists of:

1. average service time $1/\mu_k$
2. time to service customers of priority 1 to k
   (= waiting time in $M/G/1$ system without priorities)
   $$\frac{R_k}{1 - p_1 - \ldots - p_k}$$
   where R_k is the mean residual time
   $$R_k = \frac{1}{2} \sum_{i=1}^{k} \lambda_i \overline{X_i^2}$$
3. the average waiting time for new arriving customers of priority 1 to $k-1$
   $$\sum_{i=1}^{k-1} \frac{1}{\mu_i} \lambda_i T_k = T_k \sum_{i=1}^{k-1} p_i$$
   So,we get
   $$T_k = \frac{1}{\mu_k} + \frac{R_k}{(1 - p_1 - \ldots - p_k)} + T_k \sum_{i=1}^{k-1} p_i$$
   or
   $$T_k = \frac{1}{\mu_k(1 - p_1 - \ldots - p_{k-1})} + \frac{\sum_{i=1}^{k} \lambda_i \overline{X_i^2}}{2(1 - p_1 - \ldots - p_{k-1})(1 - p_1 - \ldots - p_k)}$$

2.10 $G/G/1$ system

• interarrival times and service times are all independent

The average waiting time in queue:

$$W \leq \frac{\lambda(\sigma_a^2 + \sigma_b^2)}{2(1-\rho)} \quad (*)$$

σ_a^2 variance of the interarrival times

σ_b^2 variance of the service times

$1/\lambda$ average interarrival time

$1/\mu$ average service time

$\rho = \lambda/\mu$ utilization factor

Compare with $P-K$:

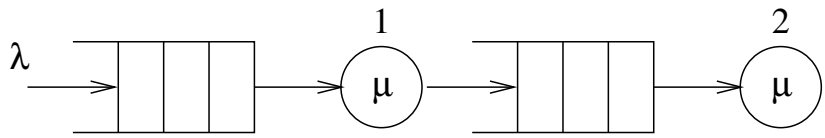
$$W = \frac{\lambda\overline{X^2}}{2(1-\rho)} \quad (**)$$

Improved (*)

$$W \leq \frac{\lambda(\sigma_a^2 + \sigma_b^2)}{2(1-\rho)} - \frac{\lambda(1-\rho)\sigma_a^2}{2} \quad (***)$$

Example: Compare (*),(**),(***) for $M/M/1$ system

2.11 Networks of transmission lines



A: 1 Poisson arrivals of rate λ pack/sec. Packets have the same length, processing (transmission) time is $= 1/\mu$ sec. We have $M/D/1$ system with waiting time given by $P - K$ formula.

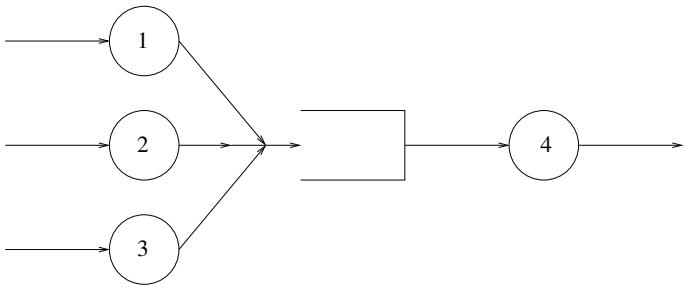
2 No waiting! Poisson assumptions are not valid. Reason: interarrival times 2 correlated with processing times 1. Indeed: interrarrival times 2 $\geq 1/\mu$

B: Now packet lengths are exponentially distributed and are independent of each other as well as of the interarrival intervals at the first queue.

1 Is $M/M/1$ queueing system

2 can not be modeled as $M/M/1$ queueing system.
Reason: interarrival times are correlated with the packet lengths.
Long packets will wait less time at 2 than short packets. Compare with slow track travelling with faster cars.

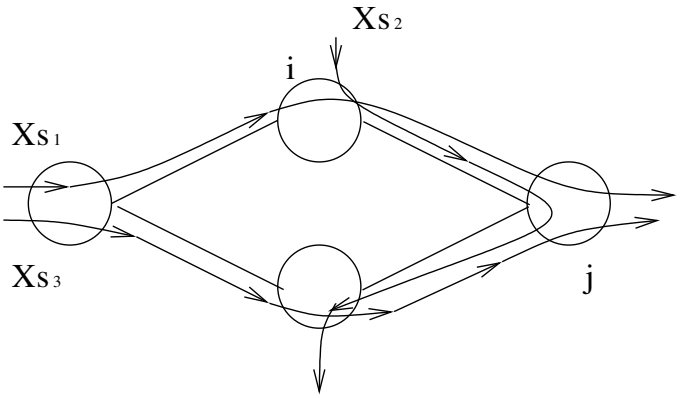
2.12 The Kleinrock independence approximation



Kleinrock(1964): Merging several packet streams on a transmission line has an effect similar to restore the independence of interarrival times and packet lengths.

As a result $M/M/1$ model can be applied.

2.13 Virtual circuit networks



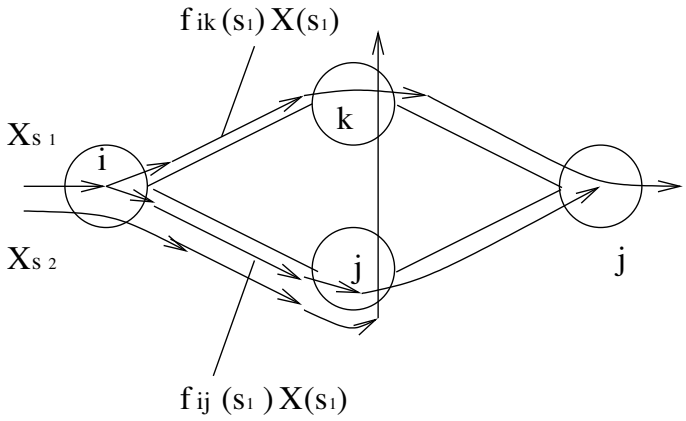
$$\lambda_{ij} = x_{s_1} + x_{s_2}$$

x_s pack/sec arrival rate of stream s.

Total arrival rate at link (i, j) is

$$\lambda_{ij} = \sum_{s \ crossing \ link \ (i,j)} x_s$$

2.14 Datagram networks



$$\lambda_{ij} = \sum_{s \text{ crossing link } (i,j)} f_{ij}(s) x_s$$

We use $M/M/1$ model for every link (i, j) (Kleinrock approximation)

Average number of packets in queue or service at (i, j) is

$$N_{ij} = \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}}$$

$1/\mu_{ij}$ average packet transmission time on link (i, j)

Total number of packets:

$$N = \sum_{(i,j)} \frac{\lambda_{ij}}{\mu_{ij} - \lambda_{ij}}$$

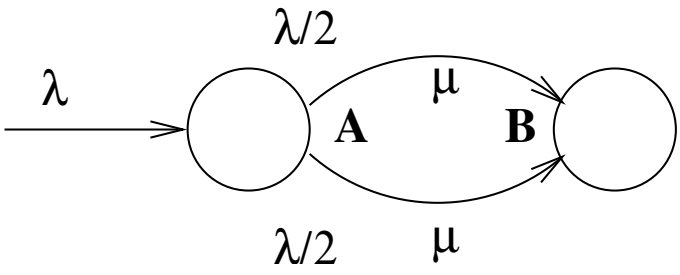
Average delay per packet neglecting processing and propagation delays (Little)

$$T = \frac{N}{\sigma}$$

where $\sigma = \sum_s x_s$ total arrival rate in the system.

In many networks packets are distributed not exponentially. If queues are independent use $P - K$.

Example:



Poisson process with rate λ divided among two links with service (transmission) rate μ.

1. Randomization Using a fair coin.
   Each of 2 queues behaves like $M/M/1$ (arrivals independent of packet length) and average delay per packet:

   $$T_R = \frac{1}{\mu - \lambda/2} = \frac{2}{2\mu - \lambda}$$

   (corresponds to Kleinrock approximation)

2. Metering Arriving packet is assigned to a queue having less number of bits. This behaves like $M/M/2$ system with arrival rate λ and each link is a server.

   Average delay

   $$T_\mu = \frac{2}{(2\mu - \lambda)(1 + \rho)} \quad (Example)$$

   with $\rho = \frac{\lambda}{2\mu}$

Metering is better than randomization, but destroys Poisson approximation. One can use $G/G/1$ system.

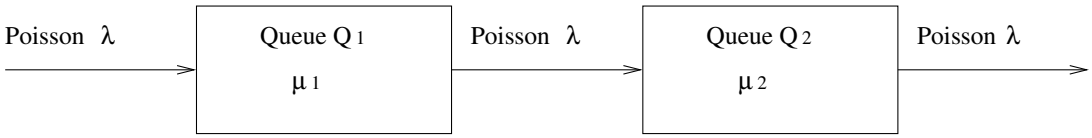
2.15 Burke's Theorem

Consider $M/M/1$, $M/M/m$ or $M/M/\infty$ system with arrival rate λ. Suppose the system starts in steady-state. Then

(a) the departure process is Poisson with rate λ.

(b) at time t, the number of customers in the system is independent of the sequence of departure times prior to t.

Example: Two queues in tandom



The service times at queues are exponentially distributed (μ_1, μ_2) and mutually independent (compare with transmission lines)

From Burke's Theorem we can show that the two queues behave as if they are independent $M/M/1$ queues in isolation.

Example:

$$P(n \text{ at } Q1,\ m \text{ at } Q2) = P(n \text{ at } Q1)P(m \text{ at } Q2)$$

Chapter 3

HierarchicalStructureofNetworkFunctionalit

Protocol: Description, how multiple parties (peer entities) interact when thay are jointly working on a single job.

Problems:

- Communication protocols are mainly implemented in software. Protocol implementations are complex-like most other software
- Communication systems often use many protocols in parallel
- Many protocols have internal states. When several protocols interact the overall statespace is equal to the Cartesian product of the individual protocol statespace. Hence, the overall state-space may become extremely huge.

The key problem for network designers in protocol description is complexity.

Counter measure: **divide and conquer** approach

- Divide whole system into sub-systems, sub-systems into sub-sub-systems,...
- Define mutually independent protocol functions (see OSI model)
- Use protocol description languages
  - UML unified modeling language
  - SDL system description language (functional description)
  - ASN.1 abstract syntax notation 1 (data structures)

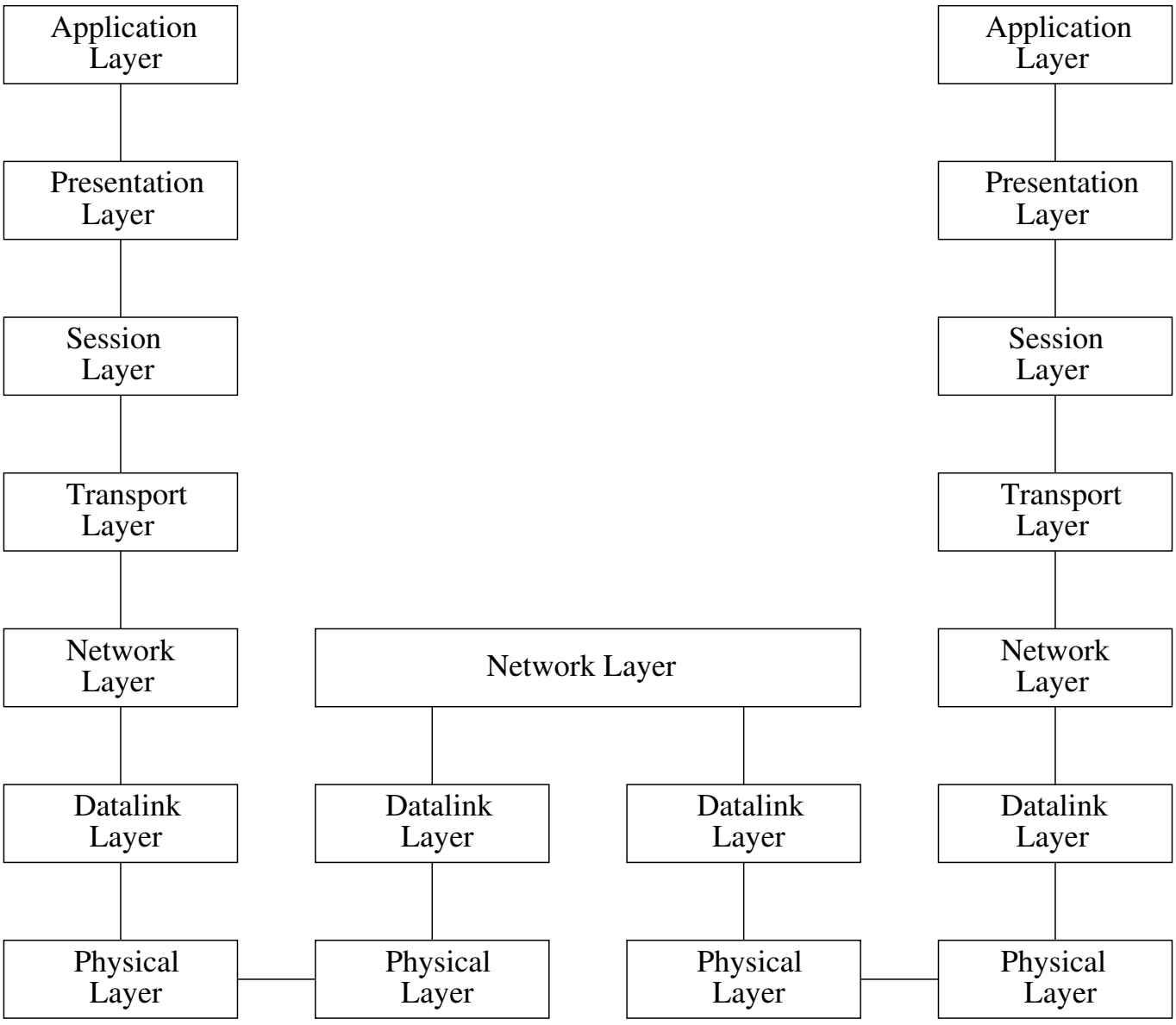
3.1 OSI Model (Open Systems Interconnections)

Design principles:

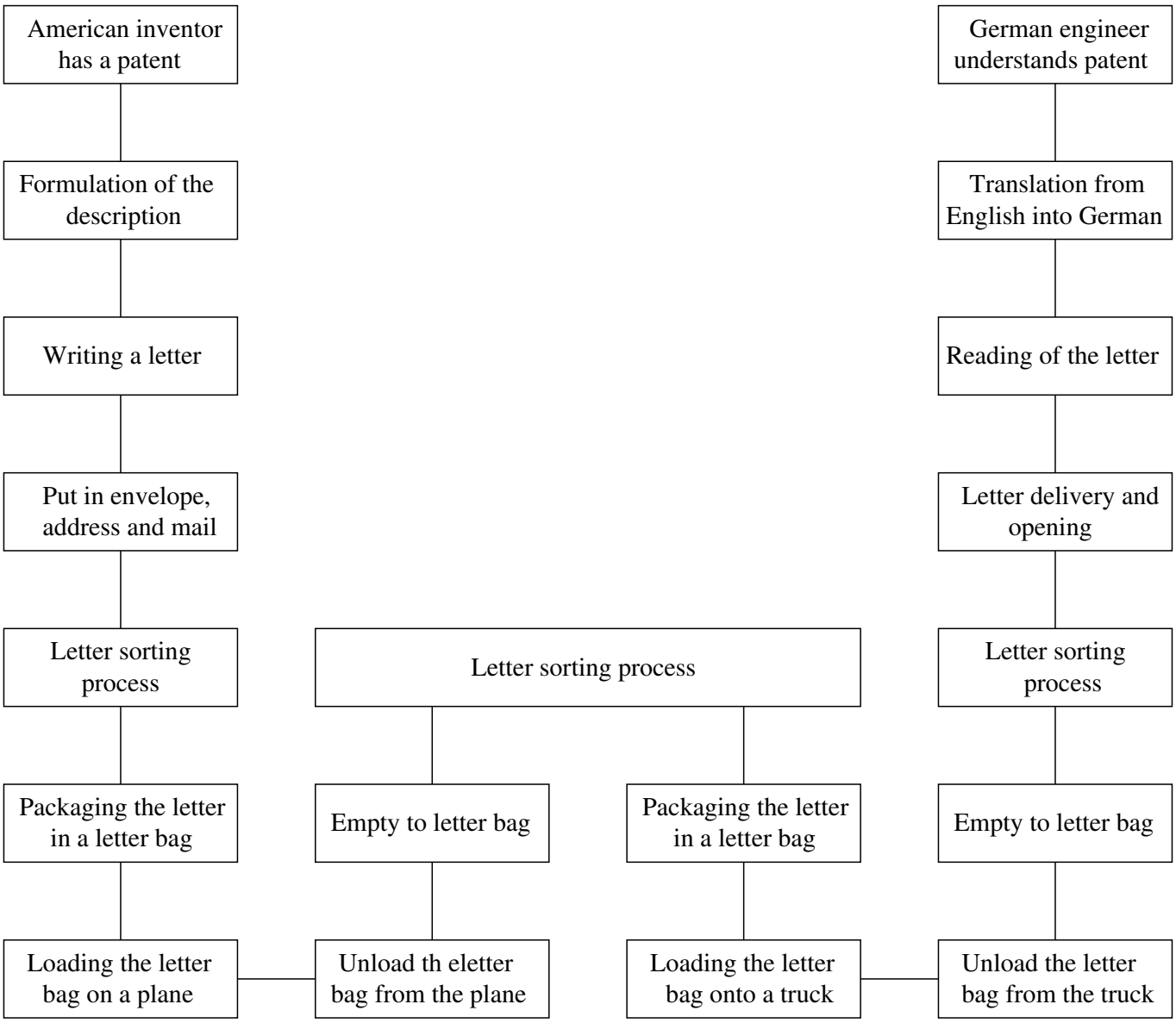
- Well-defined functions in each layer.
- Each function is completely handled in a single layer.
- Restricted inter-layer communication to avoid sides-effects (service primitives)
- Symmetry: Transmitter and receiver side of a function are always located in the same layer.

The OSI model is standardized by ISO (International Standards Organization). It is based on ideas similar to TCP/IP. In practice, the TCP/IP protocol suite is more important than the OSI model. Nevertheless, the OSI reference model and the underlying design methodology is frequently used in standardization and network design.

Hierarchical structure of the OSI model:


Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Datalink Layer
Physical Layer
Network Layer
Datalink Layer
Physical Layer
Datalink Layer
Physical Layer
Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Datalink Layer
Physical Layer

Example 1:


American inventor has a patent
Formulation of the description
Writing a letter
Put in envelope, address and mail
Letter sorting process
Packaging the letter in a letter bag
Loading the letter bag on a plane
Letter sorting process
Empty to letter bag
Unload th eletter bag from the plane
Packaging the letter in a letter bag
Loading the letter bag onto a truck
German engineer understands patent
Translation from English into German
Reading of the letter
Letter delivery and opening
Letter sorting process
Empty to letter bag
Unload the letter bag from the truck

Example 2:

James Bond meets Number One on the 7th floor of the spy headquarters building. Number One gives Bond a secret message that must get through to the US Embassy across town.

Bond proceeds to the 6th floor where the message is translated into an intermediary language, encrypted and miniaturized.
Bond takes the elevator to the 5th floor where Security checks the message to be sure it is all there and puts some checkpoints in the message so his counterpart at the US end can be sure he's got the whole message.

On the 4th floor, the message is analyzed to see if it can be combined with some other small messages that need to got the US end. Also if the message was very large it might be broken into several small packages so other spies can take it and have it reassembled on the other end.

The 3rd floor personnel check the address on the message and determine who the addressee is and advising Bond of the fastest route to the Embassy.

On the 2nd floor the message is put into a special courier pouch (packet). It contains the message, the sender and dstination ID. It also warns the recipient if other pieces are still coming.

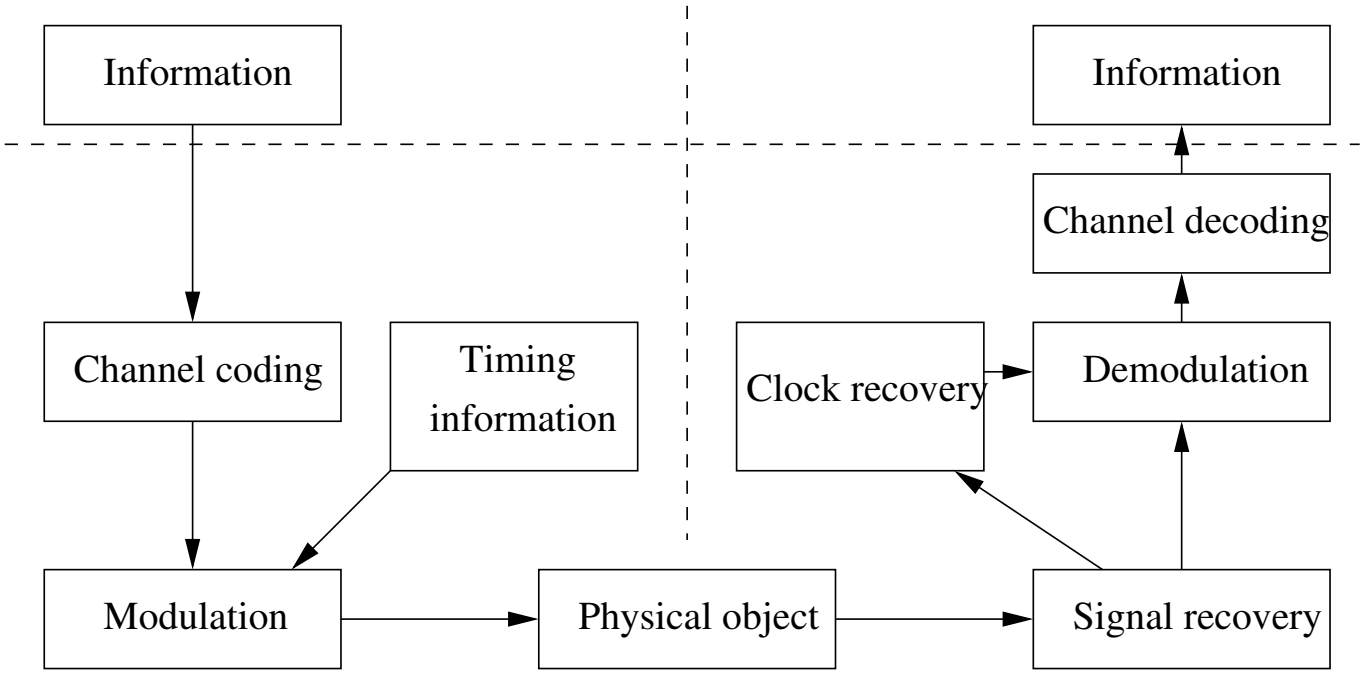
Bond proceeds to the 1st floor where Q has prepared the Aston Martin for the trip to the Embassy.

Bond departs for the US Embassy with the secret packet in hand. On the other end the process is reversed. Bond roceeds from floor to floor where the message is decoded.
The US Ambassador is very grateful the message got through safely.
"Bond, please tell Number One I'll be glad to meet him for dinner tonight."

3.1.1 Physical Layer

Main goal:

- Mapping of information onto a signal, i.e., change the state of a physical object according to the information.
- Transport the physical object.
- Recover information from the state of the received physical object.



Second goal: physical channel multiplexing, i.e. a single physical medium is used by several independent links in parallel (divide a channel into several independent sub-channels).

Example: Frequency multiplex of TV channels.

3.1.2 Data Link Control Layer

The DLC has two sub-layers:

3.1.2a Medium access control

Medium access protocols avoid that several users disturb each other‘s data transmission and assign transmission capacity in a fair way.

- Multiplex: separate sub-channels, independent usage
- Multiple access: single channel, shared usage
- Examples: Ethernet, Token Ring, ALOHA

3.1.2b Logical link control

Main goal: Provide reliable point-to-point link (no bit-errors)
Other goals: Link related ciphering and message integrity protection

Logical channel: Super imposed to physical channel, considers only information and no signal or physical objects.

