INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

COMPUTER NETWORKS

Prepared by:

Dr. Y Mohana Roopa

Mr. P Ravinder

Ms. N M Deepika

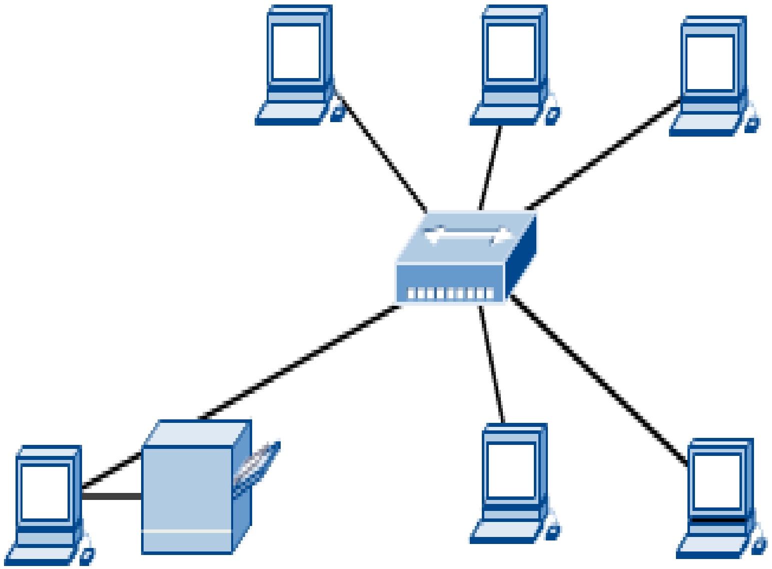
Ms. B Jaya Vijaya

UNIT-1

INTRODUCTION TO COMPUTER NETWORKS

Computer Networks

- **Computer network connects two or more autonomous computers.**



- **The computers can be geographically located anywhere.**

LAN, MAN & WAN

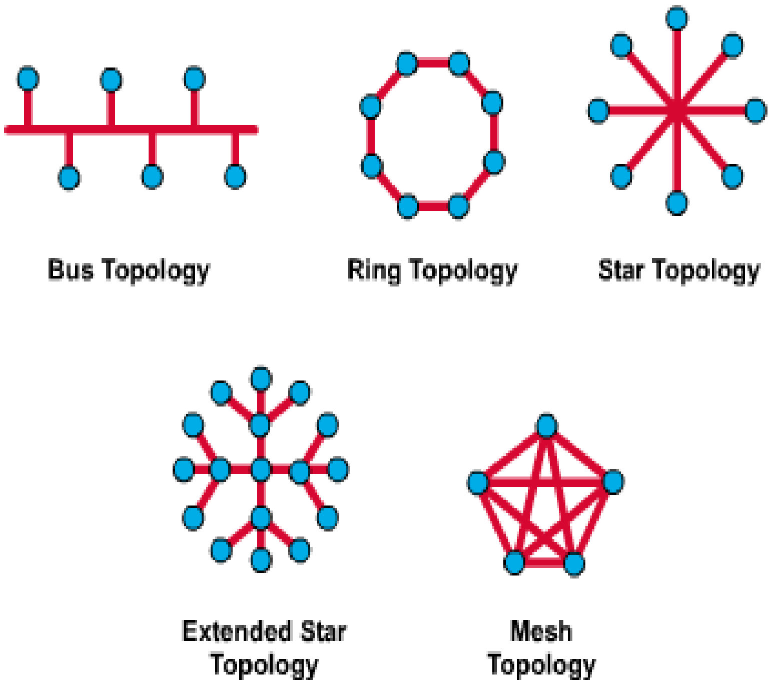
- Network in small geographical Area (Room, Building or a Campus) is called LAN (Local Area Network)
- Network in a City is call MAN (Metropolitan Area Network)
- Network spread geographically (Country or across Globe) is called WAN (Wide Area Network)

Applications of Networks

- **Resource Sharing**
  - Hardware (computing resources, disks, printers)
  - Software (application software)
- **Information Sharing**
  - Easy accessibility from anywhere (files, databases)
  - Search Capability (WWW)
- **Communication**
  - Email
  - Message broadcast
- **Remote computing**
- **Distributed processing (GRID Computing)**

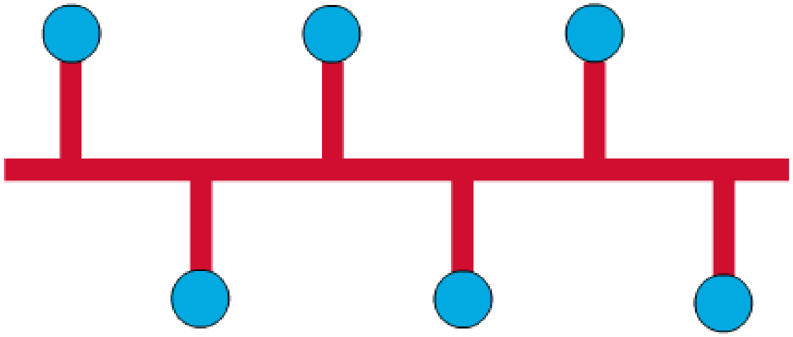
Network Topology

- **The network topology defines the way in which computers, printers, and other devices are connected. A network topology describes the layout of the wire and devices as well as the paths used by data transmissions.**



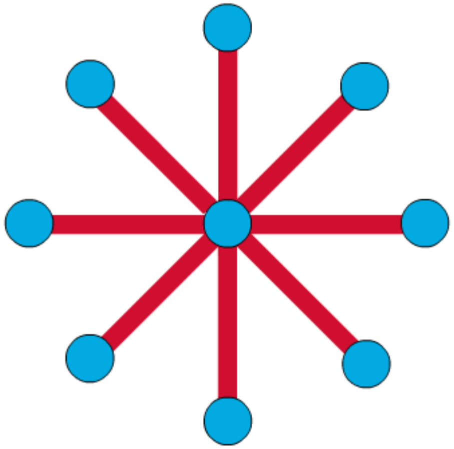
Bus Topology

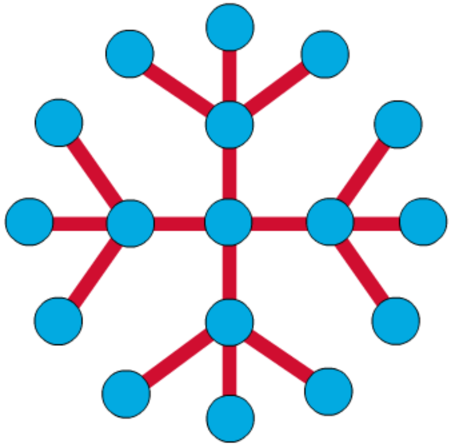
- Commonly referred to as a linear bus, all the devices on a bus topology are connected by one single cable.



Star & Tree Topology

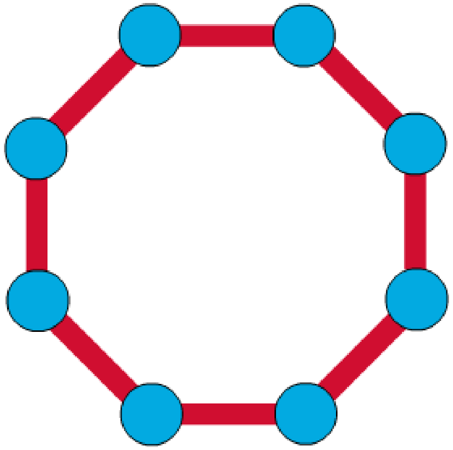
- The star topology is the most commonly used architecture in Ethernet LANs.
- When installed, the star topology resembles spokes in a bicycle wheel.
- Larger networks use the extended star topology also called tree topology. When used with network devices that filter frames or packets, like bridges, switches, and routers, this topology significantly reduces the traffic on the wires by sending packets only to the wires of the destination host.

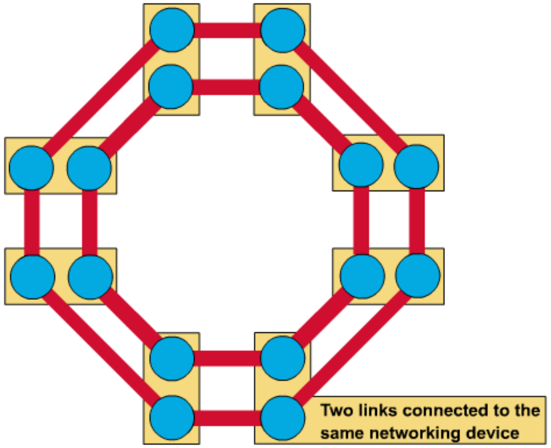




Ring Topology

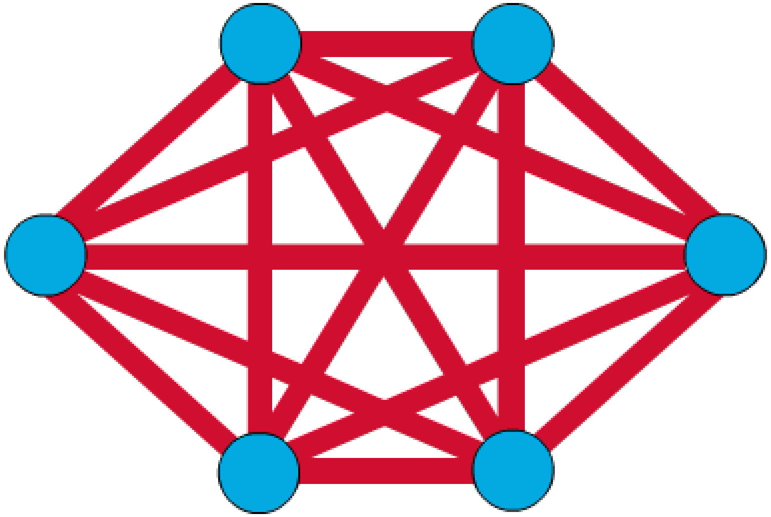
- **A frame travels around the ring, stopping at each node. If a node wants to transmit data, it adds the data as well as the destination address to the frame.**
- **The frame then continues around the ring until it finds the destination node, which takes the data out of the frame.**
  - Single ring – All the devices on the network share a single cable
  - Dual ring – The dual ring topology allows data to be sent in both directions.





Mesh Topology

- The mesh topology connects all devices (nodes) to each other for redundancy and fault tolerance.
- It is used in WANs to interconnect LANs and for mission critical networks like those used by banks and financial institutions.
- Implementing the mesh topology is expensive and difficult.

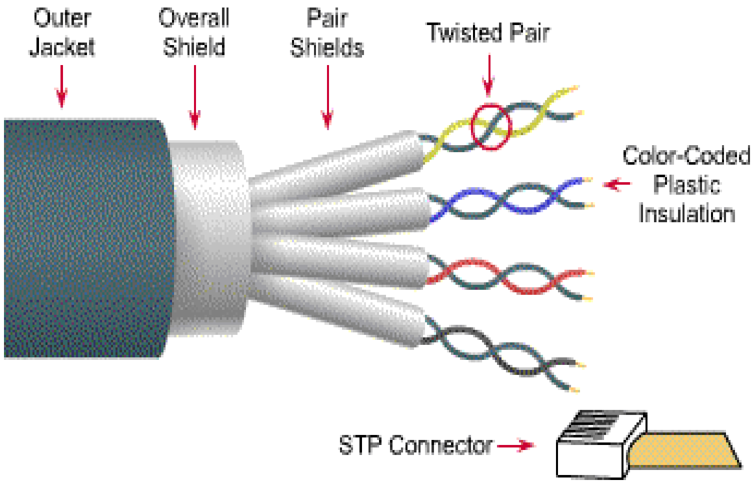


Network Components

- Physical Media
- Interconnecting Devices
- Computers
- Networking Software
- Applications

Networking Media

- **Networking media can be defined simply as the means by which signals (data) are sent from one computer to another (either by cable or wireless means).**



- Speed and throughput: 10-100 Mbps
- Cost per node: Moderately expensive
- Media and connector size: Medium to Large
- Maximum cable length: 100m (short)

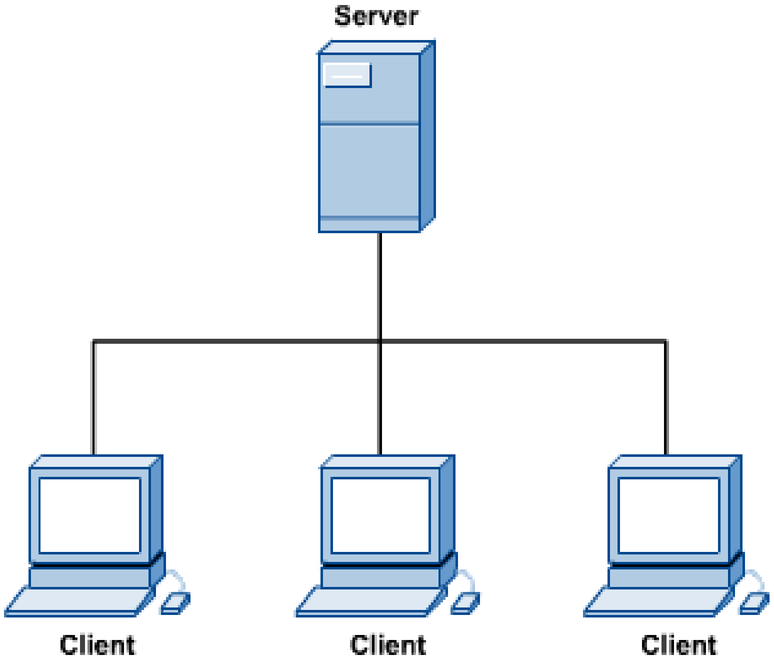
Networking Devices

- **HUB, Switches, Routers, Wireless Access Points, Modems etc.**

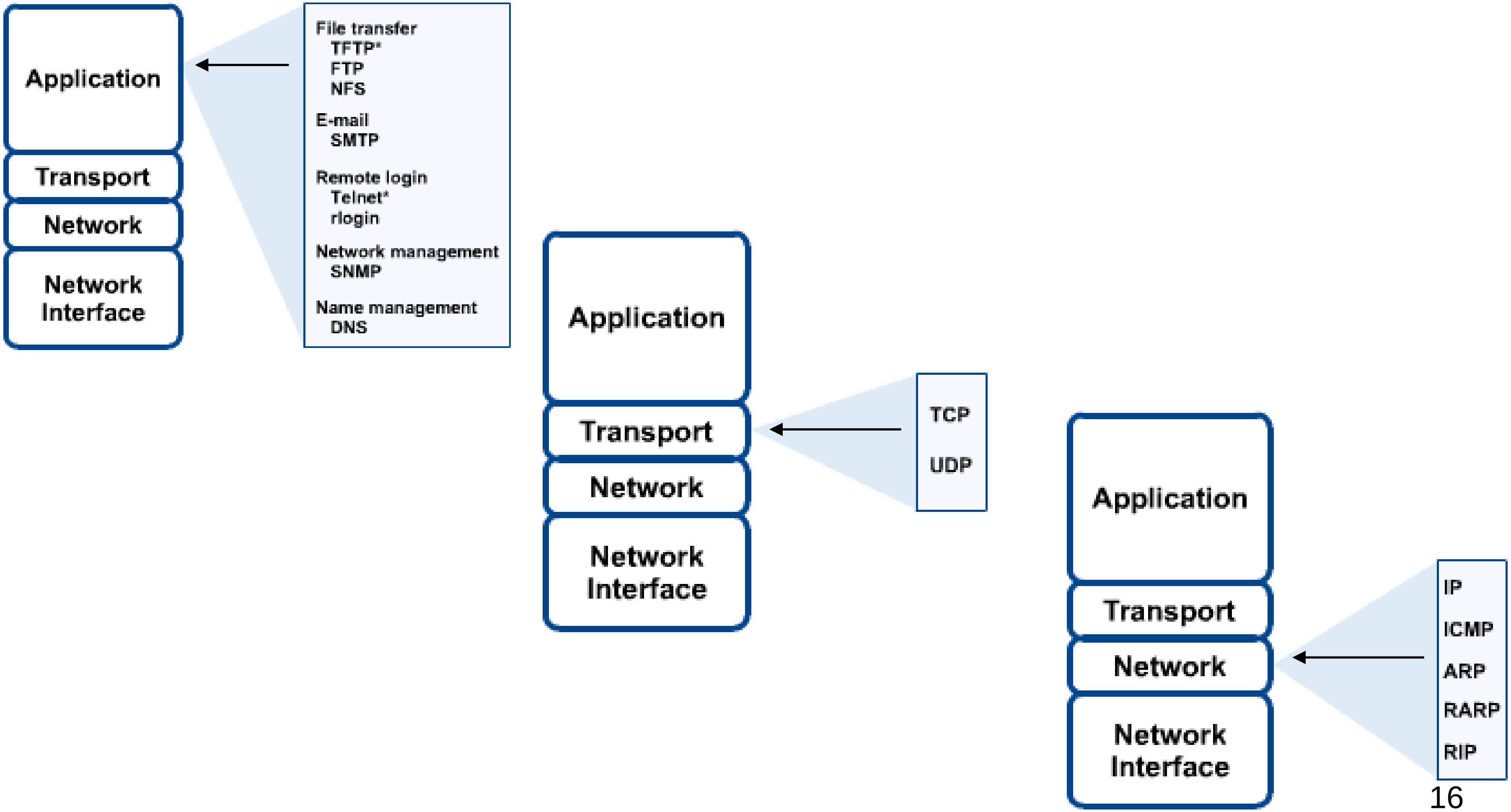
Computers: Clients and Servers

- In a client/server network arrangement, network services are located in a dedicated computer whose only function is to respond to the requests of clients.

- The server contains the file, print, application, security, and other services in a central computer that is continuously available to respond to client requests.



Networking Protocol: TCP/IP



Application

Transport

Network

Network Interface

File transfer
- TFTP*
- FTP
- NFS

E-mail
- SMTP

Remote login
- Telnet*
- rlogin

Network management
- SNMP

Name management
- DNS

TCP

UDP

IP

ICMP

ARP

RARP

RIP

Applications

- E-mail
- Searchable Data (Web Sites)
- E-Commerce
- News Groups
- Internet Telephony (VoIP)
- Video Conferencing
- Chat Groups
- Instant Messengers
- Internet Radio

Networking

Computer network

A collection of computing devices connected in order to communicate and share resources

Connections between computing devices can be physical using wires or cables or wireless using radio waves or infrared signals

Can you name some of the devices in a computer network?

Networking

Node (host)

Any device on a network

Data transfer rate (bandwidth)

The speed with which data is moved from one place to another on a network

Why is bandwidth so key?

Networking

Computer networks have opened up an entire frontier in the world of computing called the **client/server model**

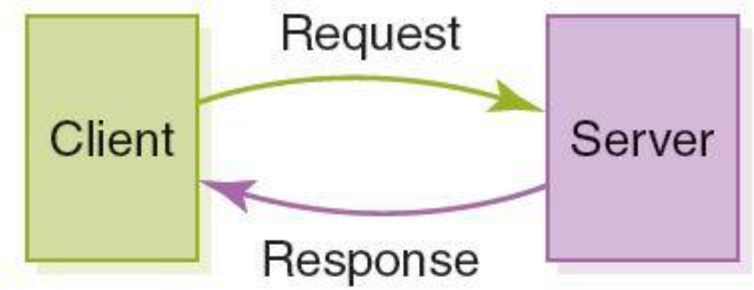


FIGURE 15.1 Client/server interaction

Networking

Protocol

A set of rules that defines how data is formatted and processed on a network

File server

A computer dedicated to storing and managing files for network users

Web server

A computer dedicated to responding to requests for web pages

P2P model

A decentralized approach that shares resources and responsibilities among many “peer” computers

Types of Networks

Local-area network (LAN)

A network that connects a relatively small number of machines in a relatively close geographical area

- **Ring topology** connects all nodes in a closed loop on which messages travel in one direction
- **Star topology** centers around one node to which all others are connected and through which all messages are sent
- **Bus topology** nodes are connected to a single communication line that carries messages in both directions

Types of Networks

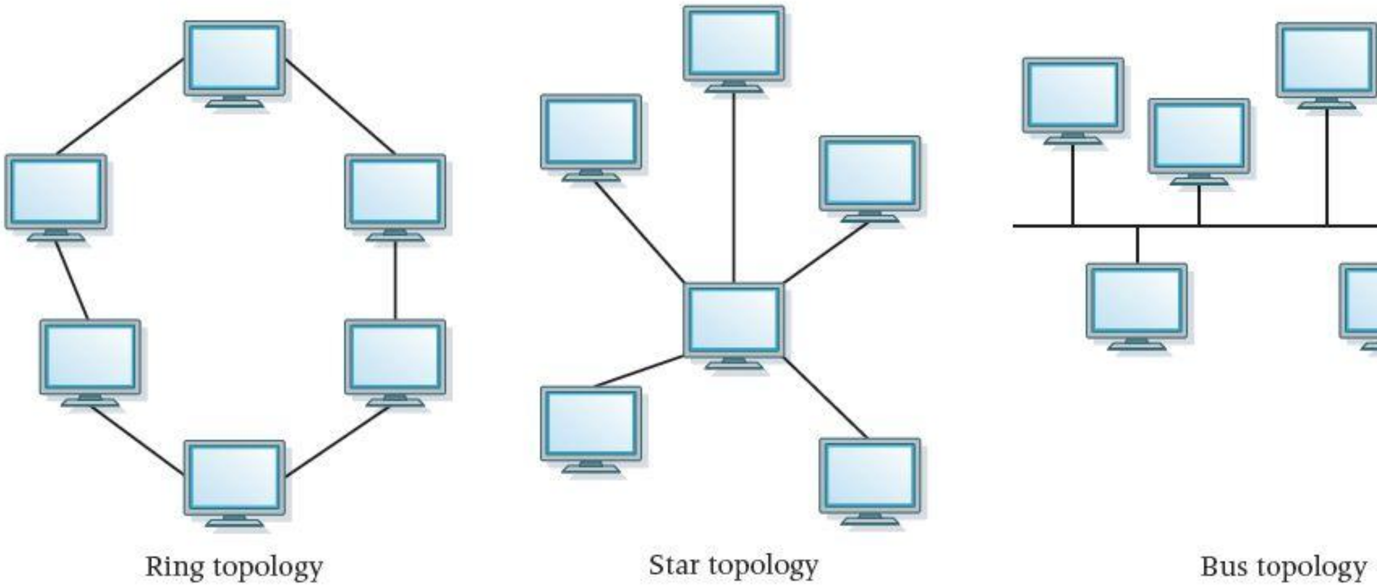


FIGURE 15.2 Network topologies

Ethernet

The industry standard bus technology for local-area networks

Types of Networks

Wide-area network (WAN)

A network that connects local-area networks over a potentially large geographic distance

Metropolitan-area network (MAN)

The communication infrastructures that have been developed in and around large cities

Gateway

One particular set up to handle all communication going between that LAN and other networks

Types of Networks

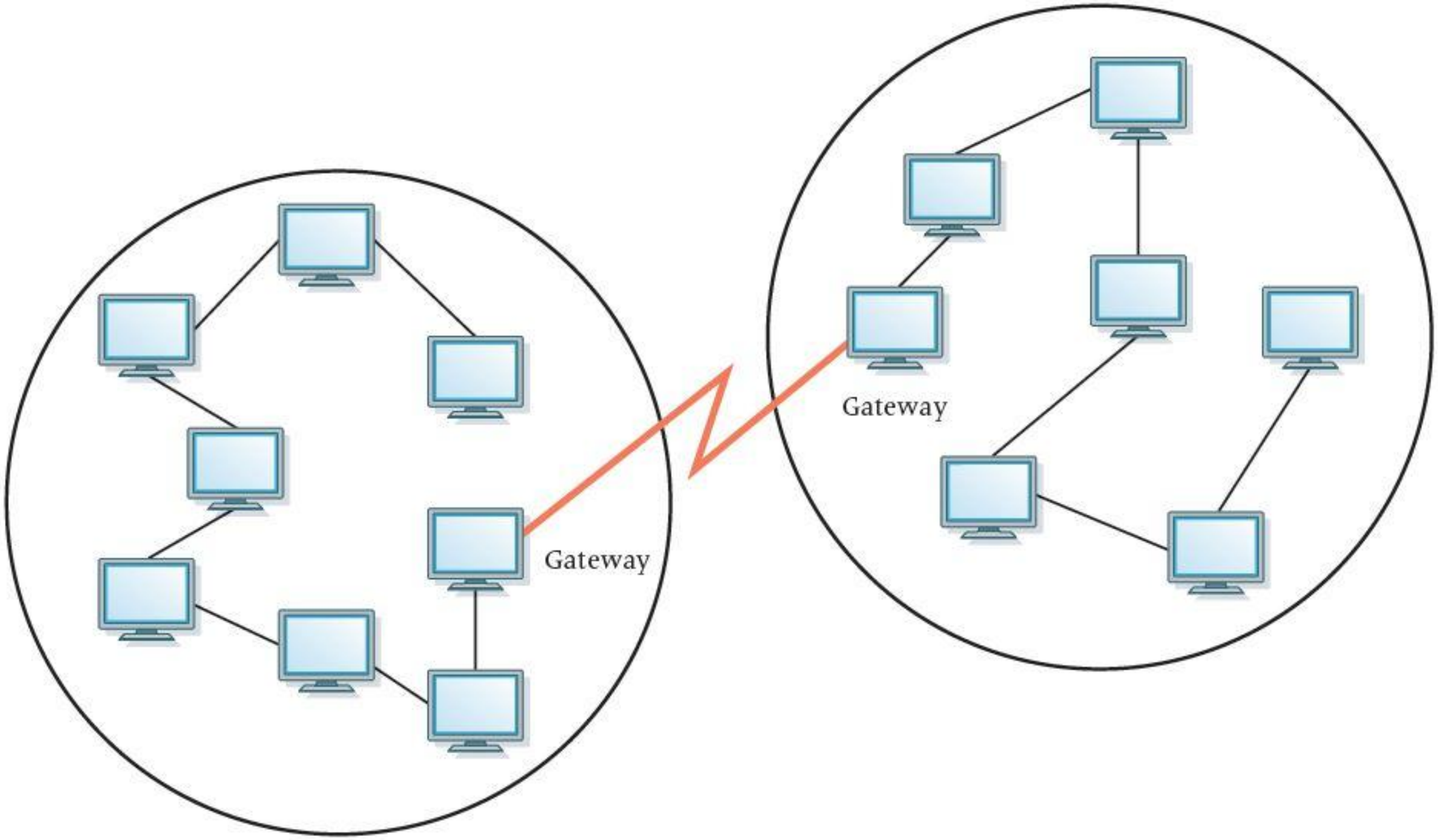


FIGURE 15.3 Local-area networks connected across a distance to create a wide-area network

Types of Networks

Internet

A wide area network that spans the planet

So, who owns the Internet?

Internet Connections

Wireless network

A network in which devices communicate with other nodes through a wireless access point

Bluetooth

A technology used for wireless communication over short distances

Internet Connections

Internet backbone

A set of high-speed networks that carry Internet traffic, provided by companies such as AT&T, Verizon, GTE, British Telecom, and IBM

Internet service provider (ISP)

An organization providing access to the Internet

Internet Connections

Various technologies available to connect a home computer to the Internet

- **Phone modem** converts computer data into an analog audio signal for transfer over a telephone line, and then a modem at the destination converts it back again into data
- **Digital subscriber line (DSL)** uses regular copper phone lines to transfer digital data to and from the phone company's central office
- **Cable modem** uses the same line that your cable TV signals come in on to transfer the data back and forth

Internet Connections

Broadband

A connection in which transfer speeds are faster than 768 kilobits per second

- DSL connections and cable modems are broadband connections
- The speed for **downloads** (getting data from the Internet to your home computer) may not be the same as **uploads** (sending data from your home computer to the Internet)

Packet Switching

Packet

A unit of data sent across a network

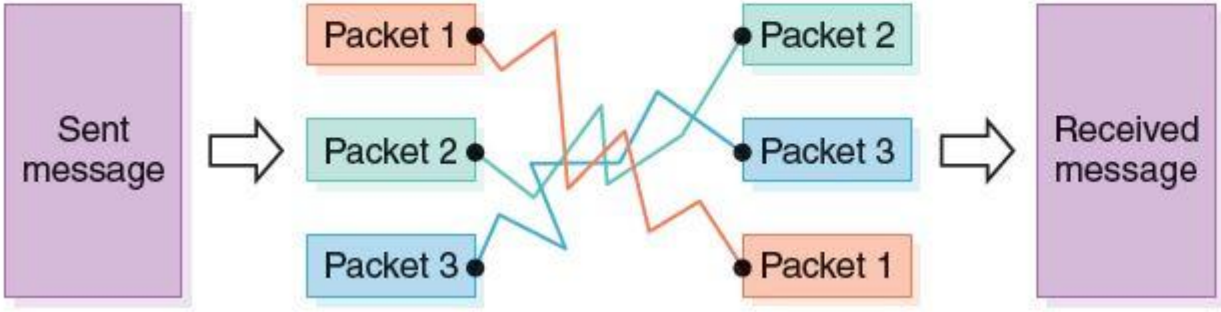
Router

A network device that directs a packet between networks toward its final destination

Packet switching

Messages are divided into fixed-sized, numbered packets; packets are individually routed to their destination, then reassembled

Packet Switching



Message is divided into packets

Packets are sent over the Internet by the most expedient route

Packets are reordered and then reassembled

FIGURE 15.4 Messages sent by packet switching

Take a message, break it into three packets, and simulate this process

Open Systems

A logical progression...

