Computer Networks - Overview:

- Computer Network Terms
- Common functions and devices in computer networks
- Structure of computer networks
- Many examples with some detailed information

—

Computer Networks - The Plan:

- now - spring break: HDLC, Ethernet (“Low-Level”)
- spring break - end: IP, TCP and applications (“High-Level”)

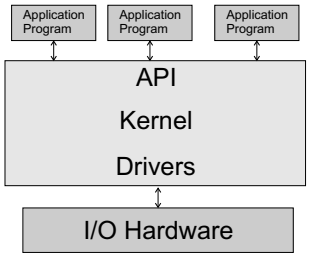
—

Operating Systems: Missing things:

- API: Application Programming Interface
- Files, Filedescriptors

—

APIs:



- Operating systems define an API
- Application programs communicate with the OS only by using the API

—

APIs:

- APIs can be very different in size:
  - Win32: 2016 Functions
  - Linux: 190 System Calls (Linux 2.2)
  - Minix: 53 System Calls
- Some common UNIX system calls: read, write, open, close, fork, kill, sbrk
- System calls are initiated in the application program but executed by the operating system: context switch (implemented through trap)

—

Examples for (unix) API functions:

- Memory management: malloc, free, sbrk
- I/O: open, close, read, write, fstat, ioctl
- Network: socket, bind, connect, shutdown, send, recv
- signals and process control: signal, kill, fork, execvp
- synchronisation: sema_post, sema_wait, sema_trywait

—

File Systems and APIs:

- Almost all general purpose OS control disk devices through file systems
- Exceptions: Some database server systems access disk devices directly
- A file system is a method of assigning names to sequences of disk blocks and controlling access to them.
- Examples: Dos Fat32, Linux Ext3, ReiserFS

—

How does it work ?:

```
h = open("thefile",O_RDWR|O_CREAT);
write(h,buffer,length_of_buffer);
close(h);
```

—

How does it really work ?:

```
h = open("thefile",O_RDWR|O_CREAT);
if (h==-1) {printf("Error %d\n",errno);
  perror("FileIO");exit(-1);}
n = write(h,buffer,length_of_buffer);
if (n==length_of_buffer)
  printf("successful write of buffer\n");
if (n==0) printf("nothing written\n");
if (n==-1) { printf("Error %d\n",errno);
  perror("FileIO"); }
n==close(h);
if (n==-1) { printf("Error %d\n",errno);
  perror("FileIO");}
```

—

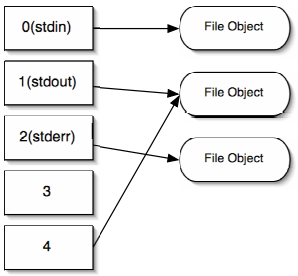
A look inside: Linux:

```
struct task_struct {
        volatile long state;    /* -1 unrunnable,
                                   0 runnable, >0 stopped */
        unsigned long flags;    /* per process flags,
                                   defined below */
        int sigpending;
...
/* filesystem information */
        struct fs_struct *fs;
/* open file information */
        struct files_struct *files;
/* namespace */
        struct namespace *namespace;
/* signal handlers */
...
```

from linux 2.4.20 include/linux/sched.h ——

A look inside: Files of a process:

Idea of the `files_struct` for a process...



—

A look inside: Linux:

```
struct files_struct {
        atomic_t count;
        rwlock_t file_lock;
        int max_fds;
        int max_fdset;
        int next_fd;
        struct file ** fd;
        fd_set *close_on_exec;
        fd_set *open_fds;
        fd_set close_on_exec_init;
        fd_set open_fds_init;
        struct file * fd_array[NR_OPEN_DEFAULT];
};
```

from linux 2.4.20 include/linux/sched.h ——

A look inside: Linux:

```
struct file {
        struct list_head        f_list;
        struct dentry           *f_dentry;
        struct vfsmount         *f_vfsmnt;
        struct file_operations  *f_op;
        atomic_t                f_count;
        unsigned int            f_flags;
        mode_t                  f_mode;
        loff_t                  f_pos;
...
```

from linux 2.4.20 include/linux/fs.h ——

A look inside: Linux:

```
struct file_operations {
    struct module *owner;
    loff_t (*llseek) (struct file *, loff_t, int);
    ssize_t (*read) (struct file *, char *, size_t, loff_t *);
    ssize_t (*write) (struct file *, const char *, size_t, loff_t *);
    int (*readdir) (struct file *, void *, filldir_t);
    unsigned int (*poll) (struct file *, struct poll_table_struct *);
    int (*ioctl) (struct inode *, struct file *, unsigned int, unsigned long);
    int (*mmap) (struct file *, struct vm_area_struct *);
    int (*open) (struct inode *, struct file *);
    int (*flush) (struct file *);
...
```

from linux 2.4.20 include/linux/fs.h

Conclusion:

- Processes keep track of their open files
- Open files are kept in a numbered list
- File operations are abstract
- OS fills file operations with implementation

Network vs. File example: Coffee!:

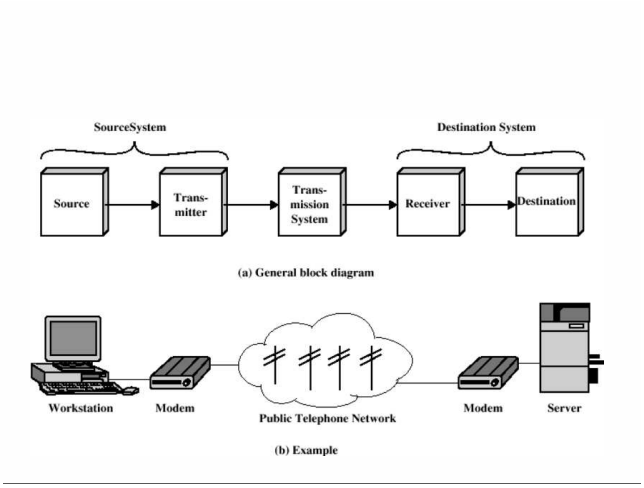
```
sprintf(buffer,"There's coffee available!");
sock=connectUDP("255.255.255.255","51966");
n=send(sock,buffer,strlen(buffer)+1,0);
close(sock);
```

```
h = open("thefile",O_RDWR|O_CREAT);
write(h,buffer,length_of_buffer);
close(h);
```

Network example: connectUDP:

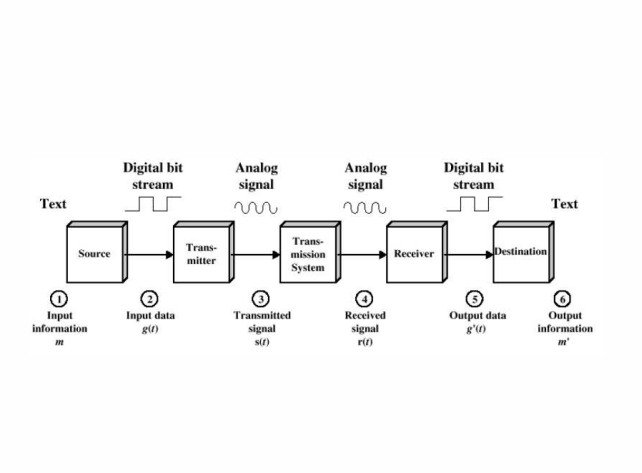
```
int connectUDP(char *host,char *service)
{
struct hostent  *phe;   /* pointer to host information entry    */
struct servent  *pse;   /* pointer to service information entry */
struct protoent *ppe;   /* pointer to protocol information entry*/
struct sockaddr_in sin; /* an Internet endpoint address         */
int     s, type;        /* socket descriptor and socket type    */
pse = getservbyname(service, "UDP"); sin.sin_port = pse->s_port;
phe = gethostbyname(host);
memcpy((char *)&sin.sin_addr,phe->h_addr, phe->h_length);
ppe = getprotobyname(protocol); type = SOCK_DGRAM;
s = socket(PF_INET, type, ppe->p_proto);
connect(s, (struct sockaddr *)&sin, sizeof(sin);
return(s);
```

Networks: An Overview:



—

Networks: Communication Model:



—

Network Types:WANs:

- Traditional: Circut Switching and Packet Switching
- Frame Relay: less error control, higher data rate
- ATM: fixed cells

—

Network Types:LANs:

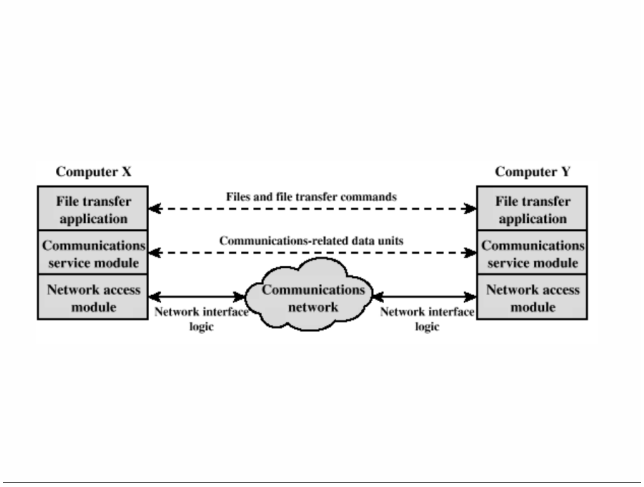
- Token Ring

- Ethernet
- Wireless LAN

—
Network Protocols:
Protocols define:

- Syntax: Data Format, Signal Levels
- Semantics: Control Information for coordination and error control
- Timings: Sequencing, Transmission Speed, Timeouts

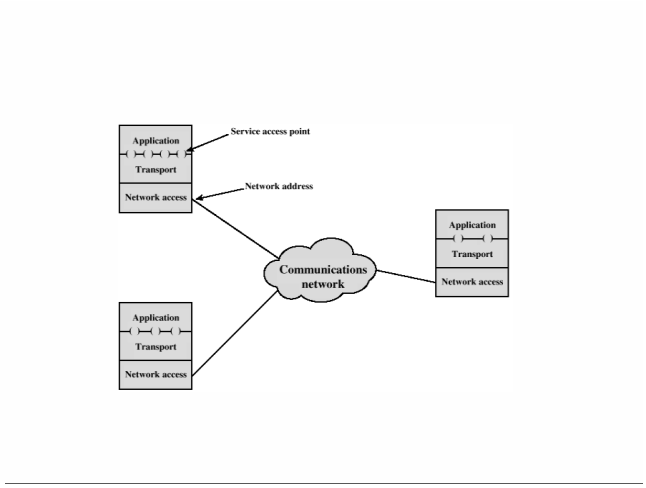
A Protocol Architecture: Set of protocols used together in a modular architecture. —
A simple protocol architecture:



—
Network Layer Model:
A Simple Model:

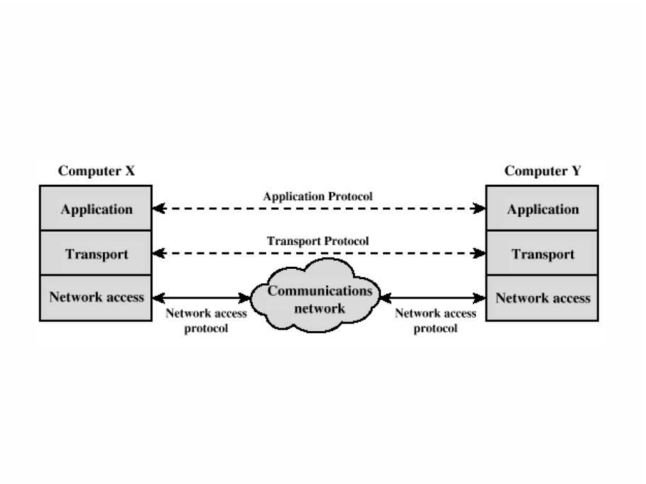
- Network Access Layer: Network (hardware) dependent, adressing
- Transport Layer: Reliable data exchange
- Application Layer: Application dependent (Service Access Points)

Each layer only communicates with adjacent layers —
A simple protocol architecture:



—

Protocols:



—

Constructing Packets:
Protocol Data Units:

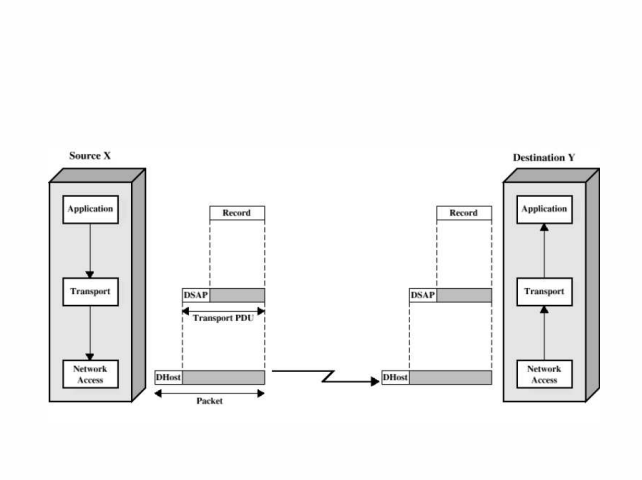
- Constructed from the data of the next higher layer and additional control information

Example:

- Transport PDU adds SAP identification, sequence number and error detection information to Application PDU.
- Network PDU adds adressing and optional priority information to Transport PDU

—

PDUs in the simple control architecture:



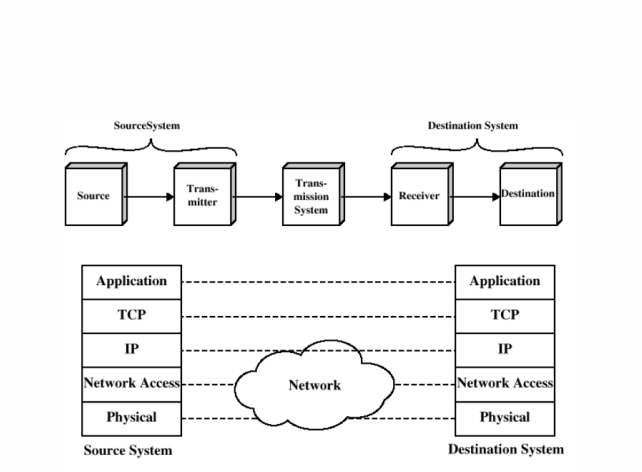
—

TCP/IP Protocol Architecture:

- 5 layer architecture: Application, Transport, Internet, Network, Physical
- TCP/IP accepts almost all Network and Physical Layers: PPP, GPRS, ATM, SONET, INMARSAT
- Application layer exists in multiple implementations: SYSV streams is one, sockets is another

—

TCP/IP Protocol Architecture:



—

OSI Protocol Architecture:

- Standardized by ISO
- 7 layer architecture: Application, Presentation, Session, Transport, Network, Data Link, Physical
- Connection-oriented
- Hardly implemented, classic example of a “failed” standardisation effort

—

OSI Protocol Architecture:

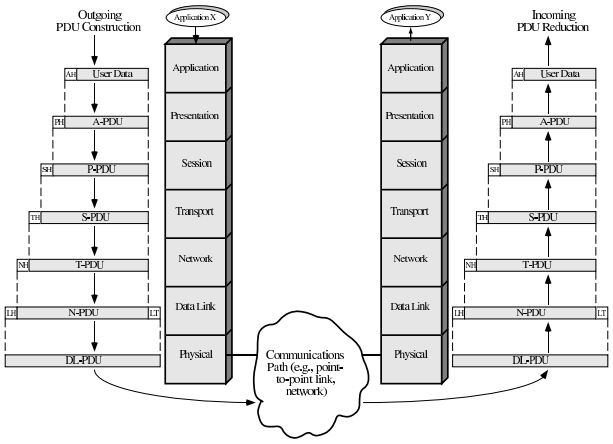
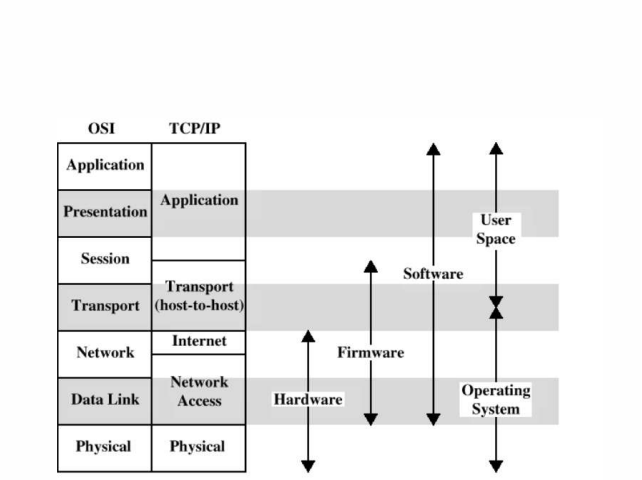


Figure 2.6 The OSI Environment

—

TCP/IP vs. OSI:



—

Standards:

- Standards Organizations: Internet Society, ISO, ITU-T, IEEE , ATM Forum,...
- Standardization Process
- Standardization is a form of business war
- Standards always come too late
- Often, De-facto standards are “promoted”. Example: Ethernet → IEEE 802.x

—
Internet Standards:

- Responsibility: Internet Society
  - Internet Architecture Board (IAB): Defines overall architecture
  - Internet Engineering Task Force (IETF): Develops new protocols
  - Internet Engineering Steering Group (IESG): Manages standardization process
  - Internet Corporation for Assigned Names and Numbers (ICANN): Maintains operational information

—
Internet Standards: Process:

- IETF Internet Draft: Review period for 6 months
- Request For Comment: Approved by IESG, Internet standard
- Examples:
  - RFC 821: Simple Mail Transfer Protocol (SMTP)
  - RFC 2616: Hypertext Transfer Protocol (HTTP)
  - RFC 2026: Standardization Process

—
Internet Standards: Process Flow:

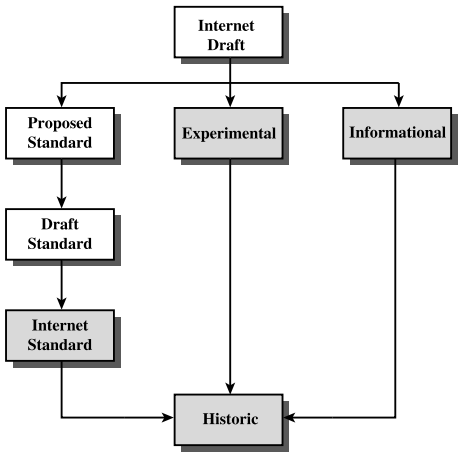


Figure 1.12 Internet RFC Publication Process

—

Physical Layer: Transmission:
(for computer scientists only. See Adv. EE for the real thing.)

- Terminology
- Data Transmission
- Problems

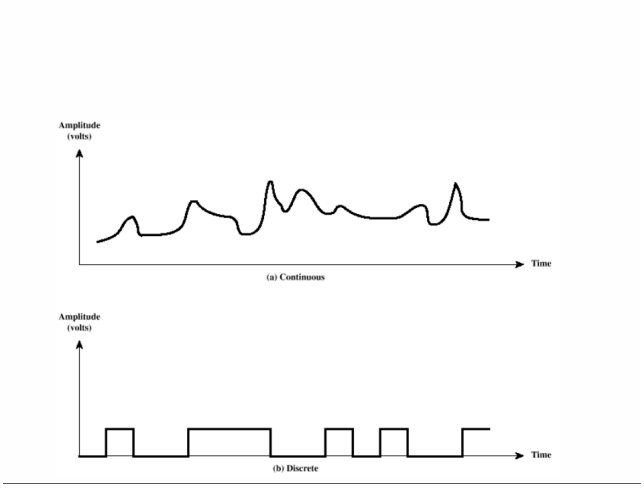
—

Terminology:

- guided media vs. unguided media
- point-to-point vs. multipoint
- simplex, half-duplex, full-duplex

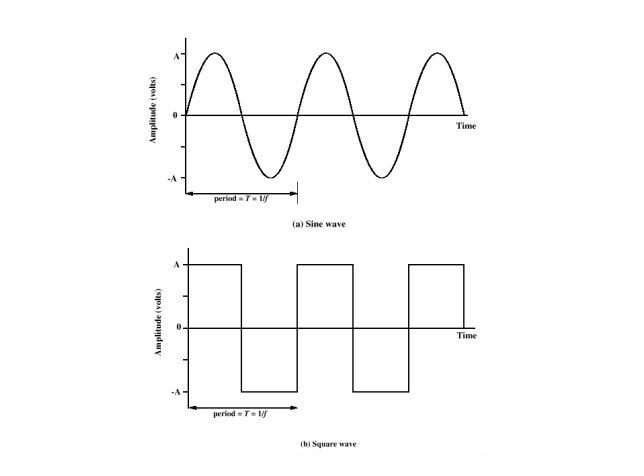
—

Discrete vs. continuous signals:



—

Periodic signals:



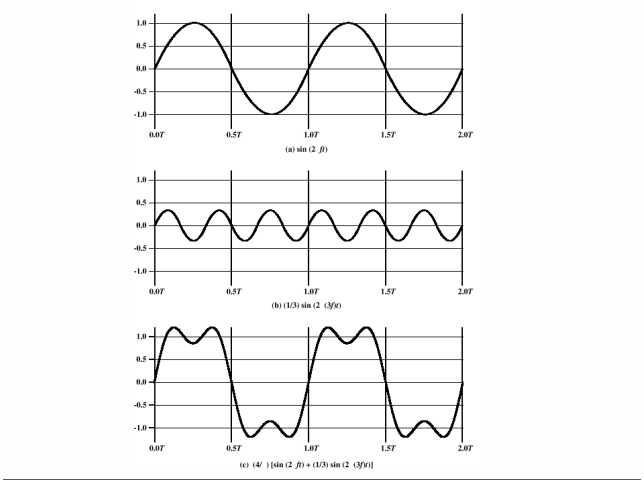
—

Frequency Domain:

- Signals can be constructed by adding signals of multiple frequencies.
- Example: $s(t) = \frac{4}{\pi}\left(\sin(2\pi f t) + \frac{1}{3}\sin(2\pi(3f)t)\right)$

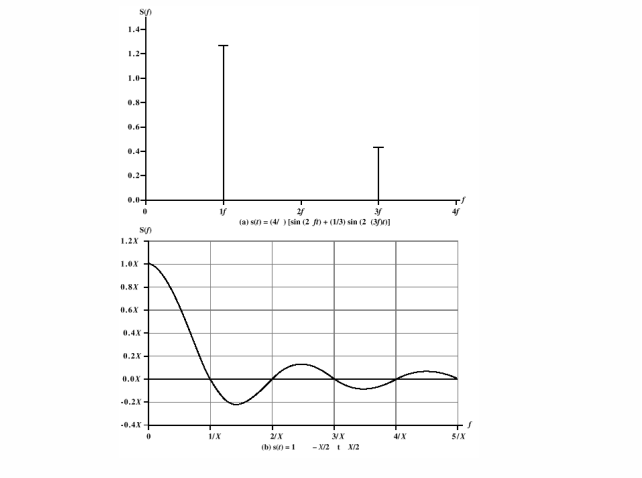
—

Addition of signals:



—

Frequency Domain Representations:



—

Terminology:

- amplitude, frequency, period, phase, wavelength
- fundamental frequency, spectrum
- absolute bandwidth, effective bandwidth
- dc component

—

Bandwidth-limited square wave:

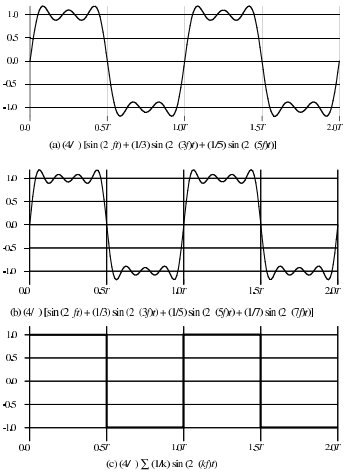


Figure 3.7 Frequency Components of Square Wave ($T = 1/f$)

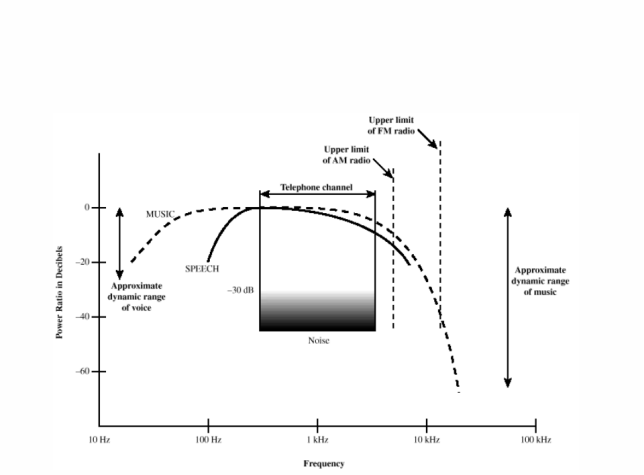
—

Data Transmission:

- Data: Thing to be transmitted
- Signal: Physical propagation
- Transmission: Sending data by signals
- Analog: Audio, Video
- Digital: text (ASCII/IRA)
- Analog over digital communication system: CODEC
- Digital over analog communication system: Modem

—

Audio Spectrum:



—

Transmission Problems:

- Attenuation
  $N = -10 \log_{10} \frac{P_{out}}{P_{in}}$
- Delay Distortion: frequency-dependent phase shift
- Noise: Thermal, intermodulation, crosstalk, impulse
- Channel capacity:
  - 4 concepts: Data rate, Bandwidth, Noise, Error rate
  - Goal: try to relate these concepts

—

Nyquist Bandwidth:

- AKA sampling theorem: With $2f$ sampling frequency, signals up to f can be reconstructed.
- Assumption: Noise-free channel
- Idea: Use the theorem "backwards": Either change signal or not $\rightsquigarrow$ two symbols per wavelength $\rightsquigarrow$ $2B$ symbols can be transmitted in B bandwidth.
- By using more than two symbols (e.g. voltages), the data rate can be furter extended: $C = 2B \log_2 M$

—

Shannon Capacity:

- Problem: Nyquist's assumption is a noise-free channel
- With noise present in the channel, "faster" transmission leads to "smaller" bits that can be "damaged" by noise more easily.
- Signal-to-Noise Ratio: $(SNR)_{db} = 10 \log_{10} \frac{\text{signal power}}{\text{noise power}}$
- Shannon: $C = B \log_2(1 + SNR)$

—

Electromagnetic Spectrum:

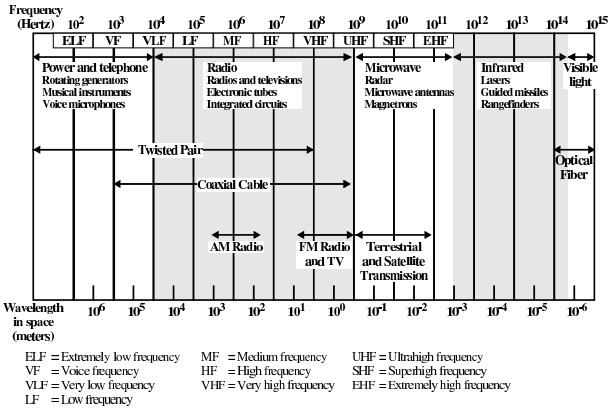


Figure 4.1 Electromagnetic Spectrum for Telecommunications

—

Transmission Media:

- Guided: Twisted-Pair cabl, Coaxial Cable, Optical Fiber
- Wireless: Terrestrial Microwave, Satellite Microwave, Broadcast, Infrared

—

Guided Media:

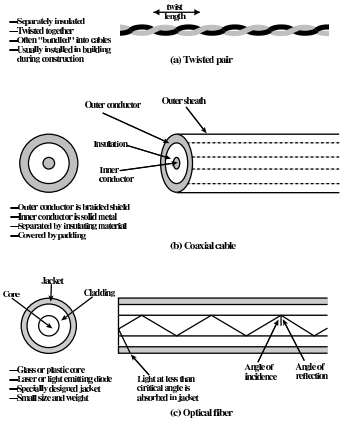


Figure 4.2 Guided Transmission Media

—

Guided Media: Bandwidth:

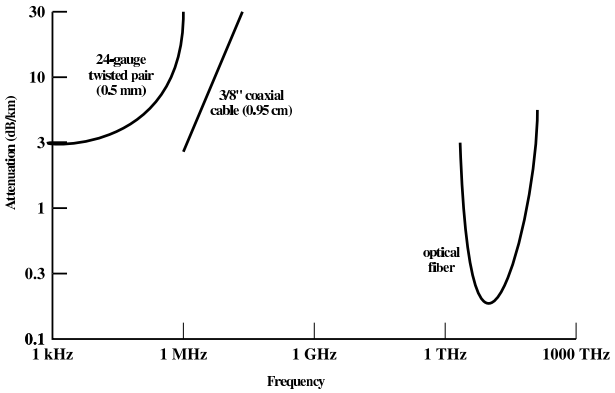


Figure 4.3 Attenuation of Typical Guided Media

—

Optical Fiber Modes:

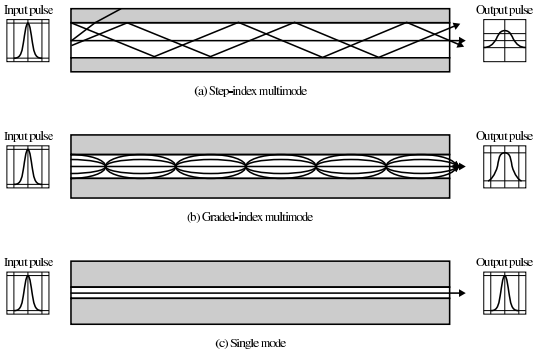


Figure 4.4 Optical Fiber Transmission Modes

—

Wireless:

- Terrestrial Microwave
  - Attenuation: $L = 10log\left(\frac{4\pi d}{\lambda}\right)^2$ dB
  - Distance Rule: $d = 7.14\sqrt{Kh}$ with $K \approx 4/3$.
- Satellite Microwave
- Broadcast Radio
- Infrared

—

Coding Schemes:

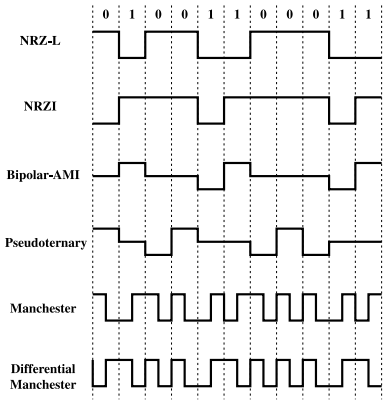


Figure 5.2 Digital Signal Encoding Formats

—

Coding Schemes: Definitions:

- NRZ-L: no voltage:0 voltage:1
- NRZI: constant on 0, invert on 1 (differential encoding)
- bipolar-AMI (alternate mark inversion): 0: no signal, 1: high or low pulse
- pseudoternary: 1: no signal, 0: high or low pulse
- Biphase: Manchester: 1: low-high, 0: high-low
- Differential Manchester: 0: with transition, 1: without transition
- Biphase coding results in higher modulation rate with similar symbol rate

—

Coding Schemes: Modulation Rate:

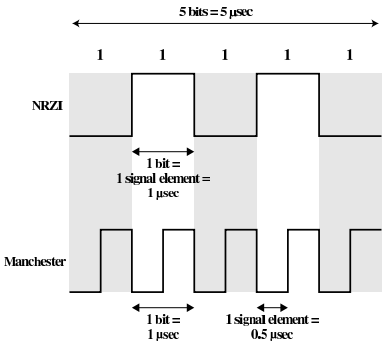


Figure 5.5 A Stream of Binary Ones at 1 Mbps

—

Coding Schemes: Terms:

- Relevant factors for communication:
  - Signal-to-noise ratio (SNR)
  - Data rate
  - Bandwidth
- An increase in data rate increases the Bit-Error-Rate (BER)
- An increase in SNR decreases the BER
- An increase in bandwidth allows an increased data rate

—

Coding Schemes: Terms:

- Coding schemes can be analyzed according to the following criteria:
  - Signal Spectrum
  - Clocking
  - Error Detection
  - Signal Interference
  - Cost and Complexity

—

Coding Schemes: Error rates:

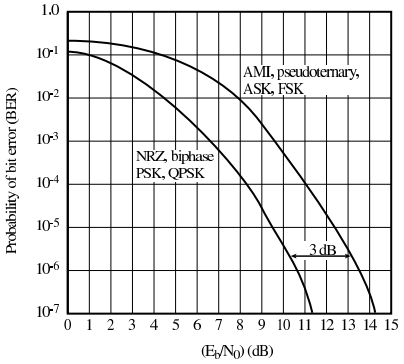


Figure 5.4 Theoretical Bit Error Rate for Various Encoding Schemes

—

Coding Schemes: Scrambling:

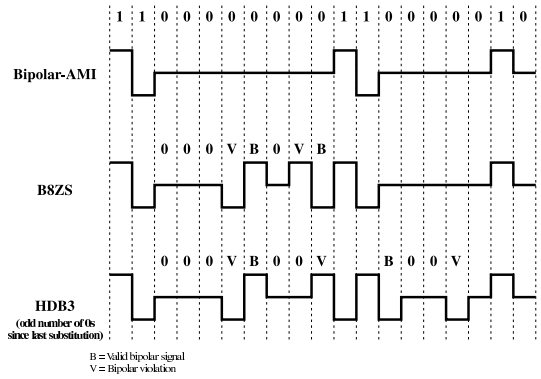


Figure 5.6 Encoding Rules for B8ZS and HDB3

—

Modulation: Digital over Analog:

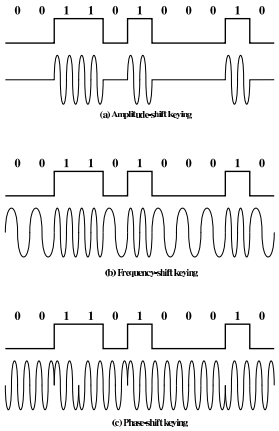


Figure 5.7 Modulation of Analog Signals for Digital Data

—

Analog over Digital:

- Pulse Code Modulation
- Delta Modulation
- Problems: Quantisation noise
- PCM Quantisation SNR: $20 \log 2^n + 1.76$dB for linear PCM
- Better: nonlinear encoding, e.g. through companding function

—

Modulation: Analog over Analog:

- Amplitude Modulation (AM)
- Angle Modulation: Frequency (FM) or Phase (PM) modulation.
- Quadrature Amplitude Modulation (QAM)
- Spread Spectrum: Frequency Hopping, Direct Sequence

—

Finally: Data Communication !:

- Synchronous, Asynchronous Transmission
- Interfaces: V.24/EIA-232-F (AKA RS-232)
- ISDN: Differential Signals

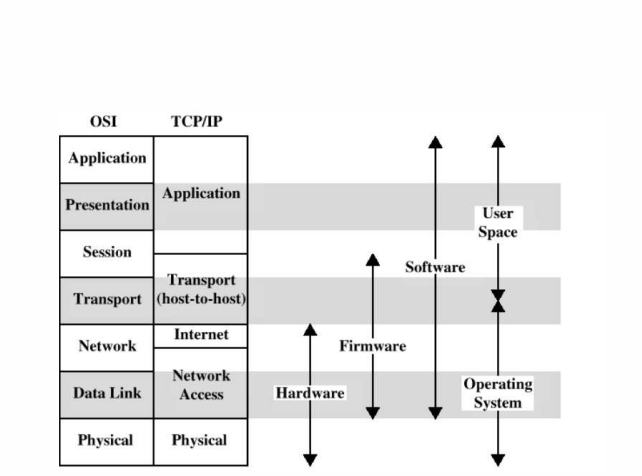
—

Finally: Data Communication !:

- Synchronous, Asynchronous Transmission
- Interfaces: V.24/EIA-232-F (AKA RS-232)
- ISDN: Synchronous, Differential Signals

—

Where are we now?:



—

Error Detection and Correction:

- Parity: add parity bit, number of 1-bits even for even parity, but only detects single-bit errors.
- Hamming Code: More parity bits, detects and corrects errors
  - m: number of data bits, p: number of redundancy bits, p+m bits to be transmitted.
  - $2^p \geq m + p + 1$, i.e. for 7-bit ASCII, 11 Bits are to be transmitted.
  - check bits are positioned at positon $1, 2, 4, \ldots, 2^n$ in the bitstream

—

Hamming Code Calculation:

- The redundancy bit at position i is calculated as the parity of all data bits that contain 2^i in their binary representation of their sequence number.
- for 7-bit ascii, the first redundancy bit contains the parity of data bits 3, 5, 9 and 11. The second redundancy bit contains the parity of the data bits 3, 6, 7 and 11 and so on.

—

Hamming Code Calculation (2) :

- If not, we receive the binary address of the faulty bit and can correct it.
- example: send 1 0 1 1 1 0 0 1 0 0 1, received 1 0 0 1 1 0 0 1 0 0 1
- c1 (positions 1 3 5 7 9 11) =1 c2 (2 3 6 7 10 11)=1 c3(4 5 6 7)=0 c4(8 9 10 11)=0 c4 c3 c2 c1 = (0 0 1 1) =3 -> bit 3 wrong.
- $\rightarrow$ The Hamming code can correct 1-bit-errors.
- Keyword: Forward-Error-Correction (FEC)

—

Error Detection and Correction:

- Arithmetic Checksum: Add all byte values (modulo 255), calculate negative of the sum, add as checksum. On the receiver side, add all including checksum.
- Drawback: does not detect sequence errors
- Cyclic Redundancy Check (CRC)

—

CRC: The idea:

- Cyclic Redundancy Check (CRC)
  - Given k-bit message, the transmitter creates a n-bit sequence (Frame Check Sequence) so that the resulting k + n bit frame is divisible by a predetermined number.
  - The receiver then divides the block by the number. If a residue of 0 is calculated, the test passes and it is assumed that there was no error.

—

CRC: The Details:

- Modulo-2-arithmetic: add and substract without carry
- $T = (k + n)$-bit frame to be transmitted, $n < k$
- $M = k$-bit-message, the first k bits of T.
- $F = n$-bit-FCS, the last n bits of T
- $P =$ pattern of $n + 1$ bits, this is the divisor
- $T = 2^n M + F$

—

CRC: The Details (2):

$$\begin{aligned}
\frac{2^n M}{P} &= Q + \frac{R}{P} \text{ use } R \text{ as FCS} \\
T &= 2^n M + R \text{ is this OK as FCS?} \\
\frac{T}{P} &= \frac{2^n M + R}{P} \\
\frac{T}{P} &= Q + \frac{R}{P} + \frac{R}{P} \\
\frac{T}{P} &= Q + \frac{R + R}{P} = Q \text{ due to mod-2 arithmetic}
\end{aligned}$$

—

CRC: What P?:

- P is one bit longer than the desired FCS and both the high-order and the low-order bit of P should be 1.
- With a good P, the CRC can detect the following:
  - singe-bit errors
  - double-bit errors as long as P has at least 3 ones.
  - Any odd-number of errors as long as P contains 11 as a factor.
  - Any burst error for which the length of the burst is less than the length of P.
  - Most larger burst errors

—

Error Control:

- Possible errors: Lost frame, damaged frame
- Error Detection

- Positive Acknowledgement
- Retransmission after timeout
- Negative acknowledgement and retransmission
- automatic repeat request (ARQ)

—

ARQs:

- Stop-and-Wait ARQ: Each frame is acknowledged
- Go-back-N ARQ: Sliding Window Flow Control. If a damaged frame is received, the frame and all subsequent frames are retransmitted.
- Selective-Reject ARQ: Only retransmit frames with negative acknowledgements

—

Data Link Control Protocols:

- HDLC (High-level data-link control protocol) (ISO 3009, ISO 4335)
- LAP-B (Link Access Procedure, Balanced) Subset of HDLC, used for X.25
- LAP-D (Link Access Procedure, D-Channel) by ITU-T as part of ISDN
- LLC (Link Level Control) Part of the IEEE 802 protocol family
- Frame Relay
- Asynchronous Transfer Mode

—

Data Link Control Protocols: HDLC:

- Widely used, basis for many other data link control protocols (LAP-B, LAP-D)
- HDLC defines:
  - Three station types: Primary, Secondary, Combined
  - Two link configurations: Balanced, Unbalanced
  - Three transfer modes: Normal response mode (NBM), Asynchronous balanced mode (ABM), Asynchronous response mode (ARM)

—

HDLC Frame Format:

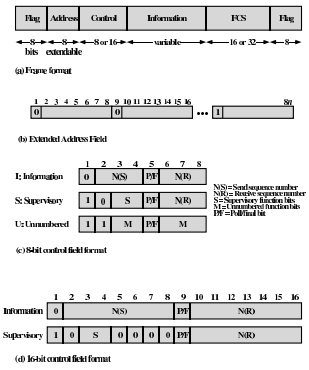
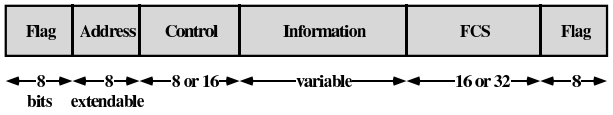


Figure 7.10 HDLC Frame Structure

—

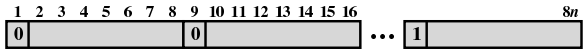
HDLC Frame Format:



(a) Frame format

—

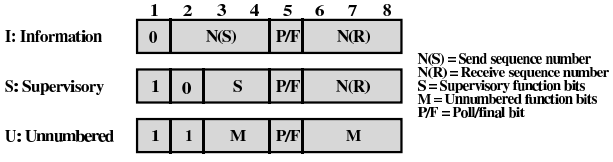
HDLC Frame Format:



(b) Extended Address Field

—

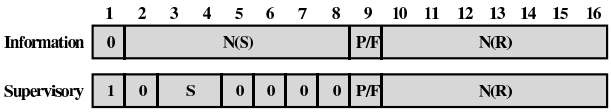
HDLC Frame Format:



(c) 8-bit control field format

—

HDLC Frame Format:



(d) 16-bit control field format

—

HDLC Bit Stuffing:

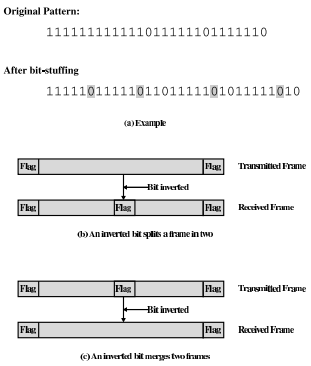


Figure 7.11 Bit Stuffing

—

HDLC Bit Stuffing:

Original Pattern:

111111111111011111101111110

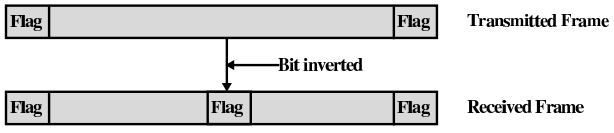
After bit-stuffing

111110111110110111110101111010

(a) Example

—

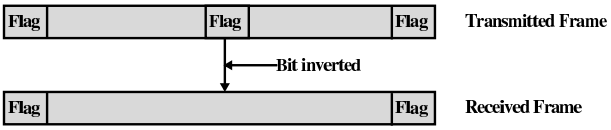
HDLC Bit Stuffing:



(b) An inverted bit splits a frame in two

—

HDLC Bit Stuffing:



(c) An inverted bit merges two frames

—

HDLC Protocol Flow:

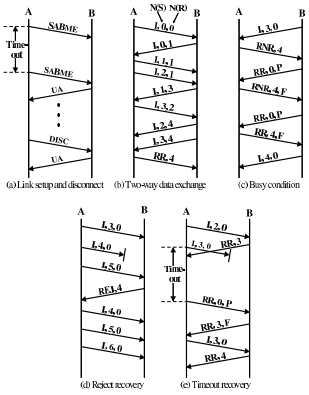
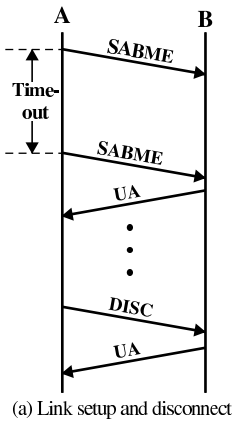


Figure 7.12 Examples of HDLC Operation

—

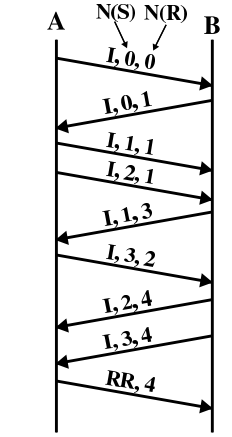
HDLC Protocol Flow:



(a) Link setup and disconnect

—

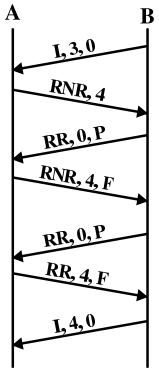
HDLC Protocol Flow:



(b) Two-way data exchange

—

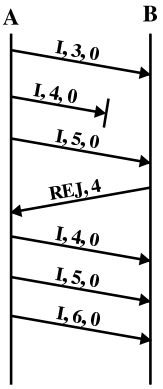
HDLC Protocol Flow:



(c) Busy condition

—

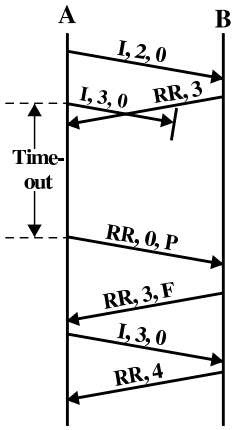
HDLC Protocol Flow:



(d) Reject recovery

—

HDLC Protocol Flow:



(e) Timeout recovery

—

Link Layer: Local Area Networks:

- Examples: Token Ring, Ethernet, FC-AL, Wireless LAN
- Applications: PC Lan, Storage Area Networks, High-performance computing
- Protocol Architecture: IEEE 802 Reference Architecture: MAC and LLC

—

IEEE 802:

- Protocol Architecture: IEEE 802 Reference Architecture
  - physical layer (as in OSI)

– medium access control (MAC) : Assemble frames, control access to communication media, disassemble frames and check for errors.
– logical link control (LLC) : Perform flow and error control, provide interface to higher layers

—

Ethernet in the protocol context:

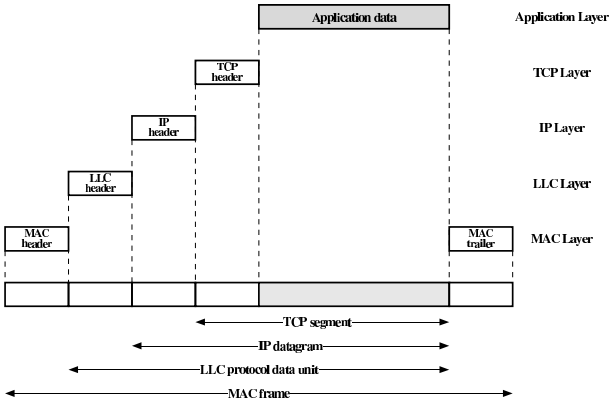


Figure 13.2 LAN Protocols in Context

—

LAN topologies:

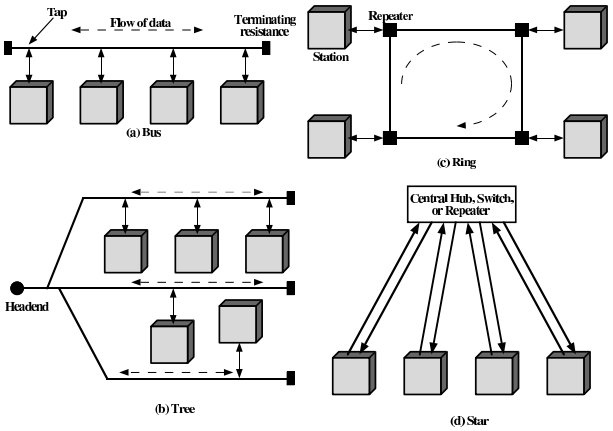
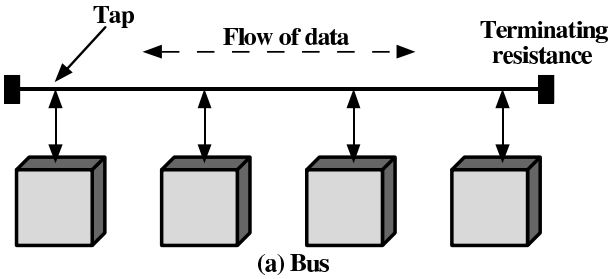


Figure 13.3 Local Area Network Topologies

—

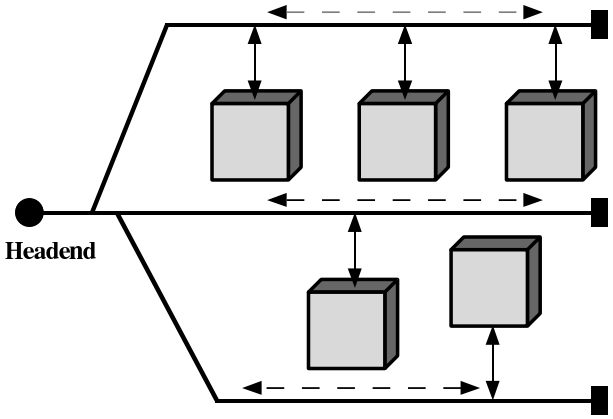
LAN topologies:



(a) Bus

—

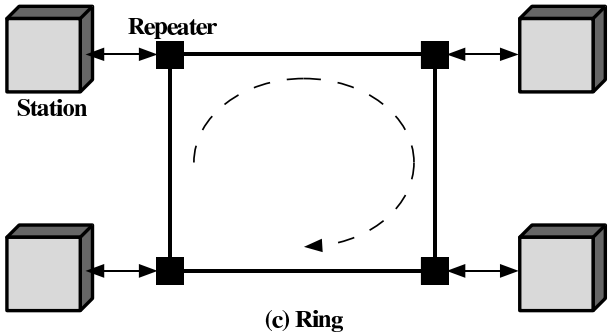
LAN topologies:



(b) Tree

—

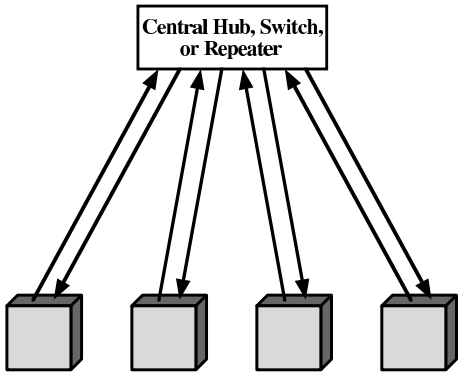
LAN topologies:



(c) Ring

—

LAN topologies:



(d) Star

—

Medium Access Control:

- Round Robin: Token Bus (IEEE 802.4), Token Ring (IEEE 802.5). Request/priority (IEEE802.12)
- Reservation: rarely implemented in LANs (IEEE 802.6: DQDB)
- Contention: CSMA/CD (IEEE 802.3), CSMA(/CA) (IEEE 802.11)

—

Ethernet:

- Standards: IEEE 802
- Ethernet on cables: IEEE 802.3
- different standards/cables/speeds/connectors...
- 10BASE5: Yellow Cable, 10 mbps (old)
- 1000Base-SX: Multimode Glassfiber, 1000 Mbps, 770-860 nm wavelength (IUB Backbone)

—

Ethernet Frames:

- Preamble and SFD
- Destination Address: 48 Bit
- Source Address: 48 Bit
- Length/Type Code: 16 Bit
- LLC Data
- Ethernet Checksum: 32 Bits
- length: 64 to 1518 bytes (excluding SFD and preamble)

—

Ethernet Frame:

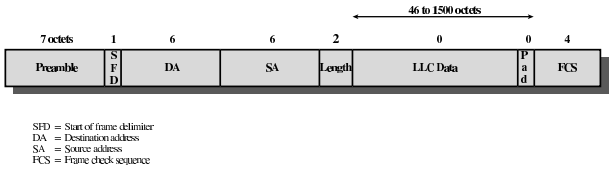


Figure 14.2 IEEE 802.3 Frame Format

—

Other things to know about Ethernet:

- Special Address: ff:ff:ff:ff:ff:ff (Broadcast address)
- CSMA/CD (Carrier Sensed Multiple Access / Collision Detect)
- Length Limits depending on cable types/ethernet standard
- Number of stations sometimes limited (in bus connections)
- Ethernet Repeaters connect segments
- Ethernet Bridges connect collision domains

—

CSMA/CD:

1. If the medium is idle, transmit
2. If the medium is busy, wait for the medium to become idle, then transmit immediately
3. Listen while transmitting. If a colision is detected, transmit a brief jamming signal (so that everybody notices the collision) and then stop.
4. Wait a random amount of time, then try again from step 1

—

CSMA/CD Operation:

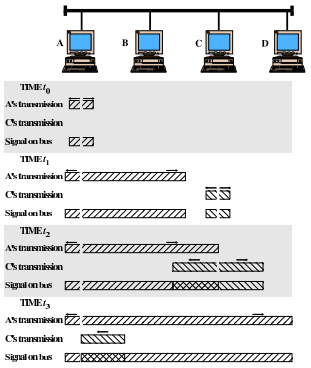


Figure 14.1 CSMA/CD Operation

—

Token Ring (IEEE 802.5):

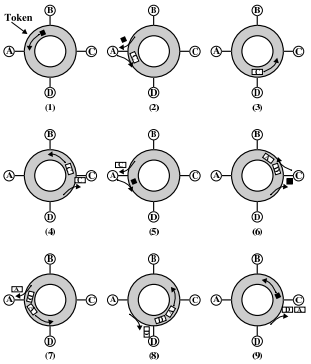
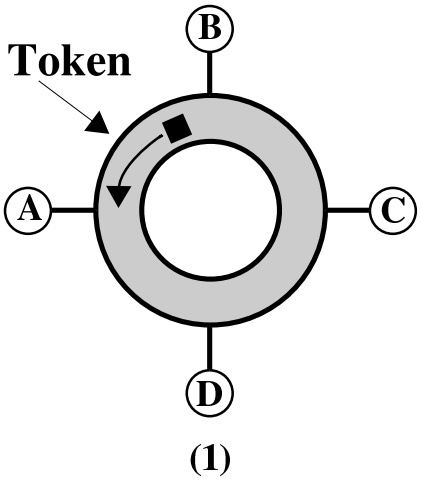


Figure 14.5 Token Ring Operation

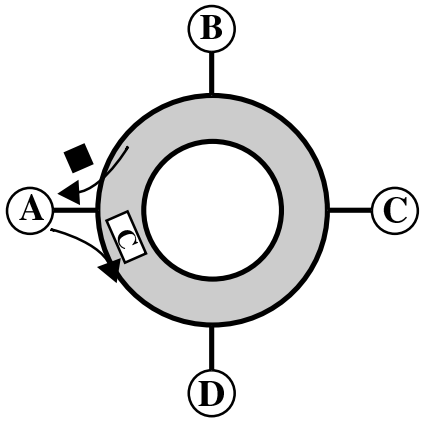
—

Token Ring (IEEE 802.5):



—

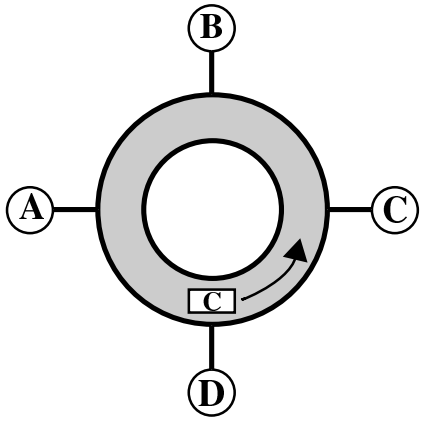
Token Ring (IEEE 802.5):



(2)

—

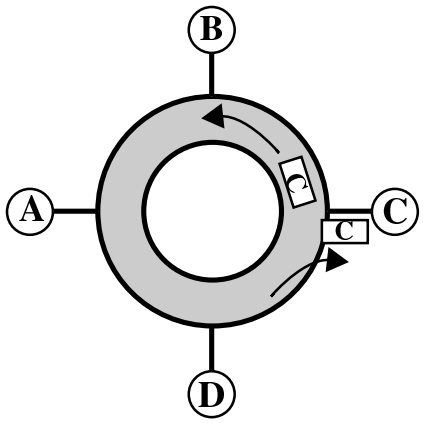
Token Ring (IEEE 802.5):



(3)

—

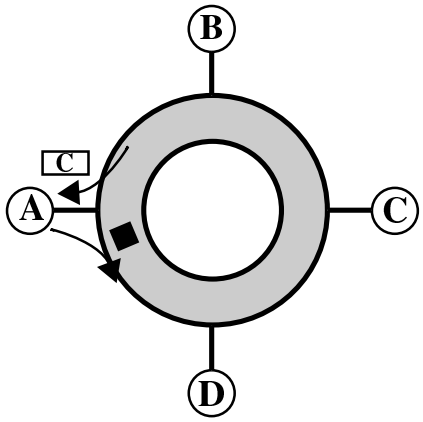
Token Ring (IEEE 802.5):



(4)

—

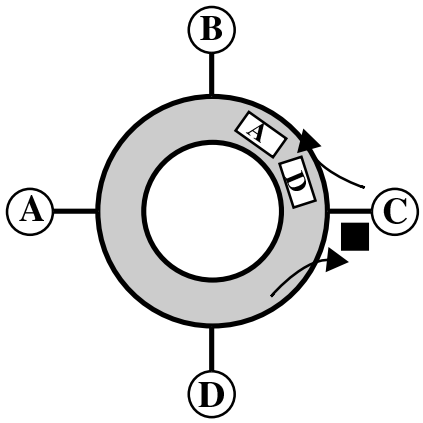
Token Ring (IEEE 802.5):



(5)

—

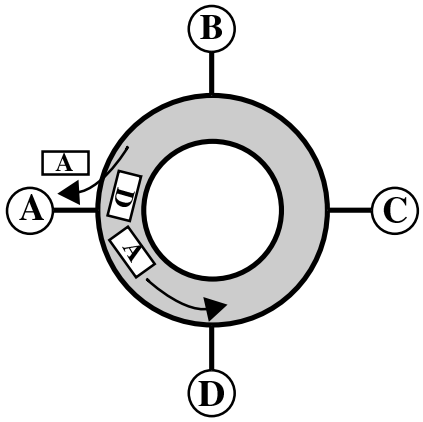
Token Ring (IEEE 802.5):



(6)

—

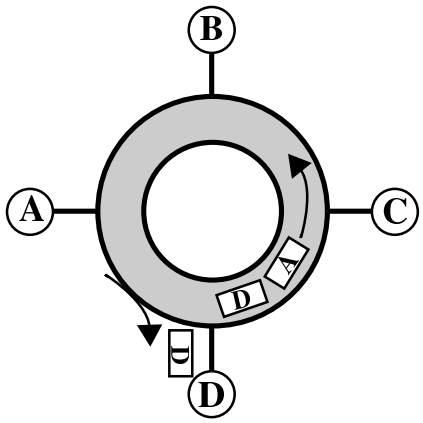
Token Ring (IEEE 802.5):



(7)

—

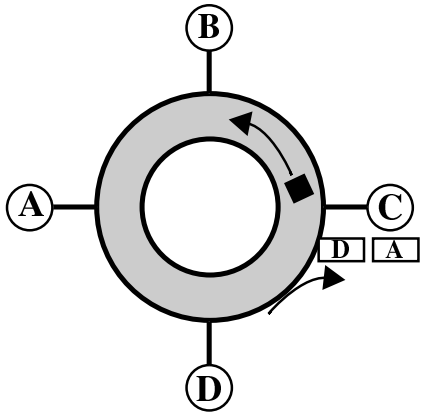
Token Ring (IEEE 802.5):



(8)

—

Token Ring (IEEE 802.5):



(9)

—

Token Ring Frame Format:

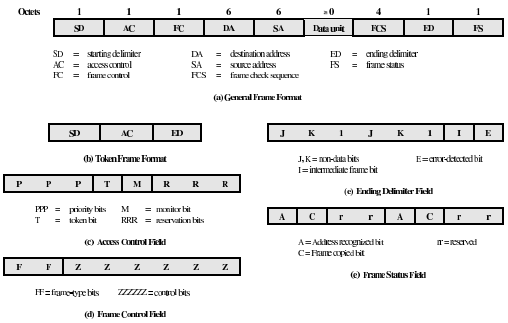


Figure 14.6 IEEE 802.5 Frame Format

—

Bridges:

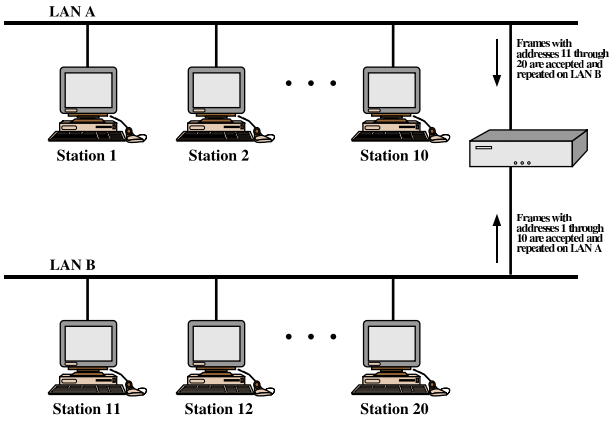


Figure 13.14 Bridge Operation

—

Bridges: Internal:

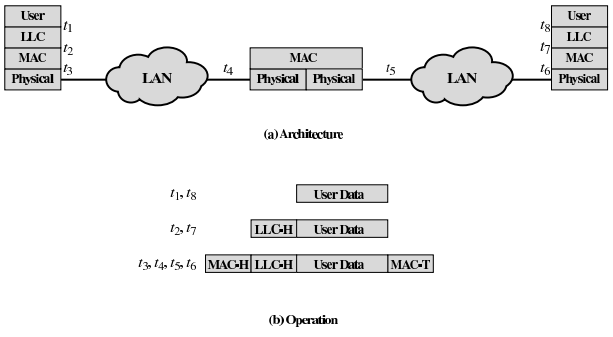


Figure 13.15 Connection of Two LANs by a Bridge

—

System of Bridges:

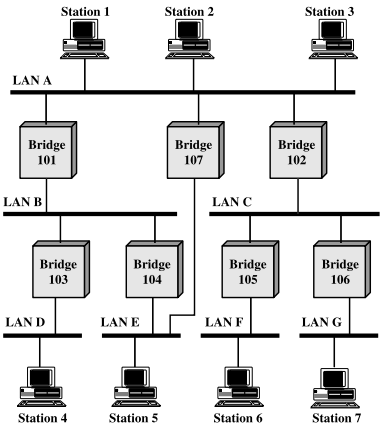


Figure 13.16 Configuration of Bridges and LANs, with Alternate Routes

—

Loop of Bridges:

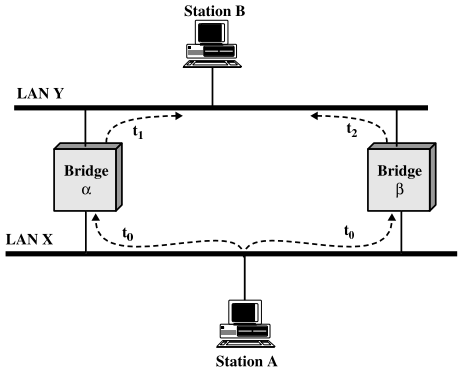


Figure 13.17 Loop of Bridges

—

TCP/IP:

- IP: Internet Protocol
- TCP: Transmission Control Protocol
- Many other protocols: ICMP,IGMP,OSPF,RSVP,FTP,BGP,UDP,ARP,SMTP,MIME, SNMP,HTTP,Sun-RPC,POP-3,IMAP,XNTP,. . .
- Standardization by IETF RFCs

- IP: Two Versions IPv4, IPv6
- IPv4: currently used, IPv6 about to be introduced

—

TCP/IP Protocol Context:

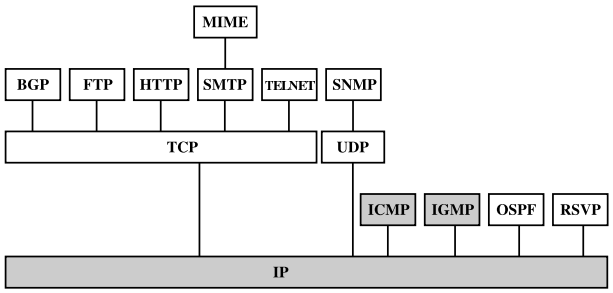
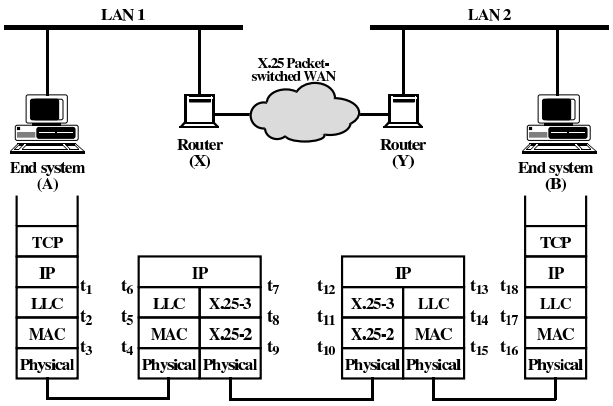


Figure 15.1 Internetworking Protocols in Context

—

TCP/IP Operation:



—

Internetworking design issues:

- Routing
- Datagram Lifetime
- Fragmentation and Reassembly
- Error Control

- Flow Control

—

IP V4 Header:

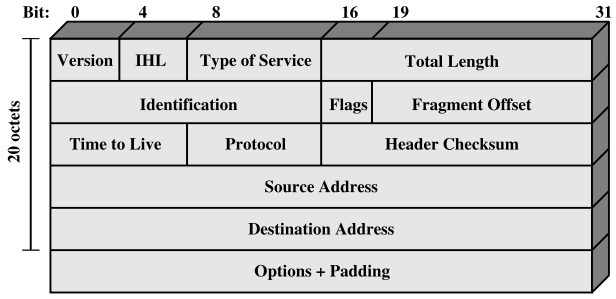


Figure 15.6 IPv4 Header

—

IP V4 Addressing:

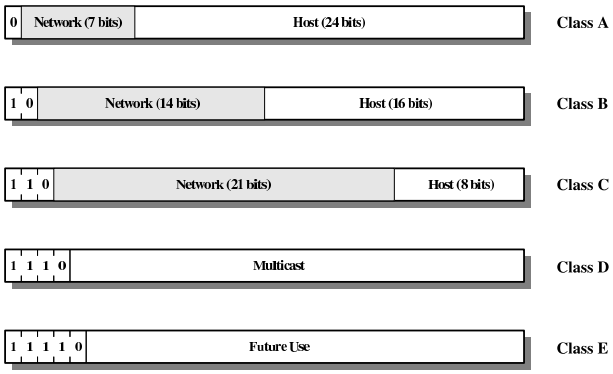


Figure 15.7 IP Address Formats

—

Subnetting:

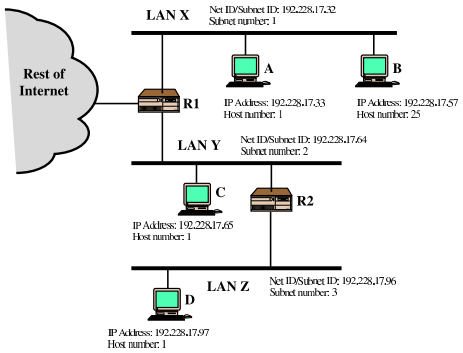


Figure 15.8 Example of Subnetworking

—

ICMP:

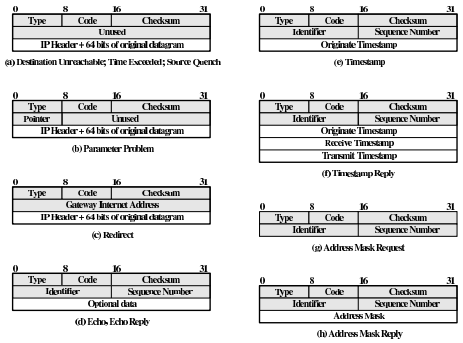


Figure 15.9 ICMP Message Formats

—

IPv6:

- 128 Bit addresses ($= 6 \cdot 10^{23}$ unique addresses per m^2 of the earth's surface !)
- Improved Option Mechanism: optional headers
- Address autoconfiguration
- Increased address flexibility
- Ressource Allocation

—

IPv6 Header Structure:

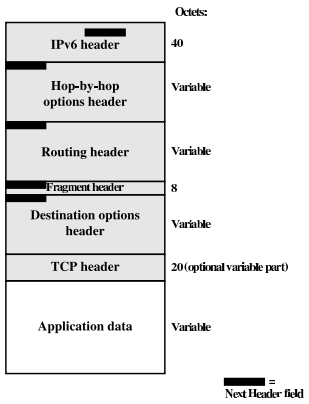


Figure 15.10 IPv6 Packet with Extension Headers (containing a TCP Segment)

—

IPv6 header:

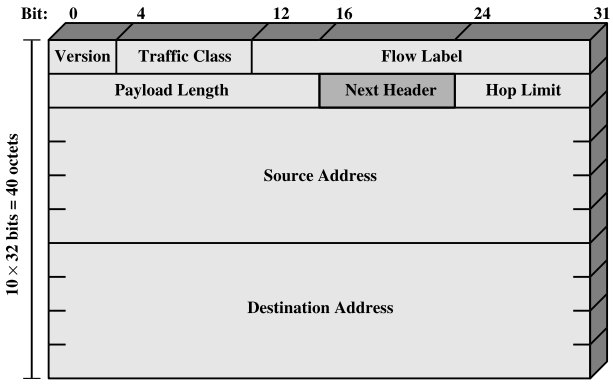


Figure 15.11 IPv6 Header

—

IPv6 optional headers:

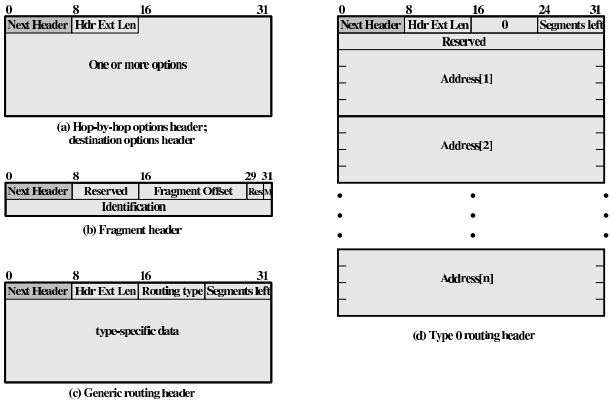


Figure 15.12 IPv6 Extension Headers

—

Multicasting:

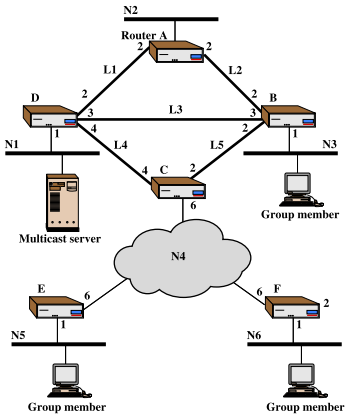


Figure 15.13 Example Configuration

—

Multicasting:

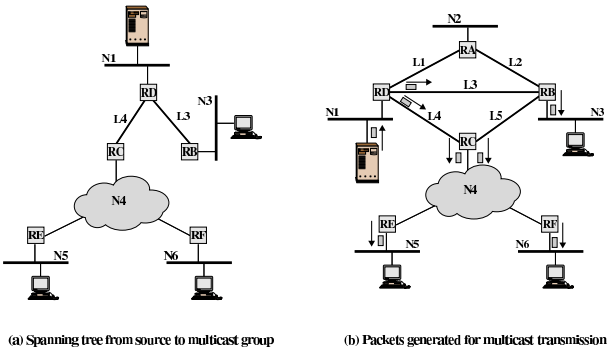


Figure 15.14 Multicast Transmission Example

—

Multicasting:

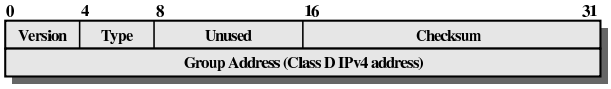


Figure 15.16 IGMP Message Format

—

Routing:Characteristics:

- Correctness
- Simplicity
- Robustness: against failures of links, overloads
- Stability: predictable behaviour
- Fairness: every packet (or station) is equal
- Optimality: e.g. maximise throughput
- Efficiency

—

Routing:Strategies:

- Fixed Routing
- Flodding
- Random Routing
- Adaptive Routing

—

Routing:ARPANET:

- First Generation (1969): Distributed adaptive routing
  - uses estimated delay (outgoing queue length) as performance criterion
  - distributed Bellman-Ford: Every node maintains one matrix row.
- Second Generation (1979): measured delay
  - uses measured delay (and link data rate) as performance criterion
  - Djikstra's Algorithm: Every node maintains full matrix row.
- Third Generation (1987): new link costs function
- smoothed estimated link ultilization as performance criterion