Non-real time data transmission:

- Split data stream into frames (frame synchronization required)
- Add checksum (parity check bits, CRC)
- Verify correctness of checksum on receiver side
- Request transmitter to repeat frames with incorrect checksum (ARQ)
- Very low residual bit-error rate
- Memory required in transmitter and in some cases also in the receiver
- Delay and reduced throughput, when frames have to be repeated

Transparent data transmission (real time services like voice or video):

- No delay permitted
- Error-detection by checksum
- Error-concealment: muting in speech transmission, repetition of previous, already correctly received image in video transmission

3.1.3 Network Layer

Main goal:

- **Routing:** Find a way from sender to receiver which satisfies the requested Quality of Service (QoS), i.e. data rate, error-rate, packet loss, delay, ...
- **Switching:** Forwarding of traffic according to a route determined before by the routing procedure. (Note: Switching differs for connection-oriented and connection-less services)
- **Flow control:** Protect network links from overload; otherwise risk of deadlock or break down of parts or even complete network. (methods: clever routing, queuing of incoming traffic, discarding/blocking low-priority traffic)

For mobile radio networks: Extended routing and flow control mechanisms

Mobility management:

- Handles changing user location
- Central location data base: HLR
- Actual location data base: VLR
- Pageing: notify user in unknown location of incoming call

Radio resource management:

- Optimize usage of radio resources
- Admission control -long term
- Load control -intermediate
- Congestion control -short term

3.1.4 Transport Layer

Main goal:

- Provide a reliable end-to-end data transport
- ARQ protocols like in LLC layer to request retransmission of erroneous or missing packets

Second goal:

- Provide high-speed end-to end connection by distributing traffic onto-multiple network connections (downward multiplexing: multiple connections used by one user, in contrast to upward multiplexing: one connection shared among multiple users)

3.1.5 Session Layer

Main goal:

- Answer the question: Where to access a wanted service in the network? E.g. find processors in the network to share a job.
- Handles the interactions between the two end points in setting up a session
- Deals with access rights

3.1.6 Presentation Layer

Main goal:

- Handles syntax and semantics of data
- Conversion of data structures, e.g. character sets
- Source coding (speech compression, data compression)
- Cryptographix end-to-end ciphering, integrity protection of documents, authentication of communication parties

3.1.7 Application Layer

Service provided to the user:

- Telecommunications: phone, fax, SMS
- Broadcast: radio, TV
- Data communications: file transfer, remote login, e-mail, HTTP

3.1.8 Common issues of all layers

Addressing:

- Networking means that resources are shared and used jointly in parallel.
- Address headers describe where to deliver certain data.

Segmentation, fragmentation:

- Split long data streams into pieces which are more convenient to handle
- Add headers and sequence numbers to each piece.
- **Segments:** Add checksum for ARQ, i.e. each segment can be retransmitted individually
- **Fragments:** No individual chechsum, hence no retransmission of pieces. Only retransmission of complete data stream after reassembly.

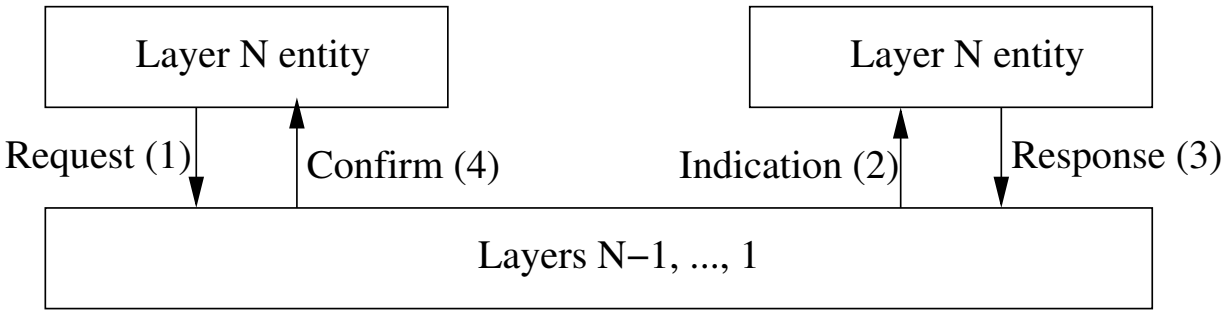
Connection setup, maintenance, release: (Connection-oriented services only)

- **Setup:** Common understanding between peer entities on parameters (modulation alphabet, data rate, QoS, frame/packet size, ...)
- **Maintenance, reconfiguration:** Check that connection is alive, adjust parameters
- **Release:** Declare resource as free again.

3.2 Communication in the OSI Model

Entity:

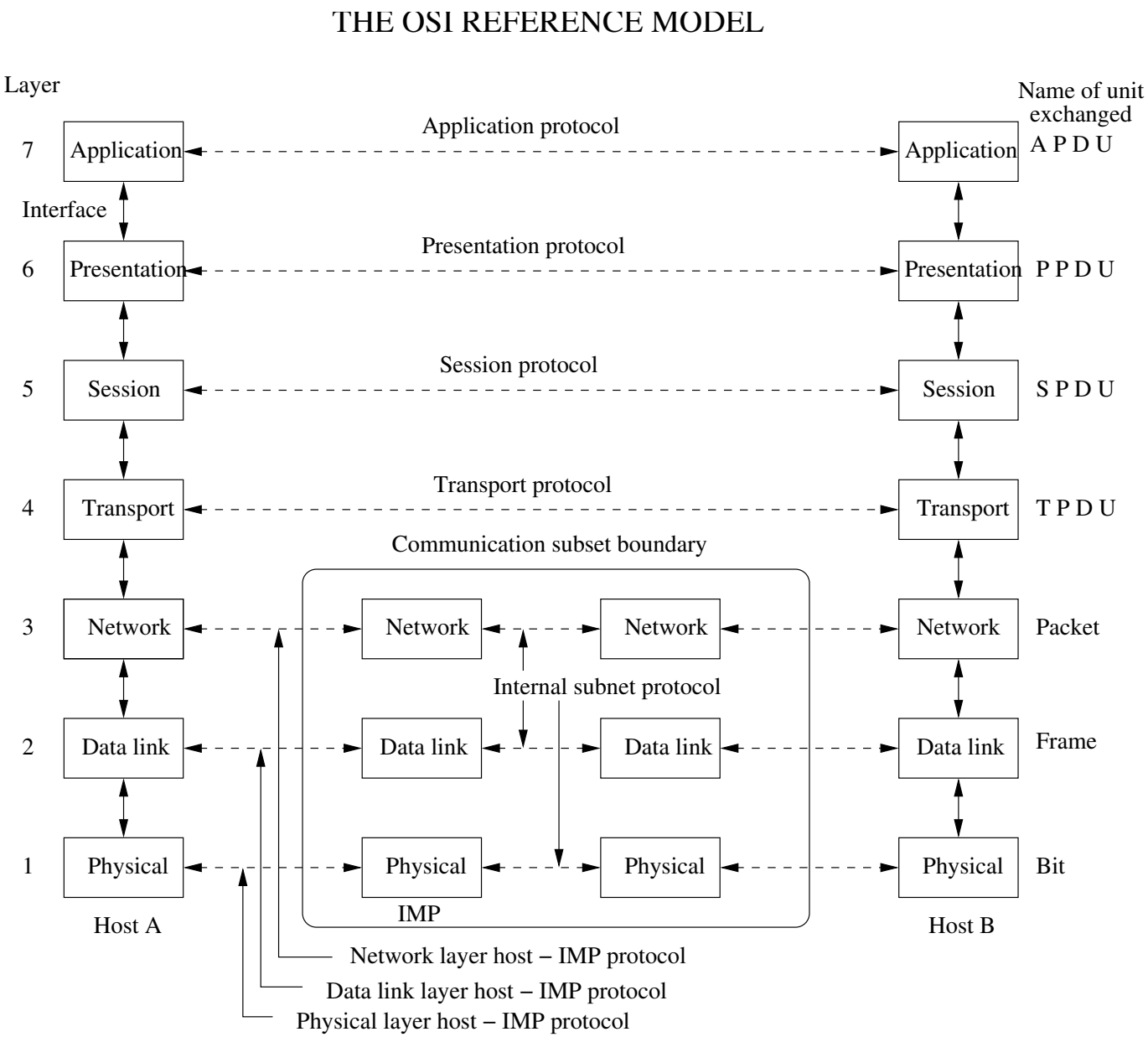
- Some parts of the OSI model have to be realized in hardwae, while some parts are realized in soft ware. However, all functional units are call **entities**.
- Hence, terminals, network nodes, processes, users, ... are entities.
- In the OSI model direct communication is only allowed between entities of adjacent layers N and N-1 using service primitives
- **A service primitive** can be considered as a call of a subroutine, i.e. a message or parameter exchange between entities
- Example of primitives:
  - connection request
  - connection confirm
  - request that data be sent
  - signal the arrival of data



Example how to use primitives:

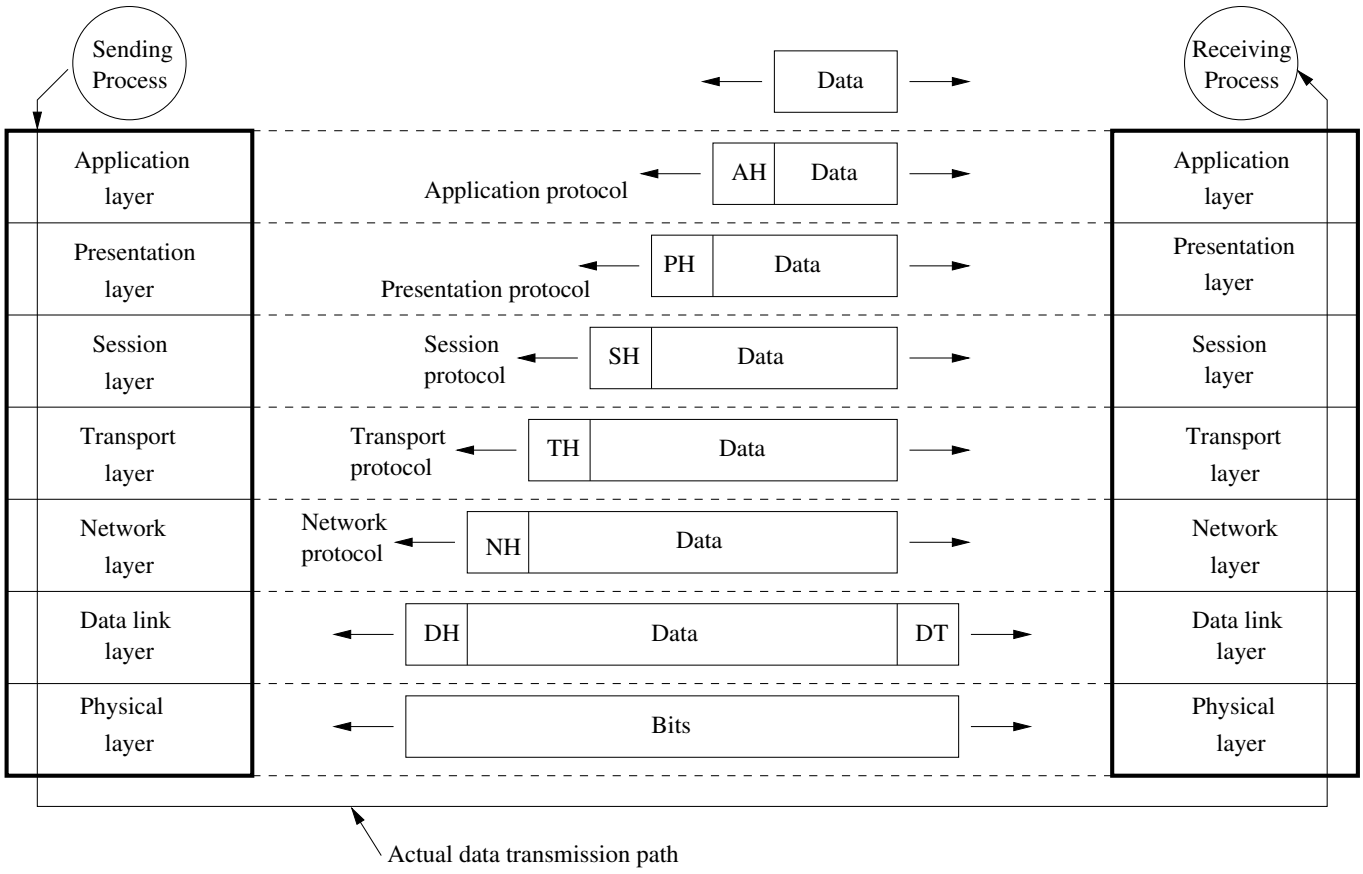
- CONNECT.request -Dial Aunt Millie‘s phone number
- CONNECT.indication -Her phone rings
- CONNECT.response -She picks up the phone
- CONNECT.confirm -You hear the ringing stop
- DATA.request -You invite her to tea
- DATA.indication -She hears your invitation
- DATA.request -She says she will be delighted to come
- DATA.indication -You hear her acceptance
- Disconnect.request -You hang up the phone
- Disconnect.indication -She hears it and hangs up too

- Entities in the same layer on different machines are called **peer entities**
- **Peer-to-peer communication (protocols)** is realized indirectly between opposite entities of the same layer. Example: ARQ protocol (layer 2) sends ACK or NAK via PHY (layer 1), without PHY understanding these data.

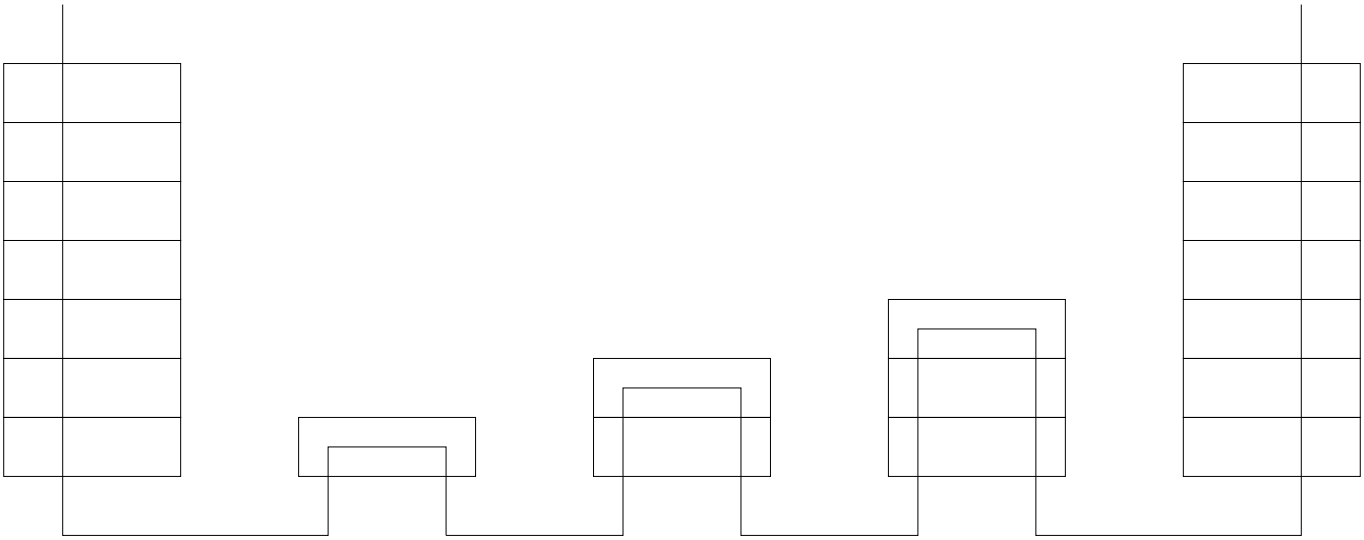


3.3 OSI Data Structure

- We distinguish between **protocol data units** (PDU) and **Service data units**(SDU)
- The PDU of layer N becomes the SDU at layer N-1.
- At layer N-1 a header is added to the SDU and the resulting packet is the PDU of layer N-1.
- The header contains information such as address, segment number, PDU length.
- The contents of a SDU is transported without being evaluated in any way.



3.4 Switching Entities in TCP/IP and OSI



Repeater: Connects two networks at physical layer

- Forwards bits from one network to another
- Two networks look like one
- Repeaters copy bits without understanding what they are doing.

Bridge: Connects two networks at data link layer

- Useful when the networks have different data link layers but the same network layer
- Example: connection of Ethernet and Token bus
- Bridges are smart
- They can be programmed to copy frames selectively and make changes.

Router: Connects two networks at network layer

- Useful when the networks have different network layers but the same transport layer
- Example: connection of X.25 and Token bus

Gateway: Connects network on higere layers

- Useful to connect networks that does not use the OSI model at all
- In many cases: connection in the application level

3.5 Remarks

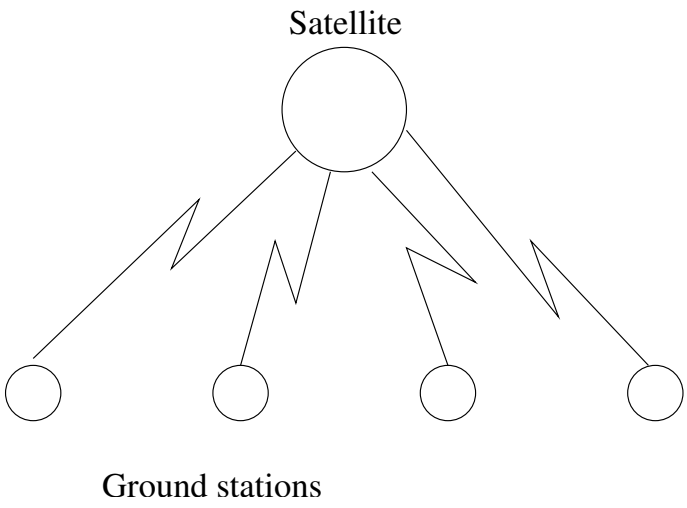
- Implementation of inter-layer communication is not specified in OSI. Hence, highly vendor specific.
- Building a protocol stack in a single physical entity from layers of multiple vendors is difficult!
- Inter-layer communication can be time consuming. This is critical in real time applications, e.g. speech transmission.
- Shortcuts by management back plane.

Chapter 4

Multiaccess communications

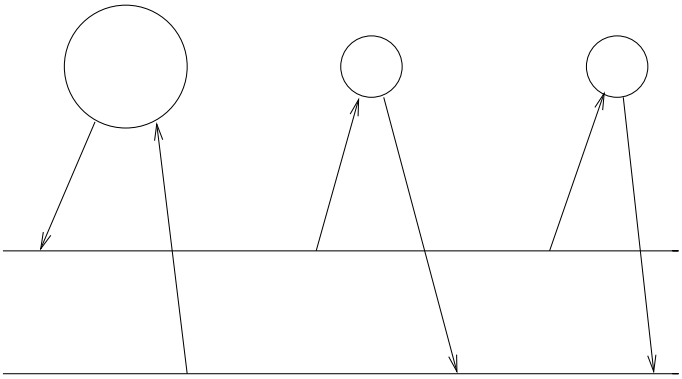
Satellite channels

FDM, TDM, Aloha



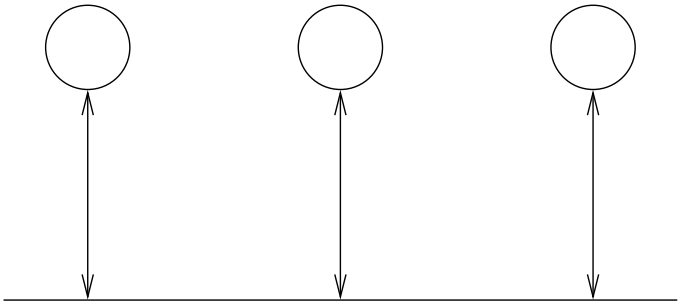
Multidrop telephone line

Polling, Aloha



Multitap bus

Round robin



Two extreme strategies:

(a) free for all (Aloha)

(b) perfectly scheduled (round robin)

Sublayer

medium access control (MAC)

sublayer of data link control (DLC)

4.1 Idealized Slotted Multiaccess Model

1. Slotted system: fixed packet length, synchronized
2. Poisson arrivals: m modes, λ/m arrival rate/node
3. Collision or perfect reception
4. $0, 1, e$ Immediate feedback: 0 packets, 1 packet, or e-error
5. Retransmission of collisions. Backlogged nodes

6a No buffering. New arrivals at busy node are discarded.

6b Infinite set of nodes ($m = \infty$)

- Approximation 6a is good for large m, small λ and small required delay.
- 6b gives an upper bound to the delay for finite m.

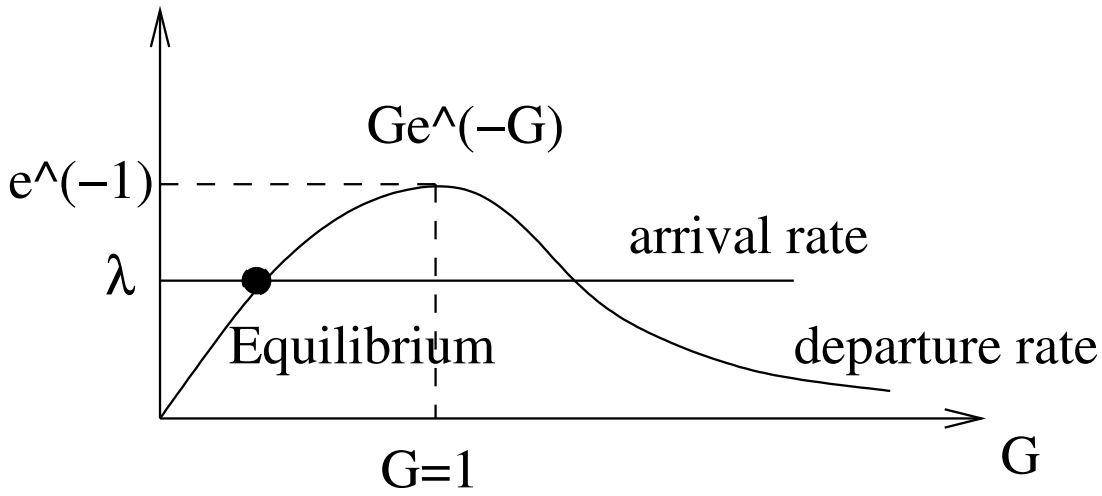
4.2 Slotted Aloha

The Aloha network (1970) Uni. of Hawaii.

• Each backlogged node retransmits at random, e.g. with probability q_r in each slot.

A total number of packets transmitted in a slot can be approximated as a Poisson random variable with parameter $G > \lambda$.

Then probability of successful transmission in a slot is Ge^{-G}.



Max departure rate at $G = 1$ is $1/e \approx 0.368$

This is maximum throughput rate of Slotted Aloha.

In equilibrium $\lambda = Ge^{-G}$

$G < 1$ many idle slots

$G > 1$ many collisions

What is dynamics of the system ?

i - number of slots until retransmission of backlogged packet.

$$p(i) = (1 - q_r)^{i-1} q_r \quad \textit{geometric distribution}$$

We use no - buffering assumption (6a)

n - number of backlogged nodes at the beginning of a slot

each of them transmits with probability q_r

each of $m - n$ unbacklogged nodes transmits with probability (if at least 1 packet arrives)

$$q_a = 1 - e^{-\lambda/m}$$

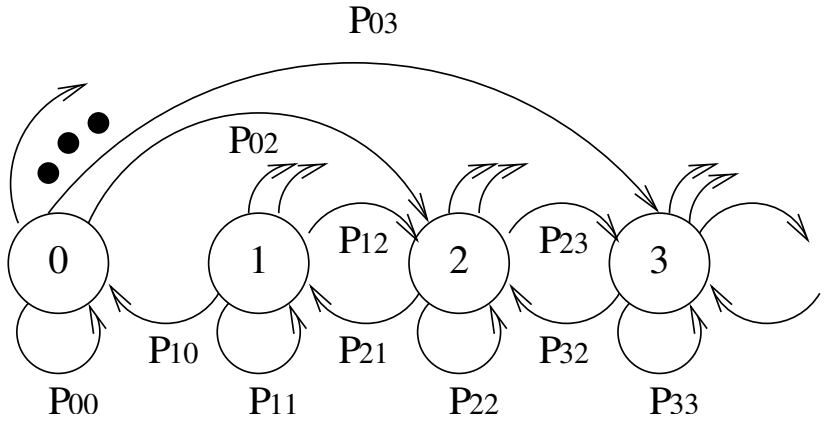
Denote: $Q_a(i, n)$ - probability that i unbacklogged nodes transmit

$$Q_a(i, n) = \begin{pmatrix} m - n \\ i \end{pmatrix} (1 - q_a)^{m-n-i} q_a^i$$

$Q_r(i, n)$ - probability that i backlogged nodes transmit

$$Q_r(i, n) = \begin{pmatrix} n \\ i \end{pmatrix} q_r^i (1 - q_r)^{n-i}$$

Markov chain (state is n)



$$P_{n,n+i} = \begin{cases} Q_a(i,n), & 2 \le i \le m-n \\ Q_a(1,n)[1-Q_r(0,n)], & i=1 \\ Q_a(1,n)Q_r(0,n) + Q_a(0,n)[1-Q_r(1,n)], & i=0 \\ Q_a(0,n)Q_r(1,n), & i=-1 \end{cases}$$

Steady-state probabilities p_n can be found recurrently from

$$\begin{cases} p_n = \sum_{i=0}^{n+1} p_i P_{in} \\ \\ \sum_{n=0}^{m} p_n = 1 \end{cases}$$

The expected number $\overline{n} = \sum_{n=0}^{m} n p_n$

From Little's Th. delay=$\overline{n}/\lambda$

The system is unstable, and if $q_r n \gg 1$ then it is locked for a long time.