Proprietary system

A system that uses technologies kept private by a particular commercial vendor

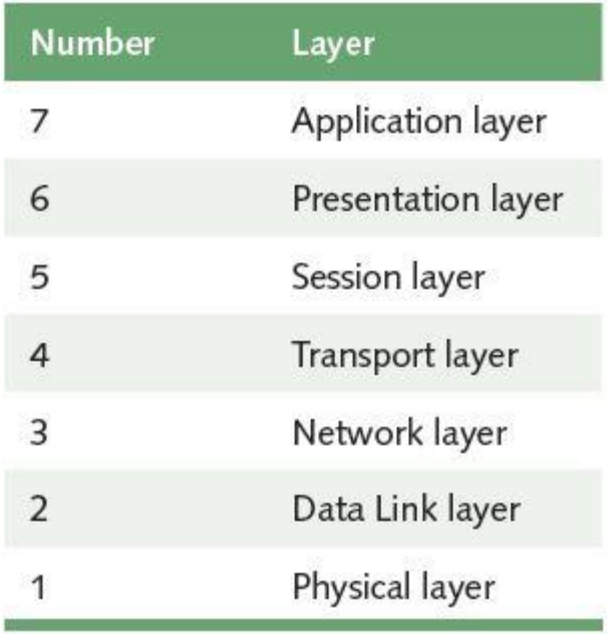
Interoperability

The ability of software and hardware on multiple machines and from multiple commercial vendors to communicate

Open systems

Systems based on a common model of network architecture and a suite of protocols used in its implementation

Open Systems



Number	Layer
7	Application layer
6	Presentation layer
5	Session layer
4	Transport layer
3	Network layer
2	Data Link layer
1	Physical layer

FIGURE 15.5 The layers of the OSI Reference Model

Open Systems Interconnection Reference Model

A seven-layer logical break down of network interaction to facilitate communication standards

Each layer deals with a particular aspect of network communication

Network Protocols

- Network protocols are layered such that each one relies on the protocols that underlie it
- Sometimes referred to as a **protocol stack**

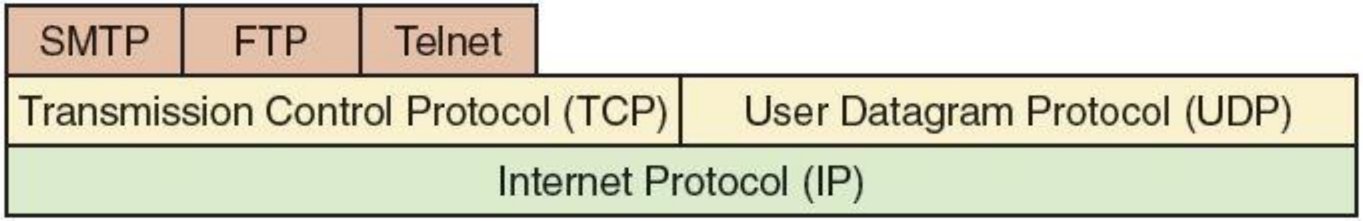

SMTP
FTP
Telnet
Transmission Control Protocol (TCP)
User Datagram Protocol (UDP)
Internet Protocol (IP)

FIGURE 15.6 Layering of key network protocols

TCP/IP

Transmission Control Protocol (TCP)

Software that breaks messages into packets, hands them off to the IP software for delivery, and then orders and reassembles the packets at their destination

Internet Protocol (IP)

Software that deals with the routing of packets through the maze of interconnected networks to their final destination

TCP/IP

User Datagram Protocol (UDP)

An alternative to TCP that is faster but less reliable

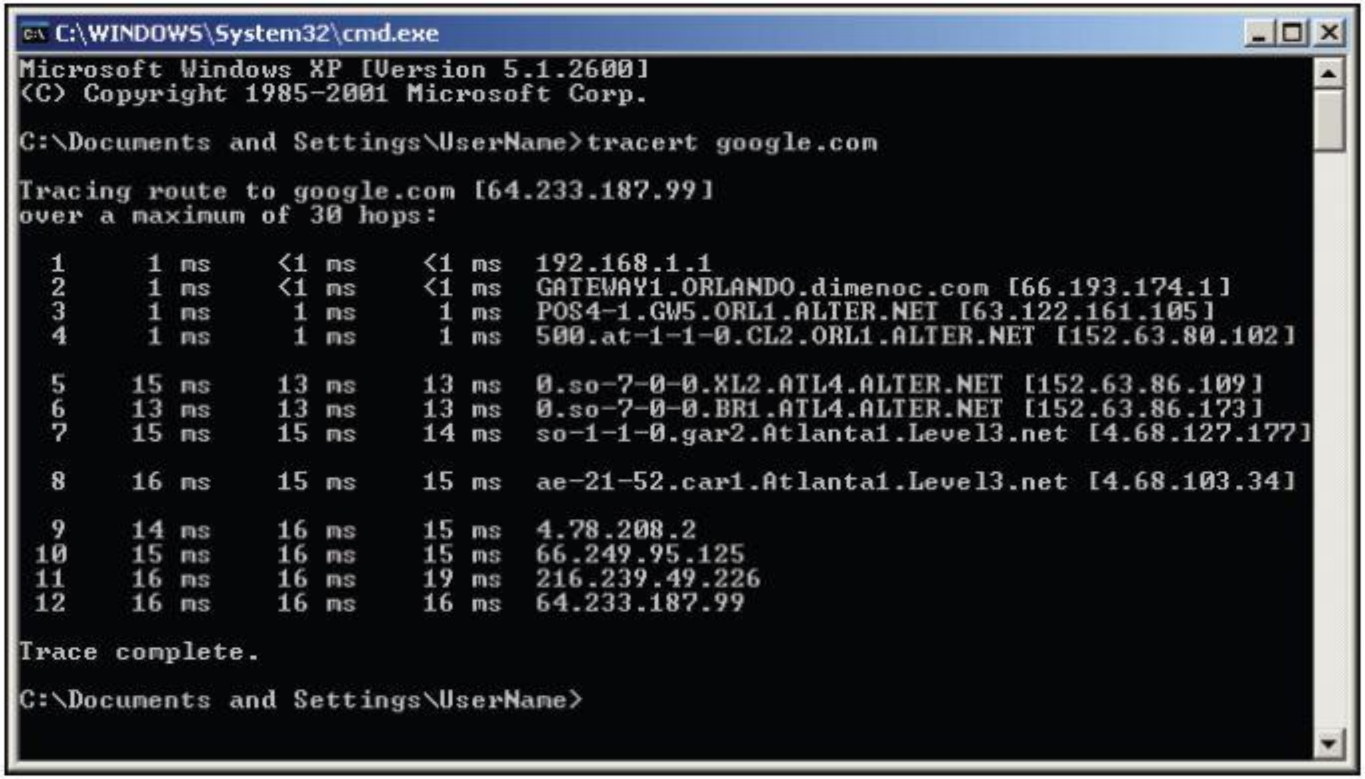
Ping

A program used to test whether a particular network computer is active and reachable

Traceroute

A program that shows the route a packet takes across the Internet

Traceroute in Action



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\UserName>tracert google.com

Tracing route to google.com [64.233.187.99]
over a maximum of 30 hops:

  1     1 ms    <1 ms    <1 ms  192.168.1.1
  2     1 ms    <1 ms    <1 ms  GATEWAY1.ORLANDO.dimenoc.com [66.193.174.1]
  3     1 ms     1 ms     1 ms  POS4-1.GW5.ORL1.ALTER.NET [63.122.161.105]
  4     1 ms     1 ms     1 ms  500.at-1-1-0.CL2.ORL1.ALTER.NET [152.63.80.102]

  5    15 ms    13 ms    13 ms  0.so-7-0-0.XL2.ATL4.ALTER.NET [152.63.86.109]
  6    13 ms    13 ms    13 ms  0.so-7-0-0.BR1.ATL4.ALTER.NET [152.63.86.173]
  7    15 ms    15 ms    14 ms  so-1-1-0.gar2.Atlanta1.Level3.net [4.68.127.177]

  8    16 ms    15 ms    15 ms  ae-21-52.car1.Atlanta1.Level3.net [4.68.103.34]

  9    14 ms    16 ms    15 ms  4.78.208.2
 10    15 ms    16 ms    15 ms  66.249.95.125
 11    16 ms    16 ms    19 ms  216.239.49.226
 12    16 ms    16 ms    16 ms  64.233.187.99

Trace complete.