—

Routing: Protocols:

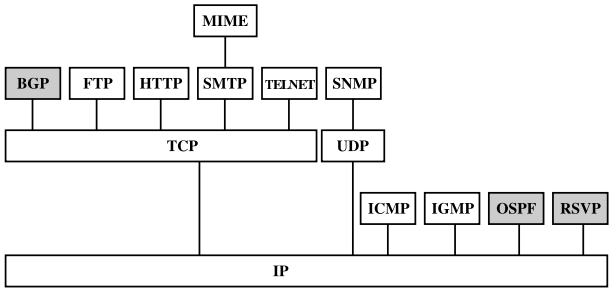


Figure 16.1 Internetworking Protocols in Context

—

Routing: Autonomous Systems(AS):

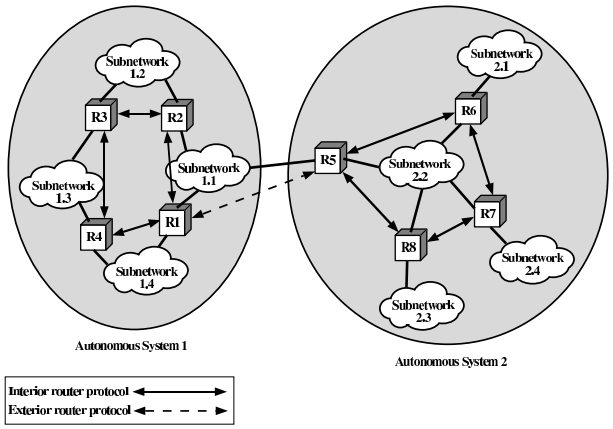


Figure 16.2 Application of Exterior and Interior Routing Protocols

—

Routing: Protocols:

- Border Gateway Protocol (BGP): RFC1771 (BGP-4) exterior router protocol for AS
- Open Shortest Path First (OSPF): RFC2328 interior router protocol for AS, successor of Routing Information Protocol (RIP)
- Integrated Service Architecture (ISA): RFC1633
- Resource Reservation (RSVP): RFC2205 for multicast applications

—

Routing: BGP:

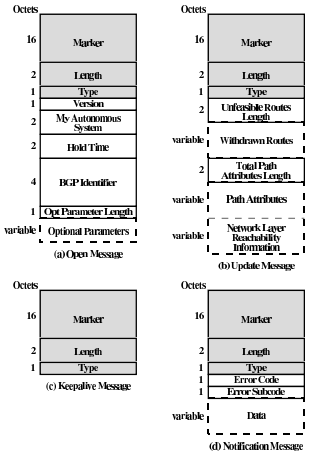


Figure 16.3 BGP Message Formats

—

Routing: OSPF example AS:

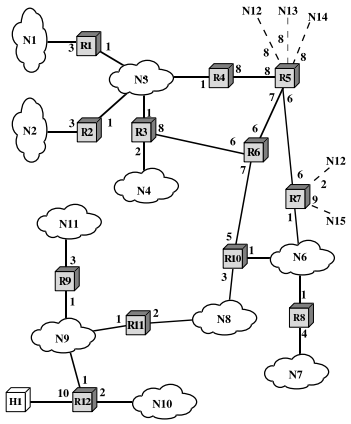


Figure 16.4 A Sample Autonomous System

—

Routing: OSPF graph for AS:

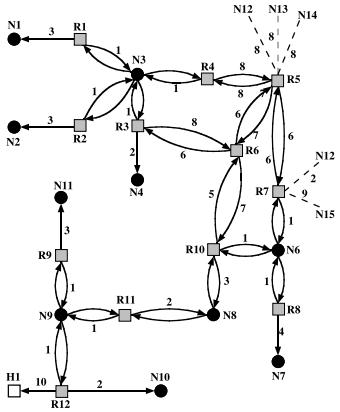


Figure 16.5 Directed Graph of Autonomous System of Figure 16.4

—

Routing: SPF tree:

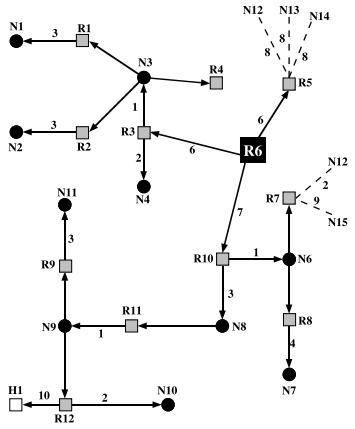


Figure 16.6 The SPF Tree for Router R6

—

IP over Ethernet: SNAP:

- SNAP: Sub Net Access Protocol is used to transmit IP over a network that already knows addresses, i.e. a sub network.
- ARP (Address Resolution Protocol) is the protocol that identifies a subnet station address for a given IP address.
- RARP (Reverse ARP) is a protocol that identifies an IP address for a given ethernet address and was one of the predecessors of DHCP (Dynamic Host Contfiguration Protocol).

—

IP over Ethernet: SNAP:

- ARP uses broadcast packets to find the ethernet address for a given IP address by “asking around”.
- In order to avoid too many ARP broadcasts, all stations use an ARP cache.

—

Security risks of ARP:

- Proxy-ARP: One station can handle traffic for another station
- ARP-Cache-Poisoning: As ARP is implemented stateless and is answered by directed packets, answers can be given before the question was even asked.

—

TCP and UDP:

- TCP and UDP use IP to transmit data
- UDP is the simpler protocol: Unreliable Datagramm Protocol
  - The UDP header just contains four fields: Source Port, Destination Port, Length and Checksum. Every field is 16 bit wide.
  - UDP adds a SAP facility to IP: Every SAP is identified with a port number.
  - UDP does not provide flow control, ARQ, in-order reception, duplicate detection or any guarantees.

—

TCP :

- TCP (Transmission Control Protocol) offers reliable connection-oriented service.
- Like UDP, it adds a SAP facility using 16 bit Port numbers.
- Unlike UDP, TCP offers:
  - Connection Establishment and Termination
  - Flow Control
  - ARQ
  - Ordered Delivery
  - Duplicate Detection

—

TCP Header:

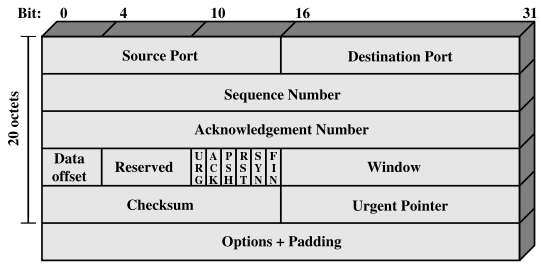


Figure 17.11 TCP Header

—

Transport Flow Control:

- Problem: On the transport layer, flow control is harder to implement than on the link layer as not only the destination may overflow but every link segment inbetween. Possible strategies (for the receiving transport entity) are:
  - Do nothing. Packets will be lost, ARQ will do resends, traffic will slow down.

– Refuse to accept packets. This is the propagated back to the link layer which will use its means of flow control.

—

Transport Flow Control:

- Use a fixed sliding-window protocol and provide buffers for all possible sequence numbers. Only send acknowledgement if a slot is empty.
- Use a credit scheme, i.e. decouple the acknowledgement from the send credit.

—

Credit Scheme:

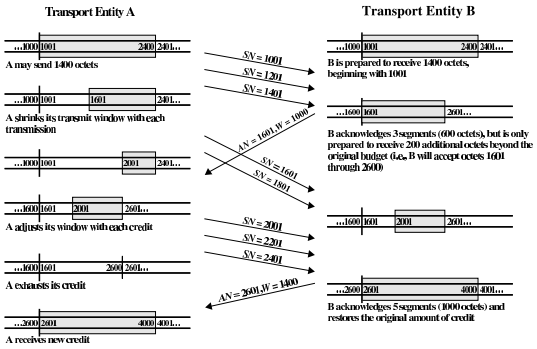


Figure 17.2 Example of TCP Credit Allocation Mechanism

—

Credit Scheme:

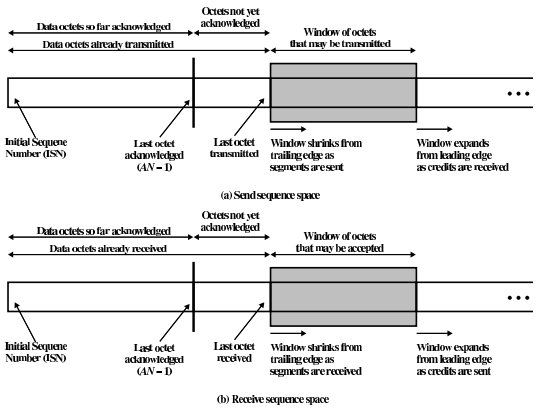


Figure 17.3 Sending and Receiving Flow Control Perspectives

—

Connection Establishment:

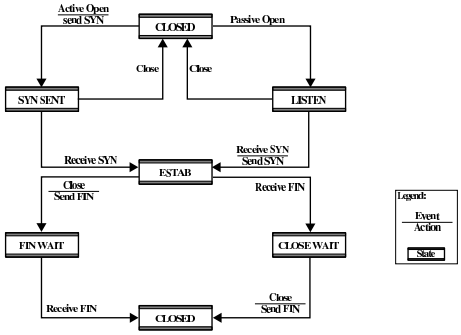


Figure 17.4 Simple Connection State Diagram

—

Possible Confusion?:

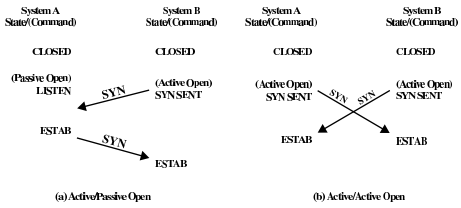


Figure 17.5 Connection Establishment Scenarios

—

TCP state diagram:

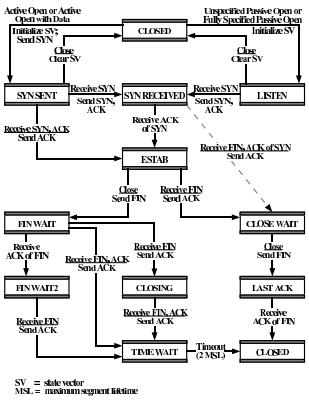


Figure 17.9 TCP Entity State Diagram

—

TCP Protocol step by step:

- Connection Establishment
- ARQ
- Flow Control
- Push
- urgent data
- Disconnection

—

Protocols using TCP:

- daytime
- test protocols: chargen, echo, discard
- SMTP
- POP3
- HTTP
- FTP, Telnet
- SSH

—