Define drift D_n in state n as expected change in n over one slot time:

$$D_n = (m-n)q_a - P_{succ}$$

where

P_{succ} - probability of successful transmission

$$P_{succ} = Q_a(1,n)Q_r(0,n) + Q_a(0,n)Q_r(1,n) \quad (*)$$

$G(n)$ [attempt rate] expected number of attempted transmissions when in state n

$$G_n = (m-n)q_a + nq_r$$

Example: For q_a and q_r small:

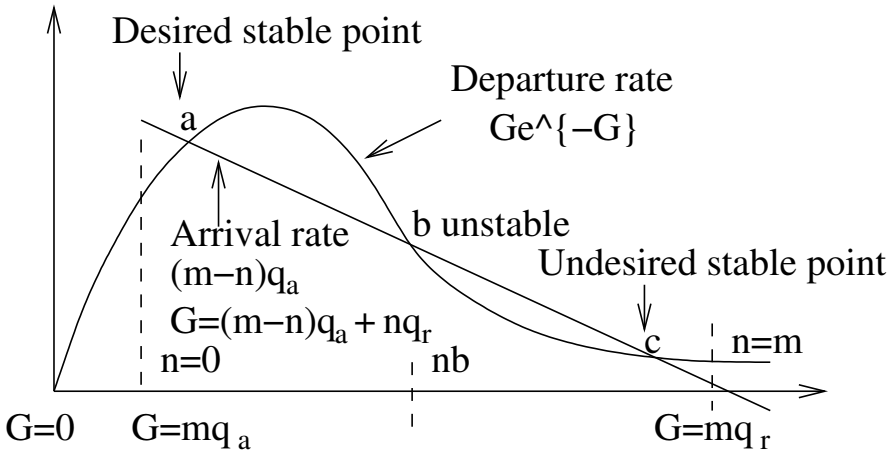
$$P_{succ} \approx G(n)e^{-G(n)}$$

[this follows from (*) and $(1-x)^y \approx e^{-xy}$]

Probability of idle slot P_{idle}

$$P_{idle} = Q_a(0,n)Q_r(0,n) \approx e^{-G(n)}$$

Thus, number of packets in a slot is well approximated by Poisson distribution.



Instability of Slotted Aloha for $q_r > q_a$

Conclusions:

- departure rate P_{succ} at most $1/e$ for large m.
- departure rate is small for long time in point c
- influence of q_r: $G(n) = mq_a + n(q_r - q_a)$

  (1) $q_r \uparrow$: retransmission delay $\downarrow$; $G(n) \uparrow$; point c reached for smaller n

  (2) $q_r \downarrow$: retransmission delay $\uparrow$; point c can disappear

For infinite - node assumption: (6b)

Attempt rate is $G(n) = \lambda + nq_r$

Arrival rate in the plot is $\lambda = const$

- point c disappears
- if $n > n_b$ (for point b), then $n \to \infty$

4.3 Stabilized Slotted Aloha

Throughput is $P_{succ} \approx G(n)e^{-G(n)}$

$max_{G(n)}\ P_{succ}$ is when $G(n) = 1$

Let us change q_r dynamically to maintain $G(n) = 1$.

Problem: n is unknown to nodes, and can only be estimated from feedback.

Main idea:

- increase q_r when idle slot
- decrease q_r when collision

We consider infinite node assumption (6b)

Definition: A multiaccess system is stable for given λ if the expected delay per packet is finite.

Theorem: Ordinary slotted Aloha is unstable $\forall \lambda > 0$

Definition: The maximum stable throughput is defined as supp λ for which the system is stable.

Theorem: The maximum stable throughput of ordinary slotted Aloha is 0.

4.4 Pseudo - Bayesian algorithm

Mikhailov(1979), Rivest(1985).

- new arrivals are backlogged on arrival.

  Attempt rate: $G(n) = nq_r$

  $$P_{succ} = nq_r(1 - q_r)^{n-1}$$

  $\widehat{n}$ - estimation of backlog n at the beginning of each slot.

- Each backlogged packet is transmitted independently with probability $q_r(\widehat{n}) = min\{1, \frac{1}{\widehat{n}}\}$
  So, attempt rate $G = nq_r \approx 1$

Updating rule:

$$\widehat{n_{k+1}} = \begin{cases} max\{\lambda, \widehat{n_k} + \lambda - 1\}, & for\ idle\ or\ success \\ \widehat{n_k} + \lambda + (e-2)^{-1} & for\ collisions \end{cases} \qquad (*)$$

Explanation: From Poisson approximation, if $\widehat{n} \approx n,\ G \approx 1$

$P_{succ} = Ge^{-G} \approx e^{-1} = \frac{1}{e}$

$P_{idle} = e^{-G} \approx e^{-1} = \frac{1}{e}$

$P_{collision} = 1 - P_{succ} - P_{idle} \approx 1 - 2e^{-1} = \frac{(e-2)}{e}$

So (*) maintains balance between n and $\widehat{n}$

λ is unknown and should be estimated.

Theorem: (Mikhailov) If fixed $\widehat{\lambda} = 1/e$ is used in the algorithm, it is stable for all $\lambda < 1/e$.

4.5 Approximate delay analysis

Assume:

- λ is known
- $P_{succ} = \begin{cases} 1 & for \quad n = 1 \\ 1/e & for \quad n > 1 \end{cases}$

Denote:

W_i delay from arrival of i^{th} packet till beginning of i^{th} successful transmission.

n_i number of backlogged packets before $i's$ arrival

R_i residual time to the beginning of the next slot

t_j interval from the end of $(j-1)^{th}$ success to the end of j^{th} success

y_i remaining interval until the beginning of i^{th} transmission.

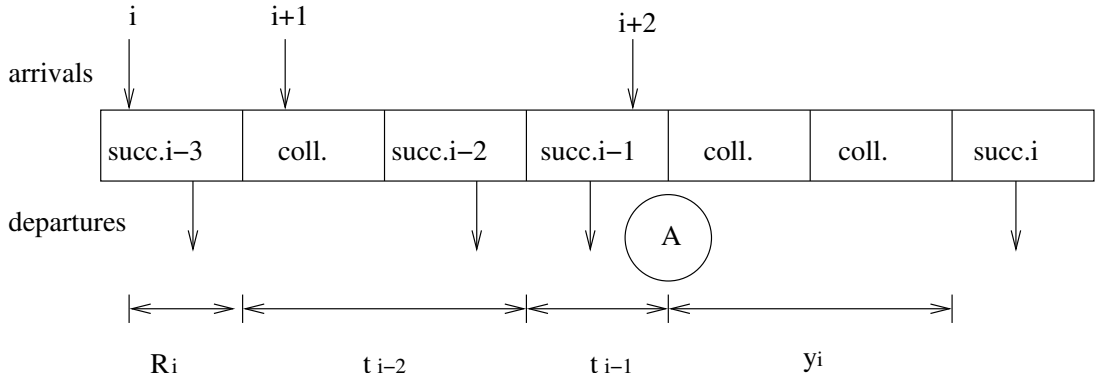


Fig. Example for $n_i = 2$

$$W_i = R_i + \sum_{j=1}^{n_i} t_{i-j} + y_j$$

For t_{i-j}, where $1 \leq j \leq n$, backlog $n \geq 2$ and $P_{succ} = 1/e$

$$E(t_j) = e$$

From Little's Theorem: $n_i = W_i \lambda$

$$W = 1/2 + W\lambda e + E(y) \qquad (w)$$

To compute $E(y)$ consider point A, where both $(i-1)$st departure and i^{th} arrival has occurred.

- if backlog $n = 1$ then $E(y_i) = 0$
- if $n > 1$ then $E(y_i) = e - 1$

Denote p_n steady-state probability that backlog is n at a slot boundary

At state 1 (which has probability p_1) transmission is successful, $y = 0$

Consider $N \to \infty$ time slots

λN - number of successful transmissions

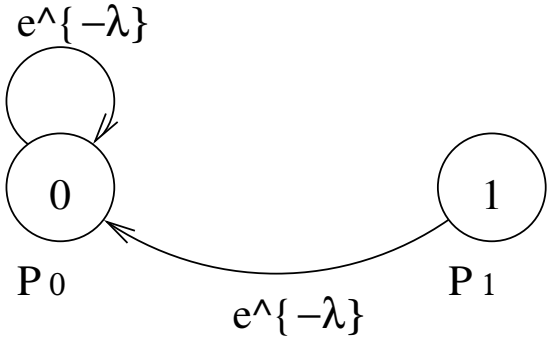
$p_1 N$ - number of successful transmissions from state 1

$\frac{\lambda N - p_1 N}{\lambda N} = \frac{\lambda - p_1}{\lambda}$ fraction of packets transmitted from states $n > 1$

$$E(y) = \frac{\lambda - p_1}{\lambda}(e - 1) \qquad (y)$$

To determine p_1:

$e^{-\lambda}$ is probability of no arrivals during one time slot.



$$p_0 = p_0 e^{-\lambda} + p_1 e^{-\lambda} = (p_0 + p_1)e^{-\lambda}$$
$$\lambda = p_1 P_{succ}(1) + (1 - p_0 - p_1)P_{succ}(> 1)$$
$$\lambda = p_1 + (1 - p_0 - p_1)/e$$

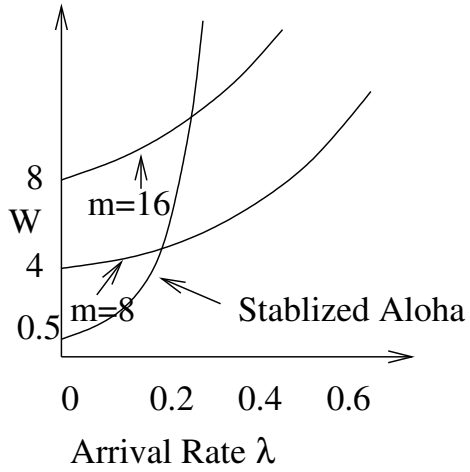
and we get

$$p_1 = \frac{(1 - \lambda e)(e^{\lambda} - 1)}{1 - (e - 1)(e^{\lambda} - 1)} \qquad (p)$$

From (w), (y), and (p) we get

$$w = \frac{e - 1/2}{1 - \lambda e} - \frac{(e^{\lambda} - 1)(e - 1)}{\lambda[1 - (e - 1)(e^{\lambda} - 1)]}$$

w is very close to simulation results.



Comparison of expected waiting time W in slots, from arrival unitl beginning of successful transmission, for stabilized Aloha and for TDM with m=8 and m=16. For small arrival rates, the delay of stabilized Aloha is little more than waiting for the next slot, whereas as the arrival rate approaches 1/e, the delay becomes unbounded.

4.6 Unslotted (pure) Aloha

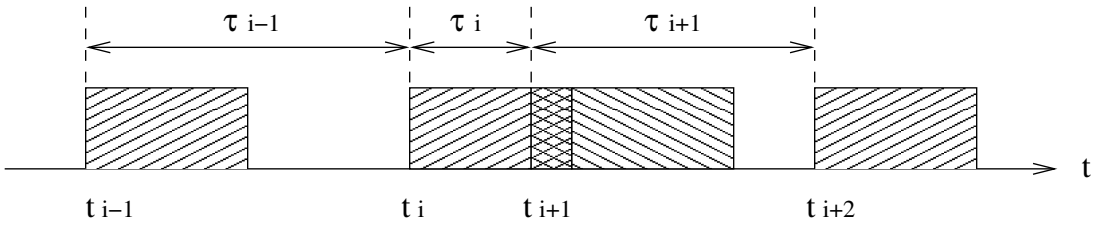
- arriving packet is transmitted immediately
- receiver rebroadcasts the received signal
- period τ until attempted retransmission is exponentially distributed $p(\tau) = xe^{-x\tau}$

x - retransmission attempt rate

λ - overall Poisson arrival rate

n - backlog at a given time

The initiation times of attempted transmissions is (time-varying) Poisson process of rate $G(n) = \lambda + nx$



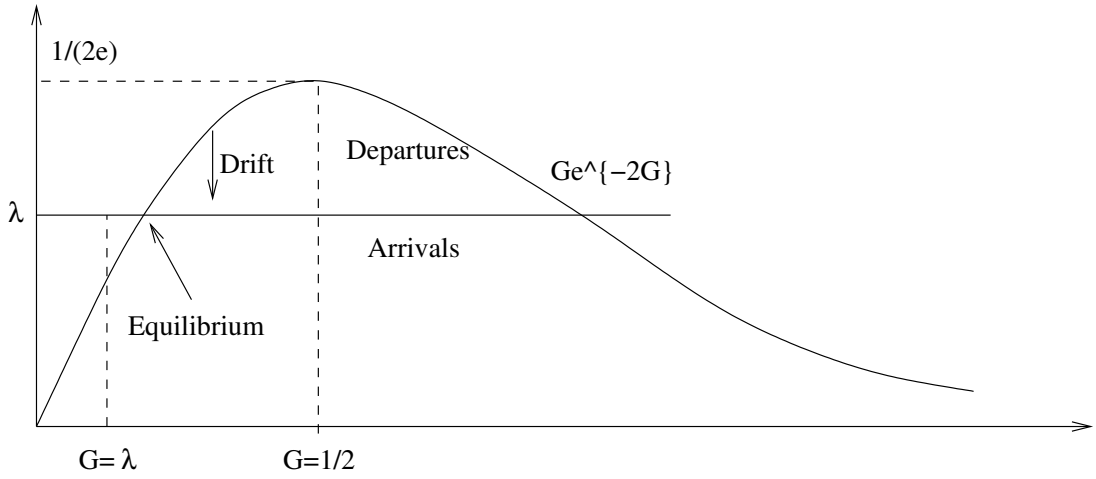
Assume unit-length packets.

The i^{th} attempt will be successful if $\tau_i > 1$, and $\tau_{i-1} > 1$.

Given backlog n for each interval, τ_i and τ_{i-1} are independent and

$$P_{succ} = [e^{-G(n)}]^2 = e^{-2G(n)}$$

4.7 Pure Aloha



Pure Aloha as a function of the attempted transmission rate G. Successful departures leave the system at a rate Ge^{−2G}, and arrivals occur at a rate λ , leading to a hypothesized equilibrium at the point shown.

- maximum throughput is $1/2e$ (compare with $1/e$ for slotted Aloha)
- pure Aloha is unstable, stabilization is not known
- can be used for variable lengths packets

4.8 Splitting algorithms

Idea: There are two packets A and B at 2 nodes, splitting with fair coin:

$Probability$	$Frame\ 1$	$Frame\ 2$
$1/4$	A	B
$1/4$	B	A
$1/4$	A, B	ϕ
$1/4$	ϕ	A, B

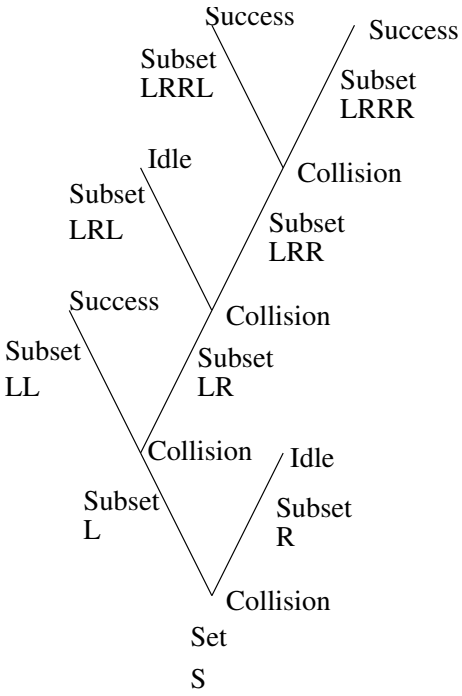
$n(i)$ - average number of slots to transmit i packets.

$n(2) = \frac{1}{4} * 2 + \frac{1}{4} * 2 + \frac{1}{4} * (n(2) + 1) + \frac{1}{4} * (n(2) + 1) = \frac{3}{2} + \frac{1}{2}n(2)$

$n(2) = 3$

Throughput $= 2/3$ (compare with $1/e$ for slotted Aloha)

4.9 Tree algorithms



Slot	Xmit Set	Waiting Sets	Feedback
1	S	−−	e
2	L	R	e
3	LL	LR,R	1
4	LR	R	e
5	LRL	LRR,R	0
6	LRR	R	e
7	LRRL	LRRR,R	1
8	LRRR	R	1
9	R	−−	0

Tree algorithm. After a collision, all new arrivals wait and all nodes involved in the collision divide into subsets. Each successive collision in a subset causes that subset to again split into smaller subsets while other nodes wait.

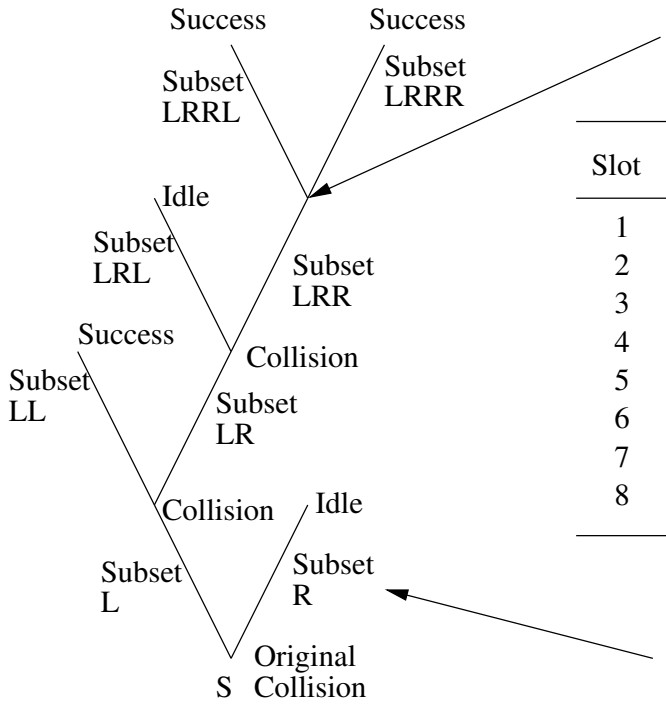
CRP - collision resolution period

What to do with new arrivals during CRP?

At the end of CRP each node estimates the number of new arrivals and determines the number j of subsets.

- Maximum throughput 0.43 packets/slot

4.9.1 Improvements to the tree algorithm



This collision is avoidable (first improvement)

Slot	Xmit Set	Waiting Sets	Feedback
1	S	−−	e
2	L	R	e
3	LL	LR,R	1
4	LR	R	e
5	LRL	LRR,R	0
6	LRRL	LRRR,R	1
7	LRRR	R	1
8	(R)	−−	(0)

This subset can be incorporated into subsequent collision resolution periods (second improvement)

Improvements in the tree algorithm. Subset LRR can be split without first being transmitted since the feedback implies that it contains two or more packets. Also, subset R is better combined with new arrivals since the number of packets in it is Poisson with an undesirably low rate.

• The first improvement gives maximum throughput 0.46 packets/slot (compare with 0.43)

• The second with the first improvements give rise to First-come first-serve (FCFS) algorithm, which is stable for $\lambda < 0.4871$

4.9.2 Variants of the tree algorithm

The tree algorithm is called a blocked stack algorithm.

Disadvantage: receivers can not be turned off.

Possible solution: (unblocked stack algorithms)

Join new arrivals at the head of the stack. Thus, only currently backlogged nodes need to monitor the channel feedback, all the rest can be turned off.

The maximum throughput is 0.40.