C:\Documents and Settings\UserName>
```

FIGURE 15.7 The traceroute utility

Used with permission from Microsoft

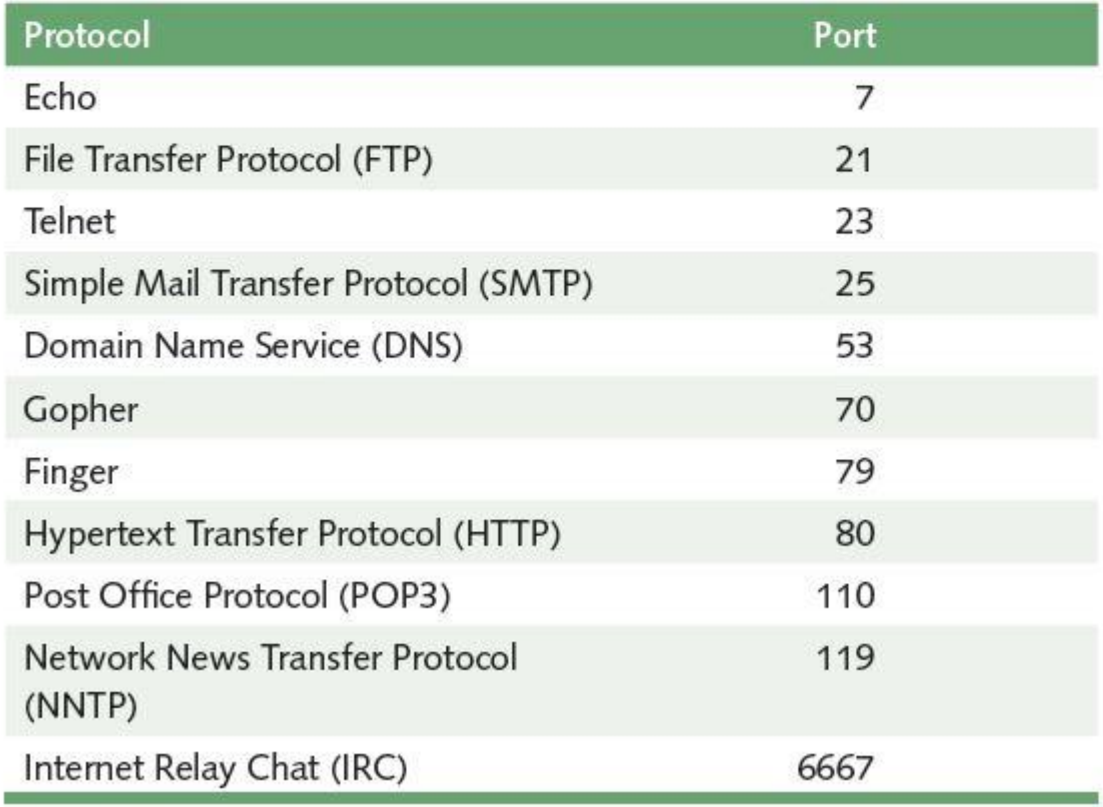
High-Level Protocols

Other protocols build on TCP/IP protocol suite

- **Simple Mail Transfer Protocol (SMTP)** used to specify transfer of electronic mail
- **File Transfer Protocol (FTP)** allows a user to transfer files to and from another computer
- **Telnet** used to log onto one computer from another
- **Hyper Text Transfer Protocol (http)** allows exchange of Web documents

Which of these have you used?

High-Level Protocols



Protocol	Port
Echo	7
File Transfer Protocol (FTP)	21
Telnet	23
Simple Mail Transfer Protocol (SMTP)	25
Domain Name Service (DNS)	53
Gopher	70
Finger	79
Hypertext Transfer Protocol (HTTP)	80
Post Office Protocol (POP3)	110
Network News Transfer Protocol (NNTP)	119
Internet Relay Chat (IRC)	6667

FIGURE 15.8 Some protocols and the ports they use

Port
A numeric designation that corresponds to a particular high-level protocol

MIME Types

MIME type

A standard for defining the format of files that are included as email attachments or on websites

What does MIME stand for?

Multipurpose Internet Mail Extension

Firewalls

Firewall

A gateway machine and its software that protects a network by filtering the traffic it allows

Access control policy

A set of rules established by an organization that specifies what types of network communication are permitted and denied

Have your messages ever been returned undelivered, blocked by a firewall?

Firewalls

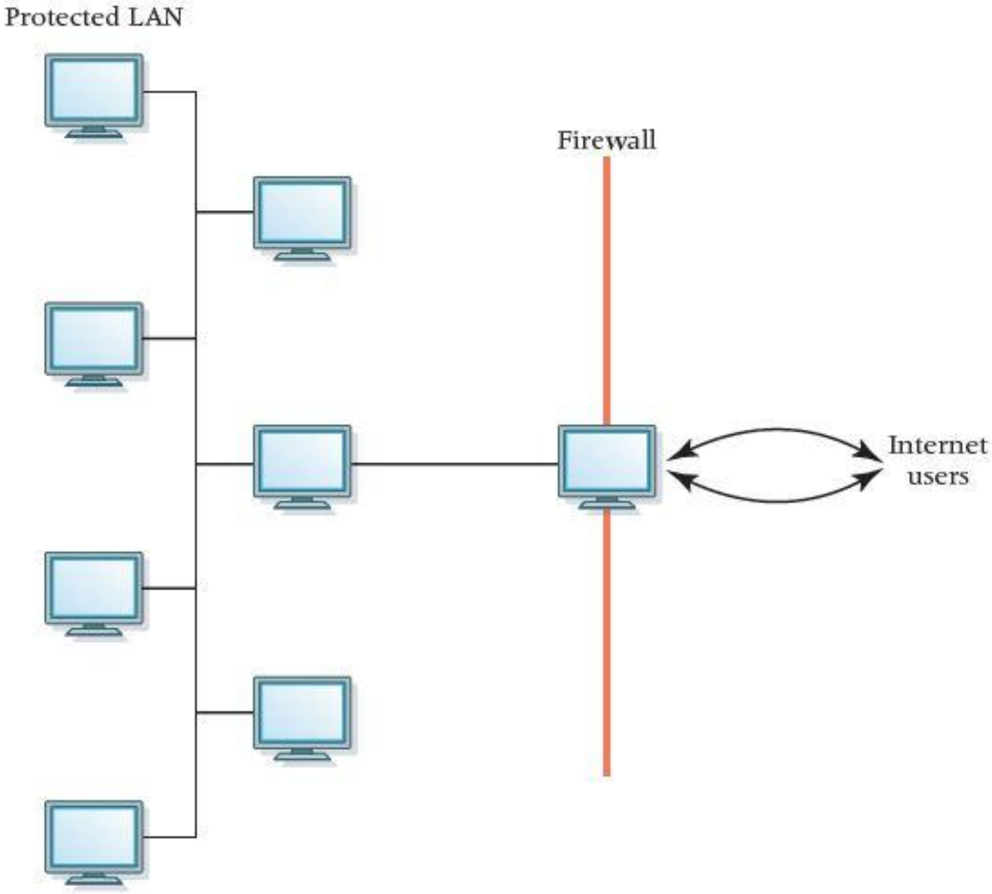


FIGURE 15.9 A firewall protecting a LAN.

Network Addresses

Hostname

A name made up of words separated by dots that uniquely identifies a computer on the Internet

IP address

An address made up of four one-byte numeric values separated by dots that uniquely identifies a computer on the Internet

Is there a correspondence between the parts of a hostname and an IP address?

Network Addresses

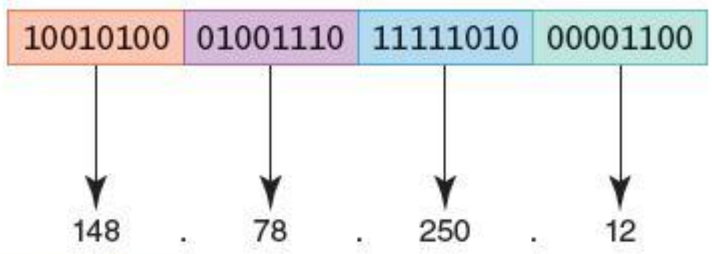


FIGURE 15.10 An IP address stored in four bytes

What is wrong with the IP4 strategy?

How did smartphones contribute to the problem?

Network Addresses

IPv4

The last block was assigned in 2011

IPv6

32 bits organized into 4 groups of 8

FE80:0000:0000:0000:0202:B3FF:FE1E:8329

They work in parallel

Domain Name System

Host number

The part of the IP address that specifies a particular host (machine) on the network *Yes, but what is it?*

Domain name

The part of a hostname that specifies a specific organization or group

Top-level domain (TLD)

The last section of a domain name that specifies the type of organization or its country of origin

Domain Name System

Domain name system (DNS)

A distributed system for managing hostname resolution

Domain name server

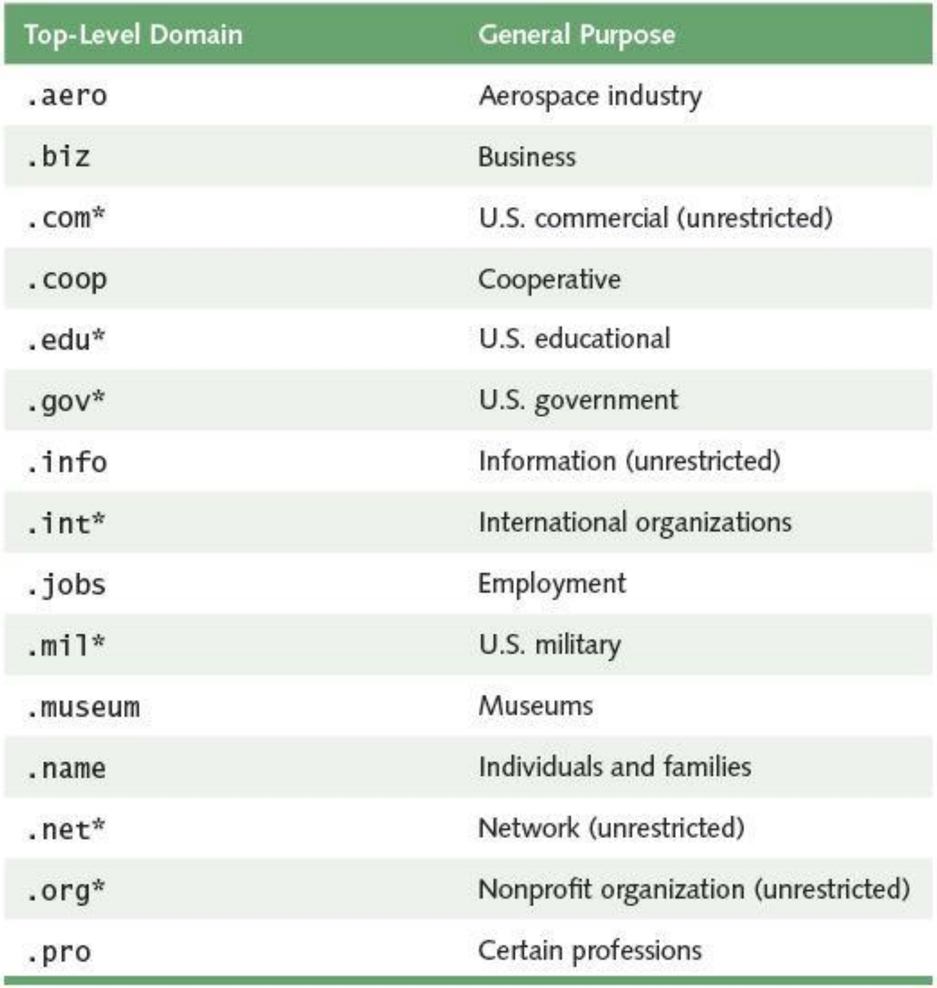
A computer that attempts to translate a hostname into an IP address

Domain Squatting

Ransoming domain names

Should the tables containing hostname/IP mappings be sorted or unsorted? Why?

Domain Name System

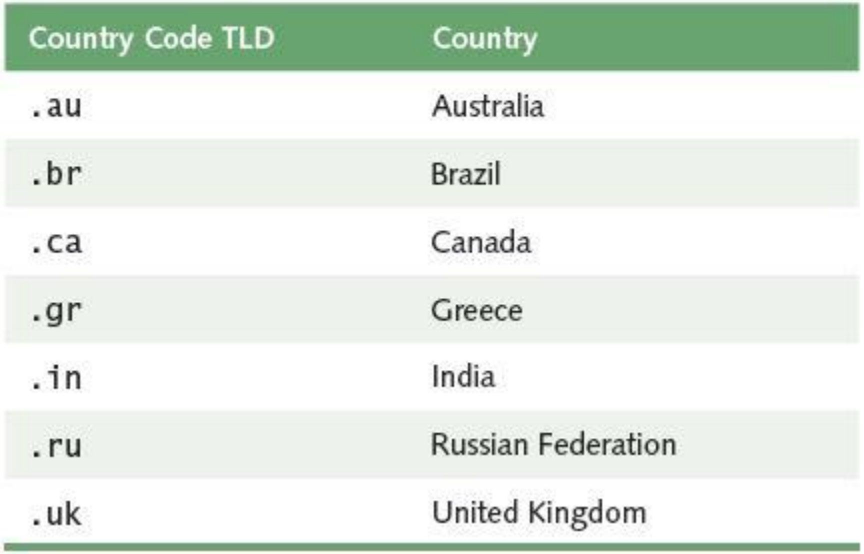


Top-Level Domain	General Purpose
.aero	Aerospace industry
.biz	Business
.com*	U.S. commercial (unrestricted)
.coop	Cooperative
.edu*	U.S. educational
.gov*	U.S. government
.info	Information (unrestricted)
.int*	International organizations
.jobs	Employment
.mil*	U.S. military
.museum	Museums
.name	Individuals and families
.net*	Network (unrestricted)
.org*	Nonprofit organization (unrestricted)
.pro	Certain professions

FIGURE 15.11 Some top-level domains and their general purpose (* indicates an original TLD)

Domain Name System

Organizations based in countries other than the United States use a top-level domain that corresponds to their two-letter country codes

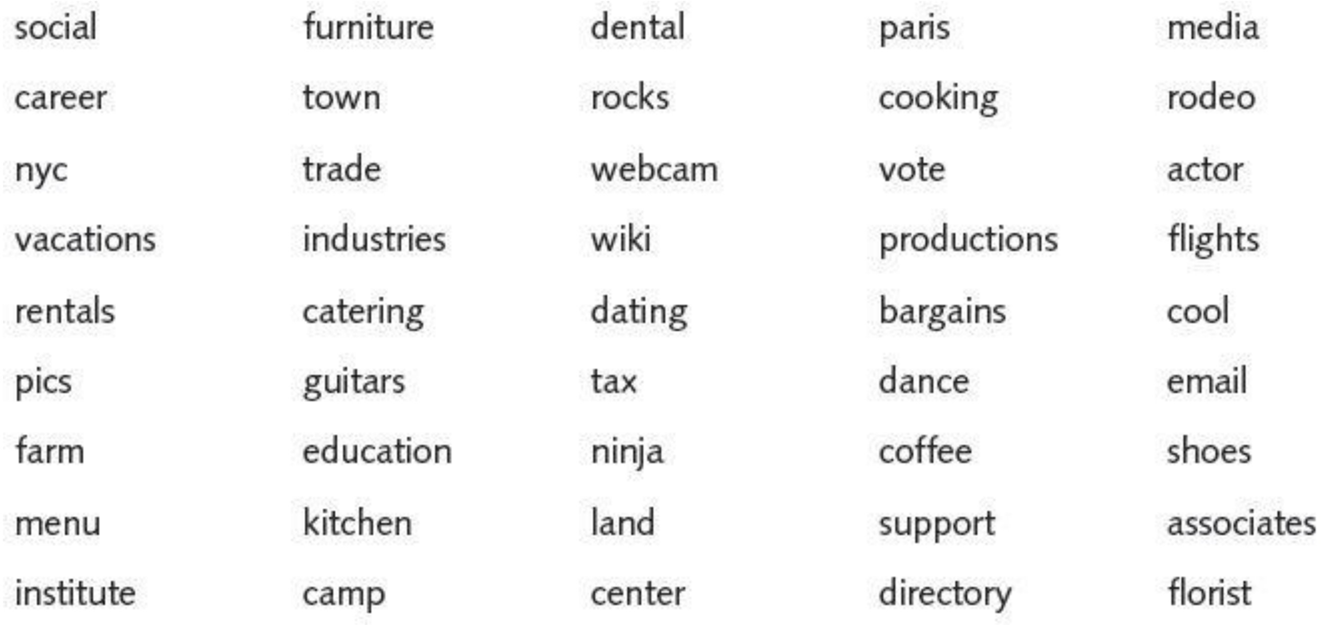


Country Code TLD	Country
.au	Australia
.br	Brazil
.ca	Canada
.gr	Greece
.in	India
.ru	Russian Federation
.uk	United Kingdom

FIGURE 15.12 Some of the top-level domain names based on country codes.

Have you emailed someone in another country?

Domain Name System



social
career
nyc
vacations
rentals
pics
farm
menu
institute
furniture
town
trade
industries
catering
guitars
education
kitchen
camp
dental
rocks
webcam
wiki
dating
tax
ninja
land
center
paris
cooking
vote
productions
bargains
dance
coffee
support
directory
media
rodeo
actor
flights
cool
email
shoes
associates
florist

A very small, random selection of new TLDs that are available as of mid-2014

Who Controls the Internet?

Control of IP addresses and domain names

- Internet began as ARPANET, a project of the US Dept. of Defense
- Control subcontracted to ICANN in 1998
- US gov’t to further reduce role as early as 2015

FCC proposal

- Would allow ISPs to provide “premium” access to certain customers, perhaps by deliberately slowing down data transfer for others
- Net neutrality - The principle that ISPs should deliver data to everyone equally, as fast as the technology allows

Cloud Computing

- **Public clouds** are accessible by any subscriber
- **Private clouds** are established for a specific group or organization
- **Community clouds** are shared among two or more organizations with the same needs
- **Hybrid clouds** are some combination of the others

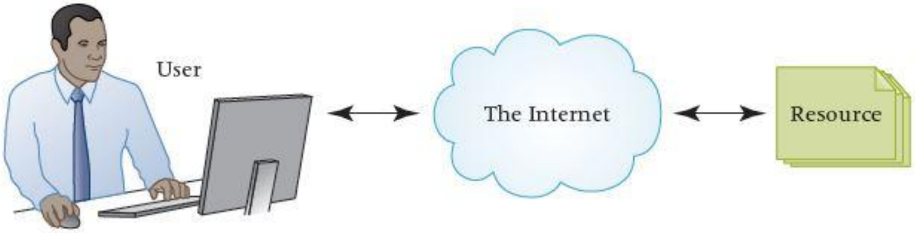


FIGURE 15.13 Internet communication depicted using a cloud

Ethical Issues

Effects of Social Networking

What are some examples of popular social networking sites?

Who uses social networking?

What are the benefits and the disadvantages of using these social networking sites?

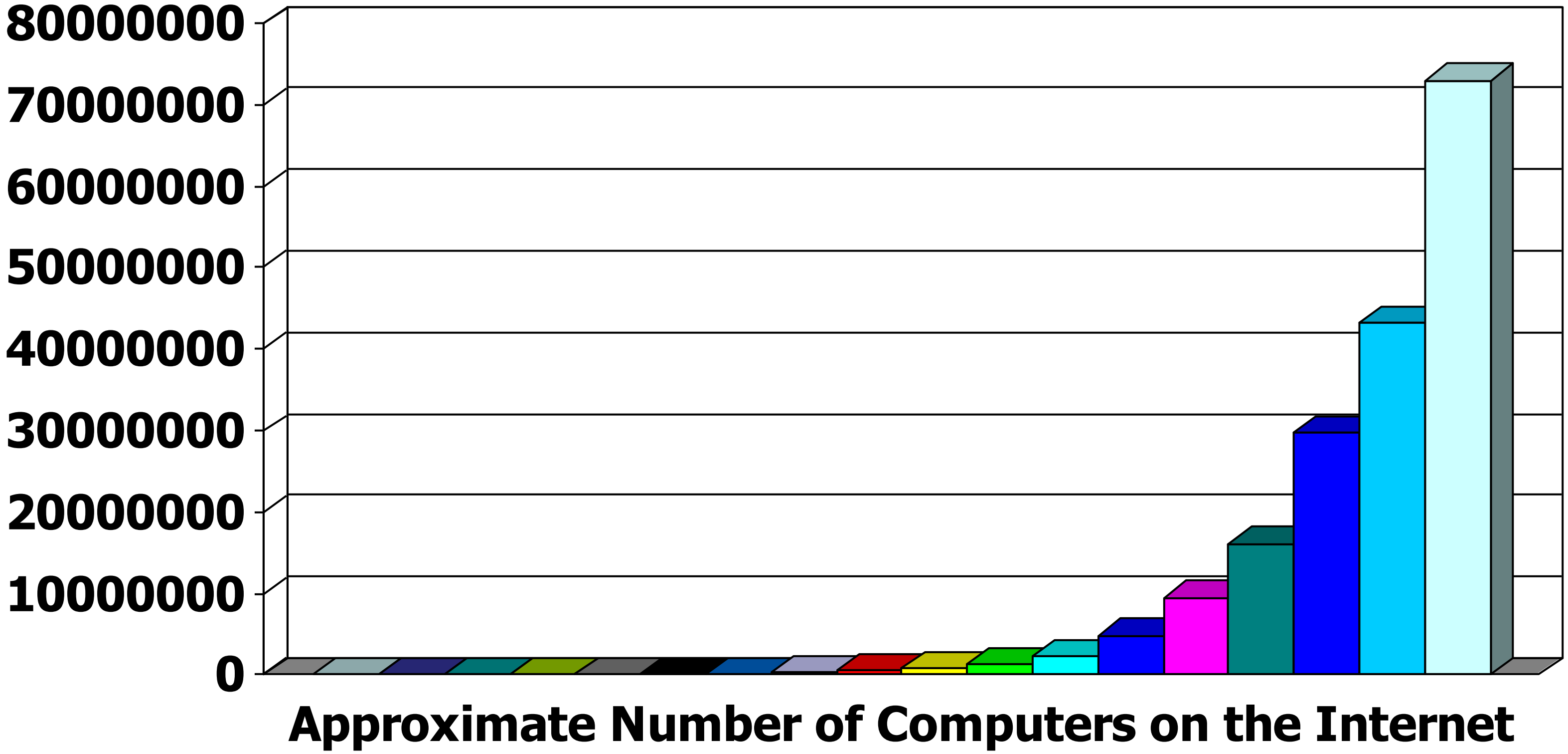
Do the benefits of social networking out weigh the potential costs?

Brief History of Internet

Internet Impact

- Check weather
- Buy goods
- Play music
- Find the shortest route
- Give a lecture…

The Incredible Growth



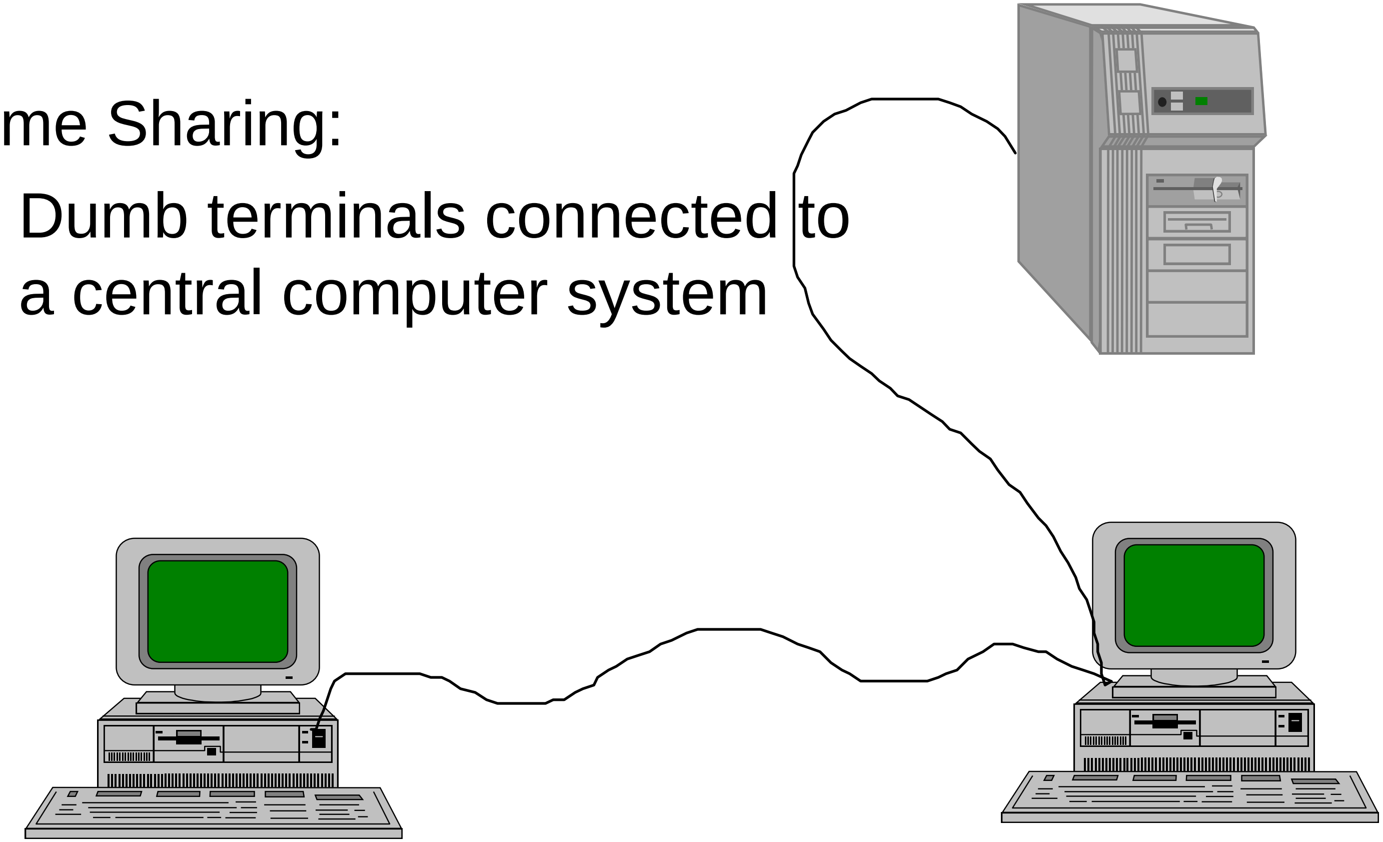
Brief Internet History

- Batch Environment - 1950s
  - No direct interaction between users and their programs during execution
- Time Sharing - 1960s
  - Users were able to interact with the computer and could share its information processing resources
  - Marked the beginning of computer communications

Brief Internet History (cont.)

Time Sharing:

Dumb terminals connected to a central computer system



Brief Internet History (cont.)

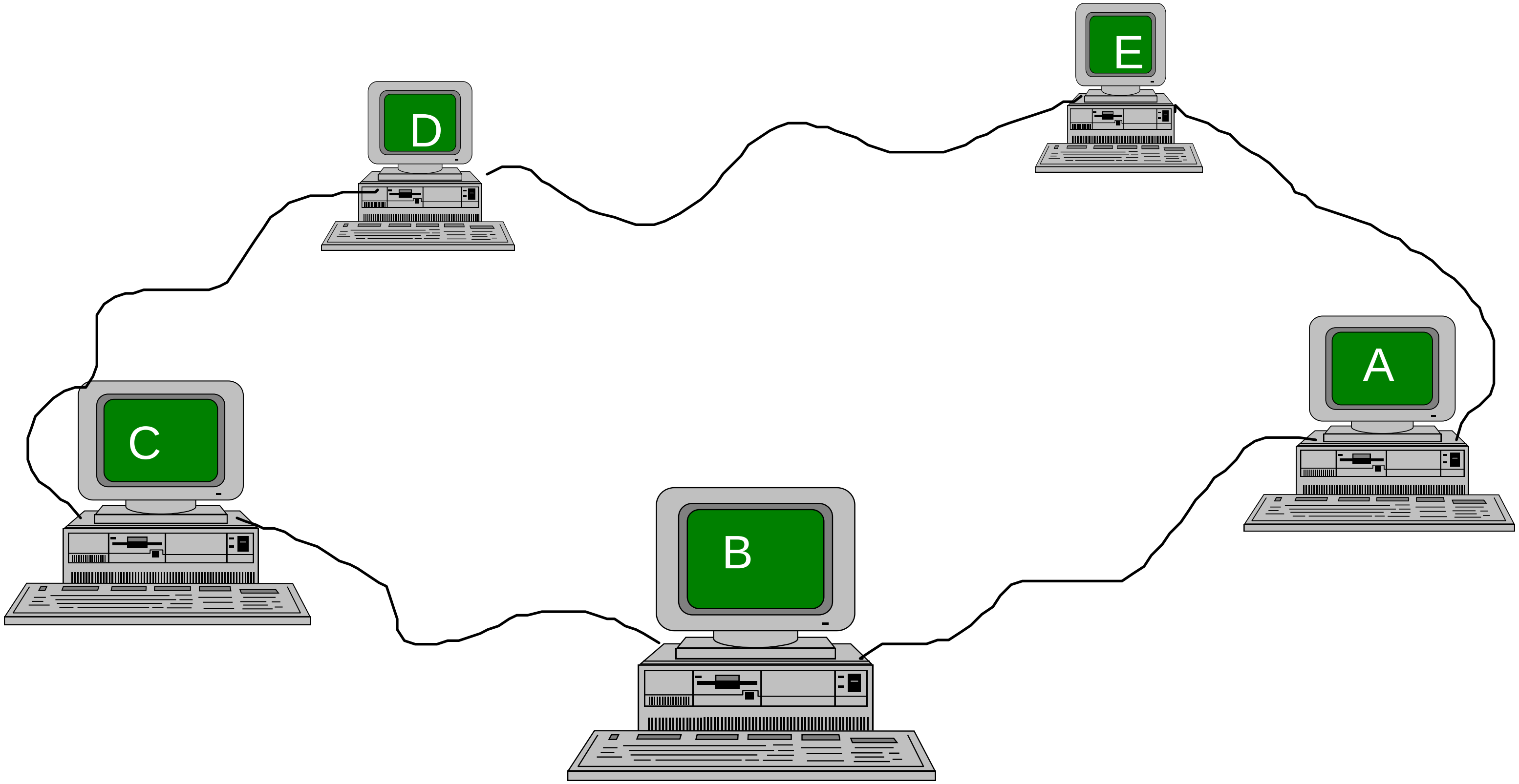
- Late 1960s: ARPANET
  - ARPA (Advanced Research Projects Agency) commissioned an experimental computer network

- 1970s:
  - Distributed Processing: minicomputers;
    - Communication between neighbor processors and applications via networks
  - Growth of ARPANET and Invention of Email

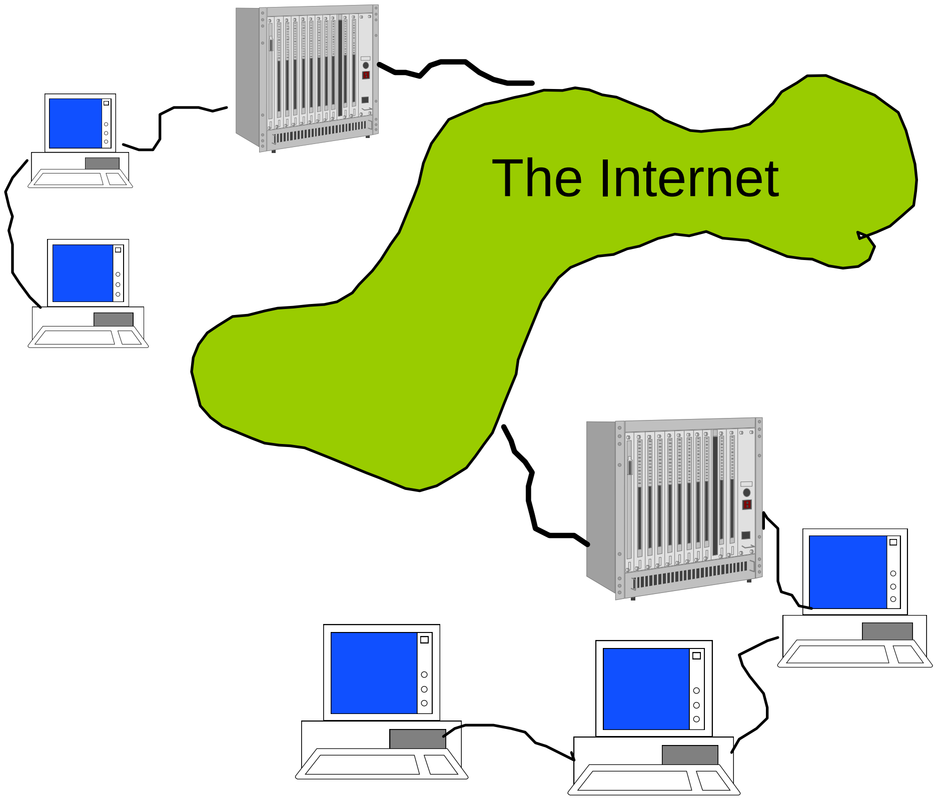
Brief Internet History (cont.)

- 1980s:
  - WAN and LAN
  - Prototype Internet
  - TCP/IP:Allows different networks to interconnect

A LAN Example



Internet: a network of networks



Brief Internet History (cont.)

- 1990s: WWW
  - – HTTP and HTML
  - – Marc Andreessen: Mosaic (all-in-one solution)
  - – Commercial traffic allowed ECommerce

Networking Questions

- Over what medium?
- At what speed?
- How to address computers?
- Which path?
- How to handle (detect & recover) errors?
- What services?
- How to address documents?
- What data format?

Outline

- Introduction
- OSI Model
- TCP/IP Model
- IPv4 vs. IPv6

What is a Protocol?

- A standard that allows entities (i.e. application programs) from different systems to communicate
- Shared conventions for communicating information
- Includes syntax, semantics, and timing

Standardized Protocol Architectures

- Vendors like standards because they make their products more marketable
- Customers like standards because they enable products from different vendors to interoperate
- Two protocol standards are well-known:
  - TCP/IP: widely implemented
  - OSI: less used, still useful for modeling/conceptualizing

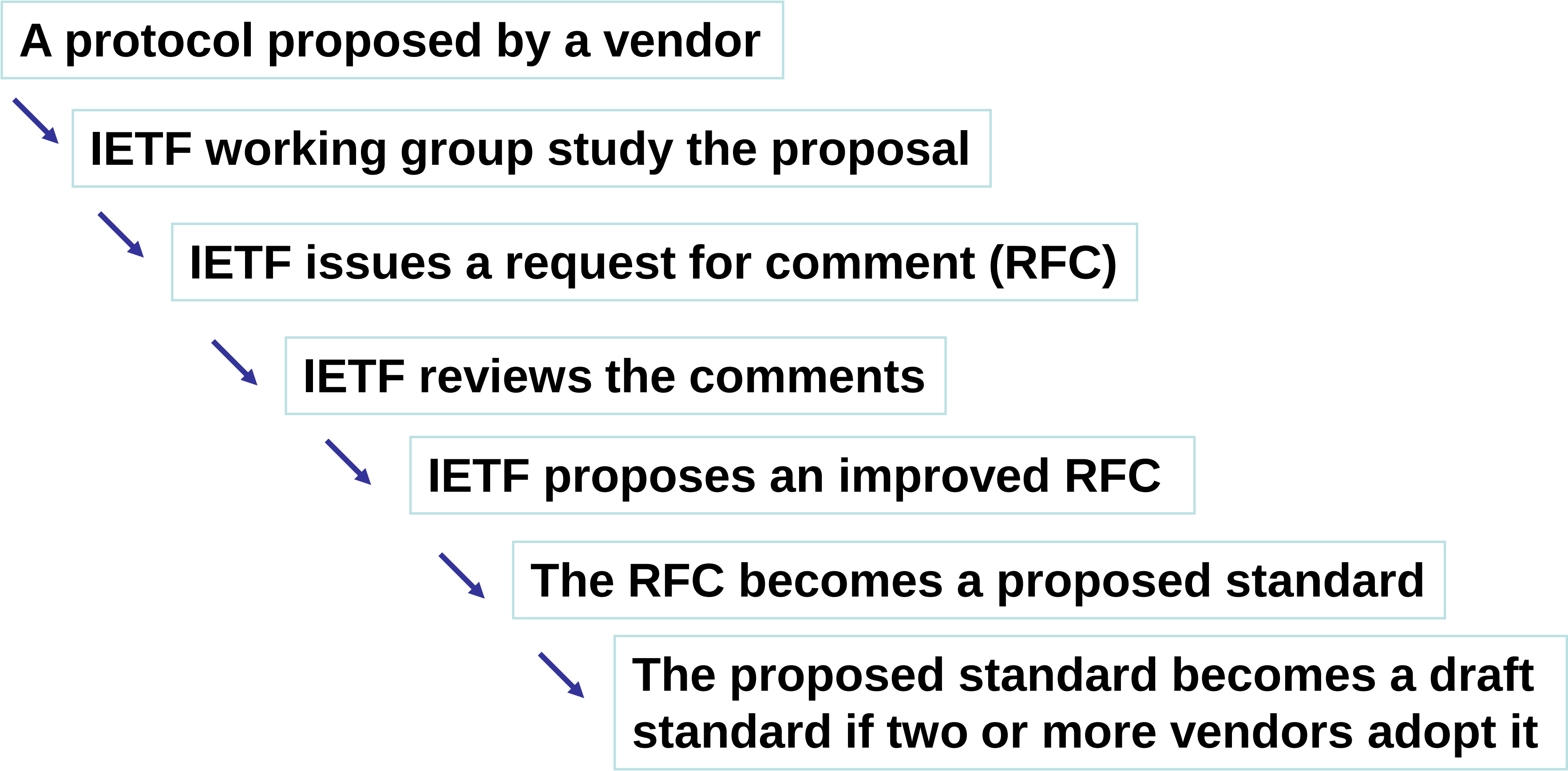
Internet Standards

- Email related standards
  - IMAP, POP, X.400, SMTP, CMC, MIME, binhex, uuencode
- Web related standards
  - http, CGI, html/xml/vrml/sgml
- Internet directory standards
  - X.500, LDAP
- Application standards
  - http, FTP, telnet, gopher, wais
- Videoconferencing standards
  - H.320, H.323, Mpeg-1, Mpeg-2

*Telecommunication Standards Organizations

- International Telecommunications Union - Telecommunication Standardization Sector (**ITU-TSS**). Formerly called the Consultative Committee on International Telegraph and Telephone (**CCITT**)
- International Organization for Standards (**ISO**). Member of the ITU, makes technical recommendations about data communications interfaces.
- American National Standards Institute (ANSI)
- Institute of Electrical and Electronics Engineers (**IEEE**)
- Internet Engineering Task Force (**IETF**)
- Electronic Industries Association (EIA)
- National Institute of Standards and Technology (NIST)
- National Exchange Carriers Association (NECA)
- Corporation for Open Systems (COS)
- Electronic Data Interchange -(EDI) of Electronic Data Interchange for Administration Commerce and Transport (EDIFACT).

*Internet Engineering Task Force



A protocol proposed by a vendor

IETF working group study the proposal

IETF issues a request for comment (RFC)

IETF reviews the comments

IETF proposes an improved RFC

The RFC becomes a proposed standard

The proposed standard becomes a draft standard if two or more vendors adopt it

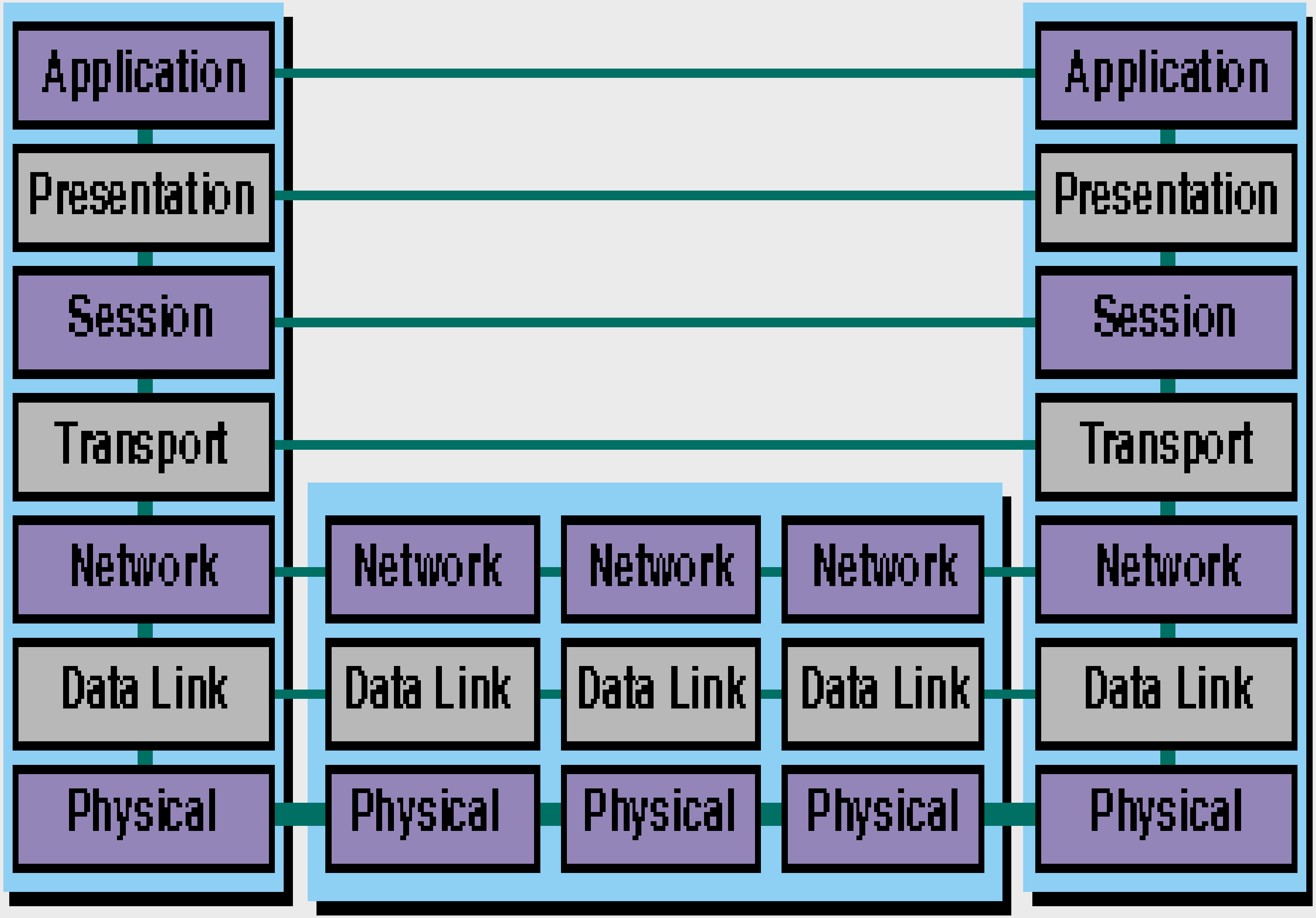
What is OSI?

- Developed by the International Organization for Standardization (ISO) in 1984
- The primary architectural model for intercomputer communications.
- A conceptual model composed of seven layers, each specifying particular network functions.
- Describes how information from a software application in one computer moves through a network medium to a software application in another computer.

Why Study OSI?

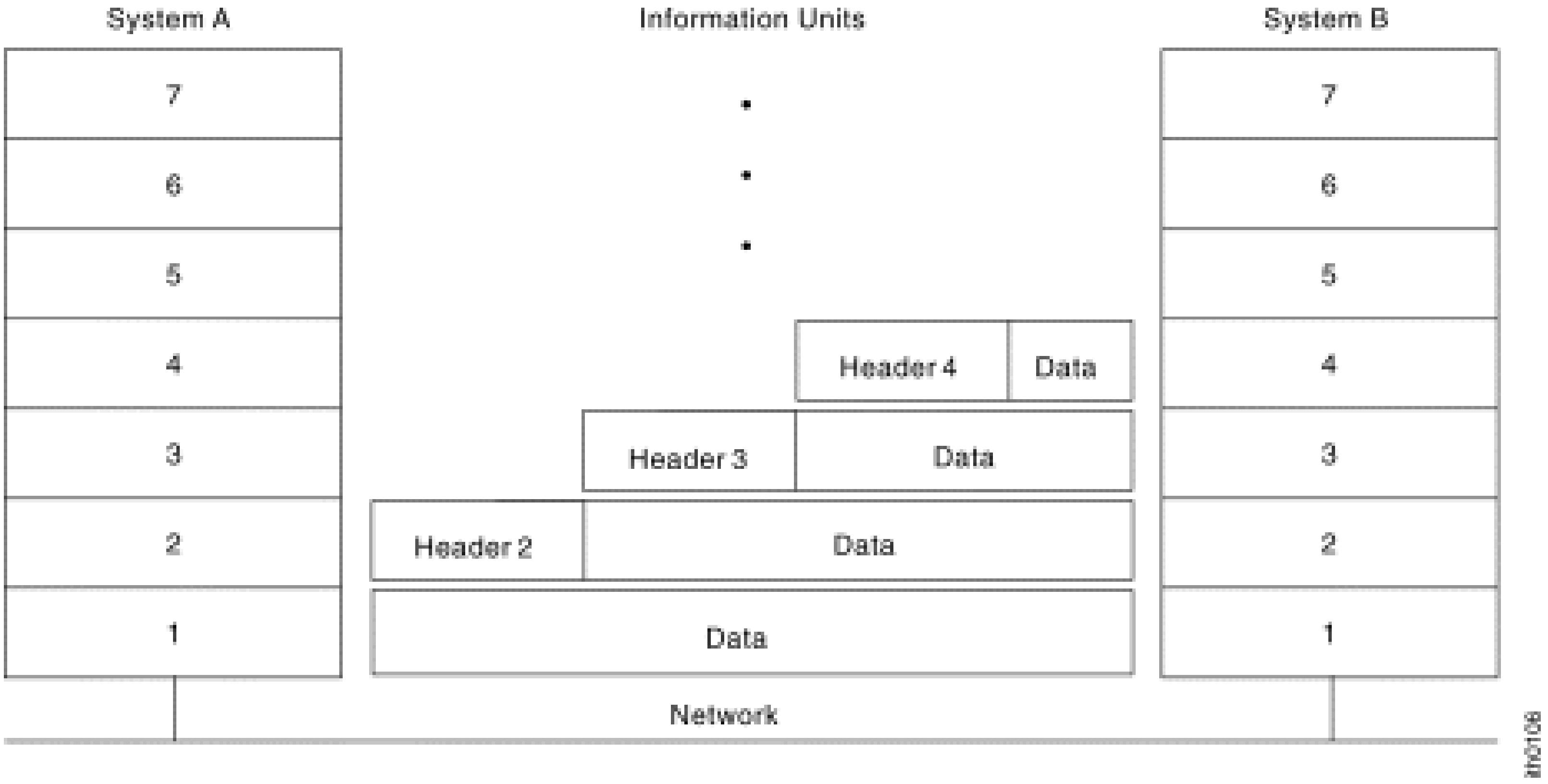
- Still an excellent model for conceptualizing and understanding protocol architectures
- Key points:
  - Modular
  - Hierarchical
  - Boundaries between layers=interfaces

OSI Stack



Intermediate nodes in a network
service (example AT&T APS)

Headers and Data



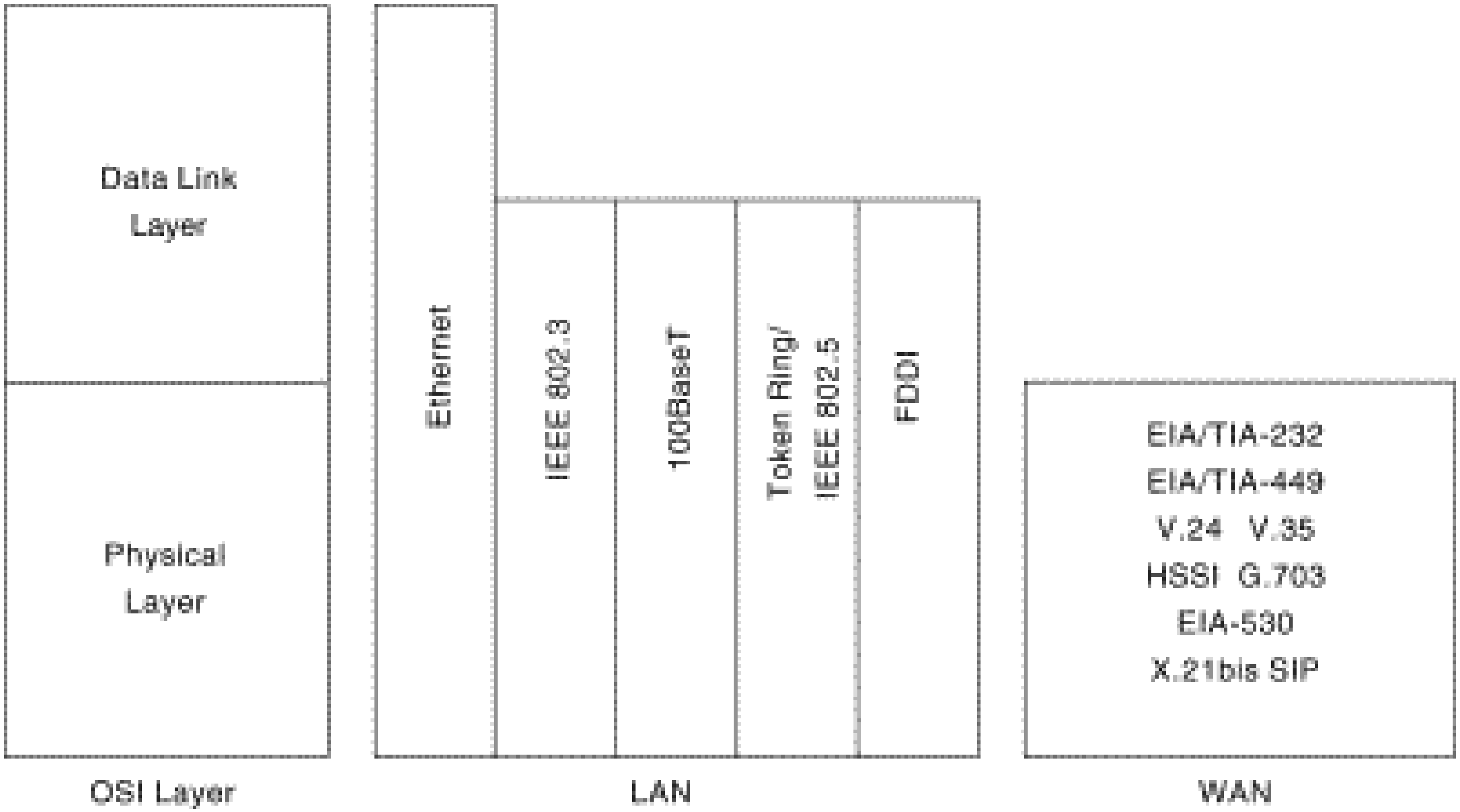
OSI Lower Layers

- Physical – Layer 1
- Data Link – Layer 2
- Network – Layer 3

OSI Physical Layer

- Responsible for transmission of bits
- Always implemented through hardware
- Encompasses mechanical, electrical, and functional interfaces
- e.g. RS-232

*Physical-layer Implementation

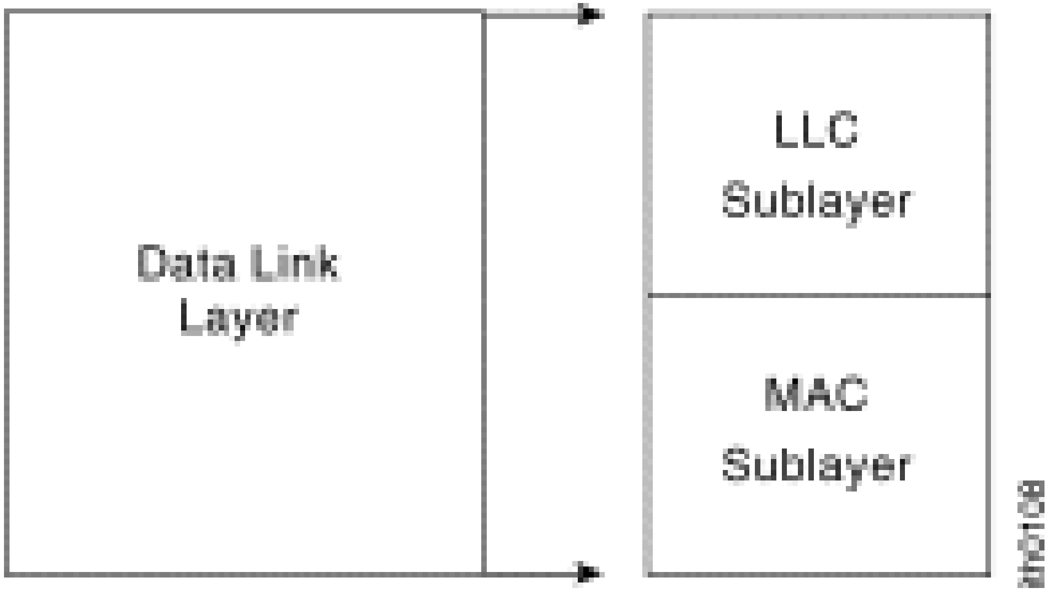


Physical Layer Implementations

OSI Data Link Layer

- Responsible for error-free, reliable transmission of data
- Flow control, error correction
- e.g. HDLC

OSI Data Link Layer



IEEE has subdivided data link layer into two sub-layers.

OSI Network Layer

- Responsible for routing of messages through network
- Concerned with type of switching used (circuit v. packet)
- Handles routing between networks, as well as through packet-switching networks

Network Access Layer

- Concerned with exchange of data between computer and network
- Includes addressing, routing, prioritizing, etc
- Different networks require different software at this layer
- Example: X.25 standard for network access procedures on packet-switching networks

OSI Upper Layers

- Transport
- Session
- Presentation
- Application

OSI Transport Layer

- Isolates messages from lower and upper layers
- Breaks down message size
- Monitors quality of communications channel
- Selects most efficient communication service necessary for a given transmission

Transport Layer

- Concerned with reliable transfer of information between applications
- Independent of the nature of the application
- Includes aspects like flow control and error checking

OSI Session Layer

- Establishes logical connections between systems
- Manages log-ons, password exchange, log-offs
- Terminates connection at end of session

OSI Presentation Layer

- Provides format and code conversion services
- Examples
  - File conversion from ASCII to EBDIC
  - Invoking character sequences to generate bold, italics, etc on a printer

OSI Application Layer

- Provides access to network for end-user
- User’s capabilities are determined by what items are available on this layer
- Logic needed to support various applications
- Each type of application (file transfer, remote access) requires different software on this layer

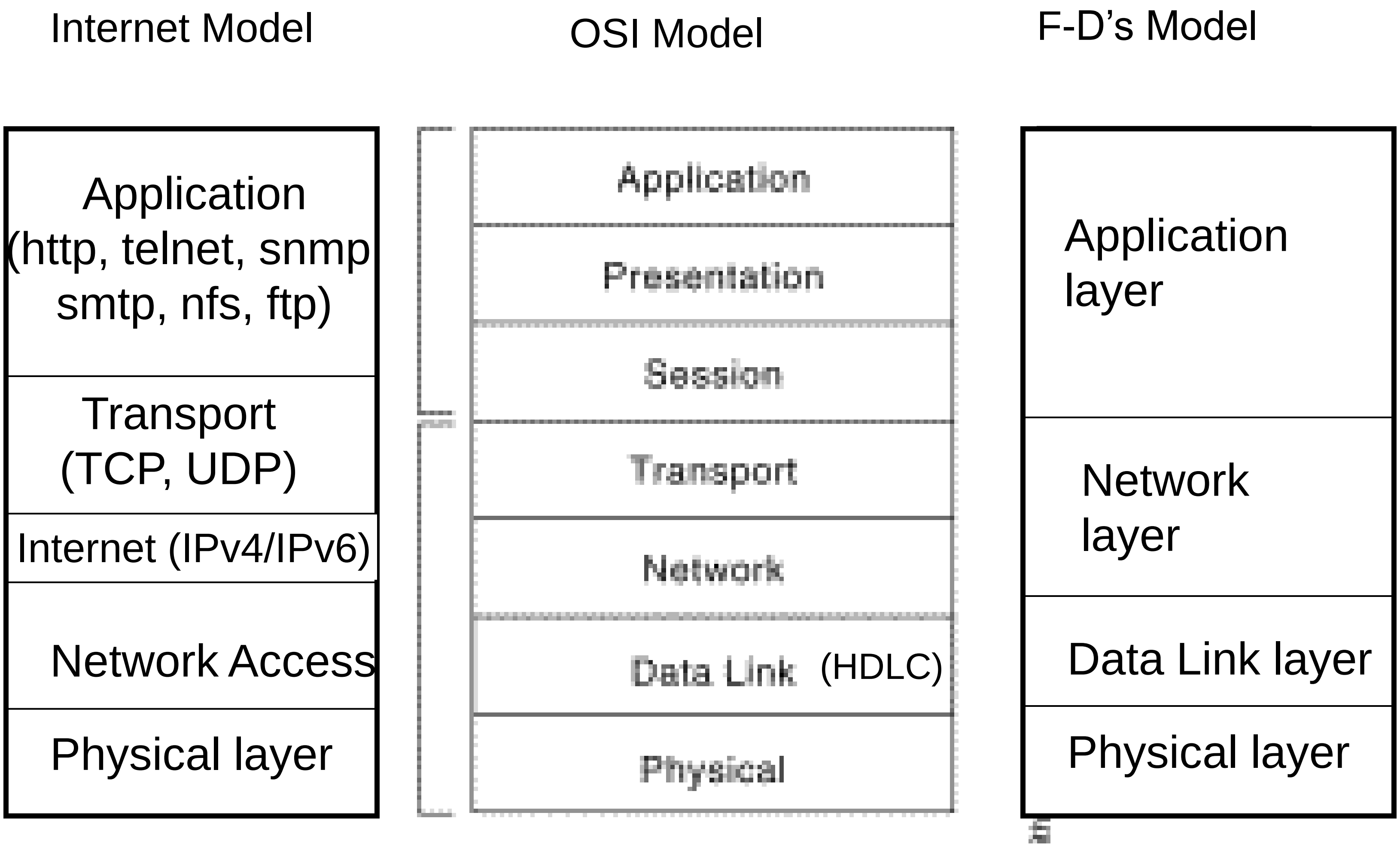
Application Viewpoint of a Network

- Distributed data communications involves three primary components:
  - – Networks
  - – Computers
  - – Applications
- Three corresponding layers
  - – Network access layer
  - – Transport layer
  - – Application layer

TCP/IP

- Transmission control Protocol/Internet Protocol
- Developed by DARPA
- No official protocol standard
- Can identify five layers
  - Application
  - Host-to-Host (transport)
  - Internet
  - Network Access
  - Physical

An OSI View of TCP/IP



Internet Model

- Application (http, telnet, snmp smtp, nfs, ftp)
- Transport (TCP, UDP)
- Internet (IPv4/IPv6)
- Network Access
- Physical layer

OSI Model

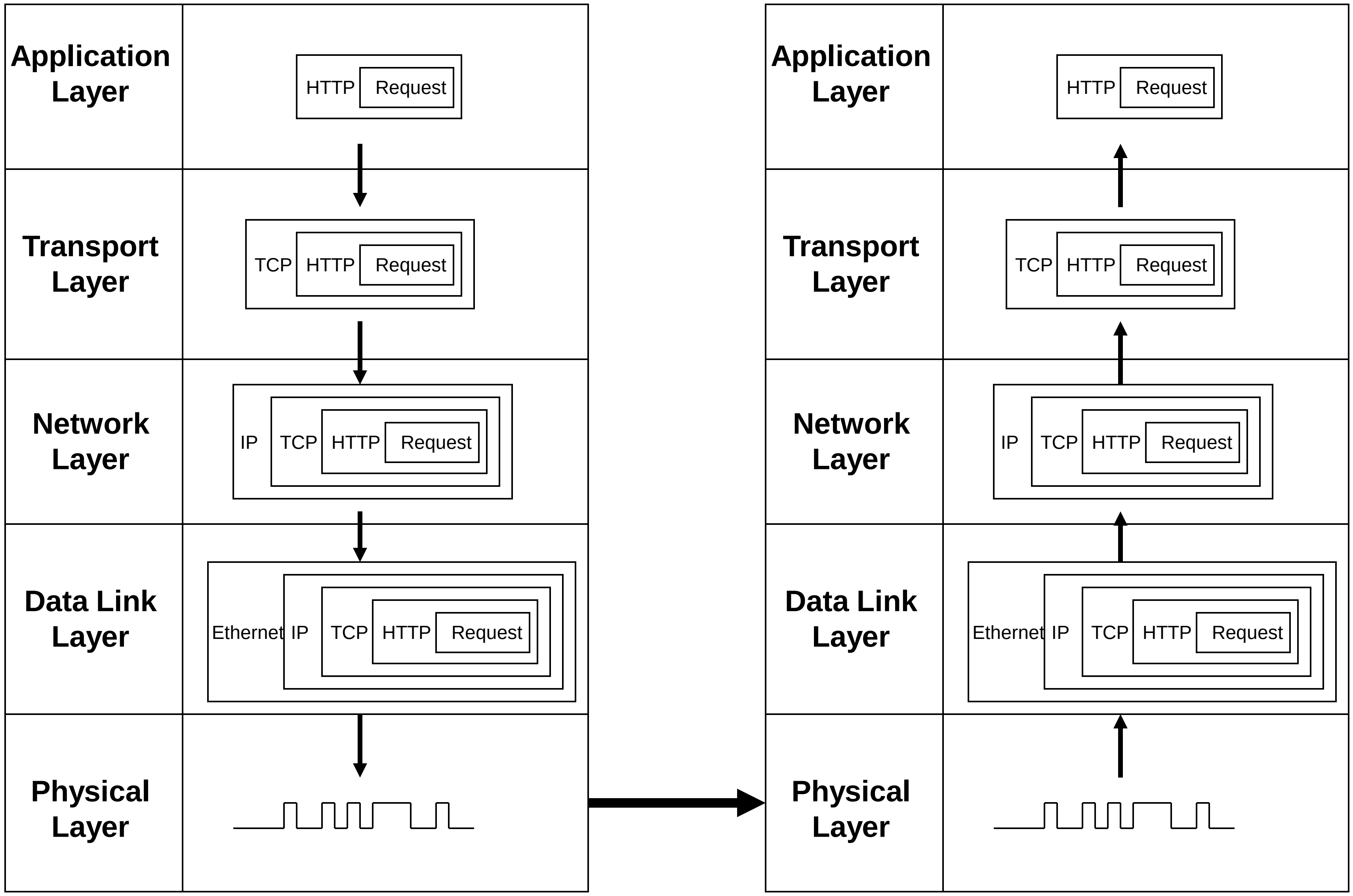
- Application
- Presentation
- Session
- Transport
- Network
- Data Link (HDLC)
- Physical

F-D's Model

- Application layer
- Network layer
- Data Link layer
- Physical layer

Sender

Receiver



Application Layer

HTTP Request

Transport Layer

TCP HTTP Request

Network Layer

IP TCP HTTP Request

Data Link Layer

Ethernet IP TCP HTTP Request

Physical Layer

Application Layer

HTTP Request

Transport Layer

TCP HTTP Request

Network Layer

IP TCP HTTP Request

Data Link Layer

Ethernet IP TCP HTTP Request

Physical Layer

TCP/IP Network Access Layer

- Exchange of data between end system and network
- Address of host and destination
- Prioritization of transmission
- Software at this layer depends on network (e.g. X.25 vs. Ethernet)
- Segregation means that no other software needs to be concerned about net specifics

TCP/IP Internet Layer

- An Internet is an interconnection of two or more networks
- Internet layer handles tasks similar to network access layer, but between networks rather than between nodes on a network
- Uses IP for addressing and routing across networks
- Implemented in workstations *and* routers

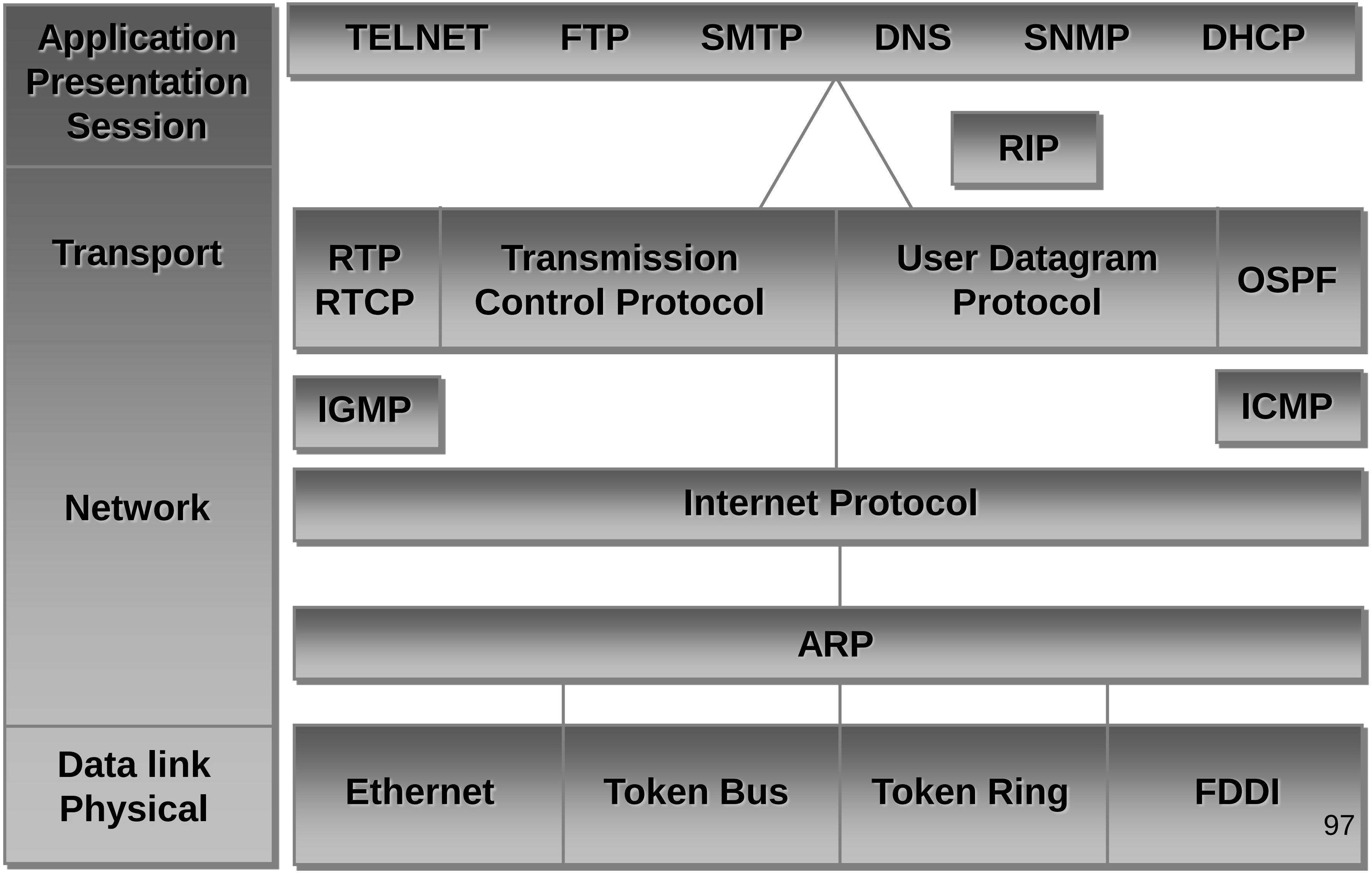
TCP/IP Transport Layer

- Also called host-to-host layer
- Reliable exchange of data between applications
- Uses TCP protocols for transmission

TCP/IP Application Layer

- Logic needed to support variety of applications
- Separate module supports each type of application (e.g. file transfer)
  - FTP
  - HTTP
  - Telnet
  - News
  - SMTP

*TCP/IP



Application
Presentation
Session
Transport
Network
Data link
Physical
TELNET
FTP
SMTP
DNS
SNMP
DHCP
RIP
RTP
RTCP
Transmission Control Protocol
User Datagram Protocol
OSPF
IGMP
ICMP
Internet Protocol
ARP
Ethernet
Token Bus
Token Ring
FDDI

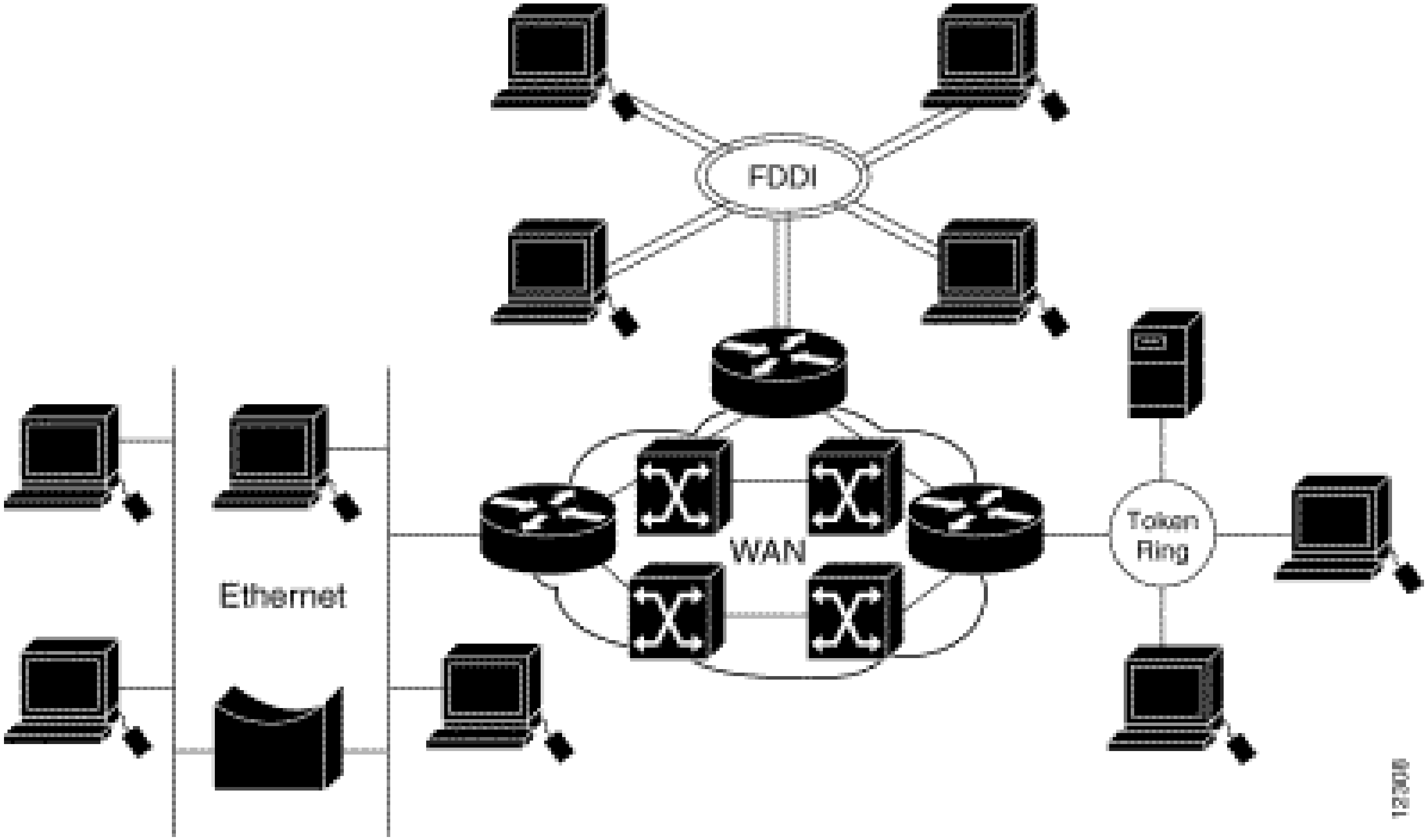
TCP & UDP

- Most TCP/IP applications use TCP for transport layer
- TCP provides a connection (logical association) between two entities to regulate flow check errors
- UDP (User Datagram Protocol) does not maintain a connection, and therefore does not guarantee delivery, preserve sequences, or protect against duplication

Internetworking

- Interconnected networks, usually implies TCP/IP
- Can appear to users as a single large network
- The global Internet is the largest example, but intranets and extranets are also examples

Internetworking



TCP Segment (TCP PDU)

- Source port (16 bits)
- Destination port (16 bits)
- Sequence number (32 bits)
- Acknowledgment number (32 bits)
- Data Offset (4 bits)
- Reserved (6 bits)
- Flags (6 bits) : URG, ACK, PSH, RST, SYN, FIN
- Window (16 bits)
- Checksum (16 bits)
- Urgent Pointer (16 bits)
- Options (variable)

The size of TCP header is 192 bits = 24 byes.

IPv4 and IPv6

- IP (IPv4) provides for 32-bit source and destination addresses, using a 192-bit header
- IPv6 (1996 standard) provides for 128-bit addresses, using a 320-bit header.
- Migration to IPv6 will be a very slow process