4.10 First-Come First - Serve (FCFS) splitting algorithm

• splitting into subsets by arrival time

$[T(k), T(k)+\alpha(k)]$ allocation interval, packets to be transmitted in slot k. $\sigma = L$ or R status

FCFS algorithm:

1 If feedback= e, then

$$T(k) = T(k-1)$$
$$\alpha(k) = \alpha(k-1)/2$$
$$\sigma(k) = L$$

2 if feedback= 1 and $\sigma(k-1) = L$, then

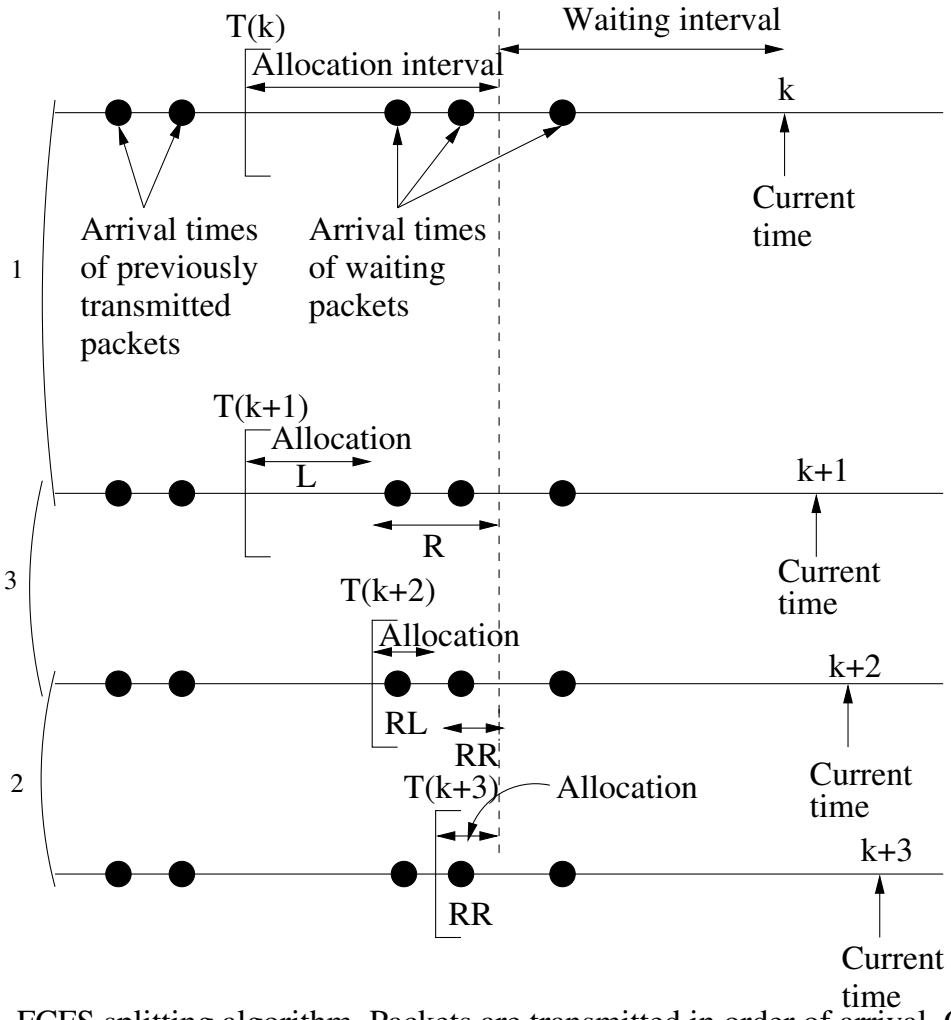
$$T(k) = T(k-1) + \alpha(k-1)$$
$$\alpha(k) = \alpha(k-1)$$
$$\sigma(k) = R$$

3 if feedback= 0 and $\sigma(k-1) = L$, then

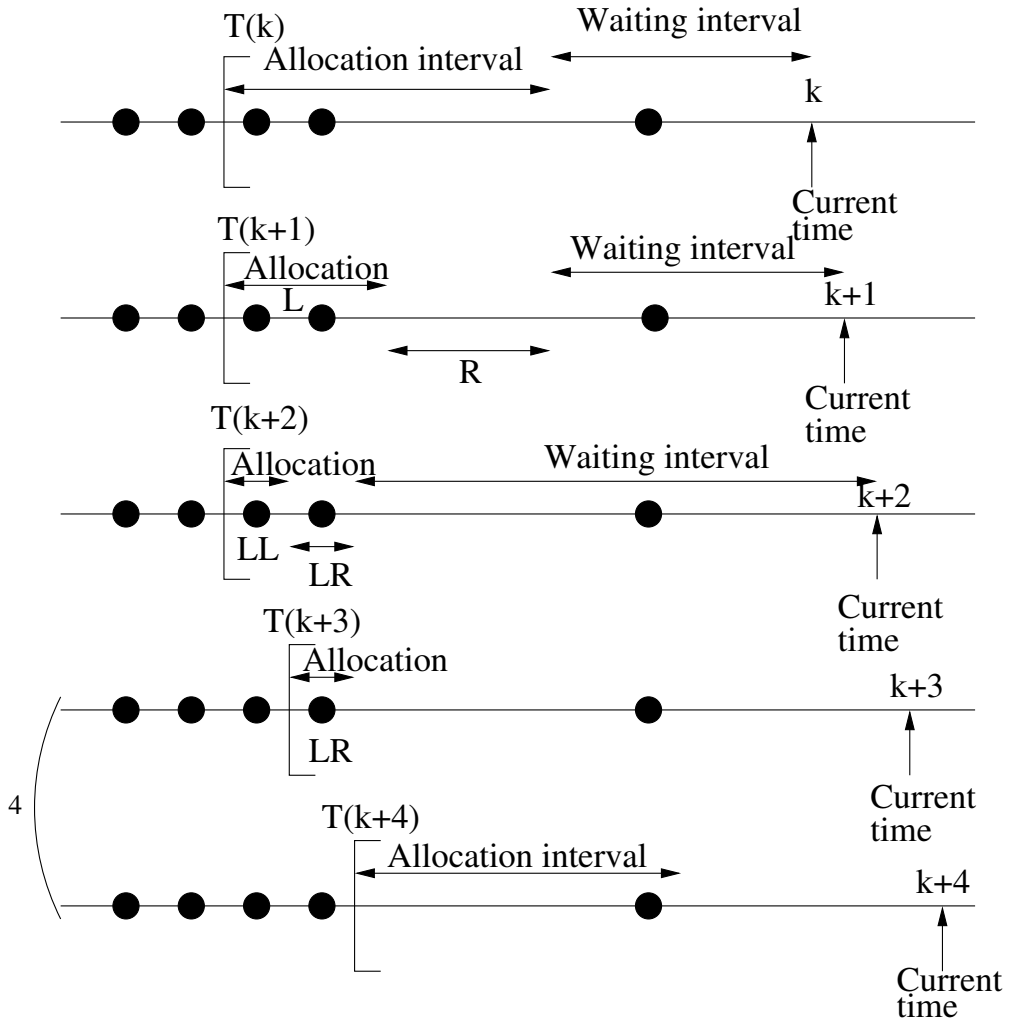
$$T(k) = T(k-1) + \alpha(k-1)$$
$$\alpha(k) = \alpha(k-1)/2$$
$$\sigma(k) = L$$

4 if feedback= 0 or 1 and $\sigma(k-1) = R$, then

$$T(k) = T(k-1) + \alpha(k-1)$$
$$\alpha(k) = min\{\alpha_0, k - T(k)\}$$
$$\sigma(k) = R$$

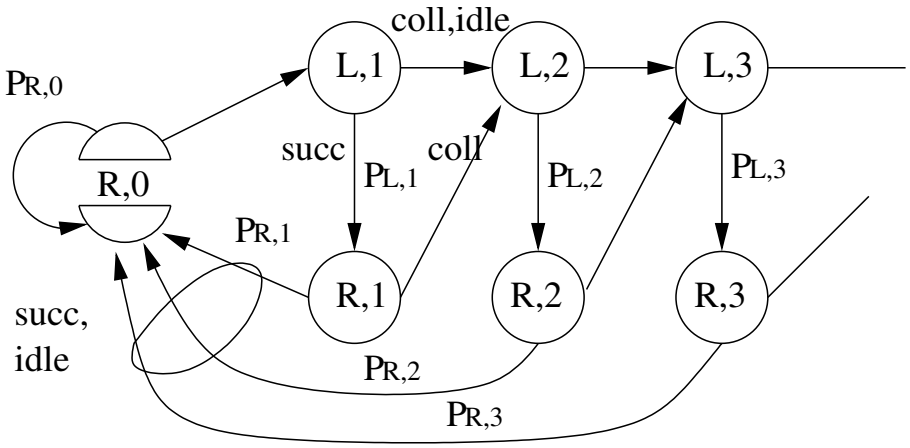


FCFS splitting algorithm. Packets are transmitted in order of arrival. On collisions, the allocation interval generating a collision is split into two subintervals, with the leftmost (earlier arrivals) transmitting first.

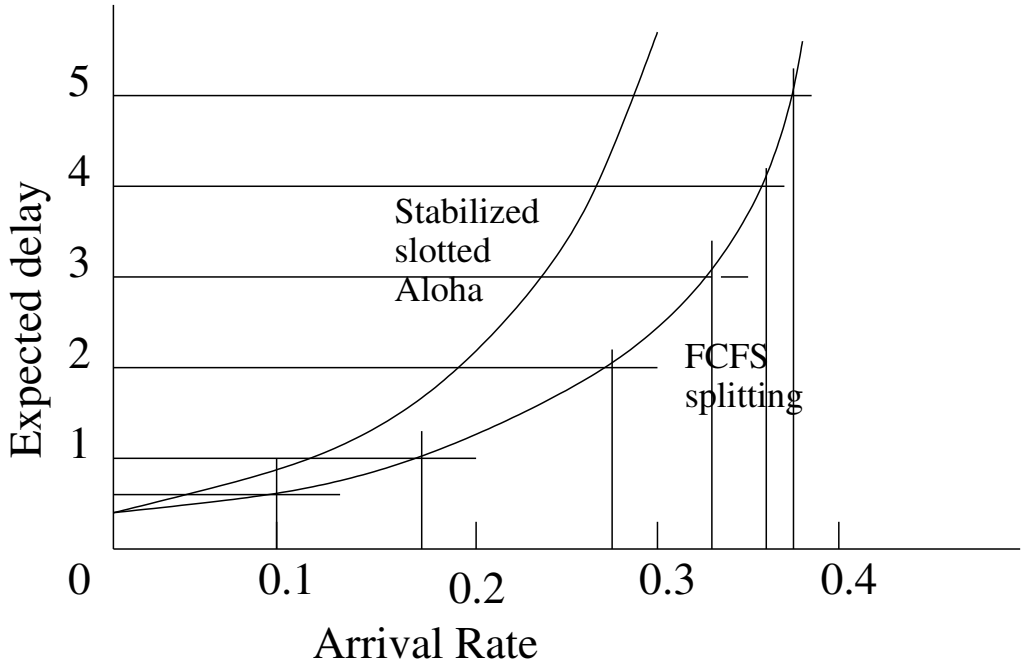


4

FCFS splitting algorithm. When a collision follows another collision, the interval on the right side of the second collision is returned to the waiting interval. The CRP is completed in slot k+3, and a new CRP is started with an allocation interval of fixed size.



Markov chain for FCFS splitting slgorithm. The top states are entered after splitting an interval and correspond to the transmission of the left side of that interval. The lower states are entered after success on the left side and correspond to transmission of the right side. Transitions from top to bottom and from bottom back to R,0 correspond to success.



Comparison of expected delay for stabilized slotted Aloha and the FCFS splitting algorithm as a function of arrival rate. One becomes unbounded as the arrival rate approaches 1/e, and the other as the arrival rate approaches 0.4871.

Improvements in the FCFS

- Splitting into equal-sized subintervals is nonoptimal.
  For optimally sized subintervals maximum stable throughput is 0.4878
- Upper bound for maximum stable throughput (under assumptions 1 to 6b) is 0.587.
- For finite set of m nodes, TDM can achieve throughput up to 1. However, expected delay (for a given λ) increases linearly with m.

4.11 Carrier sense multiple access (CSMA)

- β - propagation and detection delay (in packet transmission units)
- infinite number of nodes
- Poisson arrivals of overall intensity λ

4.11.1 CSMA Unslotted Aloha

- when a packet arrives, its transmission starts immediately if the channel is idle.
- if the channel is busy, or collision, the packet is backlogged
- retransmission with delay τ exponentially distributed $xe^{-x\tau}$

Analysis

Consider an idle period started with backlog n. The time until the first transmission starts is exp. distributed with rate

$$G(n) = \lambda + nx$$

After transmission-initiation the backlog is

n if a new arrival started transmission,

$n - 1$ if a backlogged packet started

Time until next transmission is exponentially distributed with rate $G(n)$ or $G(n-1)$.

A collision occur if transmission starts within time β. Transmission will not start with probability $e^{-\beta G(n)}$ or $e^{-\beta G(n-1)}$

Assume βx is small $\rightarrow G(n) \approx G(n-1)$

Probability of a successful transmission following an idle period

$$e^{-\beta G(n)}$$

The expected time from the beginning of one idle period until the next is

$$1/G(n) + (1+\beta),$$

where

$1/G(n)$ - time until the first transmission starts

$(1+\beta)$ - time for transmission + sensing

We have departure rate (throughput)

$$S(n) = \frac{e^{-\beta G(n)}}{1/G(n) + (1+\beta)}$$

for small β, the maximum $S(n)$ is

$$S_{max} \approx \frac{1}{1+2\sqrt{\beta}},$$

when $G(n) \approx \beta^{-1/2}$

4.11.2 Approximate delay analysis

$$W_i = R_i + \sum_{j=1}^{n_i} t_{i-j} + y_i$$

$$R = E(R_i) = \lambda/2$$

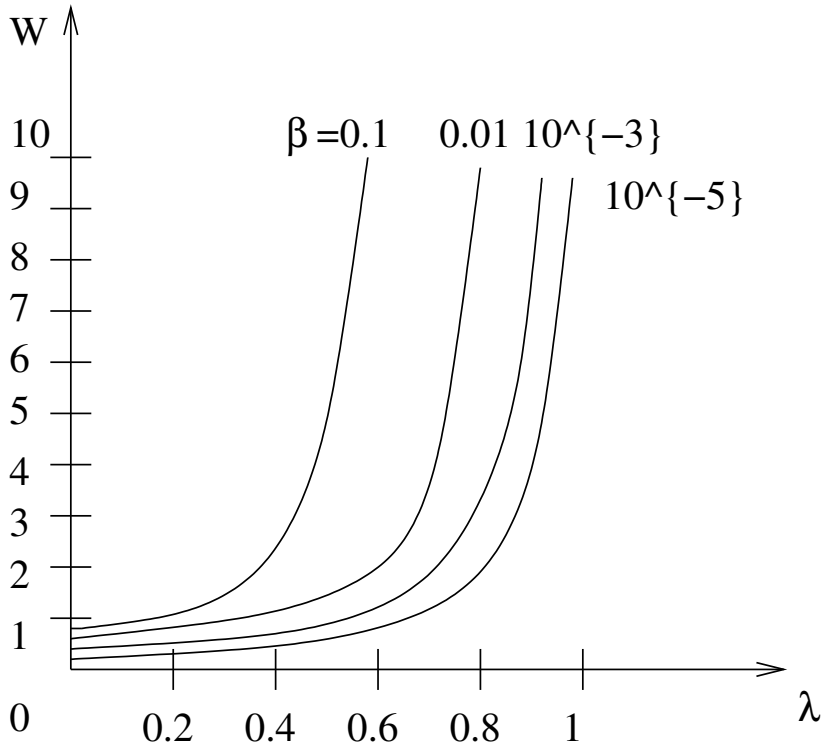
$$t = E(t_i) \approx \frac{1}{S_{max}} = 1 + 2\sqrt{\beta}$$

Little's Th $\quad n = W\lambda,\ n = E(n_i),\ W = E(W_i)$

$$y = E(y_i) = t - (1+\beta) = 2\sqrt{\beta} - \beta$$

$$W = \frac{\lambda}{2} + \lambda W(1 + 2\sqrt{\beta}) + 2\sqrt{\beta} - \beta$$

$$W = \frac{\frac{\lambda}{2} + 2\sqrt{\beta} - \beta}{1 - (1 + 2\sqrt{\beta})\lambda}$$



4.11.3 Multiaccess reservation

Idea: Send short packets in a contention mode or TDM mode and reserve long slot for data transmission

- $\nu \ll 1$ length of reservation packet (data packet has length 1)
- S_r throughput of reservation (i.e., $1/e$ for slotted Aloha)
  ν/S_r - time required for reservation

Throughput S with reservation

$$S = \frac{1}{1 + \nu/S_r}$$

If reservation packet carries also data

ν - time for reservation

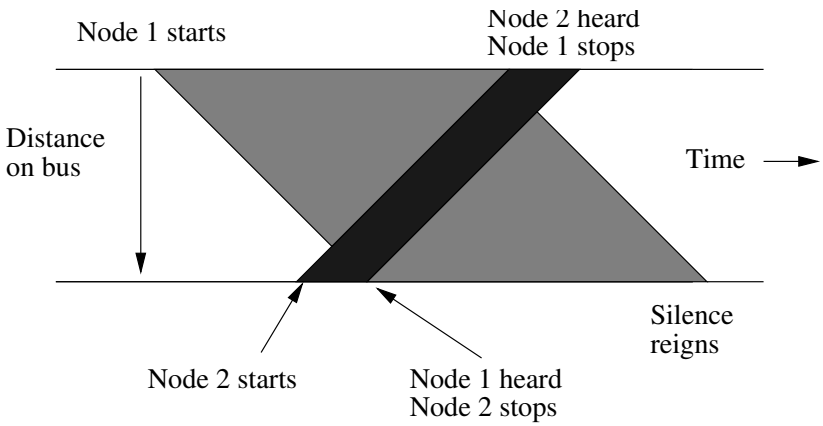
$(1 - \nu)$ - time for data (if reserv. successful)

Throughput is

$$S = \frac{1}{(1 - \nu) + \nu/S_r}$$

This is another way to compute S for CSMA Aloha.

4.11.4 CSMA/Collision Detection (CSMA/CD)



Collision detection. Node 2 starts to transmit almost β units after node 1; node 2 stops almost immediately, but node 1 continues for almost another β units before stopping

Throughput:

$$s > \frac{e^{-\beta G}}{\beta + 1/G + 2\beta(1 - e^{-\beta G}) + e^{-\beta G}}$$

$$S_{max} > \frac{1}{1 + 6.2\beta}$$

when $\beta G = (\sqrt{13} - 1)/6 = 0.43$

4.12 Framing

Problem of framing at DLC (Data Link Control) is that of deciding where frames start and stop.

Types of framing:

1. Character-based framing
2. Bit-oriented framing with flags
3. Length counts

4.12.1 Character-based framing

Character codes, e.g., ASCII:

Symbols:

S 1010011

T 1010100

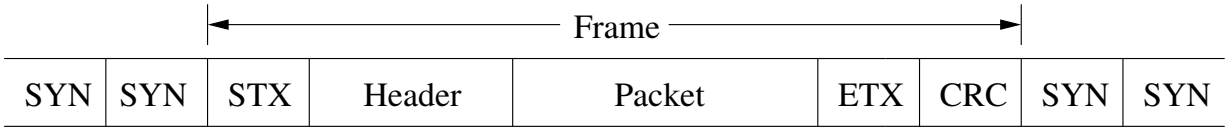
Control characters:

STX 0000011

ETX 0001000

SYN 0010110

CRC-cyclic redundancy check field



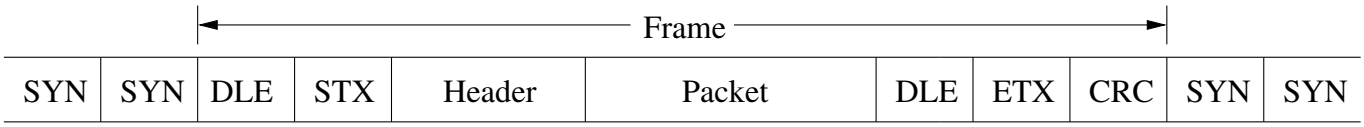
SYN = Synchronous idle
STX = Start of text
ETX = End of text

Simplified frame structure with character–based framing

Problem if the packet contains ETX

Solution: transparent mode

DLE - data link escape character



DLE = Data link escape

Character–based framing in a transparent mode as used in the ARPANET

Problem: if DLE appears in packet

Solution: (character stuffing) In the packet:

transmitter: DLE → DLE DLE

receiver: DLE DLE → DLE

If error in DLE ETX the packet is lost and should be recoverd at higher levels of OSI.

Disadvantage of character-based frames:

- long overhead
- length of frame should be integer number of characters.

4.12.2 Bit oriented framing: Flags

Like DLE ETX indicate end of frame, let us use some flag (e.g., 01111110) at the end of the frame.

Problem: packet can contain flag.

Solution: Bit-stuffing, flag=01^60

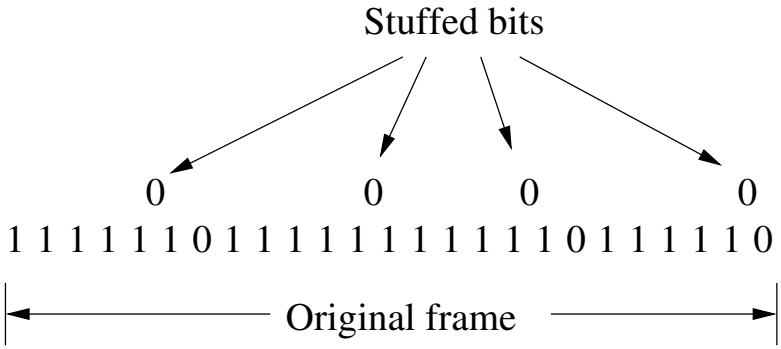
Transmitter: inserts (stuffs) a 0 after 1^5 in the frame;

Receiver: deletes 0 after 1^5

Additionally:

01^60 normal frame termination

01^61 abnormal termination



Bit stuffing. A 0 is stuffed after each consecutive five 1's in the original frame. A flag, 01111110, without stuffing, is sent at the end of the frame.

4.12.3 Average length of overhead for bit-stuffing

Frame before stuffing consists of independent random bits, with equal probability =0 or 1.

Assume flag 01^j0 (standard $j = 6$)

Insertion after i^{th} bit of original frame, $i \geq j$ if string from $i - j + 1$ to i is 01^{j-1};

probability of this: 2^{-j}

Insertion will also occer if a string from $i - 2j + 2$ to i is 01^{2j-2};

probability of this is $2^{-2j+1} \ll 2^{-j}$ and we omit this case.

Expected number of insertions in string of length K is $\approx K2^{-j}$

Expected length of overhead (with flag)

$$E(O\nu) \approx E(k)2^{-j} + j + 1$$

$min_j E(O\nu)$: find smallest j for which

$$E(k)2^{-j} + j + 1 < E(k)2^{-j-1} + j + 2$$

$$E(k)2^{-j-1} < 1$$

$$j_{opt} = \lfloor \log_2 E(k) \rfloor$$

and for this j_{opt}: $E(O\nu) \leq log_2 E(k) + 2$

Ex. $E(k) = 1000\ bit,\ j_{opt} = 9,\ E(O\nu) < 12\ bits,\ j = 6,\ E(O\nu) \approx 23\ bits$

not a big reason to change standard

4.12.4 Framing with length fields

L - packet length

K_{max} - maximum packet length

Overhead $= \lfloor \log_2 K_{max} \rfloor + 1$

If error in the length L then synchronization is lost.

4.13 How to select frame length

Questions: Given a message, in how many frames should it be divided? How long should the frames be?

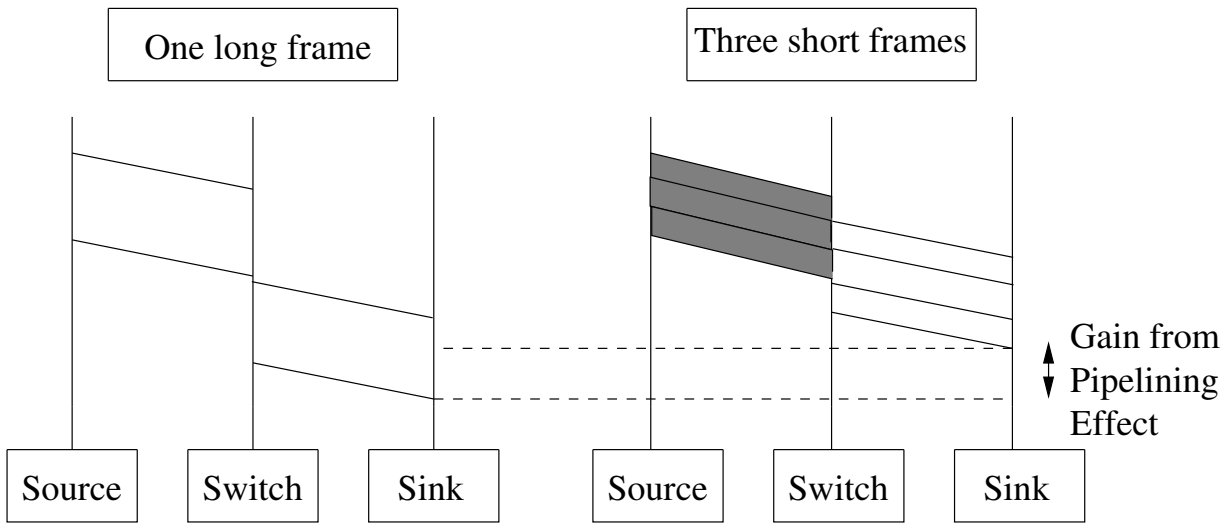
Aspects for choosing the optimum frame length:

- Overhead: More frames result in more headers, more overhead, lower throughput
- Pipelining effect: Shorter frames reduce the transmission delay
- Error rate: Long frames result in high packet error rates, many retransmissions, lower throughput

Pipelining effect:

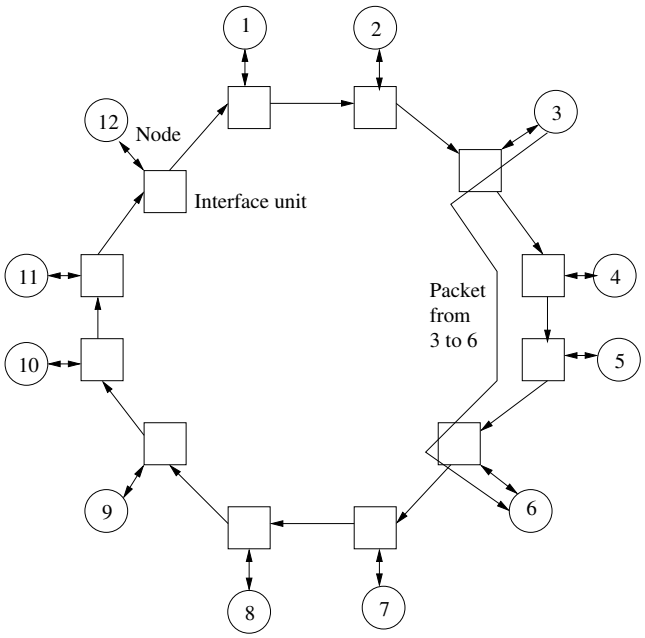
The pipelining effect occurs, when a switch has to receive a frame completely (e.g. for CRC calculation) before it can forward the frame.