*History of IPng Effort

- By the Winter of 1992 the Internet community had developed four separate proposals for IPng. These were "CNAT", "IP Encaps", "Nimrod", and "Simple CLNP". By December 1992 three more proposals followed; "The P Internet Protocol" (PIP), "The Simple Internet Protocol" (SIP) and "TP/IX". In the Spring of 1992 the "Simple CLNP" evolved into "TCP and UDP with Bigger Addresses" (TUBA) and "IP Encaps" evolved into "IP Address Encapsulation" (IPAE).
- By the fall of 1993, IPAE merged with SIP while still maintaining the name SIP. This group later merged with PIP and the resulting working group called themselves "Simple Internet Protocol Plus" (SIPP). At about the same time the TP/IX Working Group changed its name to "Common Architecture for the Internet" (CATNIP).
- The IPng area directors made a recommendation for an IPng in July of 1994 [RFC 1752].
- The formal name of IPng is IPv6

Data and Signals

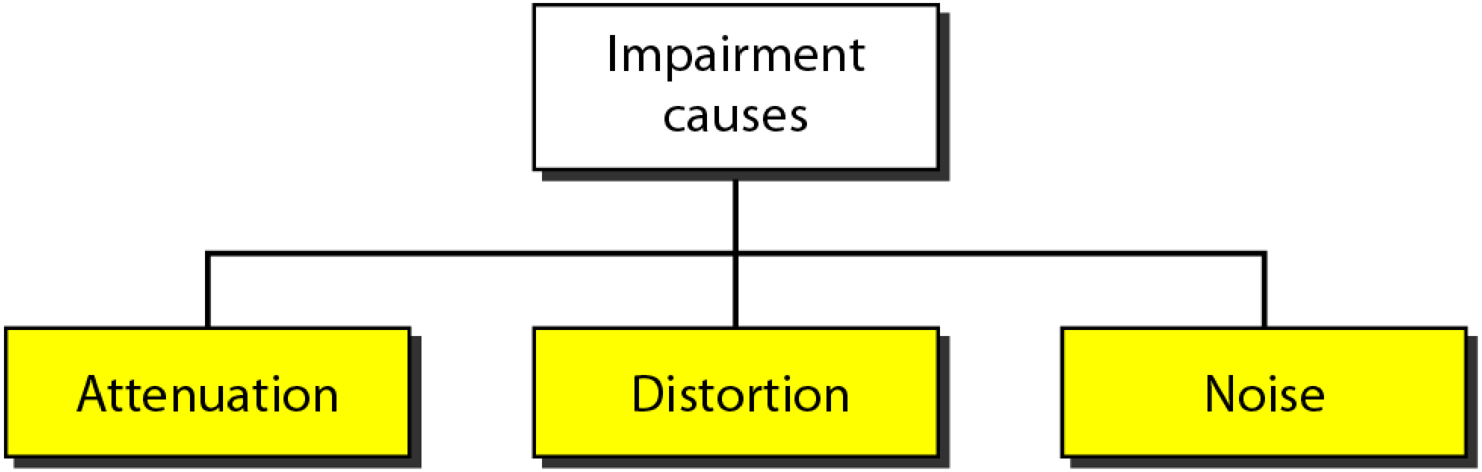
3-4 TRANSMISSION IMPAIRMENT

Signals travel through transmission media, which are not perfect. The imperfection causes signal impairment. This means that the signal at the beginning of the medium is not the same as the signal at the end of the medium. What is sent is not what is received. Three causes of impairment are attenuation, distortion, and noise.

Topics discussed in this section:

- Attenuation
- Distortion
- Noise

Figure 3.25 Causes of impairment



Impairment causes

Attenuation

Distortion

Noise

Attenuation

- Means loss of energy -> weaker signal
- When a signal travels through a medium it loses energy overcoming the resistance of the medium
- Amplifiers are used to compensate for this loss of energy by amplifying the signal.

Measurement of Attenuation

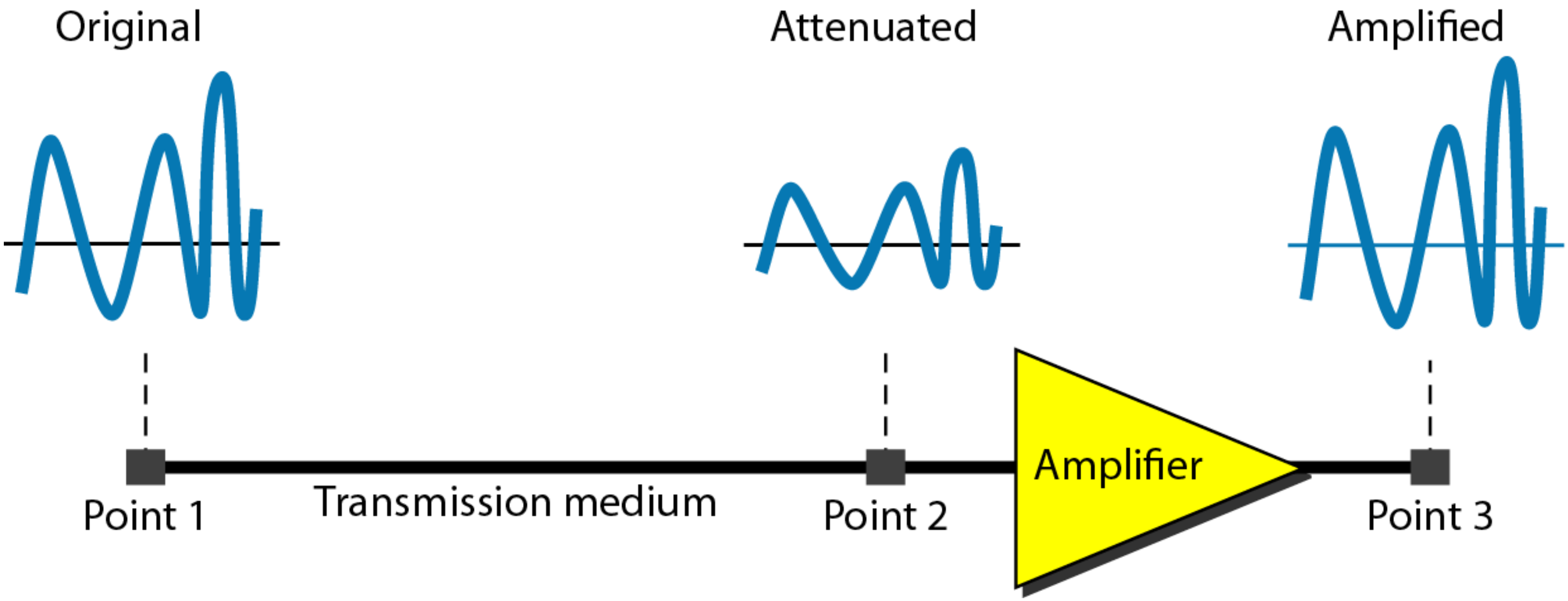
- To show the loss or gain of energy the unit “decibel” is used.

$$dB = 10\log_{10} P_2/P_1$$

P_1 - input signal

P_2 - output signal

Figure 3.26 Attenuation



Example 3.26

Suppose a signal travels through a transmission medium and its power is reduced to one-half. This means that P2 is (1/2)P1. In this case, the attenuation (loss of power) can be calculated as

$$10 \log_{10} \frac{P_2}{P_1} = 10 \log_{10} \frac{0.5 P_1}{P_1} = 10 \log_{10} 0.5 = 10(-0.3) = -3 \text{ dB}$$

A loss of 3 dB (–3 dB) is equivalent to losing one-half the power.

Example 3.27

A signal travels through an amplifier, and its power is increased 10 times. This means that $P_2 = 10P_1$. In this case, the amplification (gain of power) can be calculated as

$$\boxed{10 \log_{10} \frac{P_2}{P_1} = 10 \log_{10} \frac{10P_1}{P_1}}$$

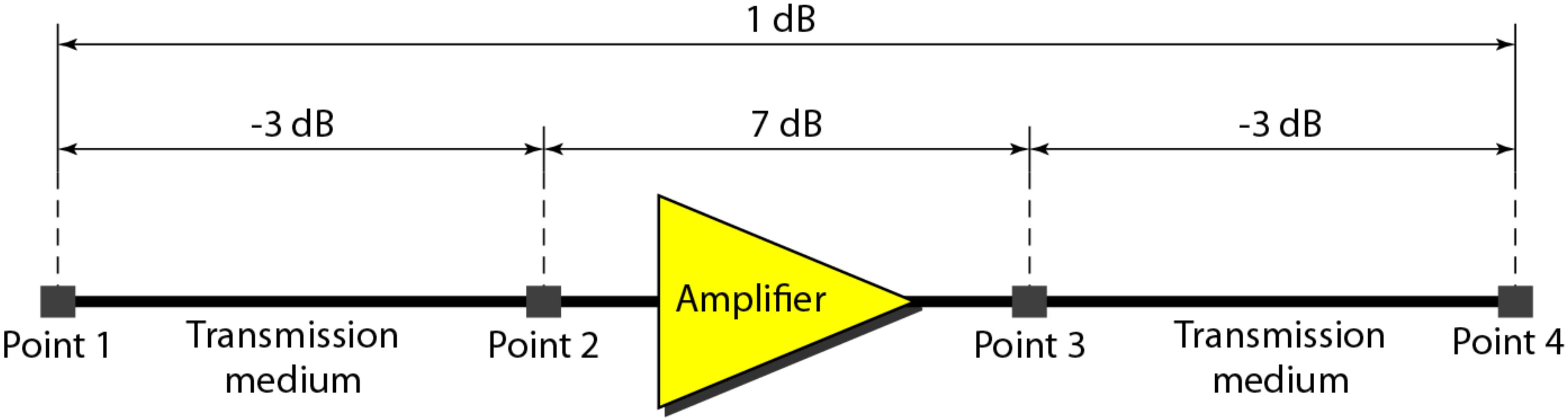
$$\boxed{= 10 \log_{10} 10 = 10(1) = 10 \text{ dB}}$$

Example 3.28

One reason that engineers use the decibel to measure the changes in the strength of a signal is that decibel numbers can be added (or subtracted) when we are measuring several points (cascading) instead of just two. In Figure 3.27 a signal travels from point 1 to point 4. In this case, the decibel value can be calculated as

$$\boxed{\text{dB} = -3 + 7 - 3 = +1}$$

Figure 3.27 Decibels for Example 3.28



Example 3.29

Sometimes the decibel is used to measure signal power in milliwatts. In this case, it is referred to as dB$_m$ and is calculated as dB$_m$ = 10 log10 P$_m$, where P$_m$ is the power in milliwatts. Calculate the power of a signal with dB$_m$ = −30.

Solution

We can calculate the power in the signal as

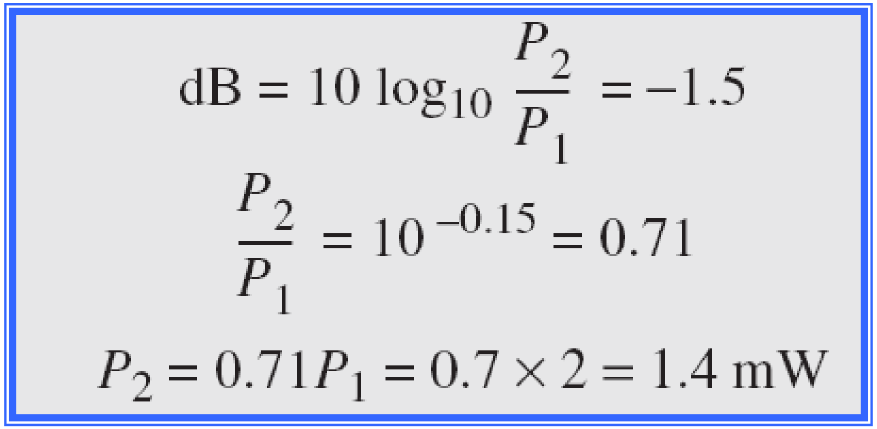
$$\boxed{\begin{gathered}\text{dB}_\text{m} = 10 \log_{10} P_m = -30 \\ \log_{10} P_m = -3 \qquad P_m = 10^{-3} \text{ mW}\end{gathered}}$$

Example 3.30

The loss in a cable is usually defined in decibels per kilometer (dB/km). If the signal at the beginning of a cable with −0.3 dB/km has a power of 2 mW, what is the power of the signal at 5 km?

Solution

The loss in the cable in decibels is 5 × (−0.3) = −1.5 dB. We can calculate the power as



$$\text{dB} = 10 \log_{10} \frac{P_2}{P_1} = -1.5$$

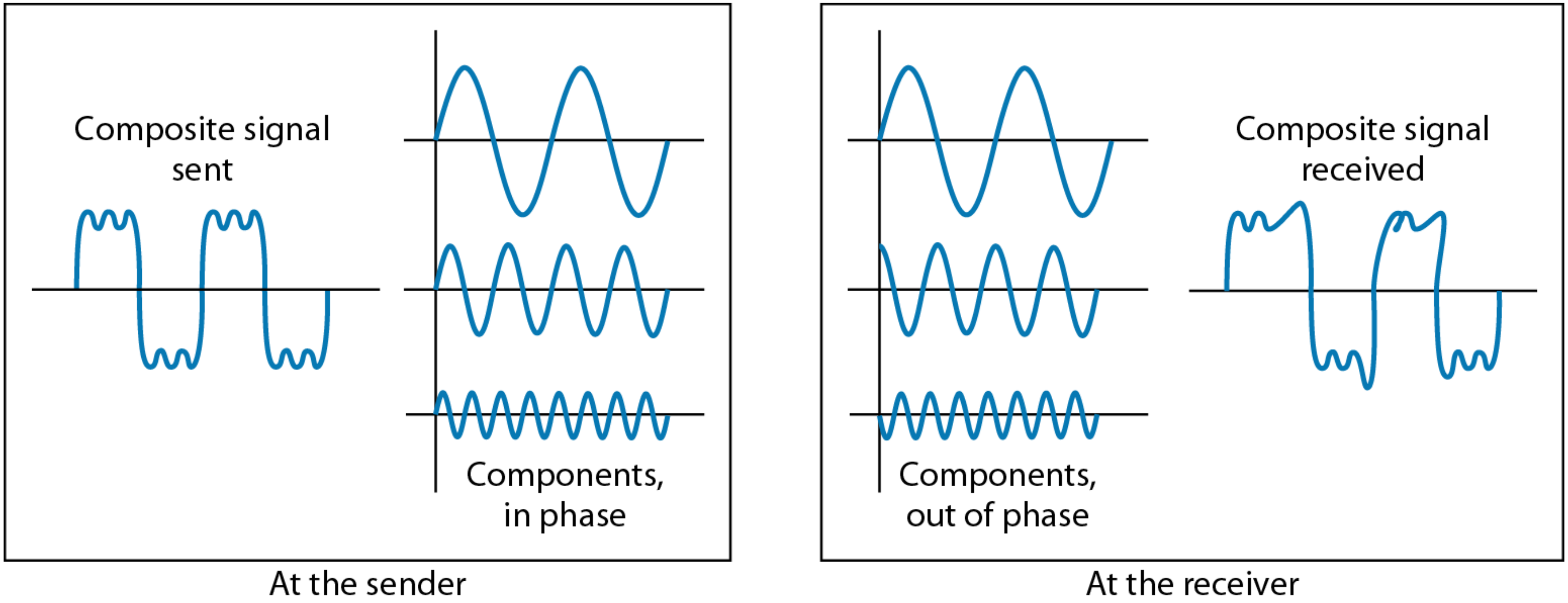
$$\frac{P_2}{P_1} = 10^{-0.15} = 0.71$$

$$P_2 = 0.71 P_1 = 0.7 \times 2 = 1.4 \text{ mW}$$

Distortion

- Means that the signal changes its form or shape
- Distortion occurs in composite signals
- Each frequency component has its own propagation speed traveling through a medium.
- The different components therefore arrive with different delays at the receiver.
- That means that the signals have different phases at the receiver than they did at the source.

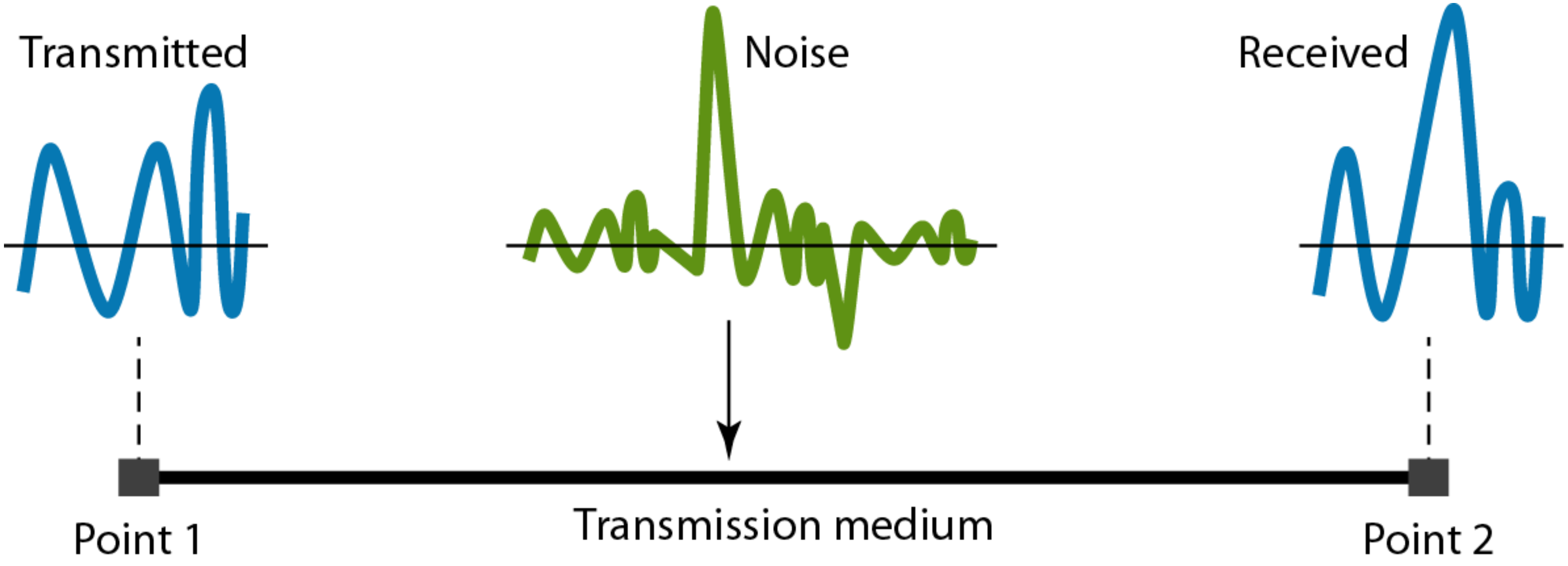
Figure 3.28 Distortion



Noise

- There are different types of noise
  - Thermal - random noise of electrons in the wire creates an extra signal
  - Induced - from motors and appliances, devices act are transmitter antenna and medium as receiving antenna.
  - Crosstalk - same as above but between two wires.
  - Impulse - Spikes that result from power lines, lighning, etc.

Figure 3.29 Noise



Signal to Noise Ratio (SNR)

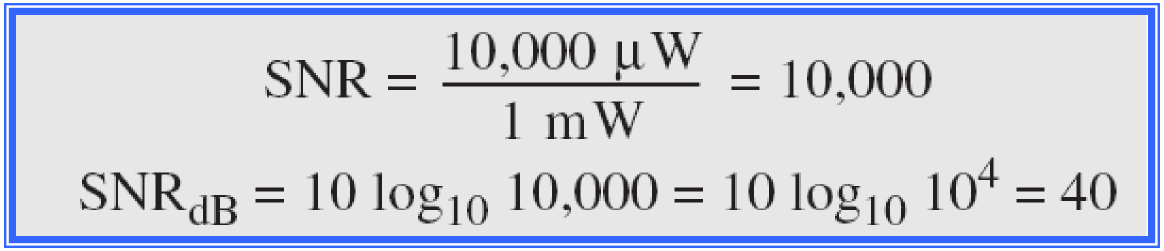
- To measure the quality of a system the SNR is often used. It indicates the strength of the signal wrt the noise power in the system.
- It is the ratio between two powers.
- It is usually given in dB and referred to as $\text{SNR}_{\text{dB.}}$

Example 3.31

The power of a signal is 10 mW and the power of the noise is 1 μW; what are the values of SNR and SNR_{dB} ?

Solution

The values of SNR and SNRdB can be calculated as follows:

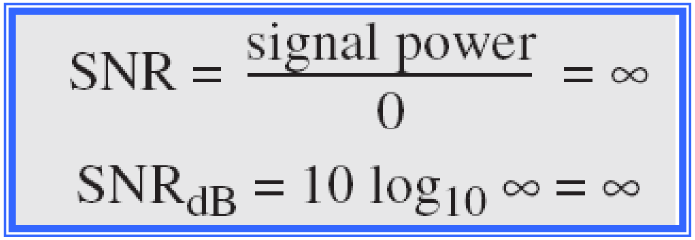


$$\text{SNR} = \frac{10{,}000\ \mu\text{W}}{1\ \text{mW}} = 10{,}000$$

$$\text{SNR}_{\text{dB}} = 10 \log_{10} 10{,}000 = 10 \log_{10} 10^4 = 40$$

Example 3.32

The values of SNR and SNRdB for a noiseless channel are

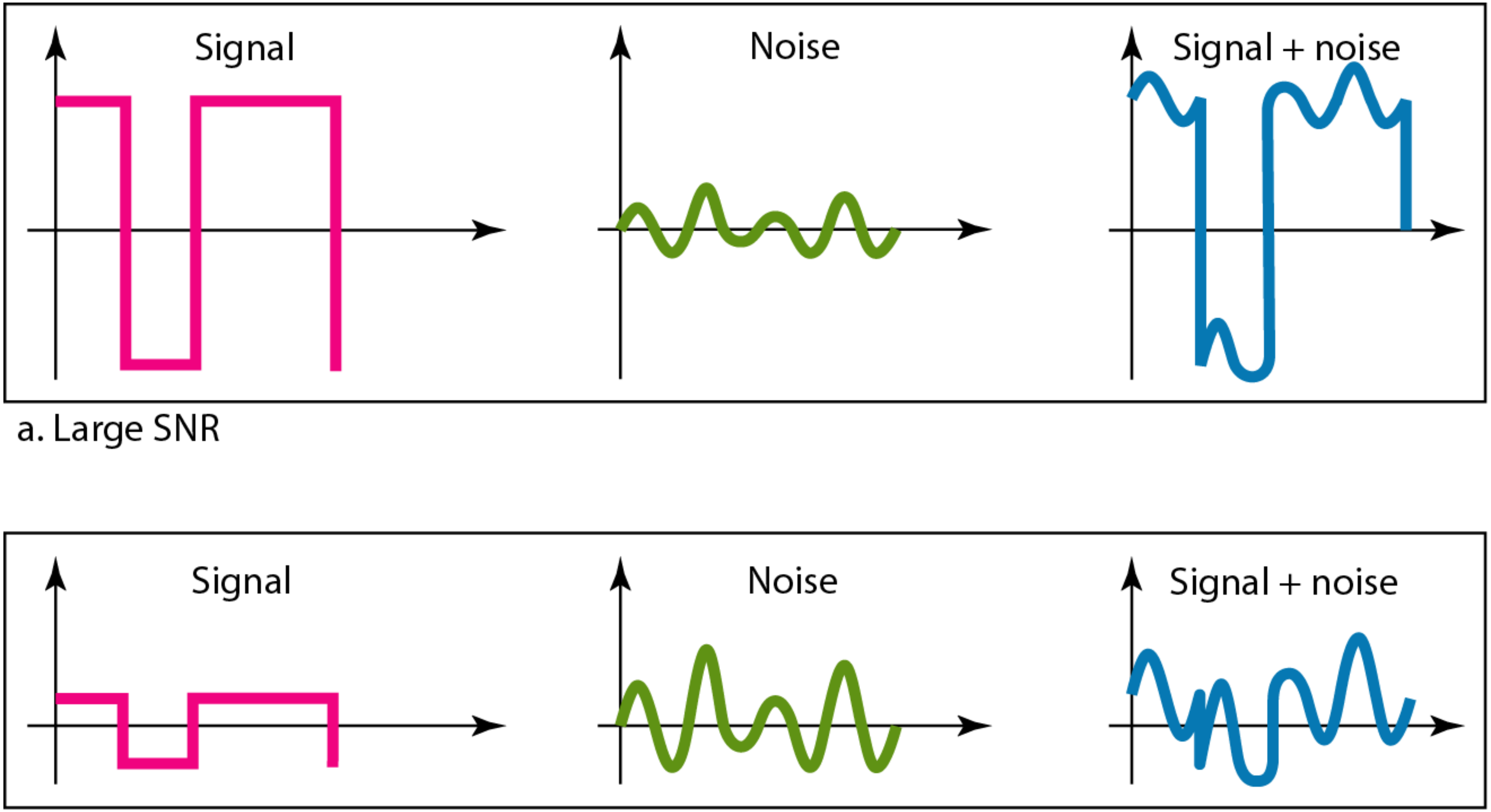


$$\mathrm{SNR} = \frac{\text{signal power}}{0} = \infty$$

$$\mathrm{SNR_{dB}} = 10 \log_{10} \infty = \infty$$

We can never achieve this ratio in real life; it is an ideal.

Figure 3.30 Two cases of SNR: a high SNR and a low SNR



a. Large SNR

b. Small SNR

Transmission Media

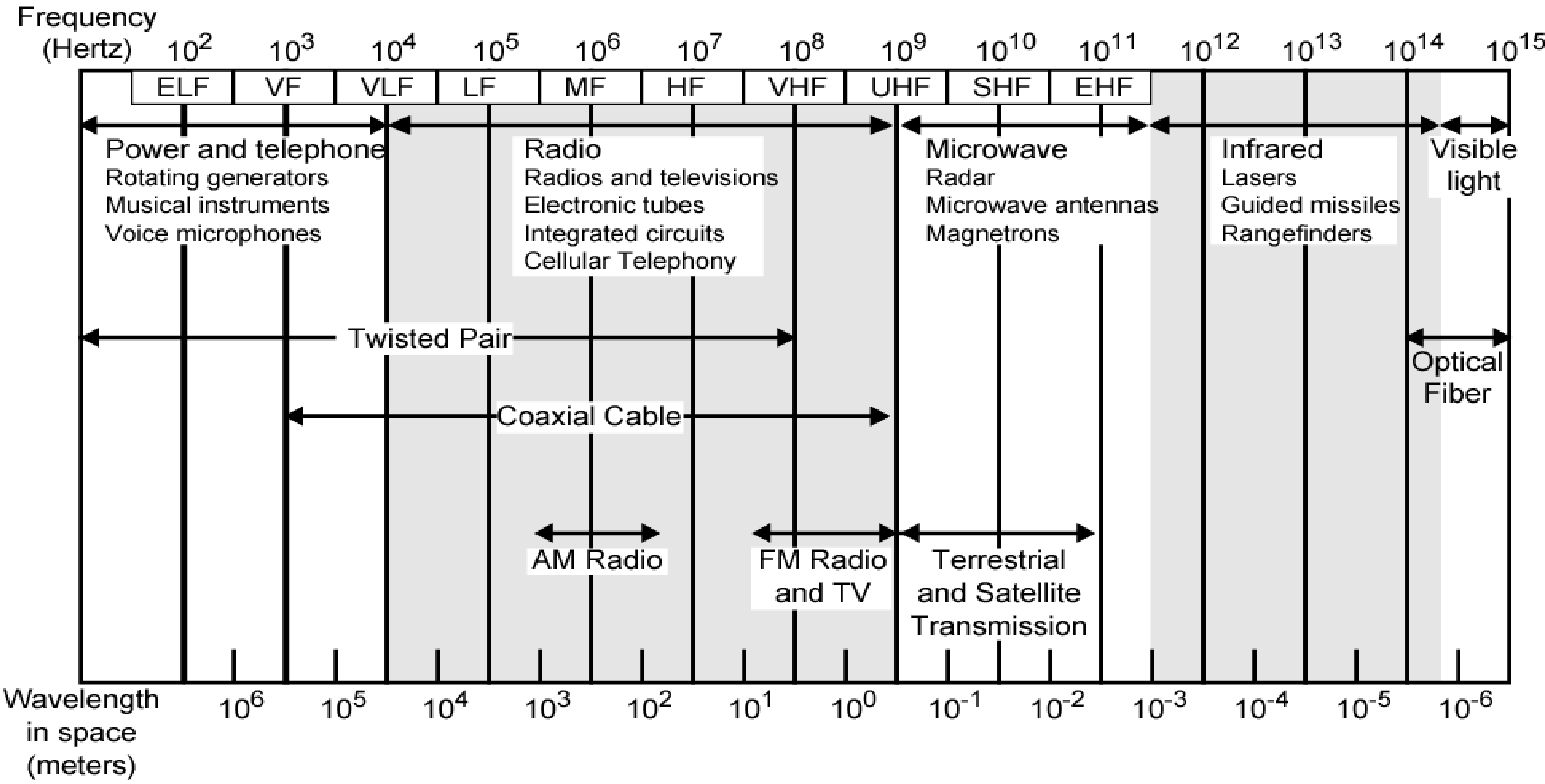
Overview

- Guided - wire
- Unguided - wireless
- Characteristics and quality determined by medium and signal
- For guided, the medium is more important
- For unguided, the bandwidth produced by the antenna is more important
- Key concerns are data rate and distance

Design Factors

- Bandwidth
  - Higher bandwidth gives higher data rate
- Transmission impairments
  - Attenuation
- Interference
- Number of receivers
  - In guided media
  - More receivers (multi-point) introduce more attenuation

Electromagnetic Spectrum



ELF = Extremely low frequency
VF = Voice frequency
VLF = Very low frequency
LF = Low frequency
MF = Medium frequency
HF = High frequency
VHF = Very high frequency
UHF = Ultrahigh frequency
SHF = Superhigh frequency
EHF = Extremely high frequency

Guided Transmission Media

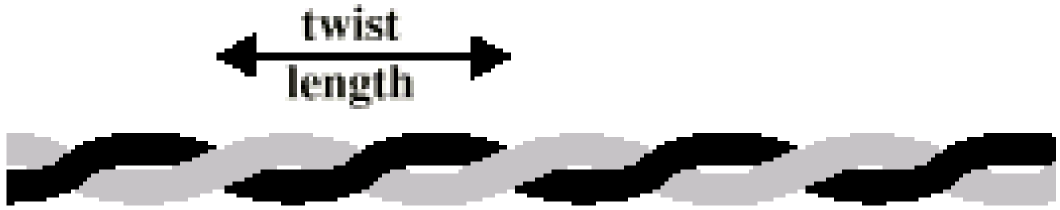
- Twisted Pair
- Coaxial cable
- Optical fiber

Transmission Characteristics of Guided Media

	Frequency Range	**Typical Attenuation**	**Typical Delay**	**Repeater Spacing**
Twisted pair (with loading)	0 to 3.5 kHz	0.2 dB/km @ 1 kHz	50 µs/km	2 km
Twisted pairs (multi-pair cables)	0 to 1 MHz	0.7 dB/km @ 1 kHz	5 µs/km	2 km
Coaxial cable	0 to 500 MHz	7 dB/km @ 10 MHz	4 µs/km	1 to 9 km
Optical fiber	186 to 370 THz	0.2 to 0.5 dB/km	5 µs/km	40 km

Twisted Pair

—Separately insulated
—Twisted together
—Often "bundled" into cables
—Usually installed in building
during construction



(a) Twisted pair

Twisted Pair - Applications

- Most common medium
- Telephone network
  - Between house and local exchange (subscriber loop)
- Within buildings
  - To private branch exchange (PBX)
- For local area networks (LAN)
  - 10Mbps or 100Mbps

Twisted Pair - Pros and Cons

- Cheap
- Easy to work with
- Low data rate
- Short range

Twisted Pair - Transmission Characteristics

- Analog
  - Amplifiers every 5km to 6km
- Digital
  - Use either analog or digital signals
  - repeater every 2km or 3km
- Limited distance
- Limited bandwidth (1MHz)
- Limited data rate (100MHz)
- Susceptible to interference and noise

Near End Crosstalk

- Coupling of signal from one pair to another
- Coupling takes place when transmit signal entering the link couples back to receiving pair
- i.e. near transmitted signal is picked up by near receiving pair

Unshielded and Shielded TP

- Unshielded Twisted Pair (UTP)
  - – Ordinary telephone wire
  - – Cheapest
  - – Easiest to install
  - – Suffers from external EM interference
- Shielded Twisted Pair (STP)
  - – Metal braid or sheathing that reduces interference
  - – More expensive
  - – Harder to handle (thick, heavy)

UTP Categories

- Cat 3
  - up to 16MHz
  - Voice grade found in most offices
  - Twist length of 7.5 cm to 10 cm
- Cat 4
  - up to 20 MHz
- Cat 5
  - up to 100MHz
  - Commonly pre-installed in new office buildings
  - Twist length 0.6 cm to 0.85 cm
- Cat 5E (Enhanced) –see tables
- Cat 6
- Cat 7

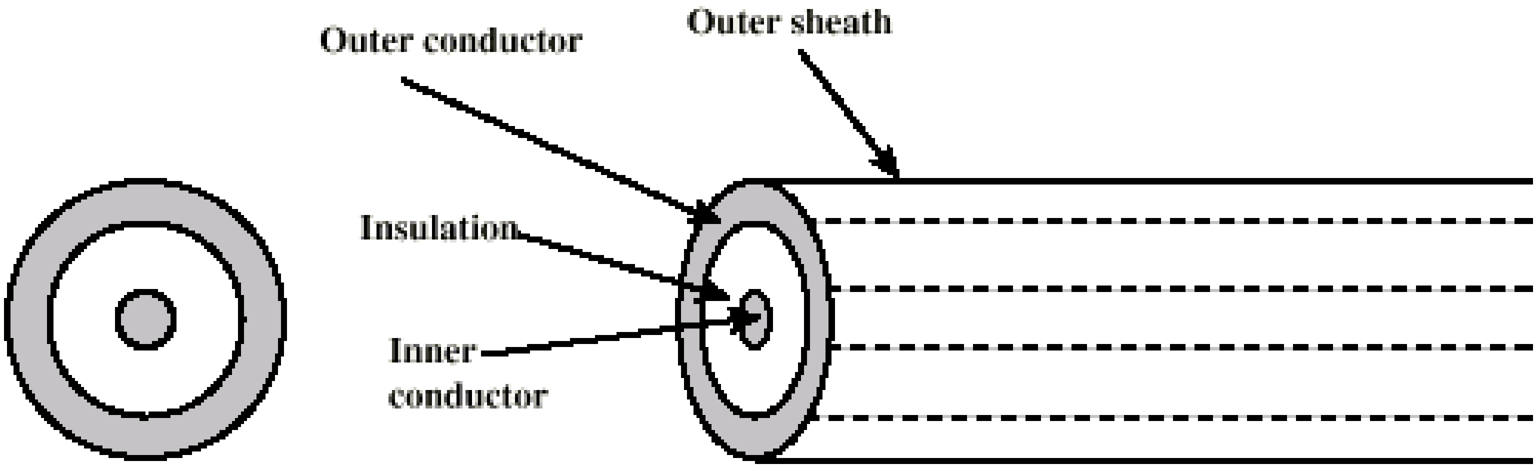
Comparison of Shielded and Unshielded Twisted Pair

	Attenuation (dB per 100 m)			Near-end Crosstalk (dB)		
Frequency (MHz)	Category 3 UTP	Category 5 UTP	150-ohm STP	Category 3 UTP	Category 5 UTP	150-ohm STP
1	2.6	2.0	1.1	41	62	58
4	5.6	4.1	2.2	32	53	58
16	13.1	8.2	4.4	23	44	50.4
25	—	10.4	6.2	—	41	47.5
100	—	22.0	12.3	—	32	38.5
300	—	—	21.4	—	—	31.3

Twisted Pair Categories and Classes

	Category 3 Class C	**Category 5 Class D**	**Category 5E**	**Category 6 Class E**	**Category 7 Class F**
Bandwidth	16 MHz	100 MHz	100 MHz	200 MHz	600 MHz
Cable Type	UTP	UTP/FTP	UTP/FTP	UTP/FTP	SSTP
Link Cost (Cat 5 =1)	0.7	1	1.2	1.5	2.2

Coaxial Cable



—Outer conductor is braided shield
—Inner conductor is solid metal
—Separated by insulating material
—Covered by padding

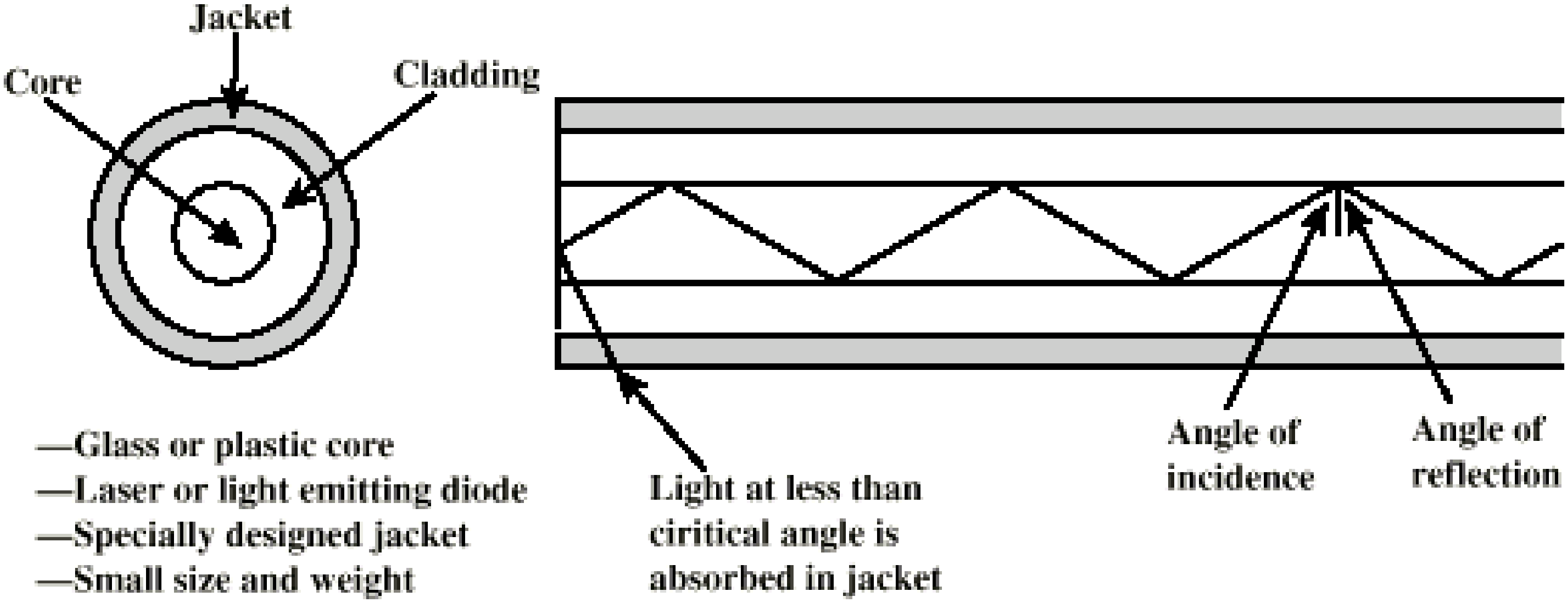
Coaxial Cable Applications

- Most versatile medium
- Television distribution
  - Ariel to TV
  - Cable TV
- Long distance telephone transmission
  - Can carry 10,000 voice calls simultaneously
  - Being replaced by fiber optic
- Short distance computer systems links
- Local area networks

Coaxial Cable - Transmission Characteristics

- Analog
  - Amplifiers every few km
  - Closer if higher frequency
  - Up to 500MHz
- Digital
  - Repeater every 1km
  - Closer for higher data rates

Optical Fiber



—Glass or plastic core
—Laser or light emitting diode
—Specially designed jacket
—Small size and weight

Light at less than ciritical angle is absorbed in jacket

Optical Fiber - Benefits

- Greater capacity
  - Data rates of hundreds of Gbps
- Smaller size & weight
- Lower attenuation
- Electromagnetic isolation
- Greater repeater spacing
  - 10s of km at least

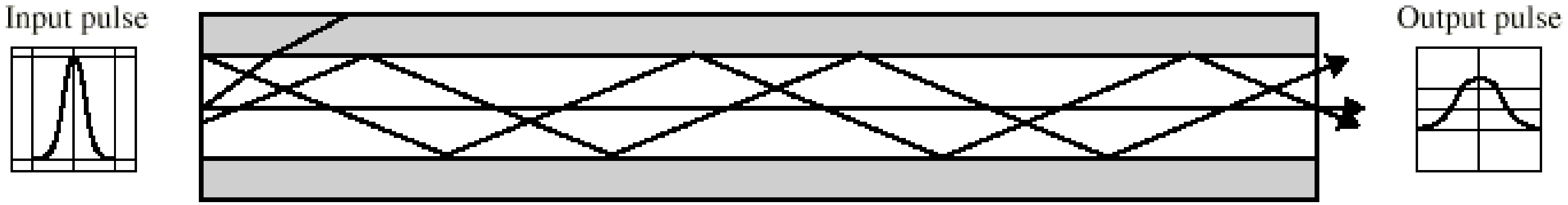
Optical Fiber - Applications

- Long-haul trunks
- Metropolitan trunks
- Rural exchange trunks
- Subscriber loops
- LANs

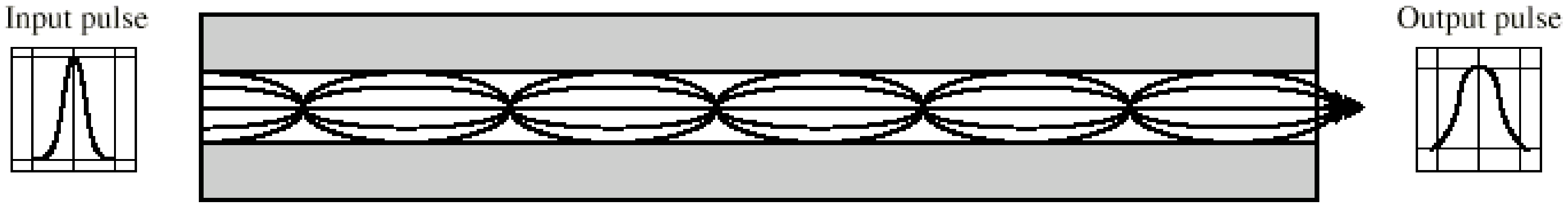
Optical Fiber - Transmission Characteristics

- Act as wave guide for 10^{14} to 10^{15} Hz
  - Portions of infrared and visible spectrum
- Light Emitting Diode (LED)
  - Cheaper
  - Wider operating temp range
  - Last longer
- Injection Laser Diode (ILD)
  - More efficient
  - Greater data rate
- Wavelength Division Multiplexing

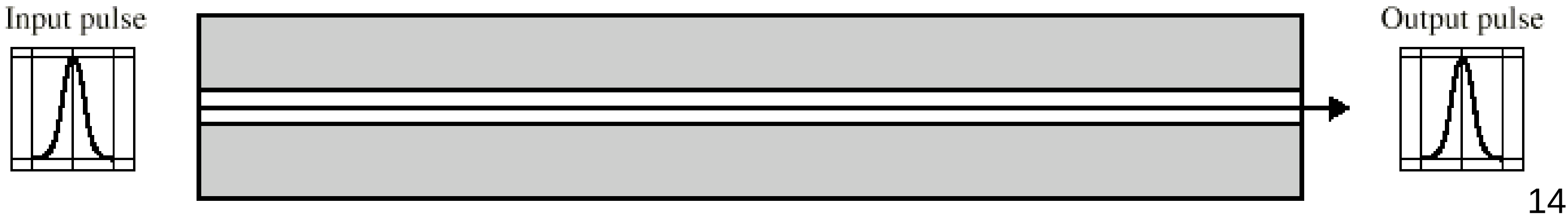
Optical Fiber Transmission Modes



(a) Step-index multimode



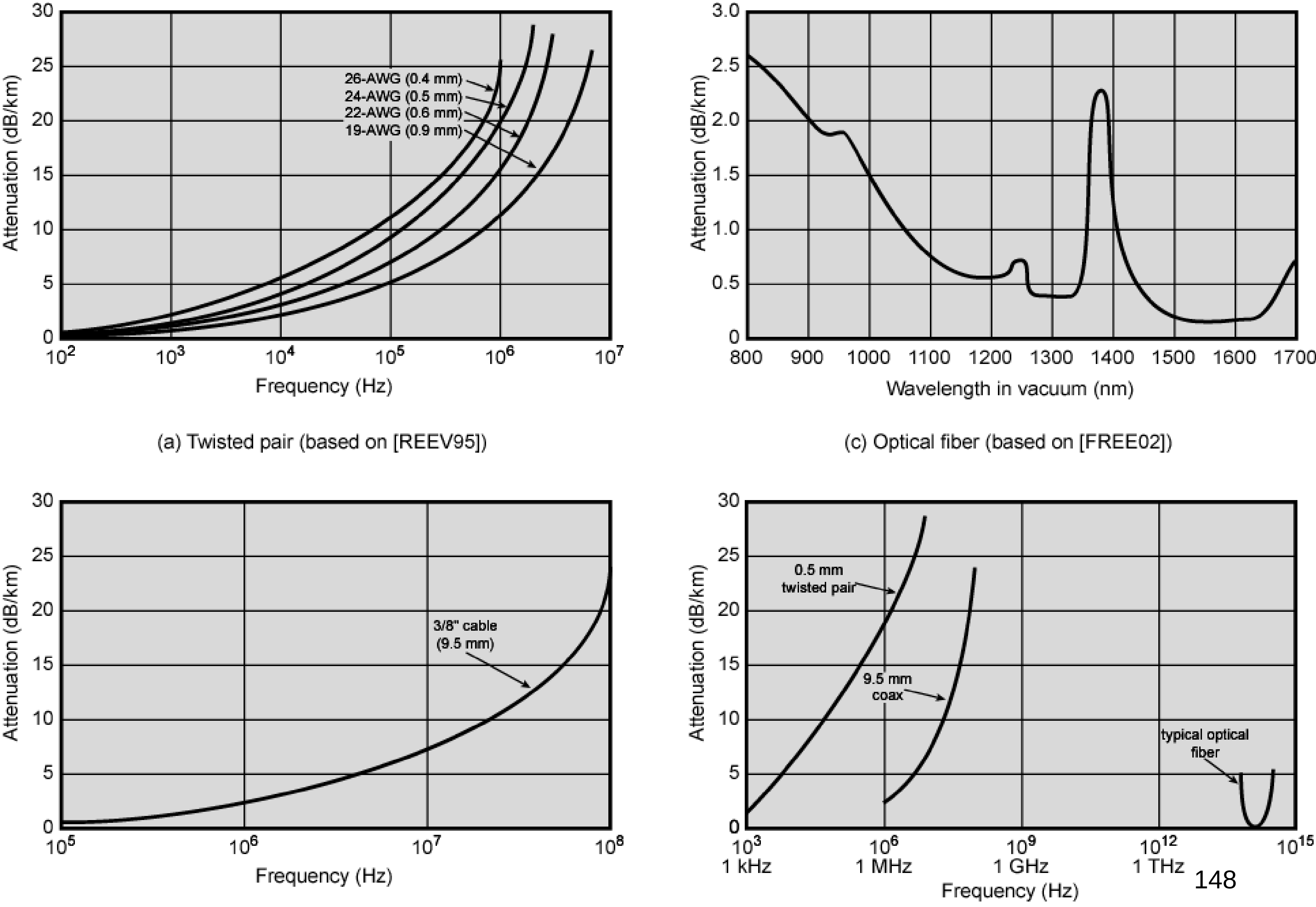
(b) Graded-index multimode



(c) Single mode

Frequency Utilization for Fiber Applications

Wavelength (in vacuum) range (nm)	Frequency range (THz)	Band label	Fiber type	Application
820 to 900	366 to 333		Multimode	LAN
1280 to 1350	234 to 222	S	Single mode	Various
1528 to 1561	196 to 192	C	Single mode	WDM
1561 to 1620	185 to 192	L	Single mode	WDM



(a) Twisted pair (based on [REEV95])

(c) Optical fiber (based on [FREE02])

(b) Coaxial cable (based on [BELL90])

(d) Composite graph

Wireless Transmission Frequencies

- 2GHz to 40GHz
  - Microwave
  - Highly directional
  - Point to point
  - Satellite
- 30MHz to 1GHz
  - Omnidirectional
  - Broadcast radio
- 3×10^{11} to 2×10^{14}
  - Infrared
  - Local

UNIT II

INTRODUCTION TO DATA LINK LAYER

Link layer

our goals:

- understand principles behind link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
  - local area networks: Ethernet, VLANs
- instantiation, implementation of various link layer technologies

Link layer, LANs: outline

5.1 introduction, services

5.2 error detection, correction

5.3 multiple access protocols

5.4 link-layer addressing

5.5 Ethernet, LANs

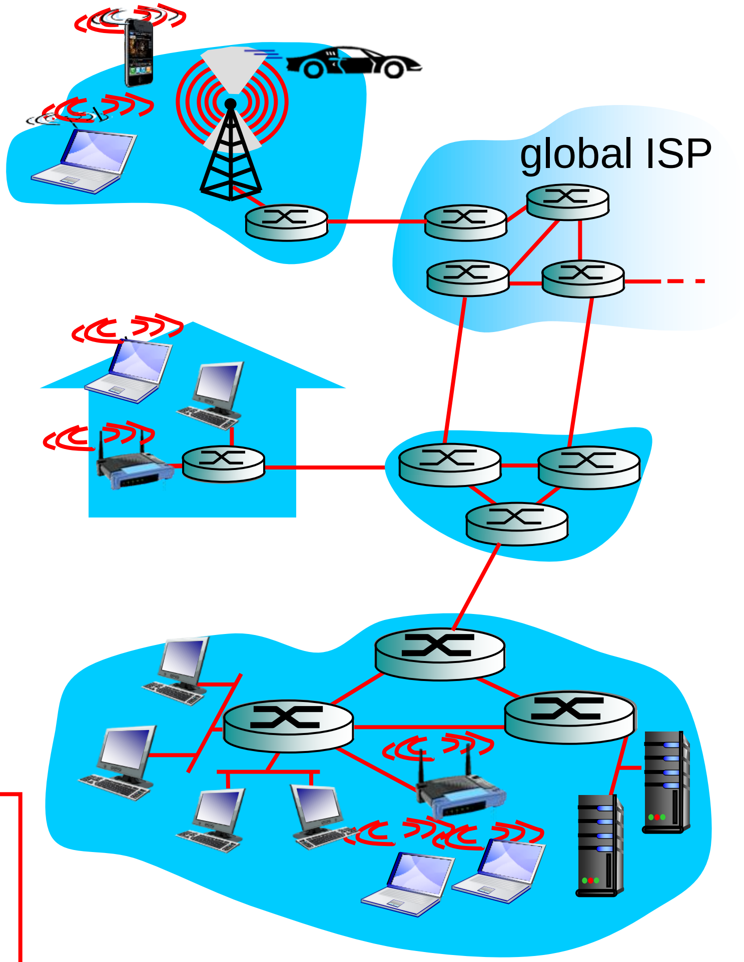
5.6 LAN switches

5.7 a day in the life of a web request

Link layer: introduction

terminology:

- hosts and routers: nodes
- communication channels that connect adjacent nodes along communication path: links
  - wired links
  - wireless links
  - LANs
- layer-2 packet: frame, encapsulates datagram



data-link layer has responsibility of transferring datagram from one node to physically adjacent node over a link

Link layer: context

- datagram transferred by different link protocols over different links:
  - e.g., Ethernet on first link, frame relay on intermediate links, 802.11 on last link
- each link protocol provides different services
  - e.g., may or may not provide rdt over link

transportation analogy:

- trip from Princeton to Lausanne
  - limo: Princeton to JFK
  - plane: JFK to Geneva
  - train: Geneva to Lausanne
- tourist = datagram
- transport segment = communication link
- transportation mode = link layer protocol
- travel agent = routing algorithm

Link layer services

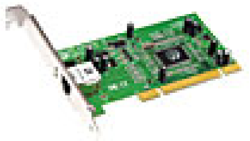
- *framing, link access:*
  - encapsulate datagram into frame, adding header, trailer
  - channel access if shared medium
  - "MAC" addresses used in frame headers to identify source, dest
    - different from IP address!
- *reliable delivery between adjacent nodes*
  - we learned how to do this already (chapter 3)!
  - seldom used on low bit-error link (fiber, some twisted pair)
  - wireless links: high error rates
    - *Q:* why both link-level and end-end reliability?

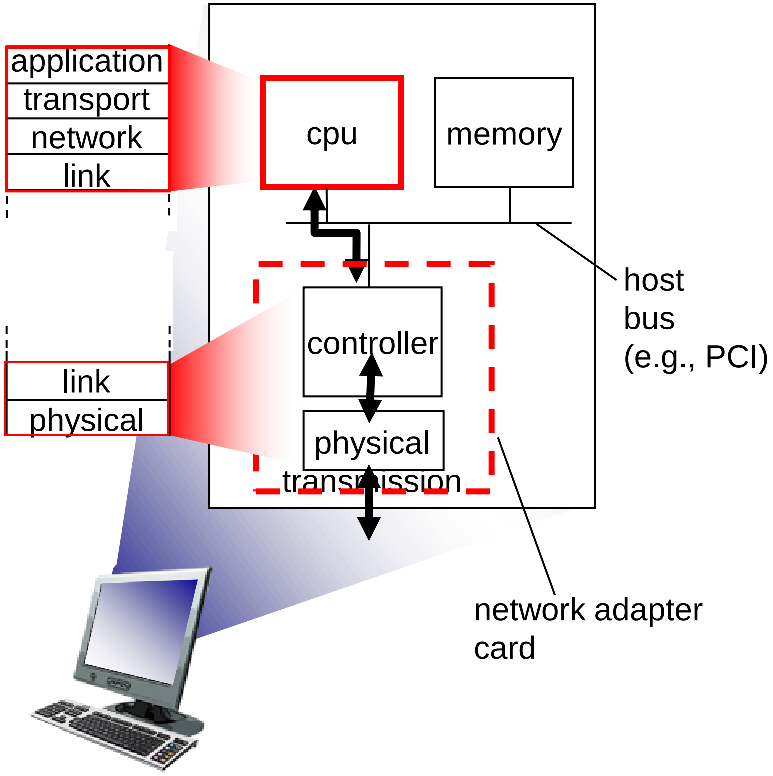
Link layer services (more)

- *flow control:*
  - pacing between adjacent sending and receiving nodes
- *error detection*:
  - errors caused by signal attenuation, noise.
  - receiver detects presence of errors:
    - signals sender for retransmission or drops frame
- *error correction:*
  - receiver identifies *and corrects* bit error(s) without resorting to retransmission
- *half-duplex and full-duplex*
  - with half duplex, nodes at both ends of link can transmit, but not at same time

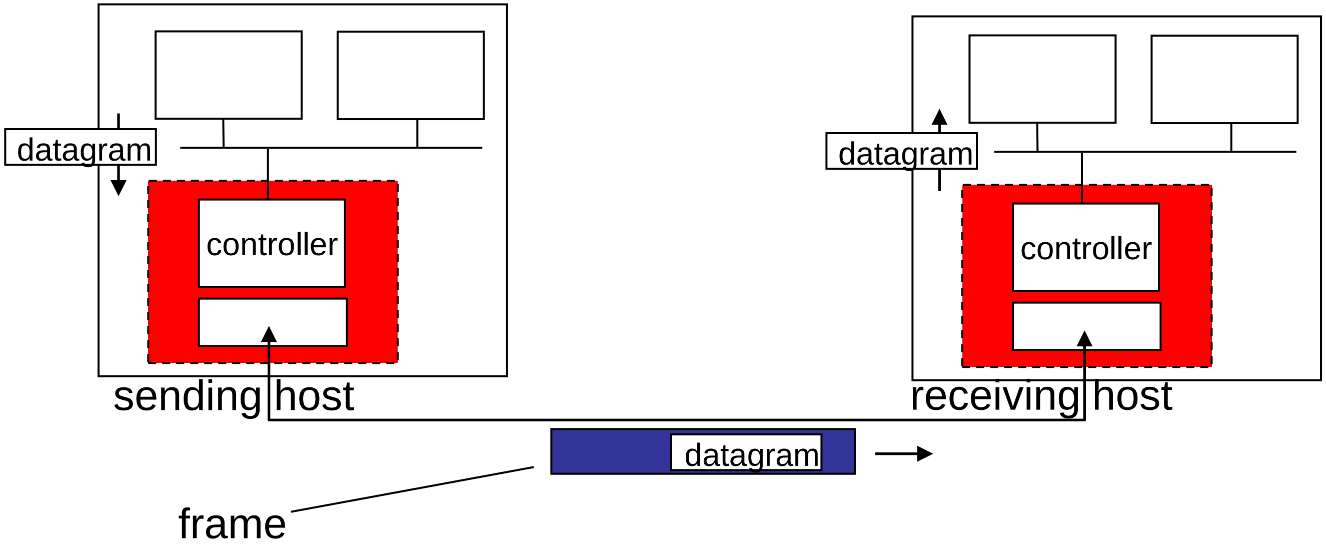
Where is the link layer implemented?

- in each and every host
- link layer implemented in "adaptor" (aka *network interface card* NIC)
  - Ethernet card, 802.11 card
  - implements link, physical layer
- attaches into host's system buses
- combination of hardware, software, firmware





Adaptors communicating



- sending side:
  - encapsulates datagram in frame
  - adds error checking bits, rdt, flow control, etc.
- receiving side
  - looks for errors, rdt, flow control, etc
  - extracts datagram, passes to upper layer at receiving side

Link layer, LANs: outline

5.1 introduction, services

5.2 error detection, correction

5.3 multiple access protocols

5.4 link-layer addressing

5.5 Ethernet, LANs

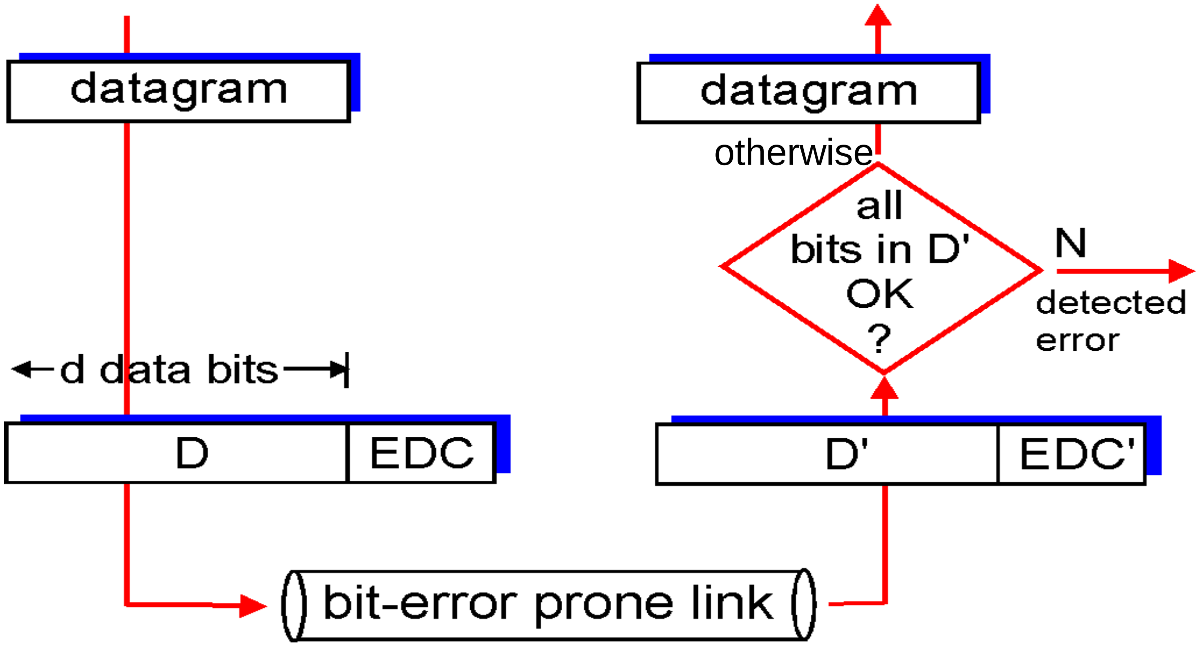
5.6 LAN switches

5.7 a day in the life of a web request

Error detection

EDC= Error Detection and Correction bits (redundancy)
D = Data protected by error checking, may include header fields

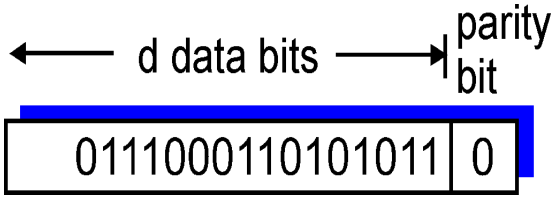
- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction



Parity checking

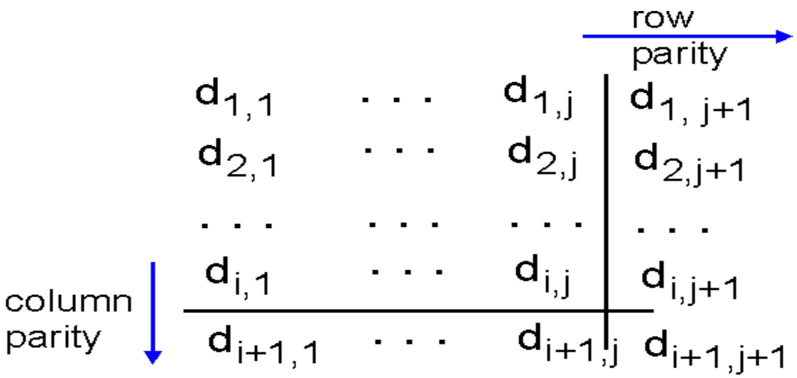
single bit parity:

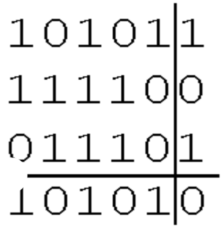
- detect single bit errors



two-dimensional bit parity:

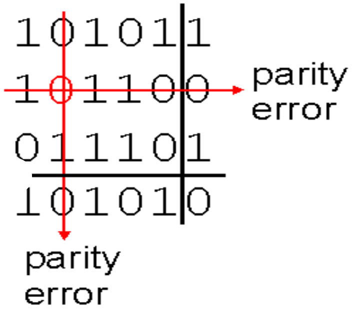
- detect and correct single bit errors


row parity
column parity
$d_{1,1}$ . . . $d_{1,j}$ $d_{1,\ j+1}$
$d_{2,1}$ . . . $d_{2,j}$ $d_{2,j+1}$
. . . .
$d_{i,1}$ . . . $d_{i,j}$ $d_{i,j+1}$
$d_{i+1,1}$ . . . $d_{i+1,j}$ $d_{i+1,j+1}$