Example: two p2p links of equal speed (data rate)



4.14 Token Passing

4.14.1 Token Ring



Ring network. Travel of data around the ring is unidirectional. Each node either relays the received bit stream to the next node with a one bit delay or transmits its own packet, discarding the incoming bit stream.

IT - Idle (or free) token 01^60

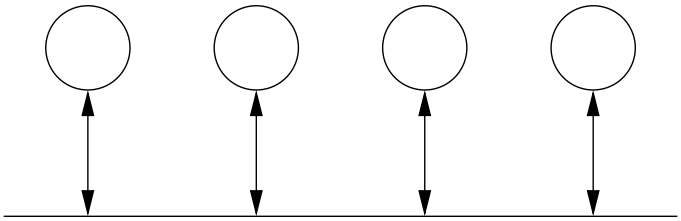
BT - Busy token 01^61

Node without token: retransmits received data, when the node receives IT, it becomes a node with token.

Node with token:

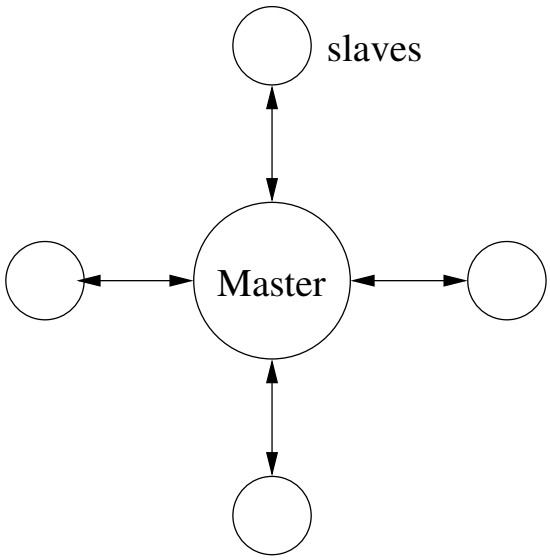
- if nothing to transmit, retransmits IT
- if it has packets to transmit, it sends BT followed by the packets.

4.14.2 Token bus



Nodes are logically connected in a ring and act as a token ring.

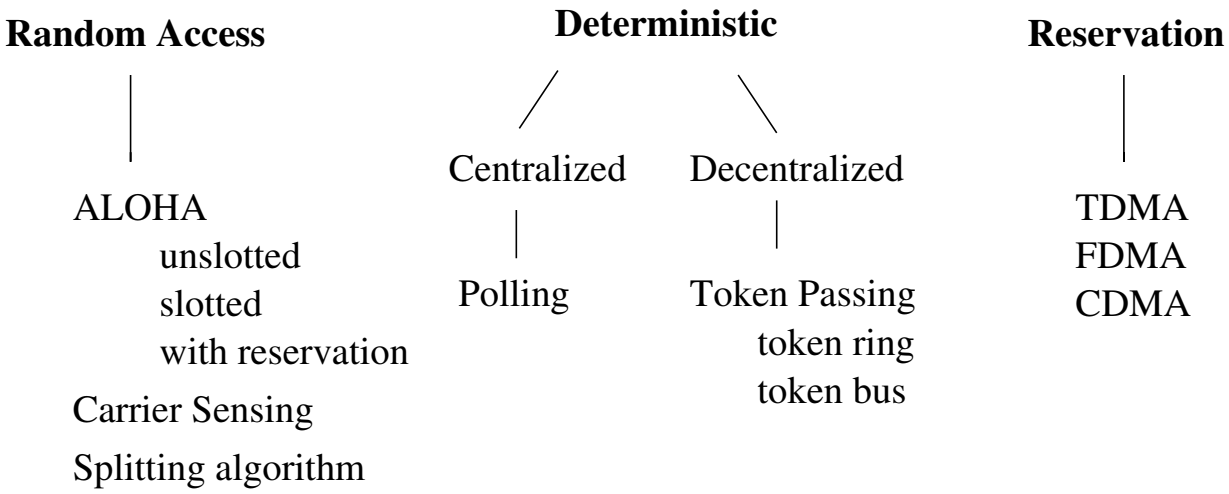
4.14.3 Polling



Master sends IT to slaves.

Token passing delay is large.

4.15 Classification of multiple access



4.15.1 Comparison

Polling:

- Static network configuration is not suitable for mobile networks.

Token Passing:

- In mobile networks: Maintenance of logical ring far too complicated.

Random access schemes:

- No special requirements on network topology. Hence, avoids complicated network maintenance.
- Fast/immediate access, when traffic load is low.
- Drawback: Network can become unstable under high traffic load.

Example GSM:

- Random access channel (RACH) for first contact to bases station.
- Traffic channel (TCH): Dedicated channel reserved for single terminal; carries voice traffic and dedicated signaling.

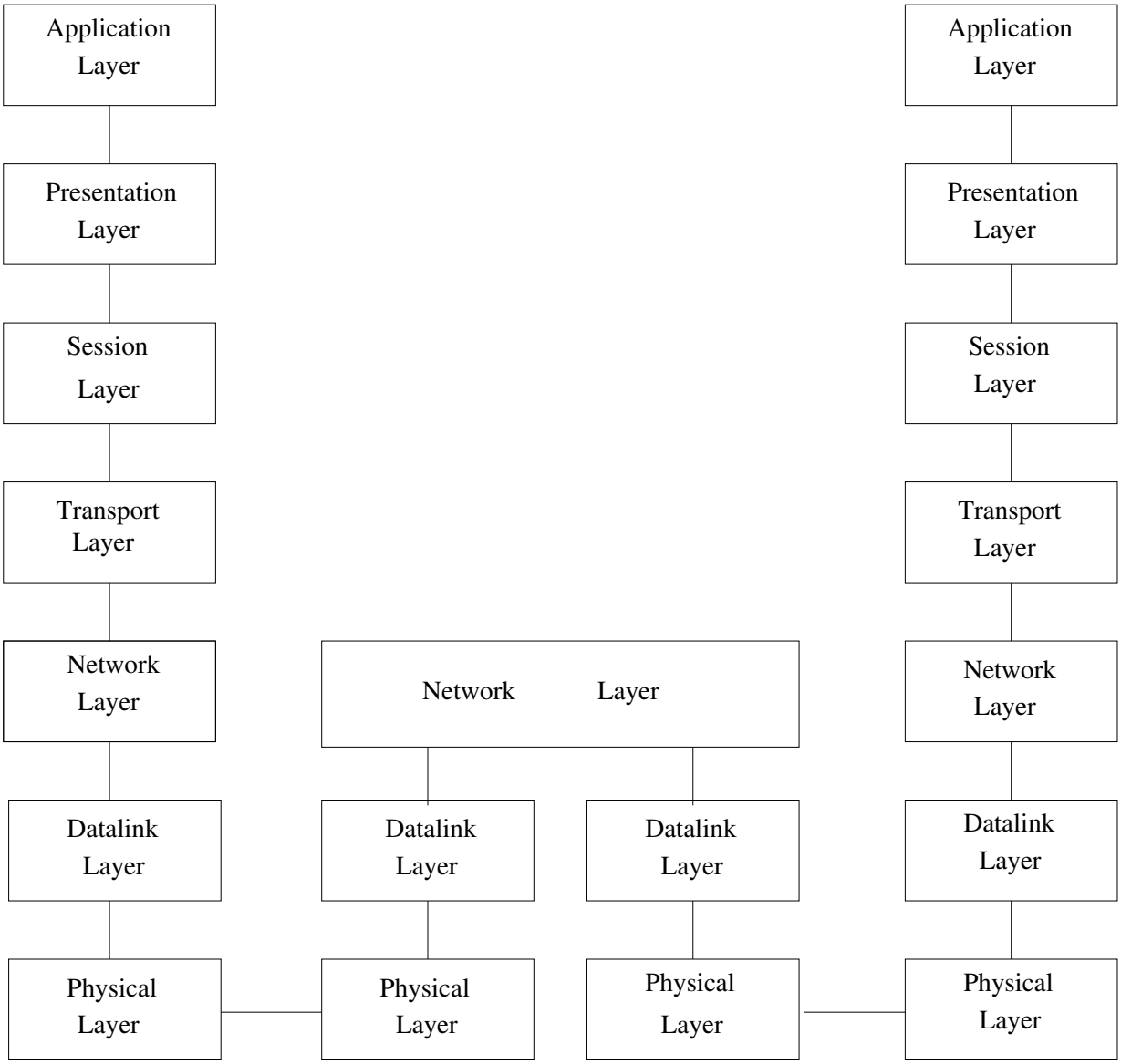
Example UMTS:

- Voice traffic like in GSM.
- Additional Common Packet Channel (CPCH): Shared channel with random access, for packet data (non-continuous traffic).

Chapter 5

Automatic Repeat Request(ARQ)

OSI model


Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Datalink Layer
Physical Layer
Network Layer
Datalink Layer
Datalink Layer
Physical Layer
Physical Layer
Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Datalink Layer
Physical Layer

5.1 Automatic Repeat Request(ARQ)

• ARQ detects frames with errors at the receiving DLC and requests retransmission.

How to detect errors?

Answer: use (n,k) binary block code.

In practice cyclic codes are used for error detection.

(CRC-Cyclic Redundancy Checks)

Probability of undetected error: $P_{CRC} \approx 2^{-(n-k)}$

We consider:

- Stop-and-Wait ARQ
- Go-back-n ARQ (modulo m)
- Selective repeat ARQ
- ARPANET ARQ
- Hybrid ARQ

Evaluation criteria for ARQ:

- Frame error rate (FER)

$$P(E) = \frac{\sharp erroneously\_accepted\_frames}{\sharp accepted\_frames}$$

- Throughput
- Delay
- Buffer size

Analysis of frame error rate:

- P_b: channel bit error probability
- $P_f = 1 - (1 - P_b)^n$: Frame error probability
- $P_{CRC} \approx \frac{1}{2^r}$: probability of CRC undetected error (r is $\sharp$CRC bits)
- $P_C = 1 - P_f = (1 - P_b)^n$: probabilit that frame is correct
- $P_r = P_f(1 - P_{CRC})$: probability that frame is in error and rejected
- $P_e = P_f P_{CRC}$: probability that frame is in error and accepted (detection error)
- $P(E) = P_e + P_r P_e + P_r^2 P_e + \ldots = P_e \sum_{i=0}^{\infty} P_r^i = \frac{P_e}{1-P_r}$

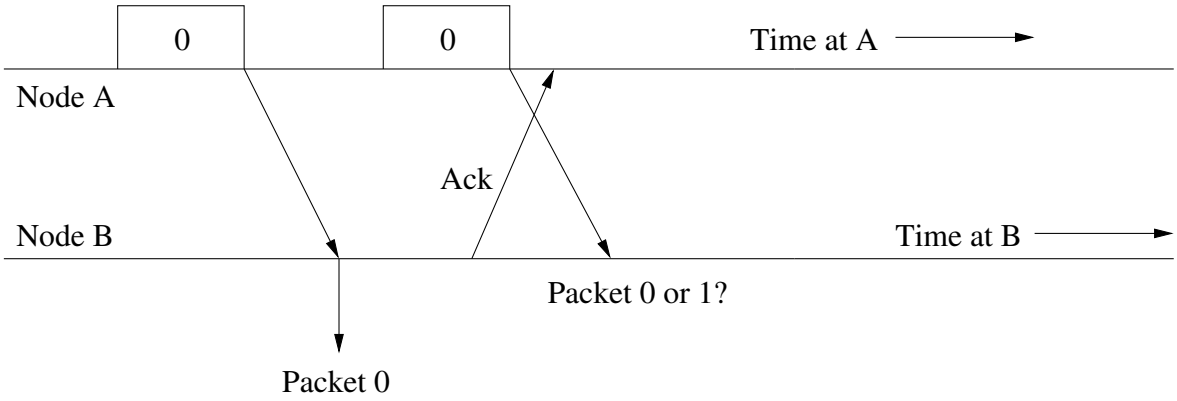
Analysis of maximum throughput:

- Θ: average number of transmission attempts per frame
- $\Theta = (1 - P_r) + 2P_r(1 - P_r) + 3P_r^2(1 - P_r) + \ldots$
- $\Theta = (1 - P_r) \sum_{i=0}^{\infty} iP_r^{i-1} = (1 - P_r)\frac{\partial}{\partial P_r} \sum_{i=0}^{\infty} P_r^i = (1 - P_r)\frac{\partial}{\partial P_r}\frac{1}{1-P_r} = \frac{1}{1-P_r}$
- maximum throughput: $\eta_{max} = \frac{1}{\Theta} R_{CRC} = (1 - P_r)\frac{k}{n}$
- throughput of specific ARQ schemes (stop and wait, go-back-n, selective repeat) will be smaller
- buffer size:
  - heavily depends on ARQ scheme
  - Interacts with throughput/delay (small buffer− >low throughput and large delay)

5.2 Stop-and-Wait ARQ

Ack-acknowledgement, Nak-negative acknowledgement.

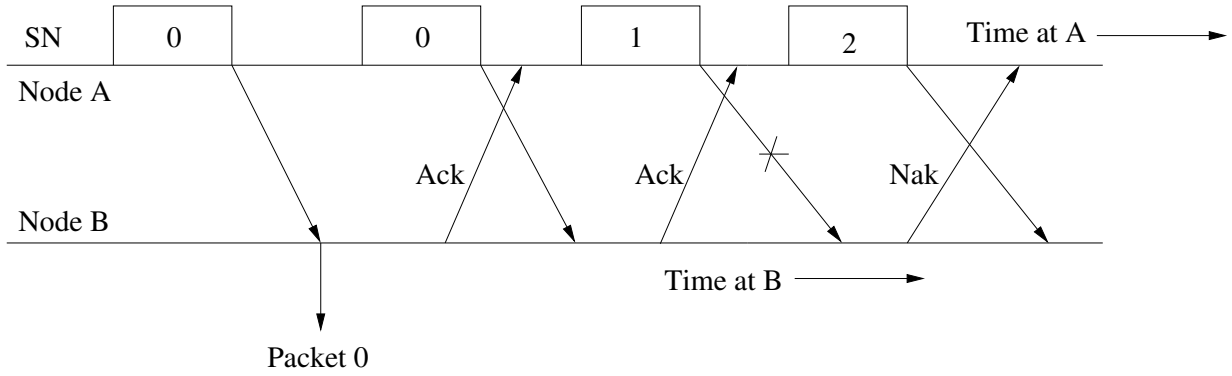
The problem with unnumbered packets:



The trouble with unnumbered packets. If the transmitter at A times−out and sends packet 0 twice, the receiver at B cannot tell whether the second frame is a retransmission of packet 0 or the first transmission of packet 1.

SN- sequence number

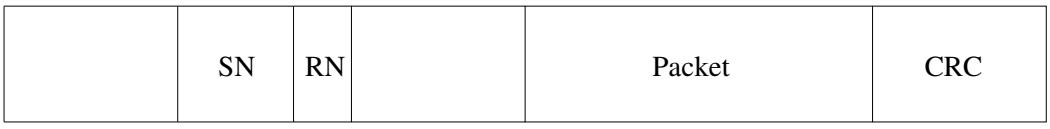
The problem with unnumbered acks:



The trouble with unnumbered acks. If the transmitter at A times−out and sends packet 0 twice, node B can use the sequence numbers to recognize that packet 0 is being repeated. It must send an ack for both copies, however, and (since acks can be lost) the transmitter cannot tell whether the second ack is for packet 0 or 1.

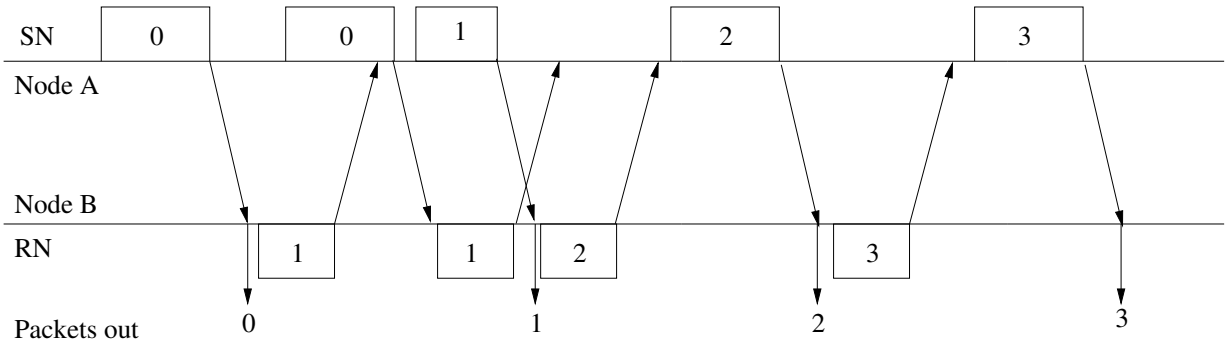
RN-request number

The header of a frame:



The header of a frame contains a field carrying the sequence number, SN, of the packet being transmitted. If piggybacking is being used, it also contains a field carrying the request number, RN, of the next packet awaited in the opposite direction.

Example of stop-and-wait (SW) algorithm:



Example of use of sequence and request numbers for stop−and−wait transmission from A to B. Note that packet 0 gets repeated, presumably because node A times−out too soon. Note also that node A delays repeating packet 1 on the second request for it. This has no effect on the correctness of the protocol, but avoids unnecessary retransmissions.

5.2.1 Stop-and-wait(SW) algorithm:

The algorithm at node A for A-to-B transmission:

1. Set the integer variable SN to 0.
2. Accept a packet from the next higher layer at A; if no packet is available, wait until it is; assign number SN to the new packet.
3. Transmit the SNth packet in a frame containing SN in the sequence number field.
4. If an error-free frame is received from B containing a request number RN greater than SN, increase SN to RN and go to step 2. If no such frame is received within some finite delay, go to step 3.

The algorithm at node B for A-to-B transmission:

1. Set the integer variable RN to 0 and then repeat steps 2 and 3 forever.
2. Whenever an error-free frame is received from A containing a sequence number SN equal to RN, release the received packet to the higher layer and increment RN.
3. At arbitrary times, but within bounded delay after receiving any error-free data frame from A, transmit a frame to A containing RN in the request number field.

SN and RN can be computed modulo 2

Evaluation:

- Buffer size:

  Transmitter - 1 frame

  Receiver - no buffer

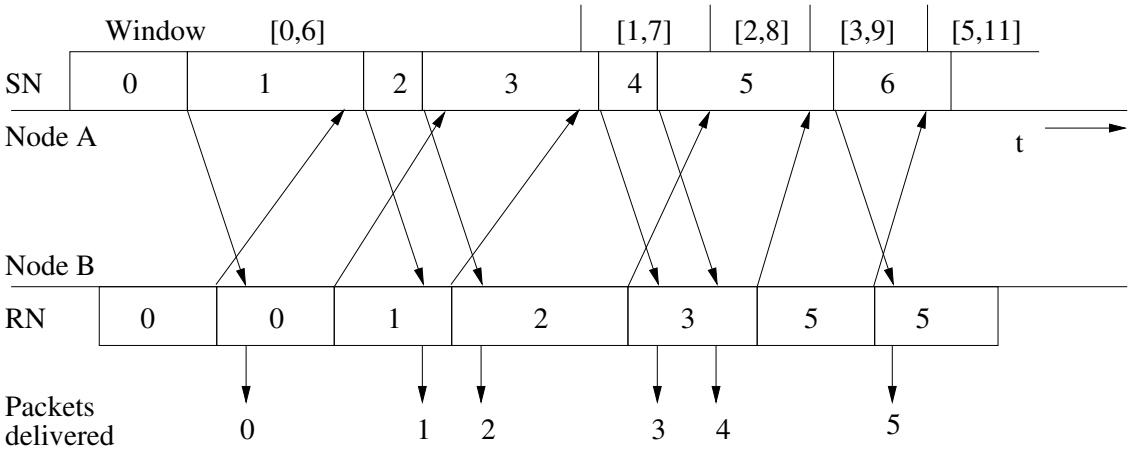
- Throughput: $\eta_{SW} = \frac{1}{\Theta} \cdot R_{CRC} \cdot \frac{\tau_{msg}}{\tau_{msg}+\tau_{idle}}$ is low due to big τ_{idle}.

5.3 Go Back n ARQ

Several successive packets can be sent without waiting for the next packet to be requested.

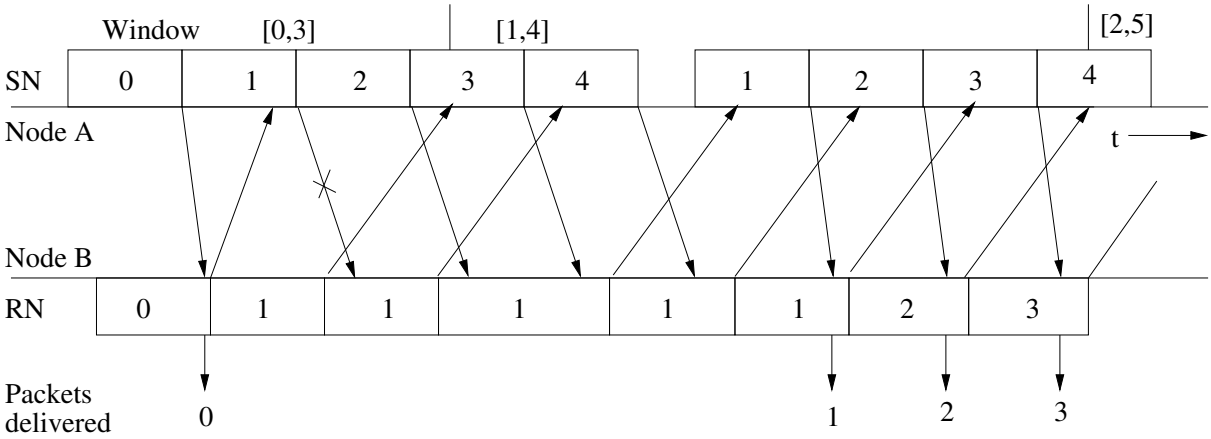
If i is the received request from B, there is a "window" of n packets $[i, i+n-1]$ that the transmitter is allowed to send. (Sliding window ARQ)

Example of go-back-7 ARQ



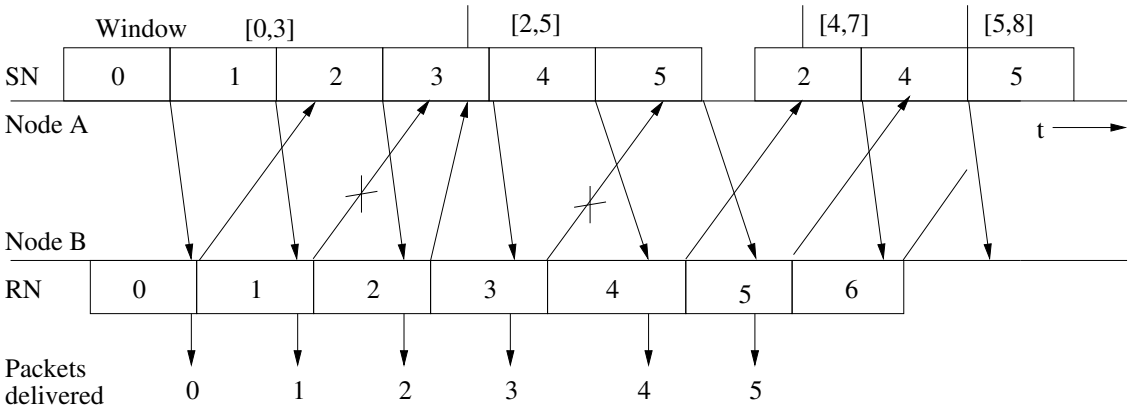
Example of go back 7 protocol for A–to–B traffic. Both nodes are sending data, but only the sequence numbers are shown for the frames transmitted from A and only the request numbers are shown for the frames from B. When packet 0 is completely received at B, it is delivered to the higher layer, as indicated at the lower left of the figure. At this point, node B wants to request packet 1, so it sends RN=1 in the next outgoing frame. When that outgoing frame is completely received at A, node A updates its window from [0,6] to [1,7]. Note that when packets 3 and 4 are both received at B during the same frame transmission at B, node B awaits packet 5 and uses RN=5 in the next frame from B to acknowledge both packets 3 and 4.

Effect for error frames on go-back-4 ARQ



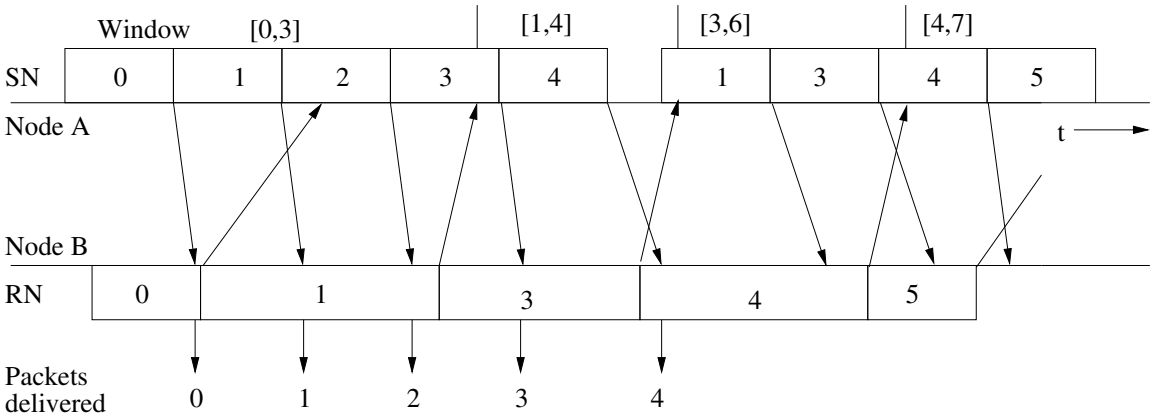
Effect of a transmission error on go back 4. Packet 1 is received in error at B, and node B continues to request packet 1 in each reverse frame until node A transmits its entire window, times−out, and goes back to packet 1.

Effect of errors in the reverse direction:



Effect of transmission errors in the reverse direction for go back 4. The first error frame, carrying RN=1, causes no problem since it is followed by an error free carrying RN=2 and this frame reaches A before packet number 3, (i.e., the last packet in the current window at A) has completed transmission and before a time−out occurs. The second error frame, carrying RN=3, causes retransmissions since the following reverse frame is delayed until after node A sends its entire window and times−out. This causes A to go back and retransmit packet 2.

Retransmission can occur in the absence of errors:



Effect of delayed feedback for go back 4. The frames in the B−to−A direction are much longer than those in the A−to−B direction, thus delaying the request numbers from getting back to A. The request for packet 1 arrives in time to allow packet 4 to be sent, but after sending packet 4, node A times−out and goes back to packet 1.

5.3.1 The go-back-n algorithm

$SN_{min} = i$ smallest-numbered packet not acknowledged

SN_{max} next packet to be accepted from the higher layer

At node A (A→B):

1. Set the integer variables SN_{min} and SN_{max} to 0.
2. Do steps 3,4 and 5 repeatedly in any order. There can be an arbitrary but bounded delay between the time when the conditions for a step are satisfied and when the step is executed.
3. If $SN_{max} < SN_{min} + n$, and if a packet is available from the higher layer, accept a new packet into the DLC, assign number SN_{max} to it, and increment SN_{max}.
4. If an error-free frame is received from B containing a request number RN greater than SN_{min}, increase SN_{min} *to* RN.
5. If $SN_{min} < SN_{max}$ and no frame is currently in transmission, choose some number SN, $SN_{min} < SN < SN_{max}$; transmit the SNth packet in a frame containing SN in the sequence number field. At most a bounded delay is allowed between successive transmissions of packet SN_{min} over interval when SN_{min} does not change.

The go back n algorithm at node B for A-to-B transmission:

1. Set the integer variable RN to 0 and repeat steps 2 and 3 forever.
2. Whenever an error-free frame is received from A containing a sequence number SN equal to RN, release the received packet ot the higher layer and increment RN.
3. At arbitrary times, but within bounded delay after receiving any error-free data frame from A, transmit a frame to A containing RN in the request number field.

5.3.2 The go-Back-n algorithm module m ($m > n$)

The go back n algorithm at node A for modulo m operation, $m > n$:

1. Set the modulo m variables SN_{min} and SN_{max} to 0.
2. Do steps 3,4 and 5 repeatedly in any order. There can be an arbitrary but bounded delay between the time when the conditions for a step are satisfied and when the step is executed.
3. If $(SN_{max} - SN_{min}) \bmod m \ < \ n$, and if a packet is available from the higher layer, accept a new packet into the DLC, assign number SN_{max} to it, and increment SN_{max} to $(SN_{max} + 1) \bmod$ m .
4. If an error-free frame is received from B containing a request number RN, and $(RN - SN_{min}) \bmod m \ \leq \ (SN_{max} - SN_{min}) \bmod m$, set SN_{min} to equal RN.
5. If $SN_{min} \neq SN_{max}$ and no frame is currently in transmission, choose some number SN such that $(SN - SN_{min}) \bmod m \ < \ (SN_{max} - SN_{min}) \bmod$ m; transmit packet SN in a frame containing SN in the sequence number field.

The go back n algorithm at node B for modulo m operation, $m > n$:

1. Set the modulo m variable RN to 0.
2. Whenever an error-free frame is received from A containing a sequence number SN equal to RN, release the received packet ot the higher layer and increment RN to (RN+1) mod m.
3. At arbitrary times, but within bounded delay after receiving any error-free data frame from A, transmit a frame to A containing RN in the request number field.

5.3.3 Go-Back-n ARQ analysis

How to choose n?

- large enough to avoid idle periods

$$n \geq \frac{round\ trip\ delay}{frame\ transmission\ time}$$

- as small as possible to decrease overhead

Throughput:

$$\eta_{GBn} = \frac{k}{n_{CRC}} \cdot \frac{1}{1 + (\Theta - 1)n} = R_{CRC} \frac{1 - P_r}{1 + P_r(n - 1)}$$

Buffer size:

transmitter - n frames

receiver - 0 frames

5.4 Selective Repeat ARQ

Go-back-n:

$$\eta_{GBn} = R_{CRC}\frac{1 - P_r}{1 + P_r(n-1)}$$

P_r - probability of retransmission

If $P_r n \ll 1$

then $\eta_{GBn} \approx R_{CRC}(1 - P_r) = \eta_{max}$

otherwise Go-back-n algorithm can be improved using selective-repeat ARQ.

5.4.1 Basic idea of Selective-Repeat ARQ

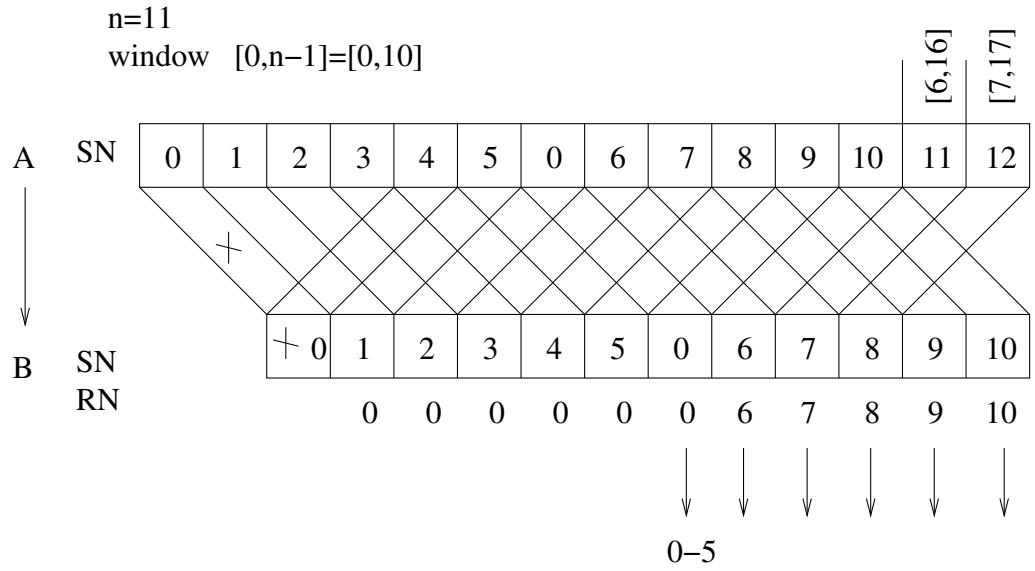
- accept out-of-order packets
- request retransmission of only wrong packets
- still window [RN, RN+n-1] for transmitter

Selective repeat algorithm $A \rightarrow B$

At node A: same as go-back-n algorithm

At node B: RN (like in go-back-n) is lowest number of packet not yet correctly received. Node B accepts all packets and deliver them ordered

Example:



Round trip delay:

$$\widehat{n} = \frac{\tau_{msg} + \tau_{AB} + \tau_{ACK} + \tau_{BA}}{\tau_{msg}} = 1 + 2 + 1 + 2 = 6$$

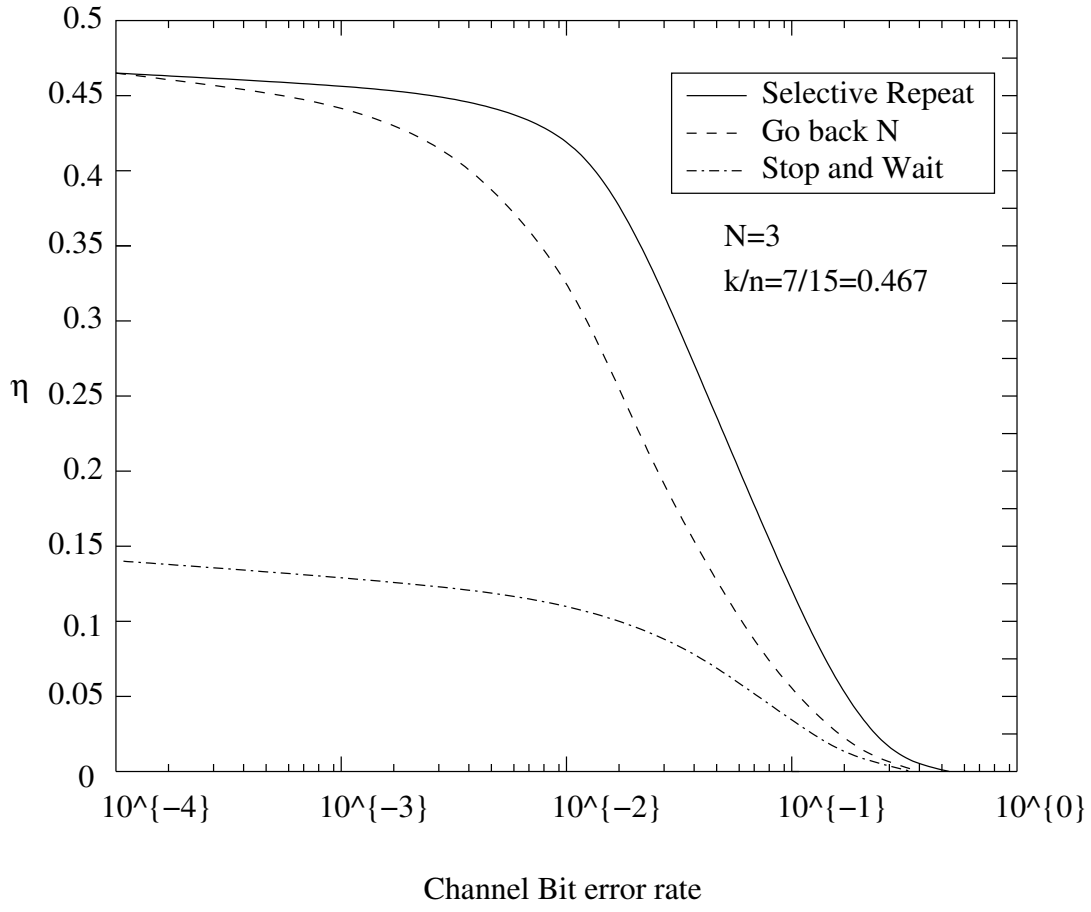
$$n \geq 2\widehat{n} - 1 = 11;\ m = n + 1 = 12 \quad \text{(module)}$$

For large n throughput

$$\eta_{SR} = R(1 - P_r) = \eta_{max}$$

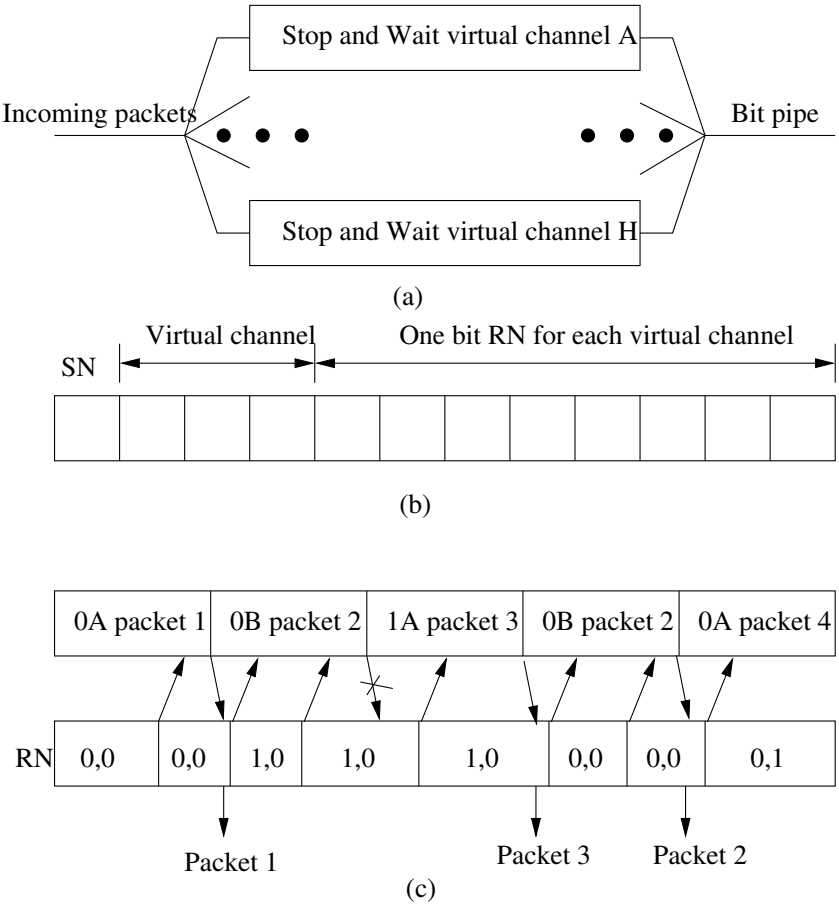
Buffer size n for both transmitter and receiver

In practice ACK or NAK is sent instead of RN.



5.5 ARPANET ARQ

(parallel channels with stop-and-wait)



ARPANET ARQ. (a) Eight multiplexed, stop–and–wait virtual channels. (b) Bits in the header for ARQ control. (c) Operation of multiplexed stop and wait for two virtual channels. Top–to–bottom frames show SN and the channel number, and bottom–to–top frames show RN for both channels. The third frame from bottom to top acks packet 1 on the A channel.

Advantages:

- repetition of ACKs leads to reliable reverse channel
- small idle time → high throughput $\approx \eta_{max}$

Disadvantages:

- delivers packets unordered

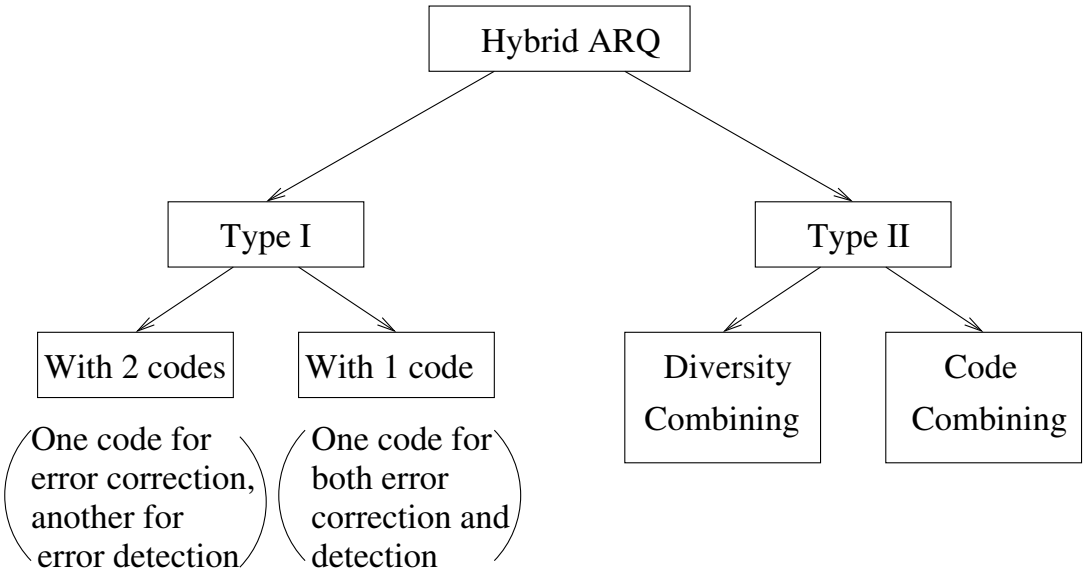
Buffersize:

transmitter 8

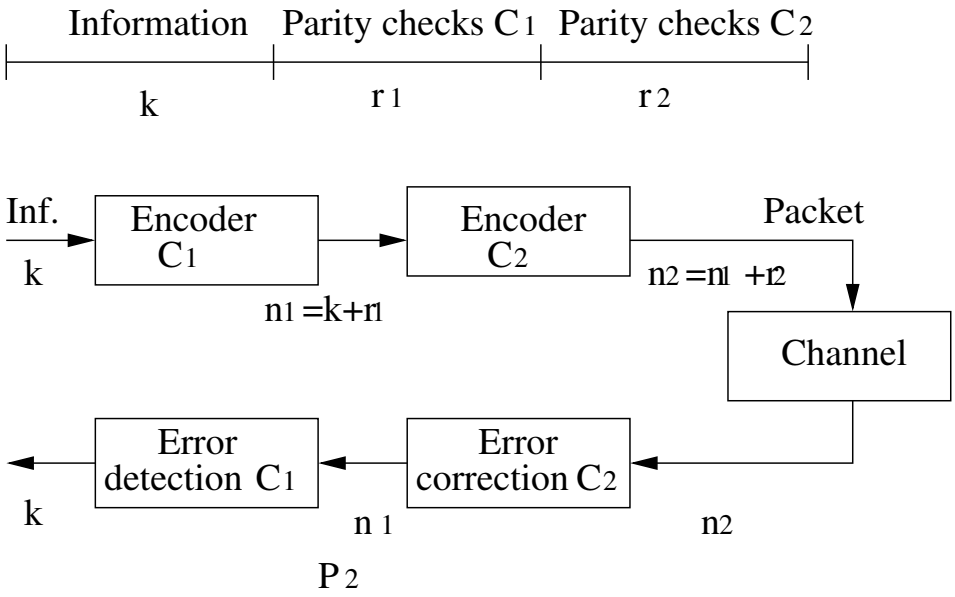
receiver 0

5.6 Hybrid ARQ

If probability of retransmission in DLC is large, use additional forward-error-correction in DLC.



5.6.1 Hybrid ARQ, Type I, with 2 codes



P_2 - frame error rate of decoder C_2 ($P_2 \ll P_f$)

Probability of retransmission $P_r^I = P_2(1 - P_{CRC}) \ll P_r$

Probability that frame in error and accepted $P_e^I = P_2 P_{CRC}$

Final frame error rate

$$P^I(E) = \frac{P_e^I}{1 - P_r^I} = \frac{P_2 P_{CRC}}{1 - P_2(1 - P_{CRC})}$$

$$\eta_{max}^I = R_1 R_2(1 - P_r^I) = R_1 R_2[1 - P_2(1 - P_{CRC})]$$

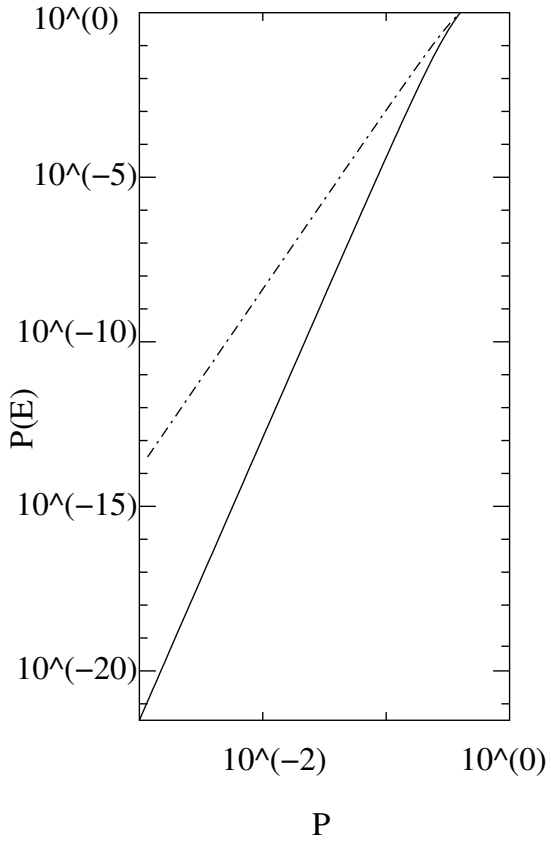
ARQ Hybrid I	Throughput
Stop-and-wait	$\eta_{max}^I \frac{\tau_{msg}}{\tau_{msg}+\tau_{idle}}$
Go-back-n	$\eta_{max}^I \frac{1}{1+P_r^I(n-1)}$
Selective-repeat	$\approx \eta_{max}^I$

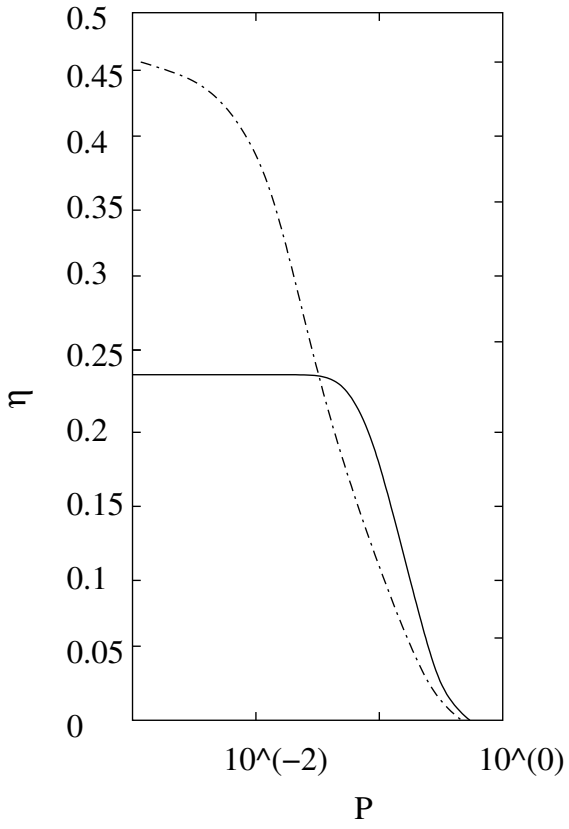
Comparison of classic ARQ with Hybrid Type I ARQ with 2 codes.

Selective-repeat strategy

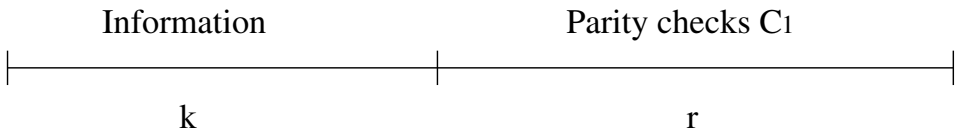
——— Hybrid

– · – · – Classic





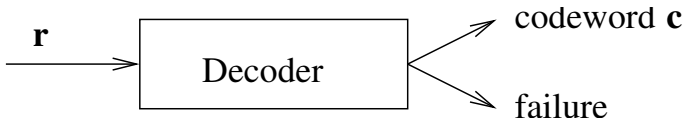
5.6.2 Hybrid ARQ Type I with 1 code



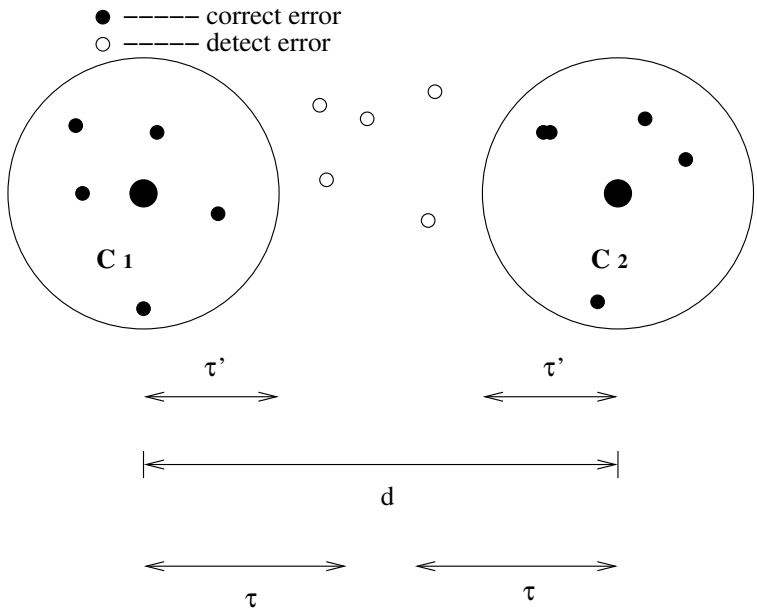
(n, k, d) code C_1 corrects $\tau = \lfloor \frac{d-1}{2} \rfloor$ errors

Take correcting radius $\tau' < \tau$

Decoding:

1. 

2. If $| \overline{r} - \overline{c} | > \tau'$ or failure then Detect error otherwise Output $\overline{c}$



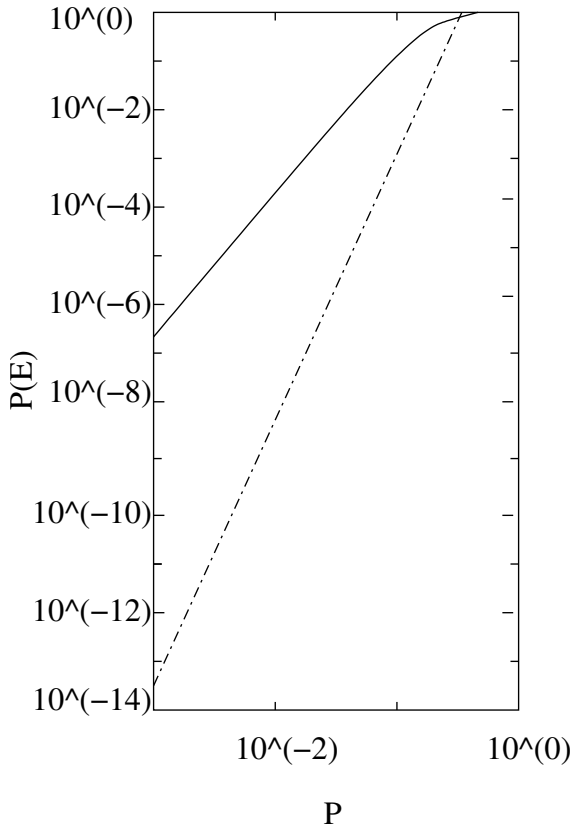
If weight distribution of C_1 is known, P_r^I and P_e^I can be computed and $P^I(E)$ and η can be found using the same formulas.

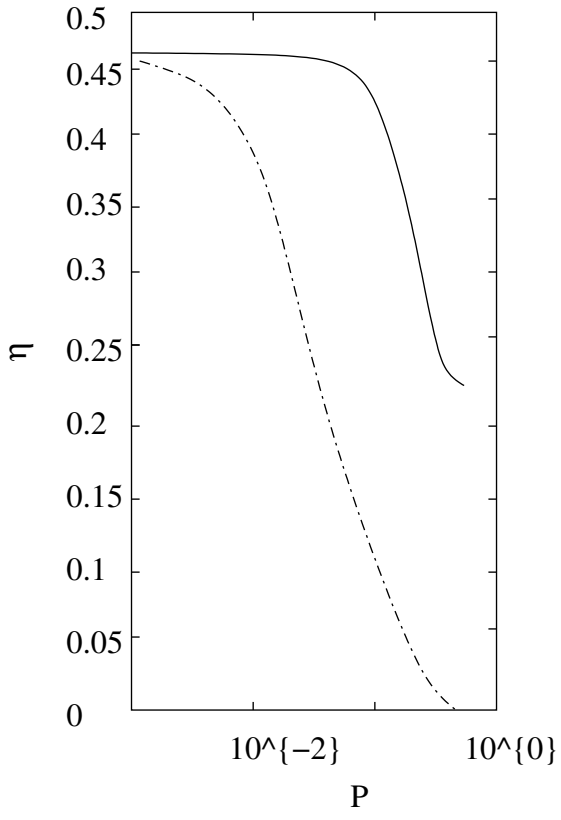
Comparison of classic ARQ with Hybrid Type I ARQ with 1 code.

Selective-repeat strategy

———— Hybrid

– · – · – Classic



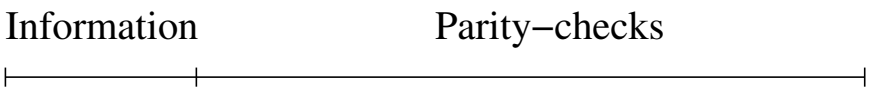


5.6.3 Hybrid ARQ Type II

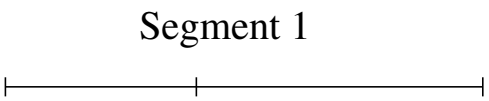
Diversity combining:

- Joint decoding of all retransmissions of one packet.

Code combining



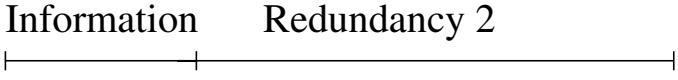
1. Transmit first segment: use it for error correction and detection



2. If errors were detected, transmit second segment

• • •

and use both segments together for error correction and detection using more powerful code ...



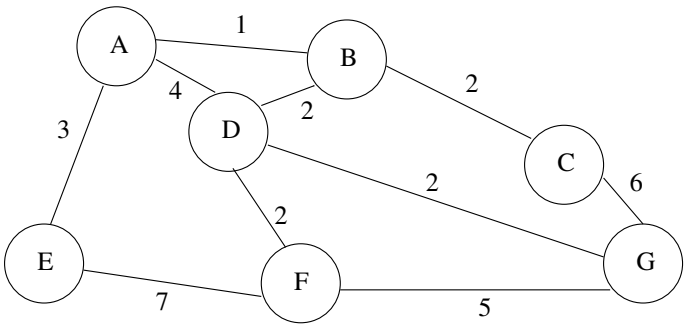
Chapter 6

Routing

Routing algorithm is the network layer protocol that guides packets to their destination.

Examples:

(1) Information about network (graph)



(2) Routers to other nodes: Routes starting from node A

(3) Routing tables for control of switches:

Switch of node A:

Destination	B	C	D	E	F	G
Next node	B	B	B	E	B	B

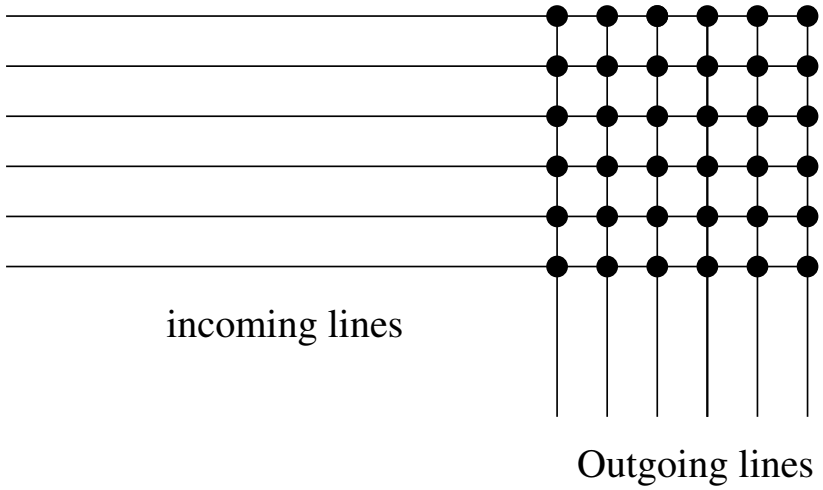
Switch of node B:

Destination	A	C	D	E	F	G
Next node	A	C	D	A	D	D

Switch of node C:

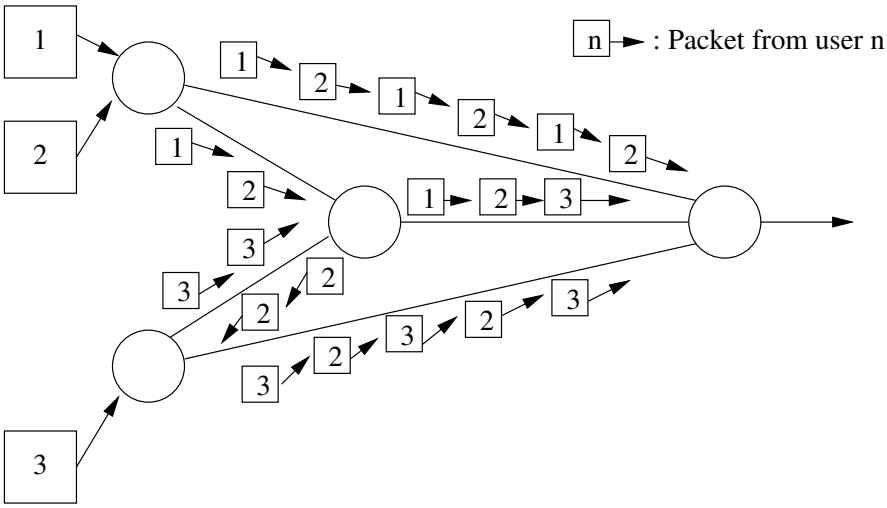
Destination	A	B	D	E	F	G
Next node	B	B	B	B	B	G

Basically: How does a switch work?

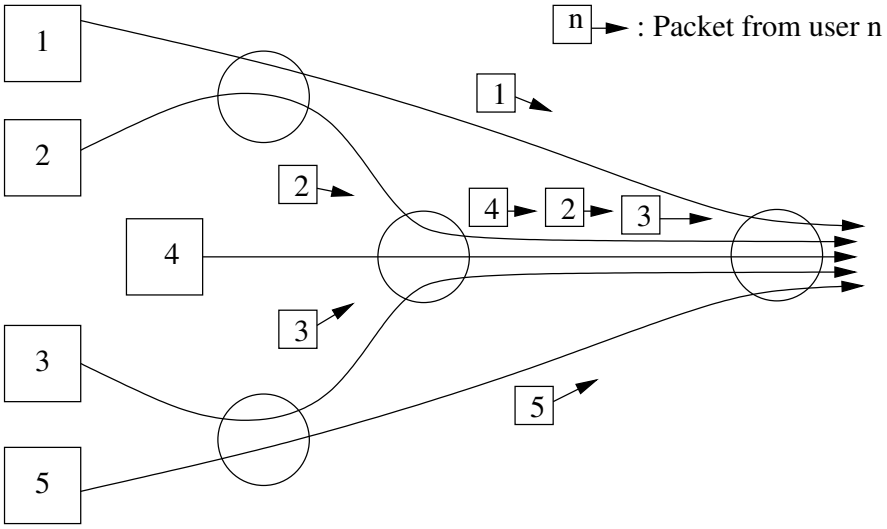


Network may use:

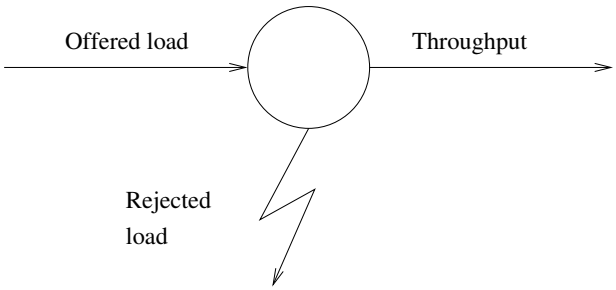
- datagrams or
- virtual circuits



Routing in a datagram network. Two packets of the same user pair can travel along different routes. A routing decision is required of each individual packet.

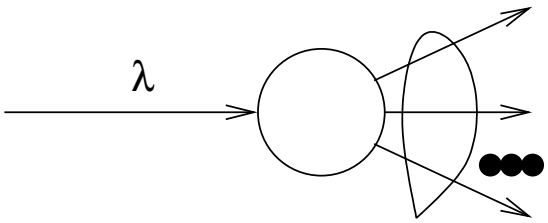


Routing in a virtual circuit network. All packets of each virtual circuit use the same path. A routing decision is required only when a virtual circuit is set up.

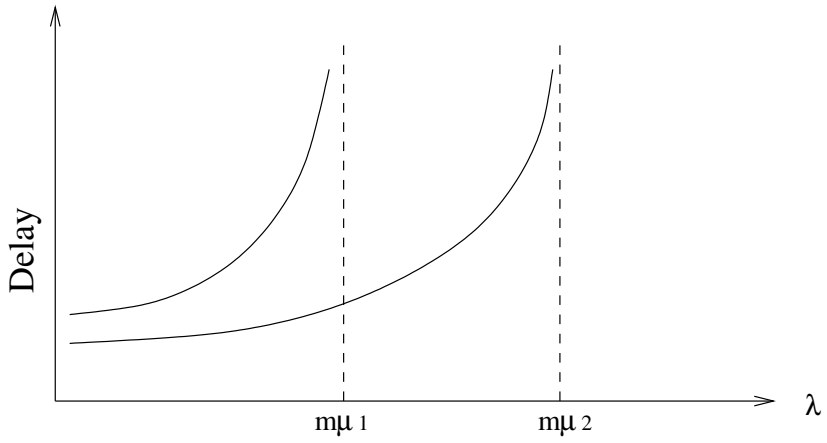


Throughput = Offered load - rejected load

Let Throughput = Offered load
From queueing theory:

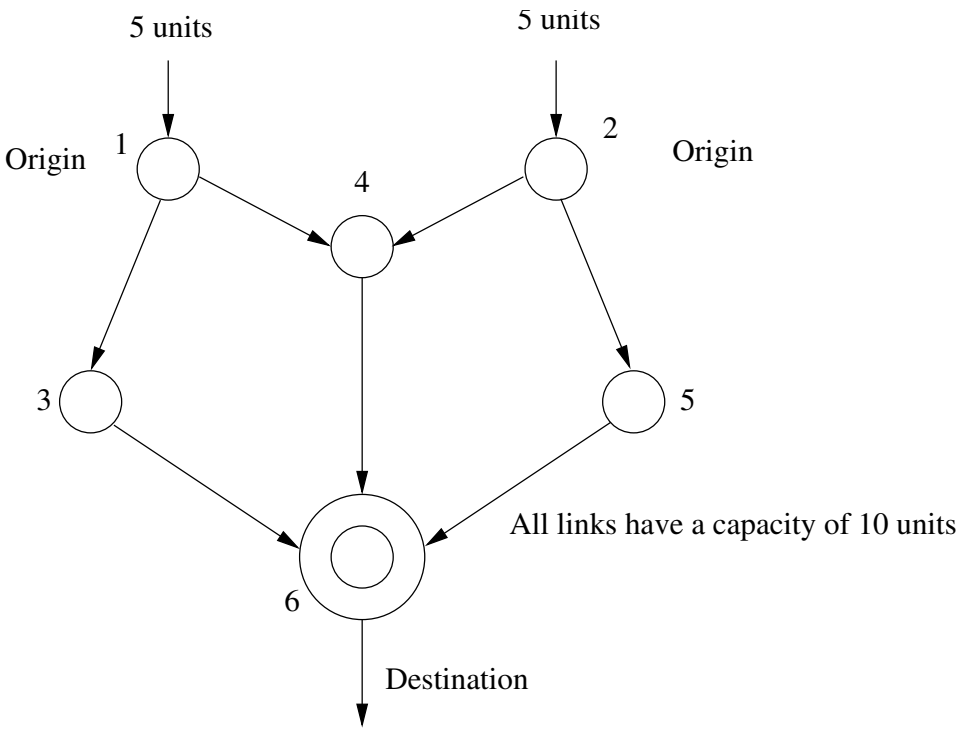


λ - arriving rate (throughput)
$m\mu$ - service rate (when busy)
$\rho = \frac{\lambda}{m\mu}$ - utilization factor
The average delay = $\frac{1}{\mu} + \frac{f(\lambda,\mu,...)}{1-\rho}$



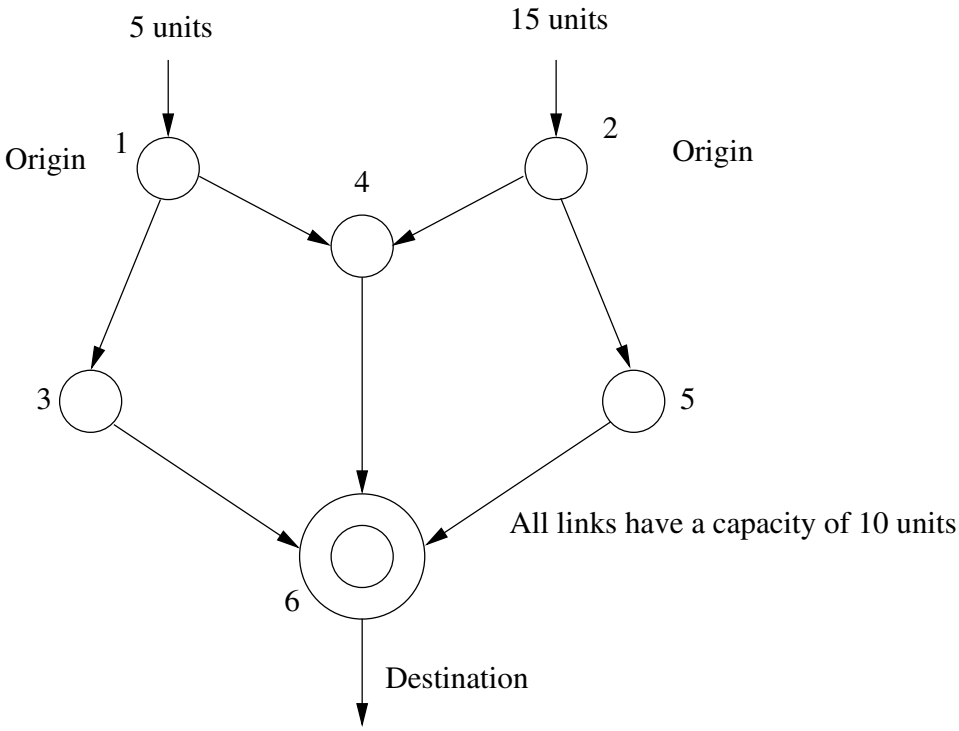
Throughput

Example I



All links have a capacity of 10 units. If all traffic is routed through the middle link (4,6), congestion occurs. If, instead, paths (1−−−3−−−6) and (2−−−5−−−6) are used, the average delay is small.

Example II Datagram routing is better.



All links have a capacity of 10 units. The input traffic can be accommodated with multiple −path routing , but at least 5 units of traffic must be rejected if a single−path routing is used.

6.1 Classification of routing algorithms

- Datagram (packet switching) versus connection-oriented switching
- Static versus adaptive
  - Static: Routes between switches/terminals are determined once (possibly at network setup)
  - Adaptive: New routes are determined frequently or on demand, taking into account the actual traffic load, delay, and buffer queue length on all individual links.
- Centralized versus distributed algorithm
  - Centralized: Bellmann-Ford algorithm, Dijkstra algorithm
  - Distributed: Ford-Fullerson algorithm, distributed Bellmann-Ford algorithm.
- Support of p2p, multicast/p2mp, broadcast connections
  - Multicast/broadcast can be useful for distributing network status information in adaptive routing.
- Address resolution methods, network topology: Routing tables with entries for all existing terminals are impossible to handle in large networks (e.g. telephone networks, Internet)
  - Hierarchical network structure
  - Name server (IP: Domain Name Server, UMTS/GSM Home Location Register, Visiting Location Register).

Example

Routing in telephone networks:

- Connection-oriented
- Hierarchical:
  - country code +49
  - city code 731
  - company 505
  - phone 4816

6.2 Basics from Graph Theory

Graph, Directed Graph (Digraph): G={N,E}

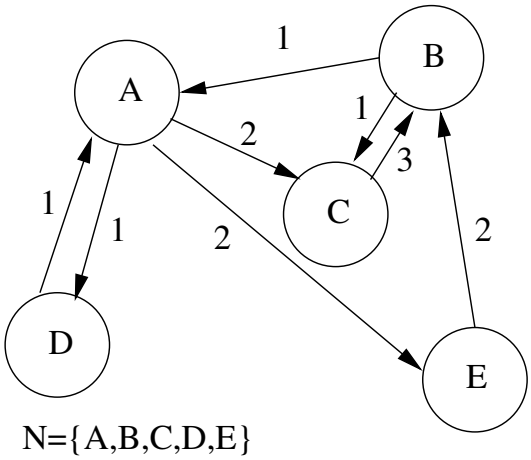
N: Set of nodes

E: Set of edges E $\subseteq$ N$\times$N, Edge e=(m,n)

- Node m is connected to node n (in graph),
- There is a link from node m to node n (in digraph)

d(e): Weight of an edge $e \in E$ (can correspond to delay, traffic load, buffer queue length,...)

Example:



N={A,B,C,D,E}
E={(A,D),(A,E),(A,C),(B,A),(B,C),(C,B),(D,A),(E,B)}
d(E)=(1,2,2,1,1,3,1,2)

Adjacency matrix:

	A	B	C	D	E
A			2	1	2
B	1		1		
C		3			
D	1				
E		2			

Incidence matrix:

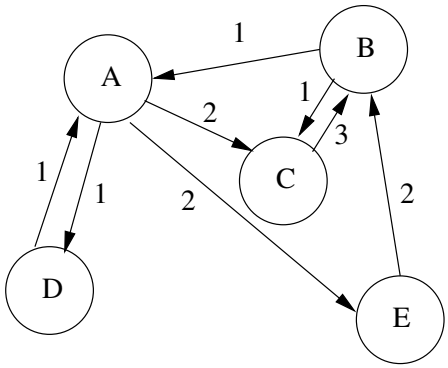
	(A,D)	(B,C)	···
A	1		
B		1	
C		−1	
D	−1		
E			

Walk: sequence of connected nodes.

Path: a walk with no repeated nodes.

Cycle: Path, whose starting node of the fist edge equals the ending node of the last edge, and all the edges are distinct.

Example:



Connected graph: there is a path connecting every two nodes.

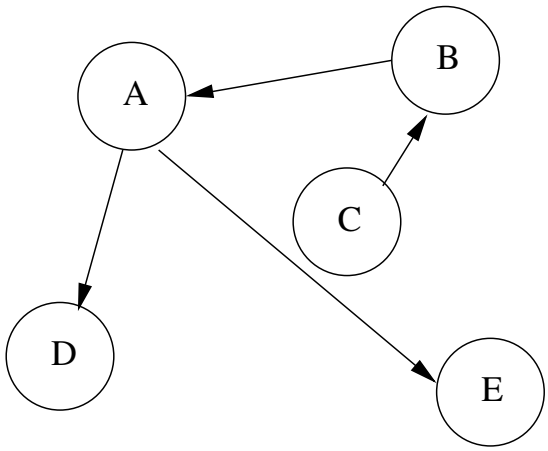
Hamiltonian cycle: cycle, which contains all nodes of the graph.

Tree: connected graph without cycles.

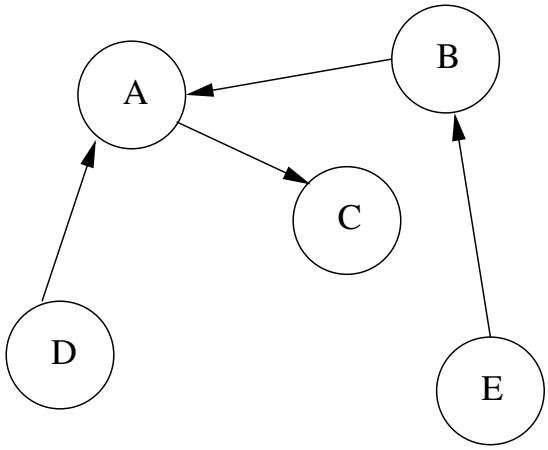
$| N |=| E_{tree} | +1$

Spanning (Complete) tree: contains all nodes of the graph.

Minimal tree: Complete tree, with minimum sum of edge weights over all complete trees (There can exist more than one minimal tree in a given graph).



Source tree



Sink tree

6.3 Routing algorithms

6.3.1 Flooding

Principle: Forward a packet to all neighboring nodes except the node which sent the packet.

Advantages:

- All nodes are reached (broadcast)
- All possible paths are used in parallel, including the shortest path from source to any destination.
- Any nod needs to know only his incoming and outgoing lines, but nothing else about the network.

Drawback: High network load!

Problems:

- Cycles: Packet will be forwarded in cycle forever.
- Same packet received on several lines-forward each of them?

Counter measures:

- Hop counter: Packet is deleted after certain number of hops. Requires appropriate counter setting.
- Packet carries source node's address and a sequence number. Switching node can recognize already known packets and discard them.

Applications:

- Distribution of network status info for other, more elaborate routing algorithms (connectivity, traffic load, link failures,...)
- For concurrent update of data bases in all network nodes.
- In military networks with extreme robustness demands.

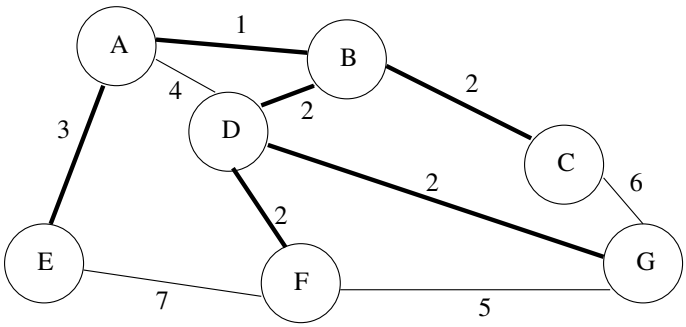
6.3.2 Complete tree and Hamiltonian cycle

Principle:

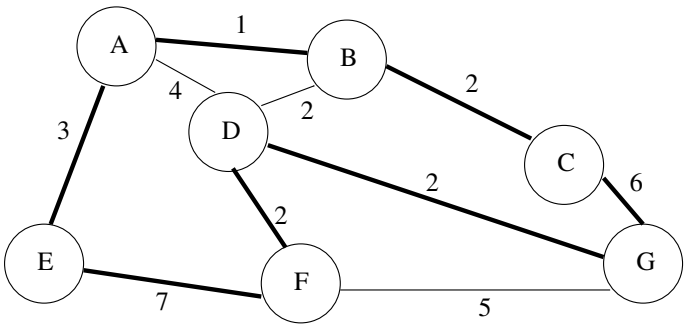
- Determine a complete source tree (with source node as root), or a Hamiltonian cycle from the graph of the network.
- Forward the packet along the tree or the cycle (preferably in both directions).

Example:

Complete tree



Hamiltonian cycle



Advantages:

- All nodes reached (broadcast).
- Significantly lower network load than with Flooding.
- No problems with cycles.

Drawbacks:

- Network graph must be known (contrary to Flooding).
- New complete tree/Hamiltonian cycle must be determined, when network topology changes.
- All nodes must know the complete tree/Hamiltonian cycle.

6.3.3 The Bellman - Ford algorithm

Problem: Find a shortest path from every node of G to destination node 1.

Let $d_{ij} = \infty$ if (i, j) is not an edge of G.

Assume that G has no cycles of negative weight.

Let D_i^h denote length of a shortest $(\leq h)$ walk from node i to 1; $D_i^h = 0 \quad \forall h$

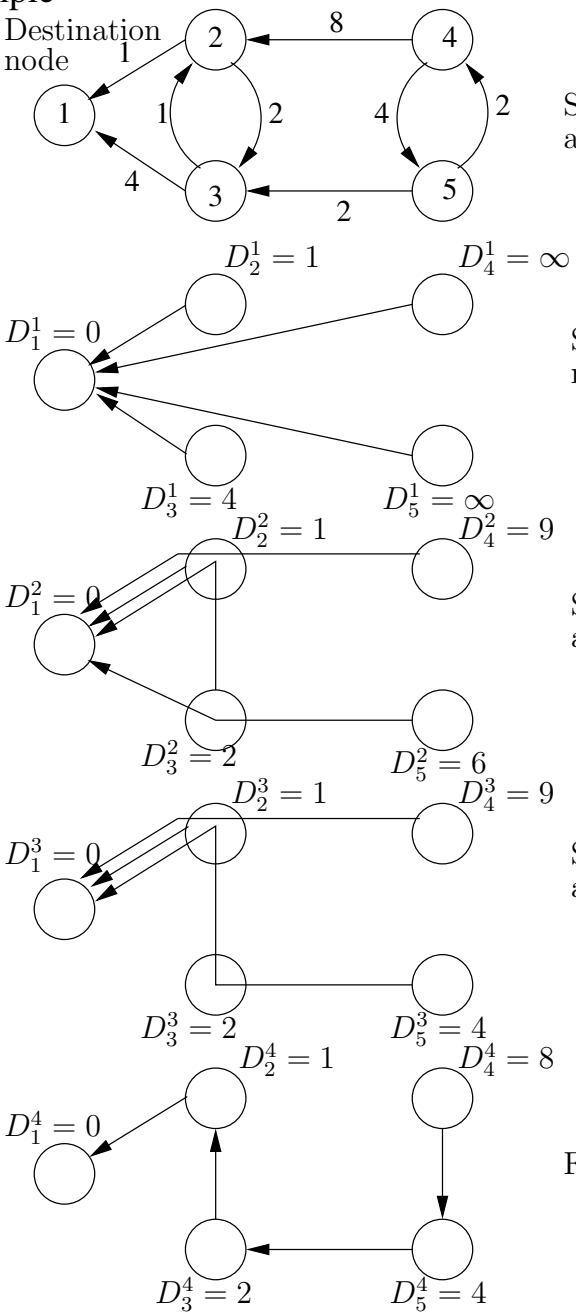
Bellman - Ford algorithm:

1. Initialization: $D_i^0 = \infty \quad \forall i \neq 1$
2. For $h = 1, 2, \ldots, \mid N \mid$ do

   $D_i^h \; =: \; \min_j [d_{ij} + D_j^{h-1}], \quad \forall i \neq 1$

   If $D_i^h = D_i^{h-1}$ then stop.

Example



Succesive iterations of the Bellman-Ford method. In this example, the shortest ($\leq$h) walks are paths because all arc lengths are positive and therefore all cycles have positive length. The shortest paths are found after N-1 iterations, which is equal to 4 in this example.

Theory: The BF algorithm terminates after a finite number $h \leq \mid N \mid$ iterations and D_i^h is the shortest path length from i to 1.

Proof: (Sketch) (For simplicity assume $d_{ij} > 0$)

Induction on h.

1. For $h = 1 \quad D_i^1 = d_{i1} \quad \forall i \neq 1$
2. Suppose $\forall i \; D_i^k$ is length of the shortest ($\leq$k) path P from i to 1 $\forall k \leq h$.

   P has no cycles, otherwise it will not be a shortest.

   Let us show that D_i^{k+1} is the length for a shortest path P'

   Let $P' = (1, \nu_2, \ldots, \nu_k, i)$, then weight

$$W(P') = W(1, \nu_2, \ldots, \nu_k) + W(\nu_k, i) = \min_j [D_i^k + d_{ij}] \quad (j = \nu_k)$$

BF algorithm works in case $d_{ij} < 0$ too.

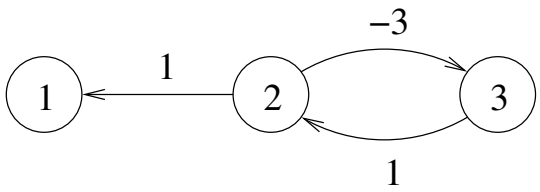
Complexity of BF algorithm: $O(\mid N \mid^3)$ or

$O(m \cdot \mid E \mid)$ where m is maximum length of the shortest path.

Bellman-Ford algorithm works with negative weights when every cycle has nonzero weight.

What if $\exists$ a cycle of negative weight?

Example:



i	D_1^i	D_2^i	D_3^i
0	0	∞	∞
1	0	1	∞
2	0	1	2
3	0	-1	2

$\mid N \mid = 3 \quad D_2^3 < D_2^2$

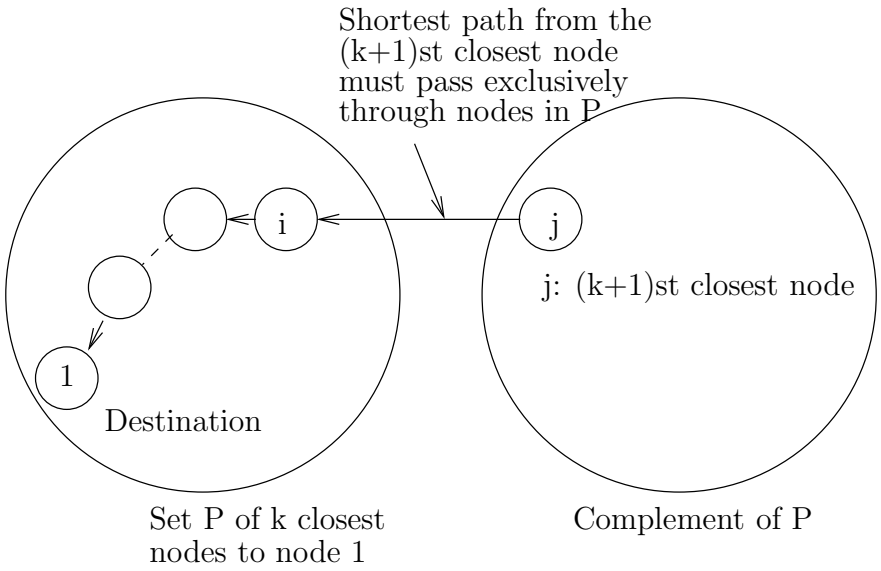
In general: If $\exists\ i \quad D_i^{|N|} < D_i^{|N|-1}$, then $\exists$ cycle of negative length.

Dijkstra's algorithm

Problem: Find a shortest path from every node of G to destination node 1.

- Assume, weight of edge $d_{ij} \geq 0 \quad \forall\ i,\ j$

General ides: iteration on path length (for BF algorithm, iteration on number of hops).



Basic idea of Dijkstra's algorithm. At the k^{th} step we have the set P of the k closest nodes to node 1 as well as the shortest distance D_i from each node i in P to node 1. Of all paths connecting some node not in P with node 1, there is a shortest one that passes exclusively through nodes in P (since $d_{ij} \geq 0$). Therefore, the (k+1)st closest node and the corresponding shortest distance are obtained by minimizing over $j! \in P$ the quantity $min_{i \in P}\{d_{ji} + D_i\}$. This calculation can be organized efficiently as discussed in the text, resulting in an $O(N^2)$ computational complexity.

Dijkstra's algorithm:

$N = \{1, 2, \ldots, \mid N \mid\}$ set of nodes

Node 1 is destination

D_i-shortest path length to node 1

P-set of closest nodes to node 1

d_{ij}-weight of edge (i, j)

Initialization

$$P = \{1\},\ D_1 = 0,\ D_j = d_{j1} \quad \forall j \neq 1$$

Step 1 (Find the next closest node)

$i = \arg\min_{j \neq P} D_j$

Set $p := P \bigcup \{i\}$

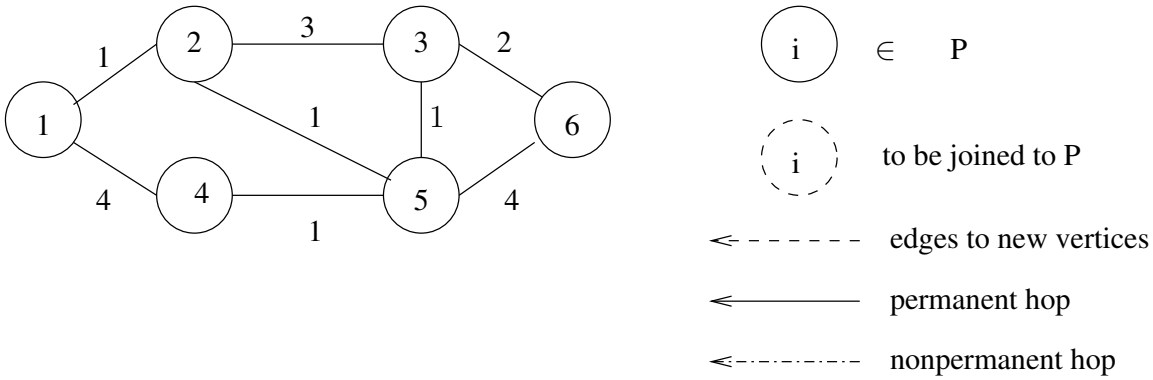
If $P = N$, then stop.

Step 2 (Updating of labels)

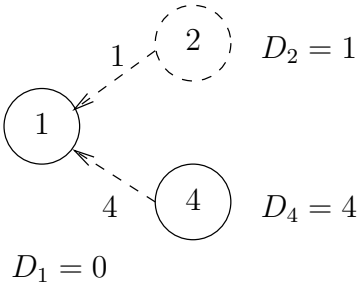
$\forall j \neq p \quad D_j := min\{D_j,\ d_{ji} + D_i\}$

Go to Step 1.

Example: Dijkstra's algorithm

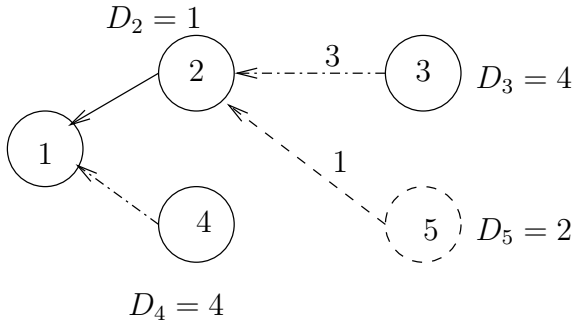


Initialization: $D_1 = 0,\ D_2 = 1,\ D_4 = 4\ (D_3 = D_5 = D_6 = \infty)$



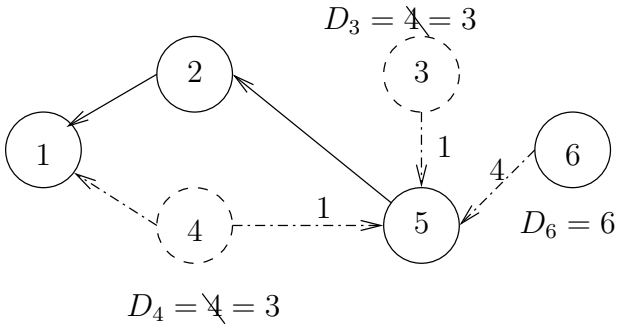
1 $i = \arg\min\{D_2,\ D_4\} = 2;\ P = \{1, 2\}$

2 $\forall j! \in P\ D_j := \min\{D_j,\ d_{ji} + D_i\}$



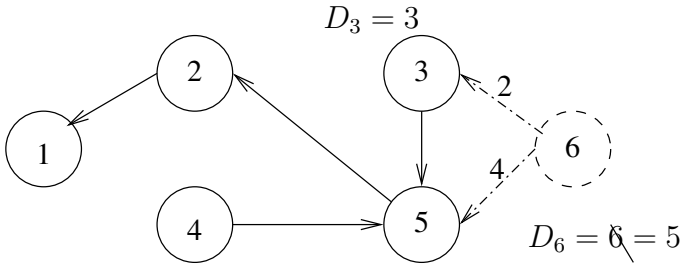
1 $i = \arg\min\{D_3,\ D_4,\ D_5\} = 5;\ P = \{1, 2, 5\}$

2 $D_3 = \min\{D_3,\ d_{3,5} + D_5\} = \min\{4,\ 1 + 2\} = 3$
$D_4 = \min\{D_4,\ d_{4,5} + D_5\} = 3$
$D_6 = 2 + 4 = 6$



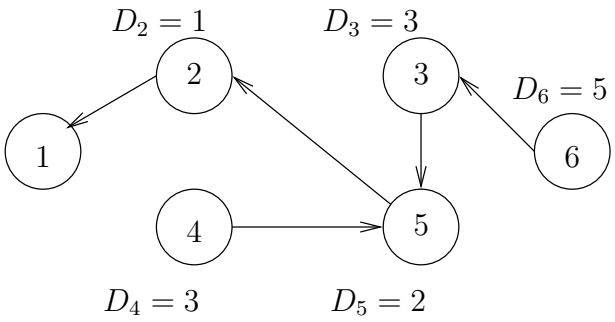
1 $i = \arg\min\{D_3,\ D_4,\ D_6\ \} = \{3, 4\};\ P = \{1, 2, 5, 3, 4\}$

2 $D_6 = \min\{D_6,\ d_{6,3} + D_3\} = 5$



1 $i = 6$; $P = \{1, 2, 3, 4, 5, 6\} = N$; Stop

Result:



Complexity of Dijkstra's algorithm $O(\mid N \mid^2)$

Correctness of Dijkstra's algorithm follows from:

At the beginnning of every step 1:

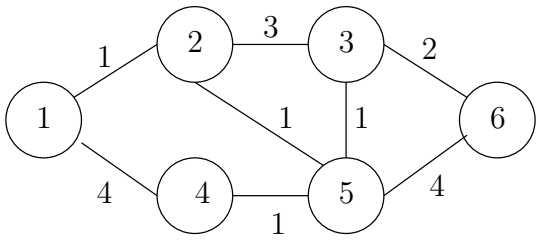
a $D_i \leq D_j \quad \forall i \in P \; j! \in P$

b D_i is the shortest path for $\forall i \in P$

At every iteration one node is added to P. So, after $\mid N \mid -1$ iterations $P = N$ and algorithm stops.

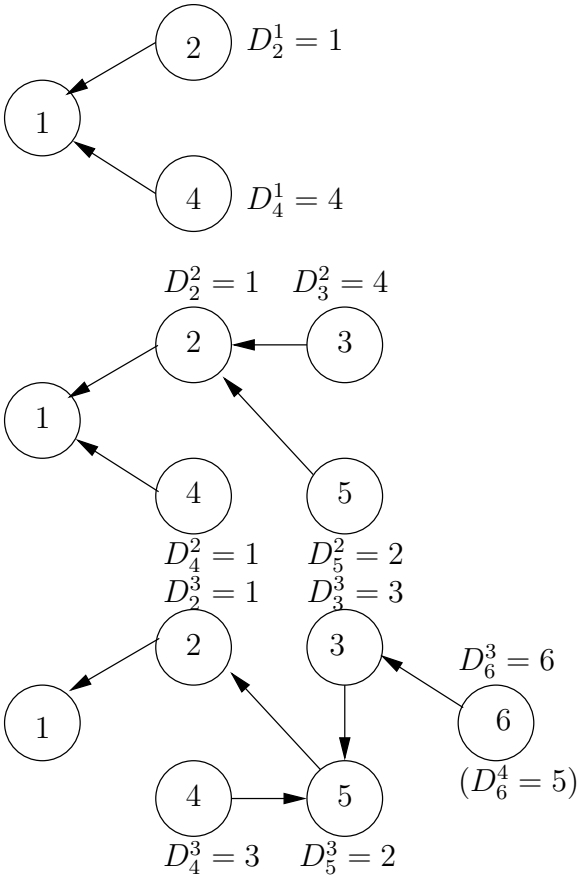
From b it follows that when $P = N$, D_i is the shortest path $\forall i$.

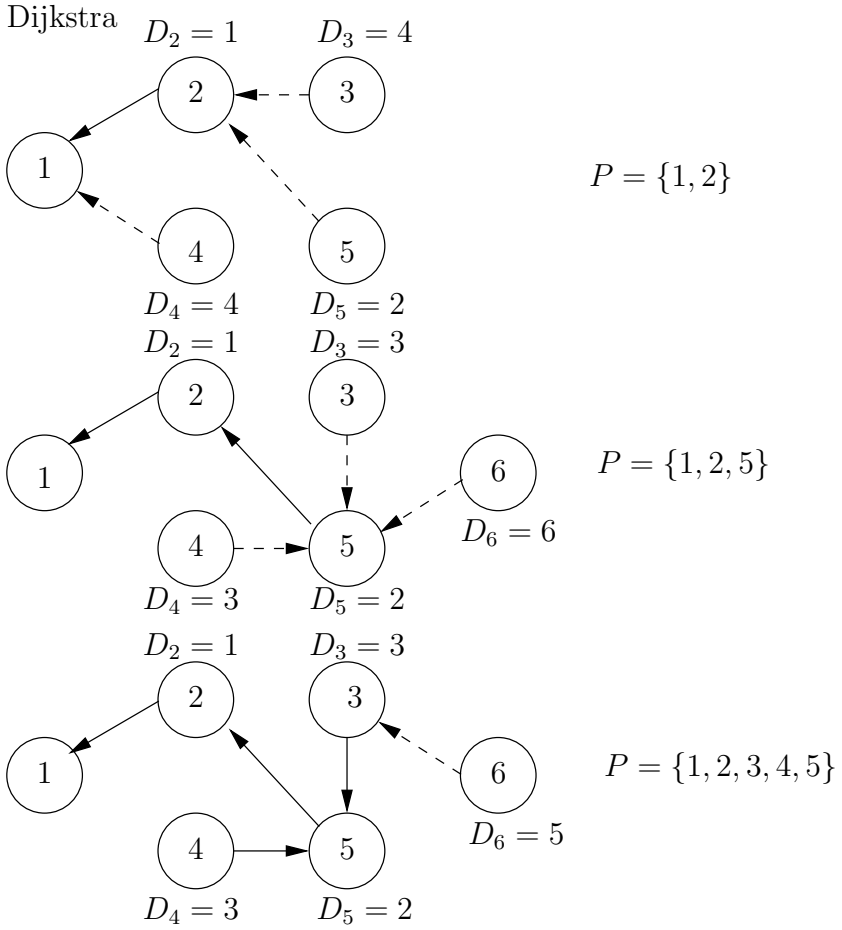
Comparison of Bellman-Ford and Dijkstra's algorithms



$d_{ij} = d_{ji}$ for all (i, j)

Bellman-Ford



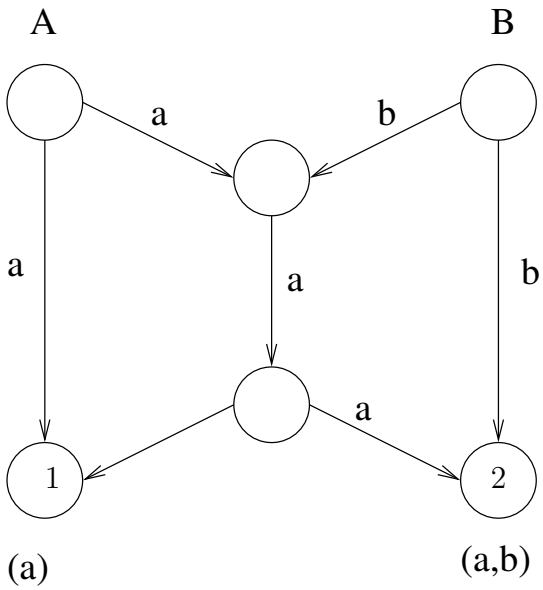


	Max. complexity	Remarks
Bellman-Ford	$O(\mid N \mid^3)$	works with negative d_{ij}
Dijkstra	$O(\mid N \mid^2)$	requires $d_{ij} \geq 0$

Coding in networks

Without coding:

Can we send a,b to nodes 1,2 in one time slot? No!



With coding:

Yes!