```
10101|1
11110|0
01110|1
------
10101|0
```

no errors



```
10101|1
10110|0
01110|1
------
10101|0
```

correctable single bit error

Internet checksum (review)

goal: detect "errors" (e.g., flipped bits) in transmitted packet (note: used at transport layer only)

sender:

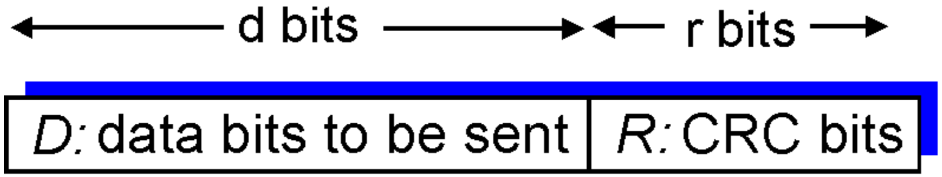
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP checksum field

receiver:

- compute checksum of received segment
- check if computed checksum equals checksum field value:
  - NO - error detected
  - YES - no error detected. *But maybe errors nonetheless?*

Cyclic redundancy check

- more powerful error-detection coding
- view data bits, D, as a binary number
- choose r+1 bit pattern (generator), G
- goal: choose r CRC bits, R, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!
  - can detect all burst errors less than r+1 bits
- widely used in practice (Ethernet, 802.11 WiFi, ATM)



d bits

r bits

D: data bits to be sent

R: CRC bits

bit pattern

$D * 2^r$ XOR R

mathematical formula

CRC example

want:

$D{\cdot}2^r$ XOR R = nG

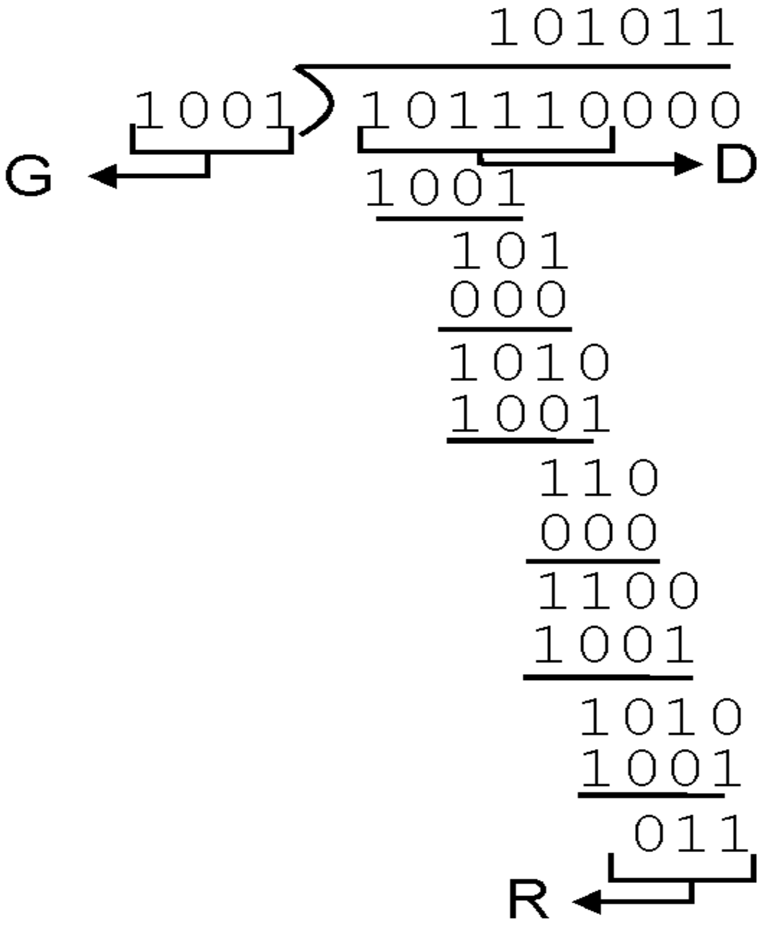
equivalently:

$D{\cdot}2^r$ = nG XOR R

equivalently:

if we divide $D{\cdot}2^r$ by G, want remainder R to satisfy:

$$R = \text{remainder}\left[\frac{D{\cdot}2^r}{G}\right]$$



Cyclic Redundancy Check (CRC)

- Properties of Generator Polynomial
  - In general, it is possible to prove that the following types of errors can be detected by a G(x) with the stated properties
    - All single-bit errors, as long as the x^k and x^0 terms have nonzero coefficients.
    - All double-bit errors, as long as G(x) has a factor with at least three terms.
    - Any odd number of errors, as long as G(x) contains the factor (x+1).
    - Any “burst” error (i.e., sequence of consecutive error bits) for which the length of the burst is less than *k* bits. (Most burst errors of larger than *k* bits can also be detected.)

Cyclic Redundancy Check (CRC)

- Six generator polynomials that have become international standards are:
  - CRC-8 = x^8+x^2+x+1
  - CRC-10 = $x^{10}+x^9+x^5+x^4+x+1$
  - CRC-12 = $x^{12}+x^{11}+x^3+x^2+x+1$
  - CRC-16 = $x^{16}+x^{15}+x^2+1$
  - CRC-CCITT = $x^{16}+x^{12}+x^5+1$
  - CRC-32 = $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

Link layer, LANs: outline

5.1 introduction, services

5.2 error detection, correction

5.3 multiple access protocols

5.4 link-layer addressing

5.5 Ethernet, LANs

5.6 LAN switches

5.7 a day in the life of a web request

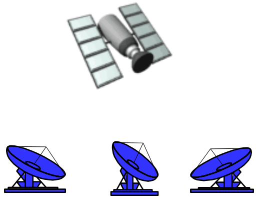
Multiple access links, protocols

two types of “links”:

- point-to-point
  - – PPP for dial-up access
  - – point-to-point link between Ethernet switch, host
- *broadcast (shared wire or medium)*
  - – old-fashioned Ethernet
  - – upstream HFC
  - – 802.11 wireless LAN

shared wire (e.g., cabled Ethernet)

shared RF (e.g., 802.11 WiFi)



shared RF (satellite)

humans at a cocktail party (shared air, acoustical)

Multiple access protocols

- single shared broadcast channel
- two or more simultaneous transmissions by nodes: interference
  - *collision* if node receives two or more signals at the same time

multiple access protocol

- distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit
- communication about channel sharing must use channel itself!
  - no out-of-band channel for coordination

An ideal multiple access protocol

given: broadcast channel of rate R bps

desiderata:

1. when one node wants to transmit, it can send at rate R.
2. when M nodes want to transmit, each can send at average rate R/M
3. fully decentralized:
   - no special node to coordinate transmissions
   - no synchronization of clocks, slots
4. simple

MAC protocols: taxonomy

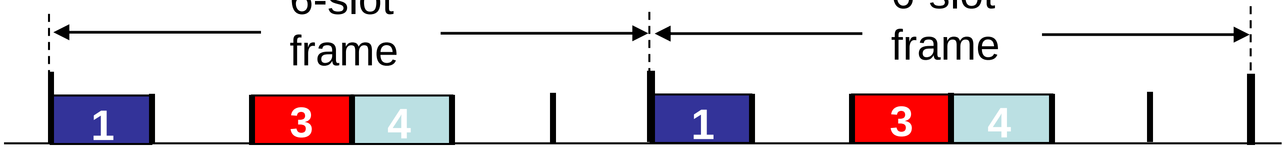
three broad classes:

- *channel partitioning*
  - divide channel into smaller "pieces" (time slots, frequency, code)
  - allocate piece to node for exclusive use
- *random access*
  - channel not divided, allow collisions
  - "recover" from collisions
- *"taking turns"*
  - nodes take turns, but nodes with more to send can take longer turns

Channel partitioning MAC protocols: TDMA

TDMA: time division multiple access

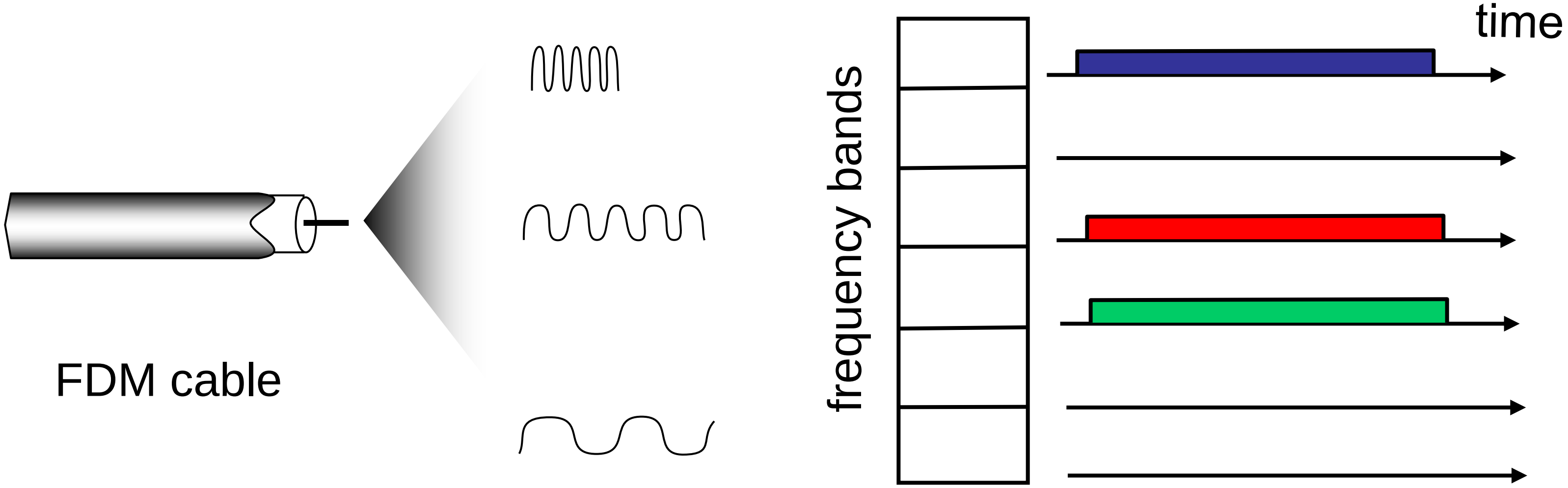
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



Channel partitioning MAC protocols: FDMA

FDMA: frequency division multiple access

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



Random access protocols

- when node has packet to send
  - transmit at full channel data rate R.
  - no *a priori* coordination among nodes
- two or more transmitting nodes ➜ "collision",
- random access MAC protocol specifies:
  - how to detect collisions
  - how to recover from collisions (e.g., via delayed retransmissions)
- examples of random access MAC protocols:
  - slotted ALOHA
  - ALOHA
  - CSMA, CSMA/CD, CSMA/CA

Slotted ALOHA

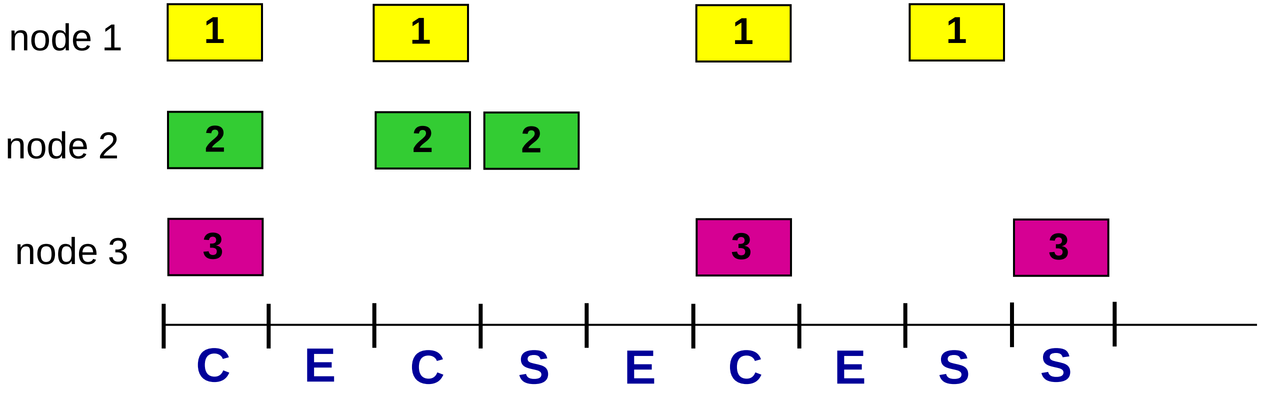
assumptions:

- all frames same size
- time divided into equal size slots (time to transmit 1 frame)
- nodes start to transmit only slot beginning
- nodes are synchronized
- if 2 or more nodes transmit in slot, all nodes detect collision

operation:

- when node obtains fresh frame, transmits in next slot
  - *if no collision:* node can send new frame in next slot
  - *if collision:* node retransmits frame in each subsequent slot with prob. p until success

Slotted ALOHA



Pros:

- single active node can continuously transmit at full rate of channel
- highly decentralized: only slots in nodes need to be in sync
- simple

Cons:

- collisions, wasting slots
- idle slots
- nodes may be able to detect collision in less than time to transmit packet
- clock synchronization

Slotted ALOHA: efficiency

efficiency: long-run fraction of successful slots (many nodes, all with many frames to send)

- *suppose: N* nodes with many frames to send, each transmits in slot with probability *p*
- prob that given node has success in a slot = $p(1-p)^{N-1}$
- prob that *any* node has a success = $Np(1-p)^{N-1}$

- max efficiency: find p* that maximizes $Np(1-p)^{N-1}$
- for many nodes, take limit of $Np^*(1-p^*)^{N-1}$ as N goes to infinity, gives:

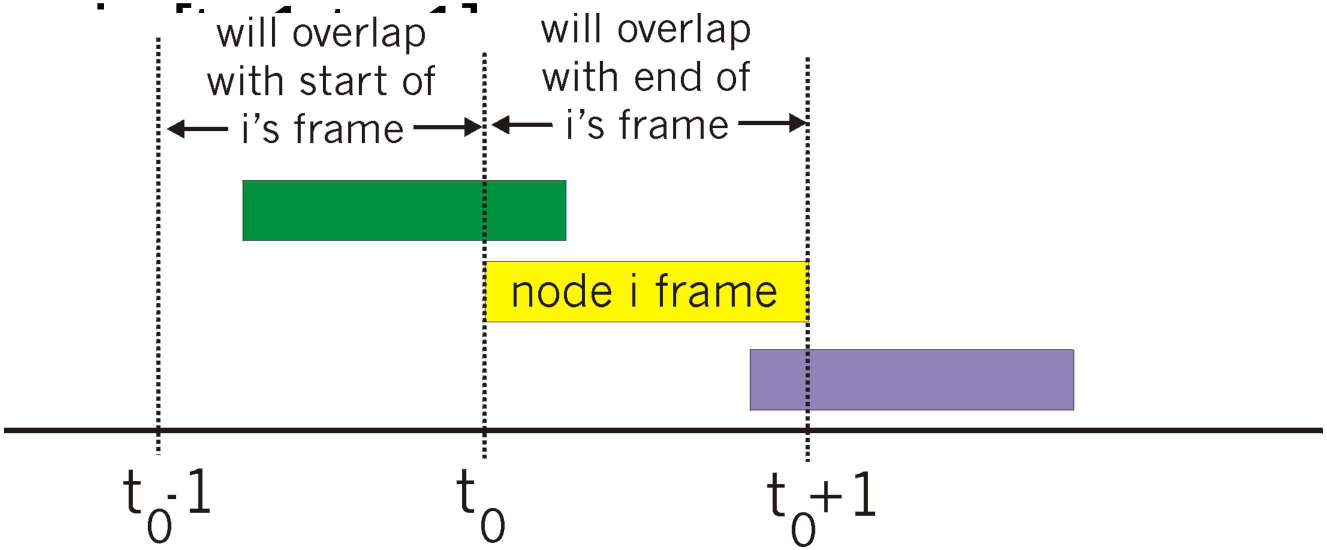
max efficiency = 1/e = .37

at best: channel used for useful transmissions 37% of time!

!

Pure (unslotted) ALOHA

- unslotted Aloha: simpler, no synchronization
- when frame first arrives
  - transmit immediately
- collision probability increases:
  - frame sent at t_0 collides with other frames sent in [t_0-1,t_0+1]


will overlap with start of i's frame
will overlap with end of i's frame
node i frame
t_0-1
t_0
t_0+1

Pure ALOHA efficiency

P(success by given node) = P(node transmits) ·

P(no other node transmits in $[t_0-1,t_0]$ ·

P(no other node transmits in $[t_0-1,t_0]$

$$= p \cdot (1-p)^{N-1} \cdot (1-p)^{N-1}$$

$$= p \cdot (1-p)^{2(N-1)}$$

… choosing optimum p and then letting n $\longrightarrow \infty$

$$= 1/(2e) = .18$$

even worse than slotted Aloha!

CSMA (carrier sense multiple access)

CSMA: listen before transmit:

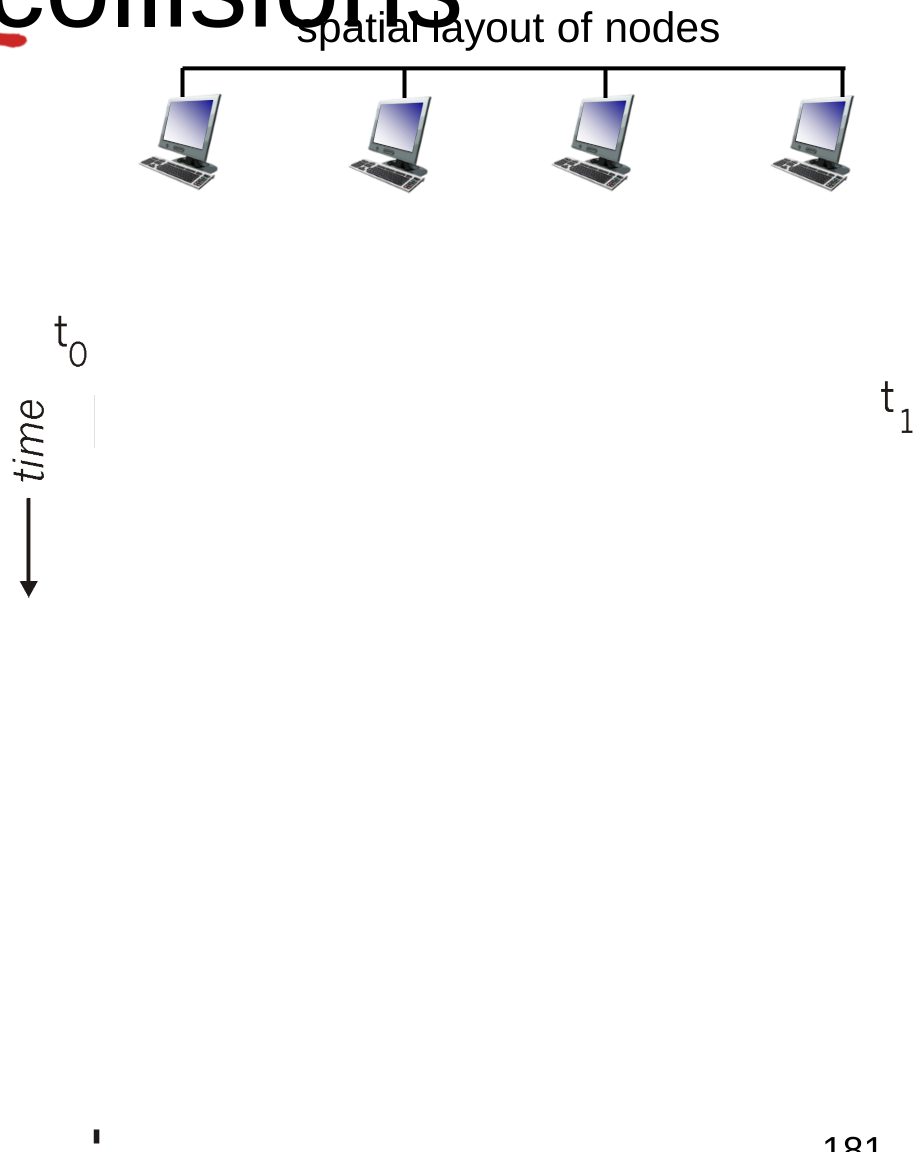
if channel sensed idle: transmit entire frame

- if channel sensed busy, defer transmission

- human analogy: don’t interrupt others!

CSMA collisions

- collisions *can* still occur: propagation delay means two nodes may not hear each other’s transmission
- collision: entire packet transmission time wasted
  - distance & propagation delay play role in in determining collision probability



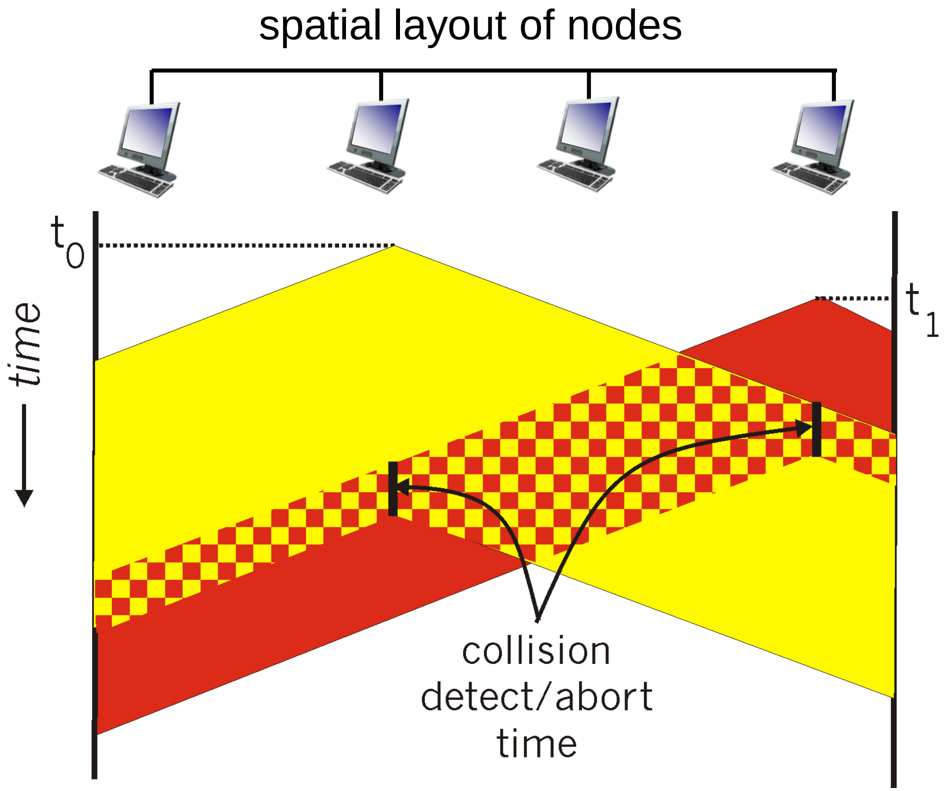
CSMA/CD (collision detection)

CSMA/CD: carrier sensing, deferral as in CSMA

- collisions *detected* within short time
- colliding transmissions aborted, reducing channel wastage

- collision detection:
  - easy in wired LANs: measure signal strengths, compare transmitted, received signals
  - difficult in wireless LANs: received signal strength overwhelmed by local transmission strength
- human analogy: the polite conversationalist

CSMA/CD (collision detection)



“Taking turns” MAC protocols

channel partitioning MAC protocols:

- share channel *efficiently* and *fairly* at high load
- inefficient at low load: delay in channel access, 1/N bandwidth allocated even if only 1 active node!

random access MAC protocols

- efficient at low load: single node can fully utilize channel
- high load: collision overhead

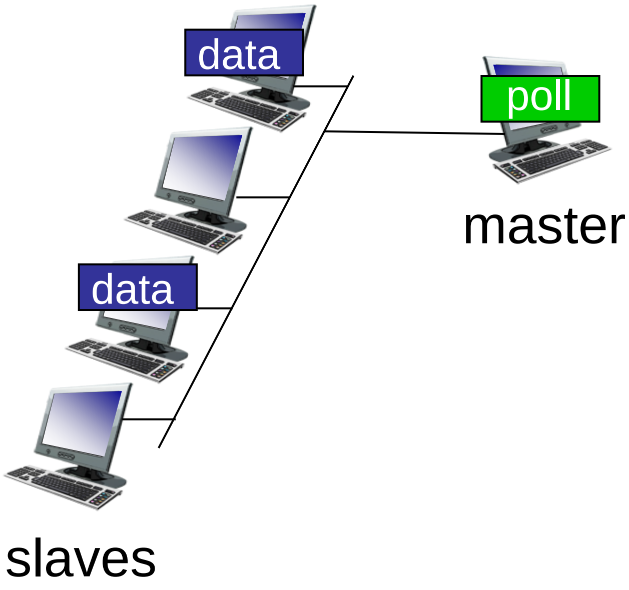
“taking turns” protocols

look for best of both worlds!

“Taking turns” MAC protocols

polling:

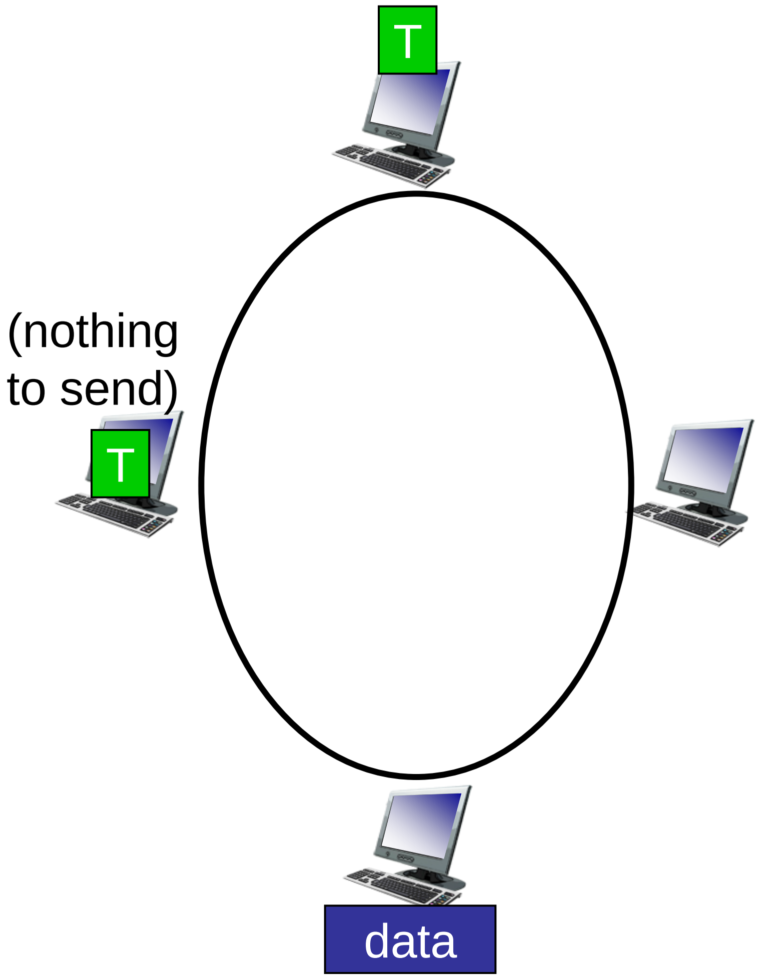
- master node “invites” slave nodes to transmit in turn
- typically used with “dumb” slave devices
- concerns:
  - polling overhead
  - latency
  - single point of failure (master)



“Taking turns” MAC protocols

token passing:

- control token passed from one node to next sequentially.
- token message
- concerns:
  - token overhead
  - latency
  - single point of failure (token)



Summary of MAC protocols

- *channel partitioning,* by time, frequency or code
  - Time Division, Frequency Division
- *random access* (dynamic),
  - ALOHA, S-ALOHA, CSMA, CSMA/CD
  - carrier sensing: easy in some technologies (wire), hard in others (wireless)
  - CSMA/CD used in Ethernet
  - CSMA/CA used in 802.11
- *taking turns*
  - polling from central site, token passing
  - bluetooth, FDDI, IBM token ring

Link layer, LANs: outline

5.1 introduction, services

5.2 error detection, correction

5.3 multiple access protocols

5.4 link-layer addressing

5.5 Ethernet, LANs

5.6 LAN switches

5.7 a day in the life of a web request

MAC addresses and ARP

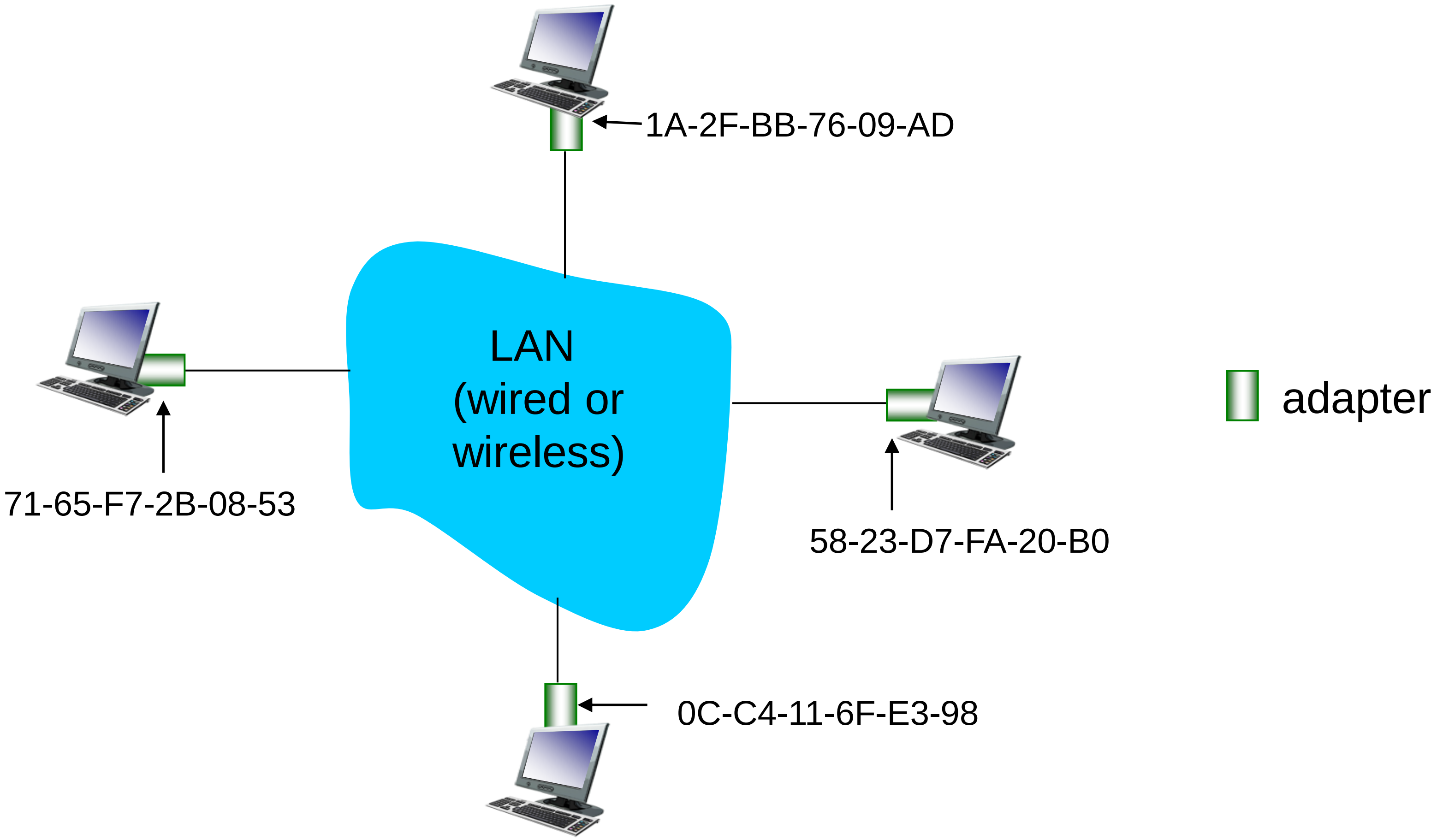
- 32-bit IP address:
  - *network-layer* address
  - datagram to destination used to get IP subnet
- MAC (or LAN or physical or Ethernet) address:
  - function: *get frame from one interface to another physically-connected interface (same network, in IP-addressing sense)*
  - 48 bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
  - e.g.: 1A-2F-BB-76-09-AD

hexadecimal (base 16) notation
(each “number” represents 4 bits)

- Why two addresses for node ??

LAN addresses and ARP

each adapter on LAN has unique LAN address

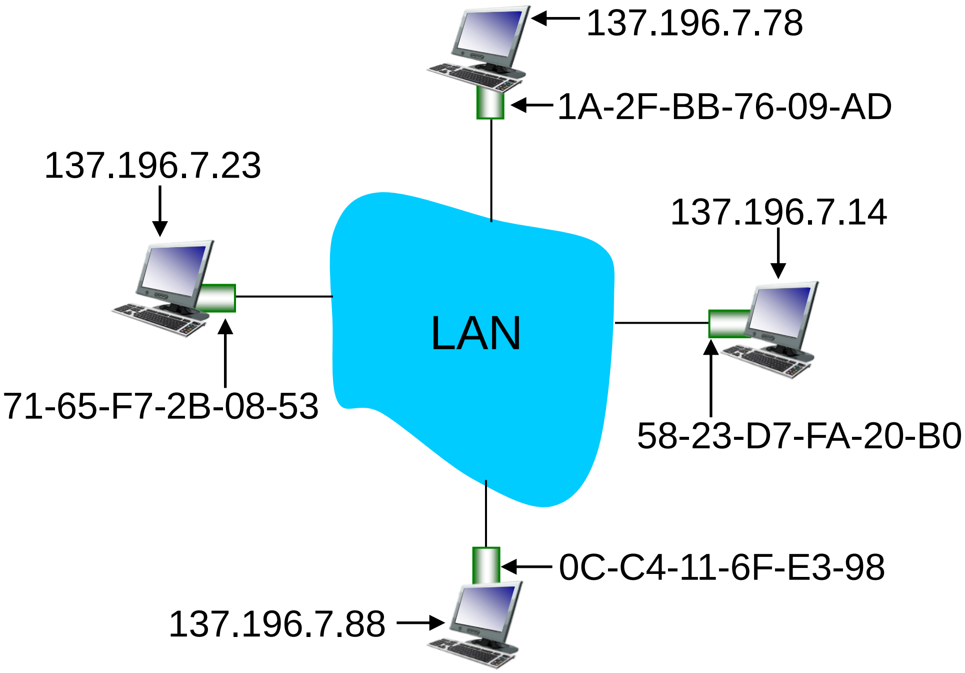


LAN addresses (more)

- MAC address allocation administered by IEEE
- manufacturer buys portion of MAC address space (to assure uniqueness)
- analogy:
  - MAC address: like Social Security Number
  - IP address: like postal address
- MAC flat address ➜ portability
  - can move LAN card from one LAN to another
- IP hierarchical address *not* portable
  - address depends on IP subnet to which node is attached

ARP: address resolution protocol

Question: how to determine MAC address of B knowing B’s IP address?



- each IP node (host, router) on LAN has *ARP* table
  - IP/MAC address mappings for some LAN nodes:

    < IP address; MAC address; TTL>
  - TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)

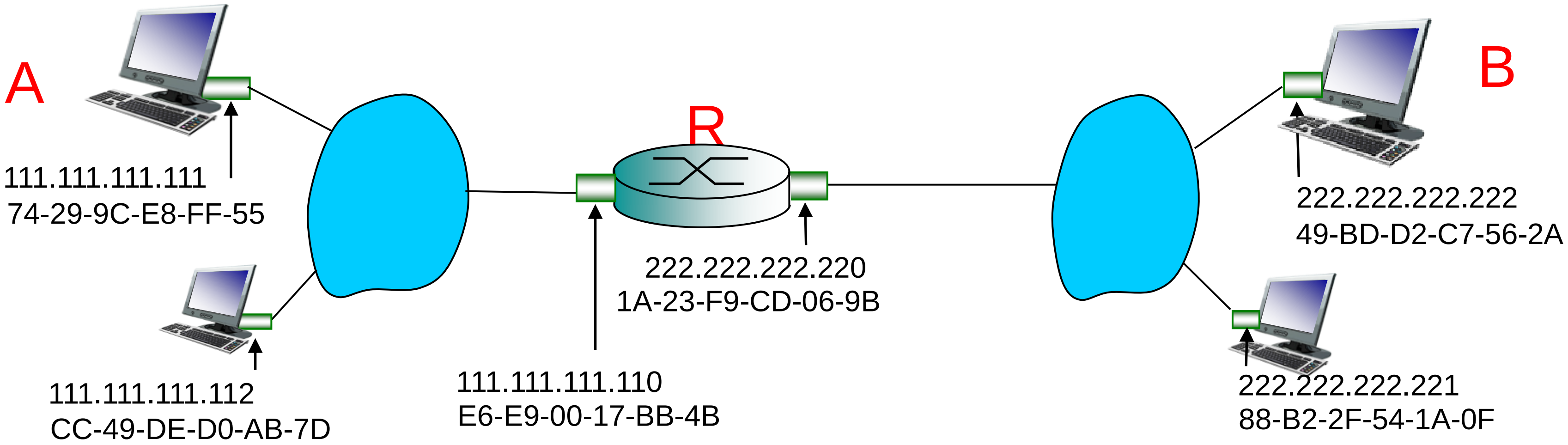
ARP protocol: same LAN

- A wants to send datagram to B
  - B's MAC address not in A's ARP table.
- A broadcasts ARP query packet, containing B's IP address
  - dest MAC address = FF-FF-FF-FF-FF-FF
  - all machines on LAN receive ARP query
- B receives ARP packet, replies to A with its (B's) MAC address
  - frame sent to A's MAC address (unicast)
- A caches (saves) IP-to-MAC address pair in its ARP table until information becomes old (times out)
  - soft state: information that times out (goes away) unless refreshed
- ARP is "plug-and-play":
  - nodes create their ARP tables *without intervention from net administrator*

Addressing: routing to another LAN

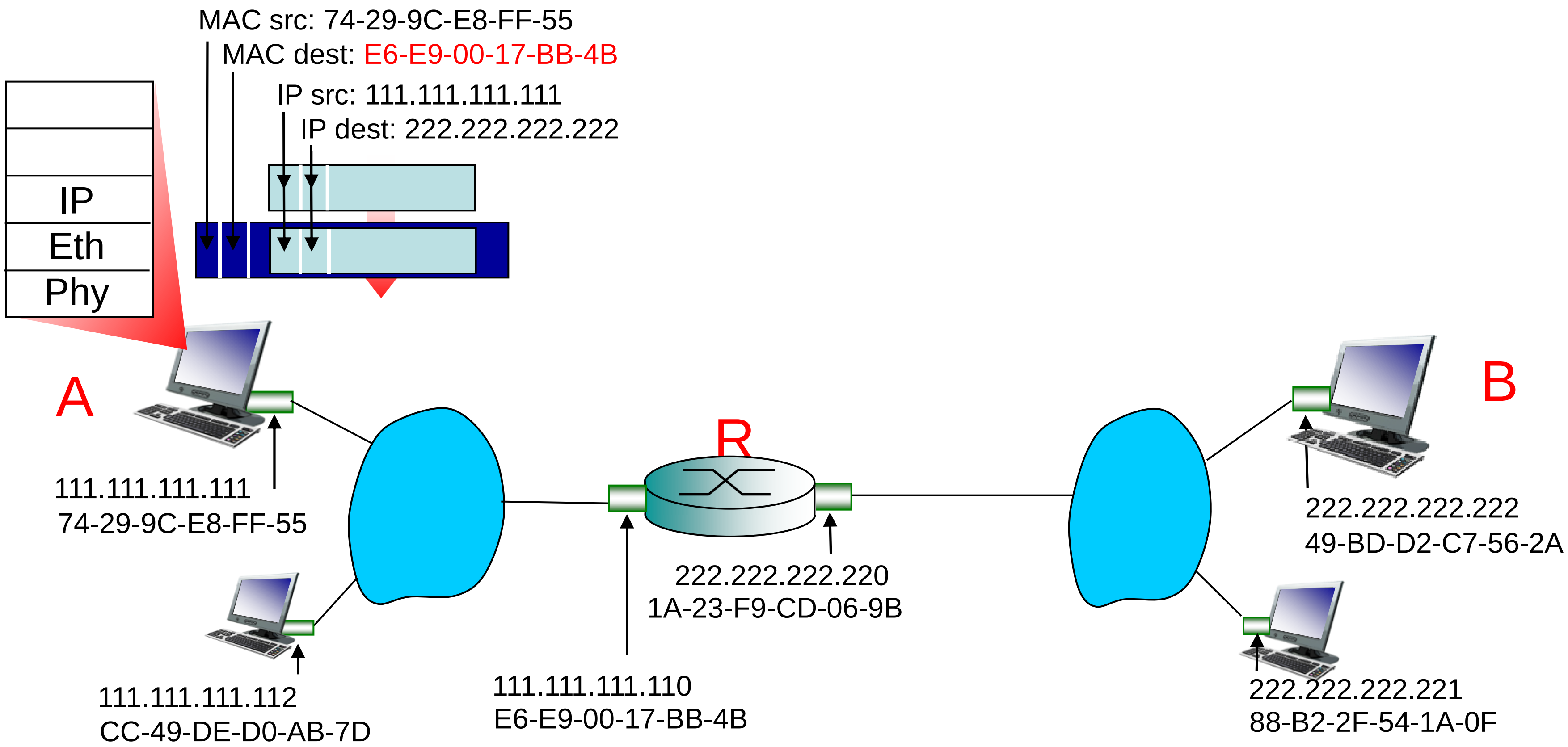
walkthrough: send datagram from A to B via R

- focus on addressing - at both IP (datagram) and MAC layer (frame)
- assume A knows B’s IP address
- assume A knows IP address of first hop router, R (how?)
- assume A knows R’s MAC address (how?)



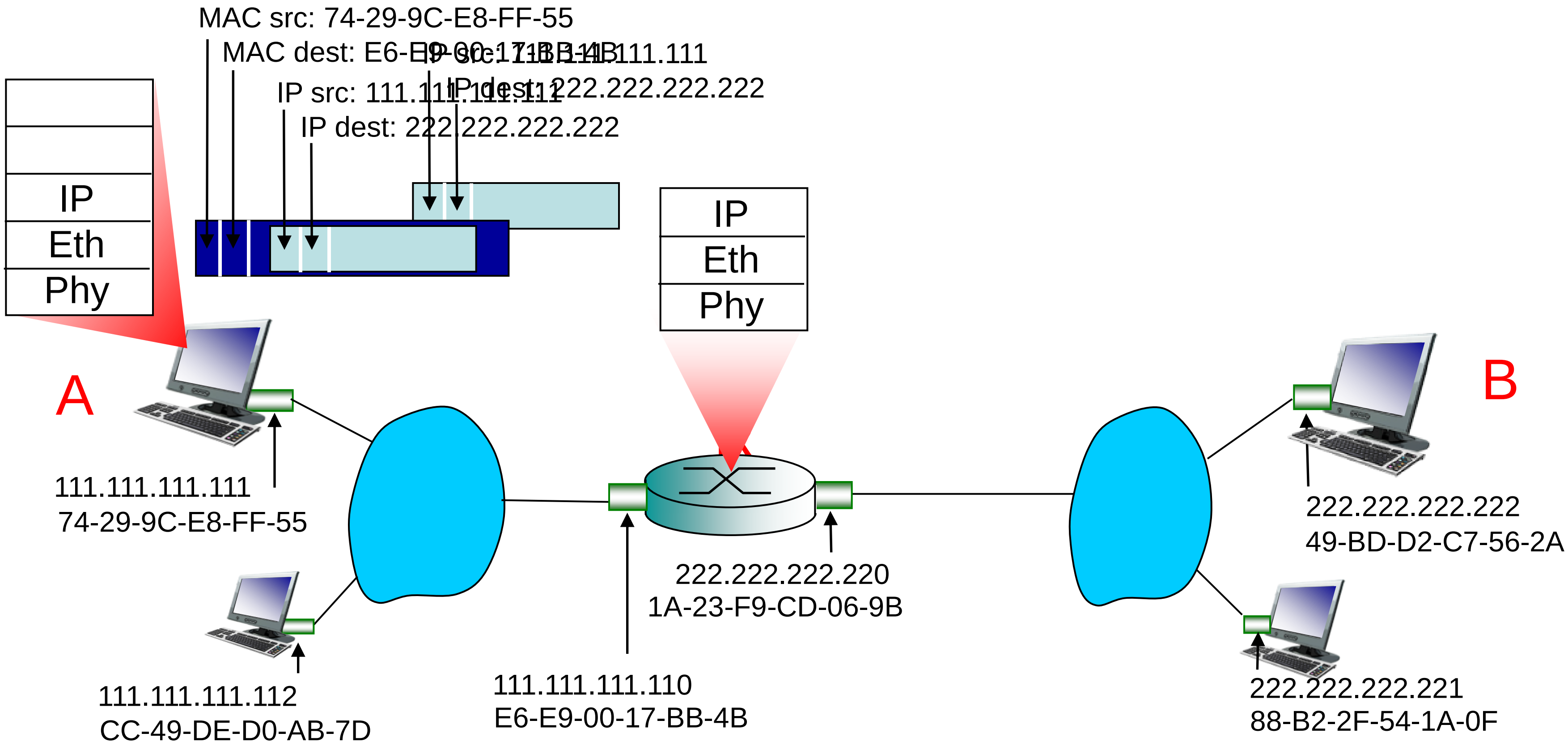
Addressing: routing to another LAN

- A creates IP datagram with IP source A, destination B
- A creates link-layer frame with R's MAC address as dest, frame contains A-to-B IP datagram



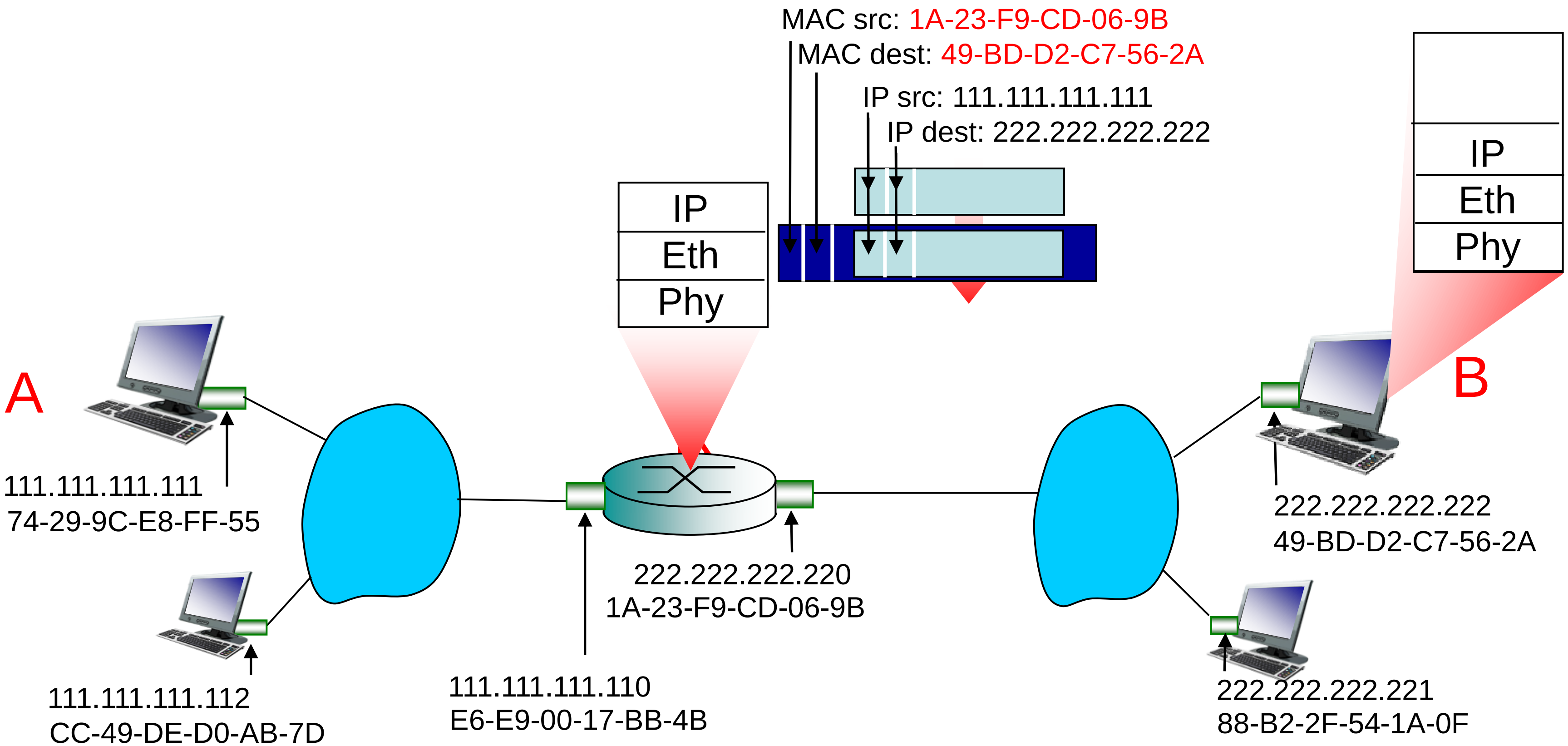
Addressing: routing to another LAN

- frame sent from A to R
- frame received at R, datagram removed, passed up to IP



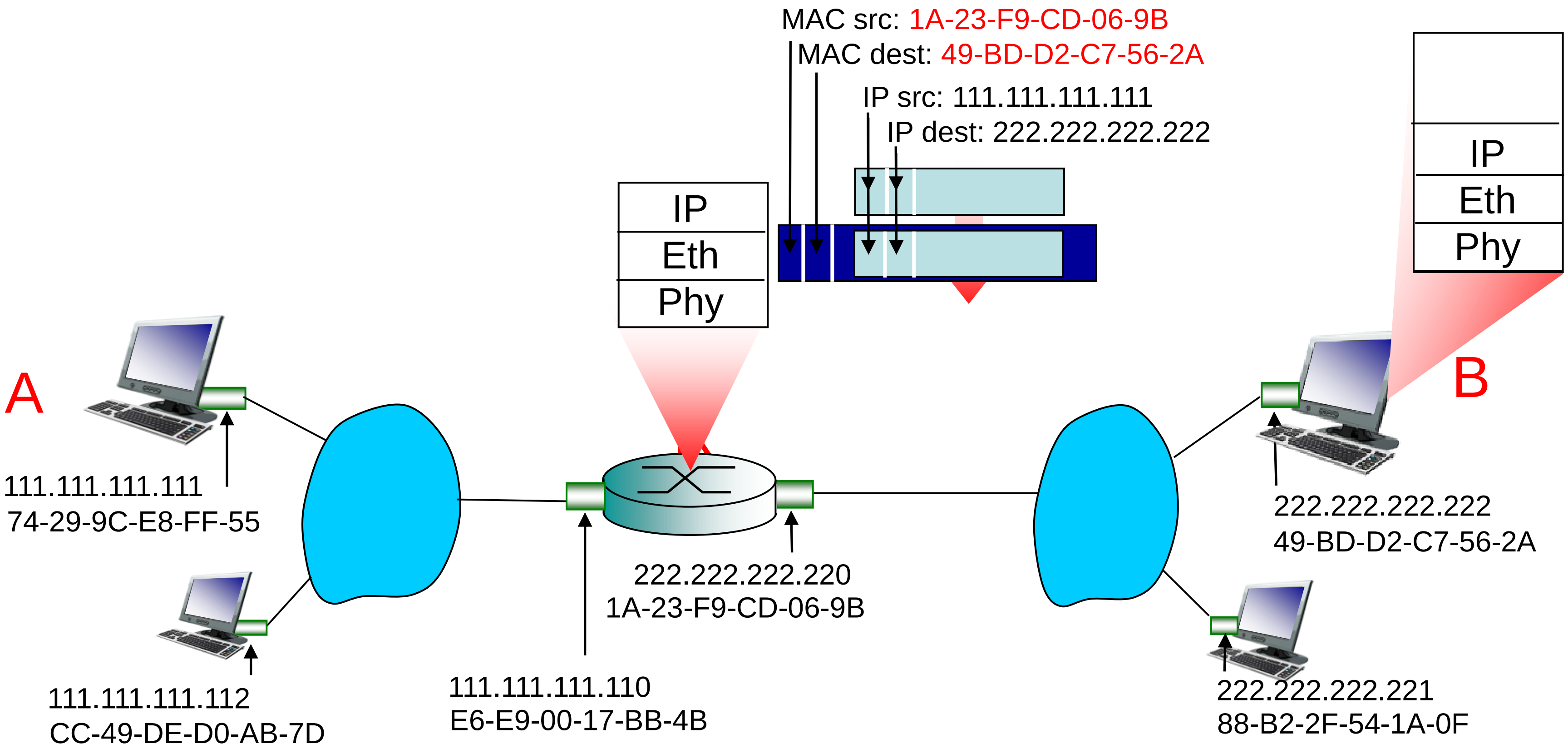
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



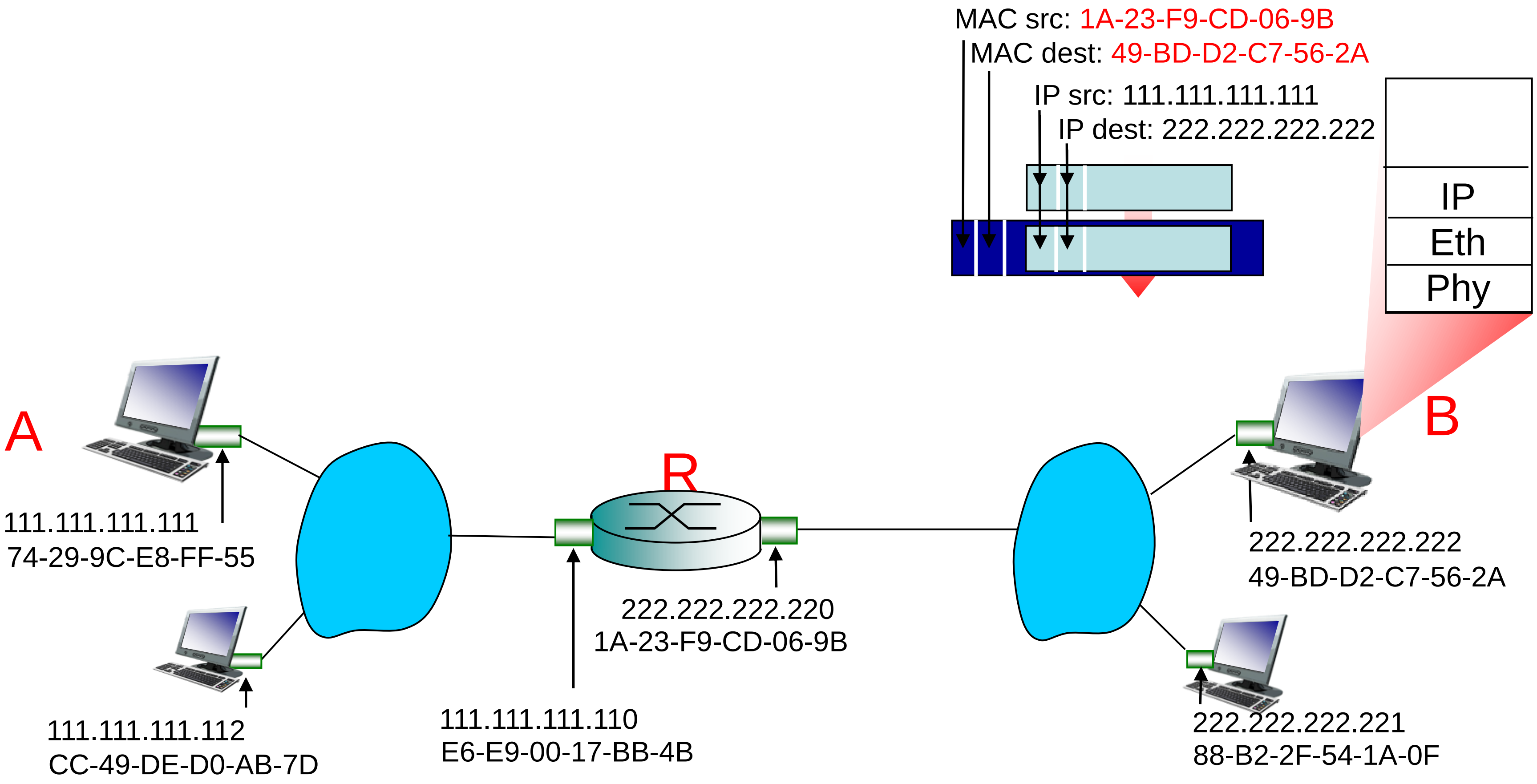
Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



Addressing: routing to another LAN

- R forwards datagram with IP source A, destination B
- R creates link-layer frame with B's MAC address as dest, frame contains A-to-B IP datagram



Link layer, LANs: outline

5.1 introduction, services

5.2 error detection, correction

5.3 multiple access protocols

5.4 link-layer addressing

5.5 Ethernet, LANs

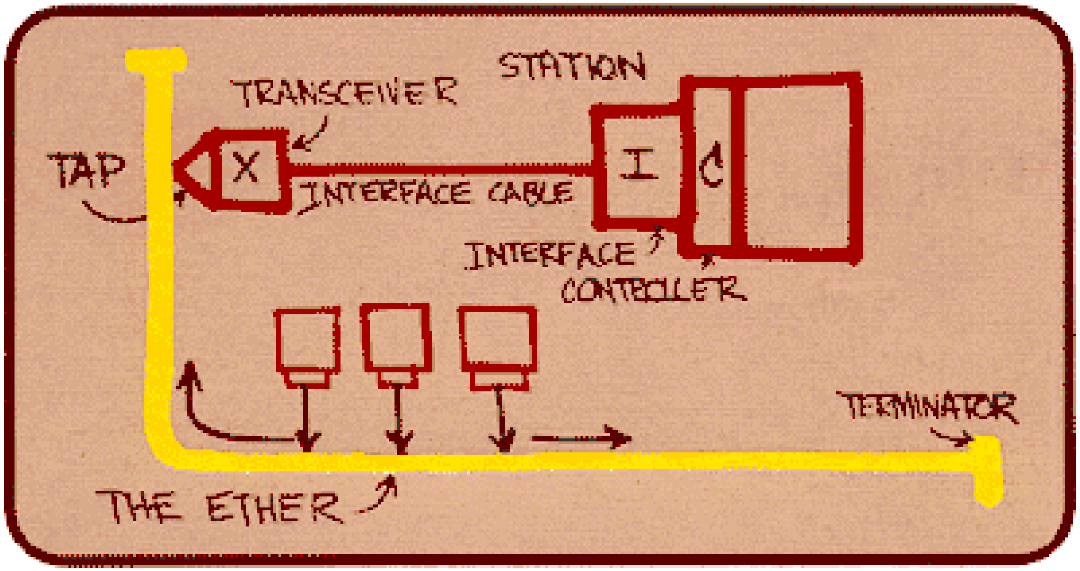
5.6 LAN switches

5.7 a day in the life of a web request

Ethernet

"dominant" wired LAN technology:

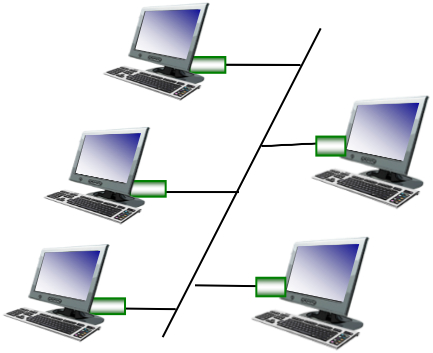
- cheap $20 for NIC
- first widely used LAN technology
- Developed in the mid-1970s by researchers at the Xerox Palo Alto Research Centers (PARC)
- simpler, cheaper than token LANs and ATM
- kept up with speed race: 10 Mbps – 10 Gbps



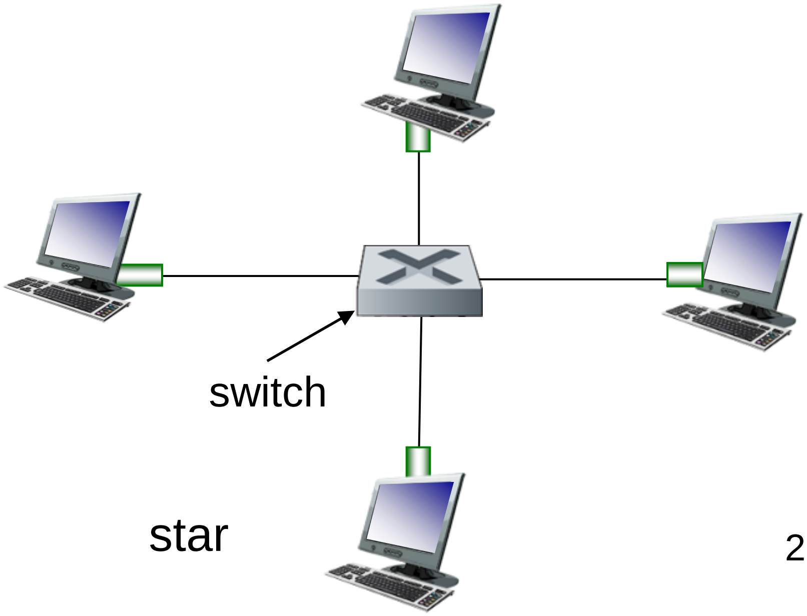
Metcalfe's Ethernet sketch

Star topology

- bus topology popular through mid 90s
  - all nodes in same collision domain (can collide with each other)
- today: star topology prevails
  - active *switch* in center
  - each “spoke” runs a (separate) Ethernet protocol (nodes do not collide with each other)

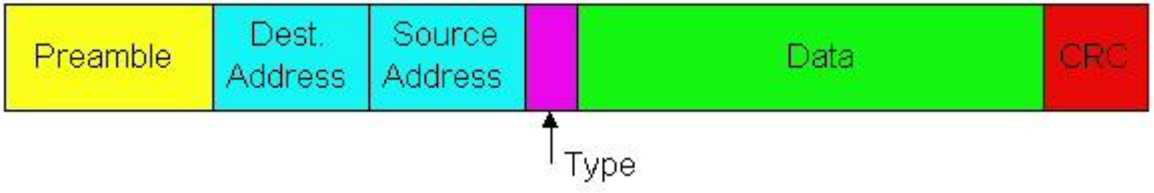


bus: coaxial cable



Ethernet frame structure

Sending adapter encapsulates IP datagram (or other network layer protocol packet) in Ethernet frame

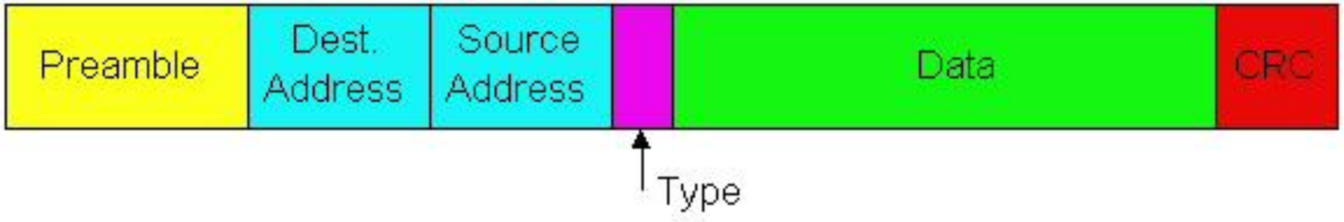


Preamble:

- 7 bytes with pattern 10101010 followed by one byte with pattern 10101011
- used to synchronize receiver, sender clock rates

Ethernet frame structure (more)

- ***addresses:*** 6 bytes
  - if adapter receives frame with matching destination address, or with broadcast address (e.g. ARP packet), it passes data in frame to network layer protocol
  - otherwise, adapter discards frame
- ***type:*** indicates higher layer protocol (mostly IP but others possible, e.g., Novell IPX, AppleTalk)
- ***CRC:*** checked at receiver, if error is detected, frame is dropped
- Data: 46 to 1500 bytes (MTU: 1500B)



Ethernet: unreliable, connectionless

- *connectionless:* No handshaking between sending and receiving NICs
- *unreliable:* receiving NIC doesn't send acks or nacks to sending NIC
  - stream of datagrams passed to network layer can have gaps (missing datagrams)
  - gaps will be filled if app is using TCP
  - otherwise, app will see gaps
- Ethernet's MAC protocol: unslotted *CSMA/CD*

Ethernet CSMA/CD algorithm

1. NIC receives datagram from network layer, creates frame
2. If NIC senses channel idle, starts frame transmission If NIC senses channel busy, waits until channel idle, then transmits
3. If NIC transmits entire frame without detecting another transmission, NIC is done with frame !
4. If NIC detects another transmission while transmitting, aborts and sends 48-bit jam signal
5. After aborting, NIC enters *exponential backoff:* after *m*th collision, NIC chooses *K* at random from $\{0,1,2,\ldots,2^m-1\}$. NIC waits K·512 bit times, returns to Step 2

CSMA/CD efficiency

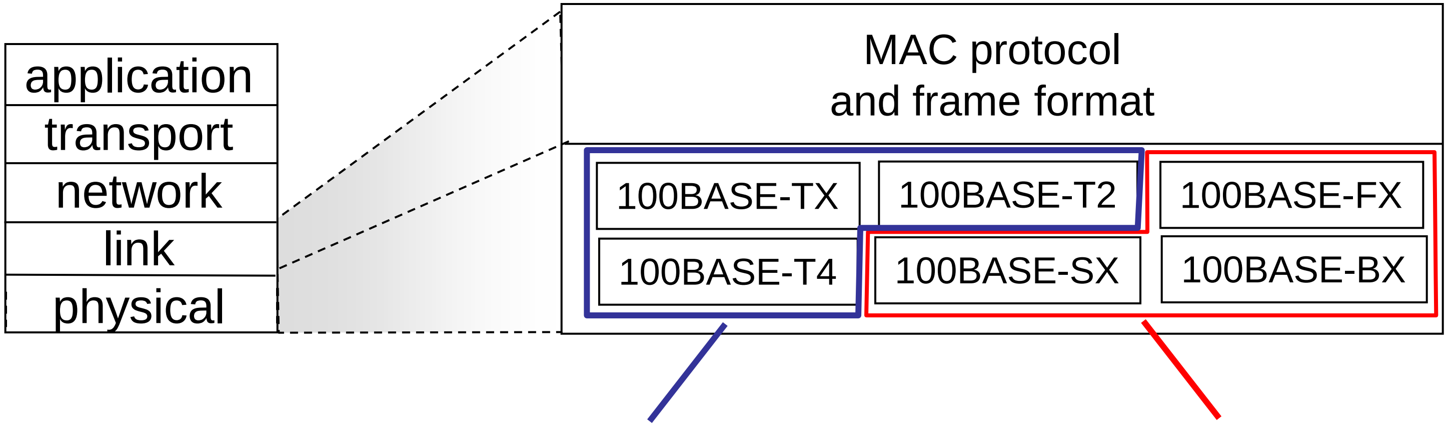
- T_{prop} = max prop delay between 2 nodes in LAN
- t_{trans} = time to transmit max-size frame

$$efficiency = \frac{1}{1 + 5t_{prop}/t_{trans}}$$

- efficiency goes to 1
  - as t_{prop} goes to 0
  - as t_{trans} goes to infinity
- better performance than ALOHA: and simple, cheap, decentralized!

802.3 Ethernet standards: link & physical layers

- *many* different Ethernet standards
  - common MAC protocol and frame format
  - different speeds: 2 Mbps, 10 Mbps, 100 Mbps, 1Gbps, 10G bps
  - different physical layer media: fiber, cable



copper (twister pair) physical layer

fiber physical layer

Link layer, LANs: outline

5.1 introduction, services

5.2 error detection, correction

5.3 multiple access protocols

5.4 link-layer addressing

5.5 Ethernet, LANs

5.6 LAN switches

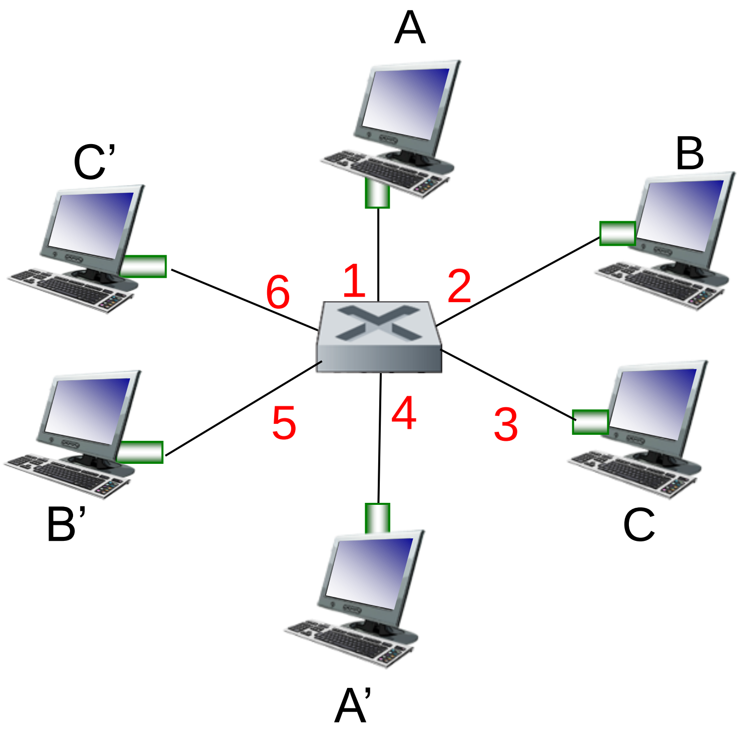
5.7 a day in the life of a web request

Ethernet switch

- link-layer device: takes an *active* role
  - – store, forward Ethernet frames
  - – examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment
- *transparent*
  - – hosts are unaware of presence of switches
- *plug-and-play, self-learning*
  - – switches do not need to be configured

Switch: *multiple* simultaneous transmissions

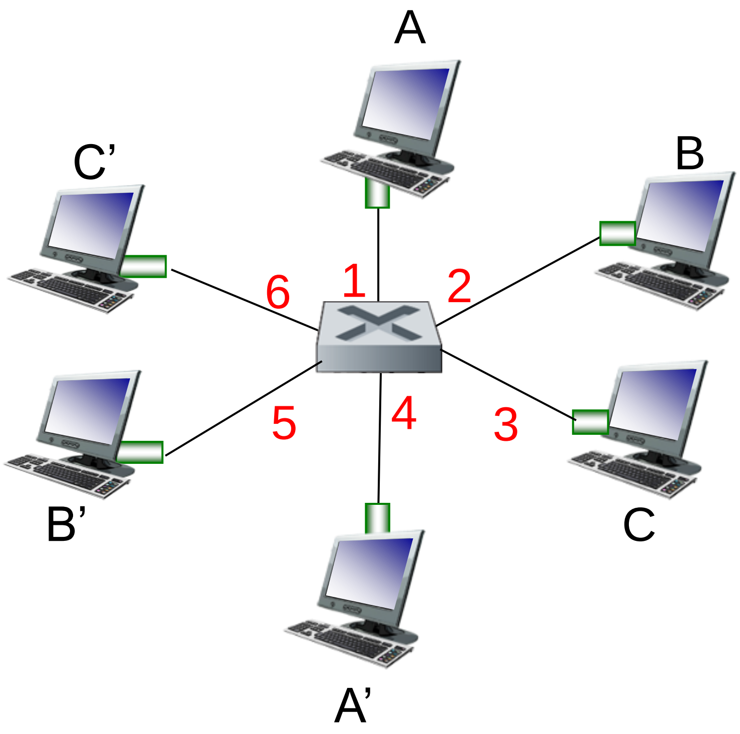
- hosts have dedicated, direct connection to switch
- switches buffer packets
- Ethernet protocol used on *each* incoming link, but no collisions; full duplex
  - each link is its own collision domain
- *switching:* A-to-A' and B-to-B' simultaneously, without collisions



switch with six interfaces (1,2,3,4,5,6)

Switch table

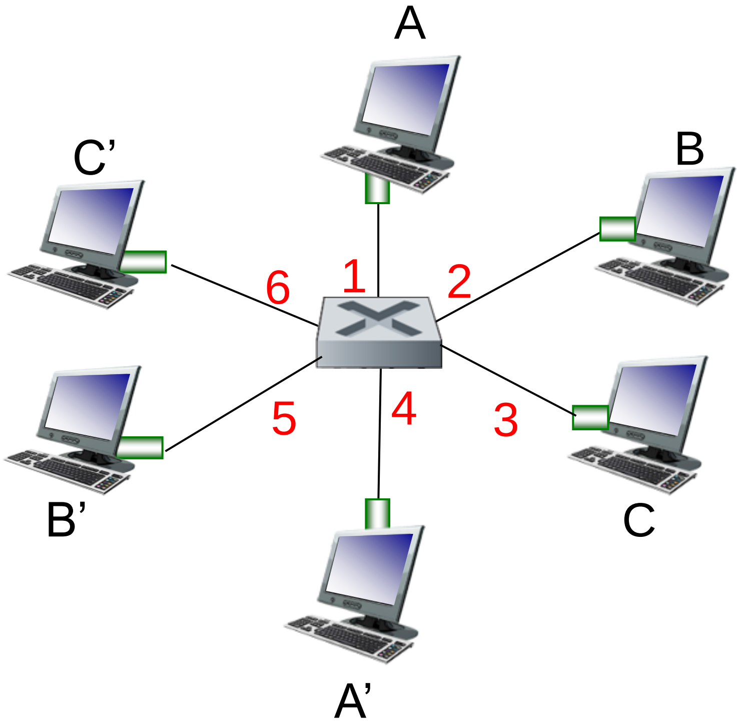
- Q: how does switch know that A' reachable via interface 4, B' reachable via interface 5?
- A: each switch has a switch table, each entry:
  - (MAC address of host, interface to reach host, time stamp)
- looks like a routing table!
- Q: how are entries created, maintained in switch table?
  - something like a routing protocol?



switch with six interfaces (1,2,3,4,5,6)

Switch table

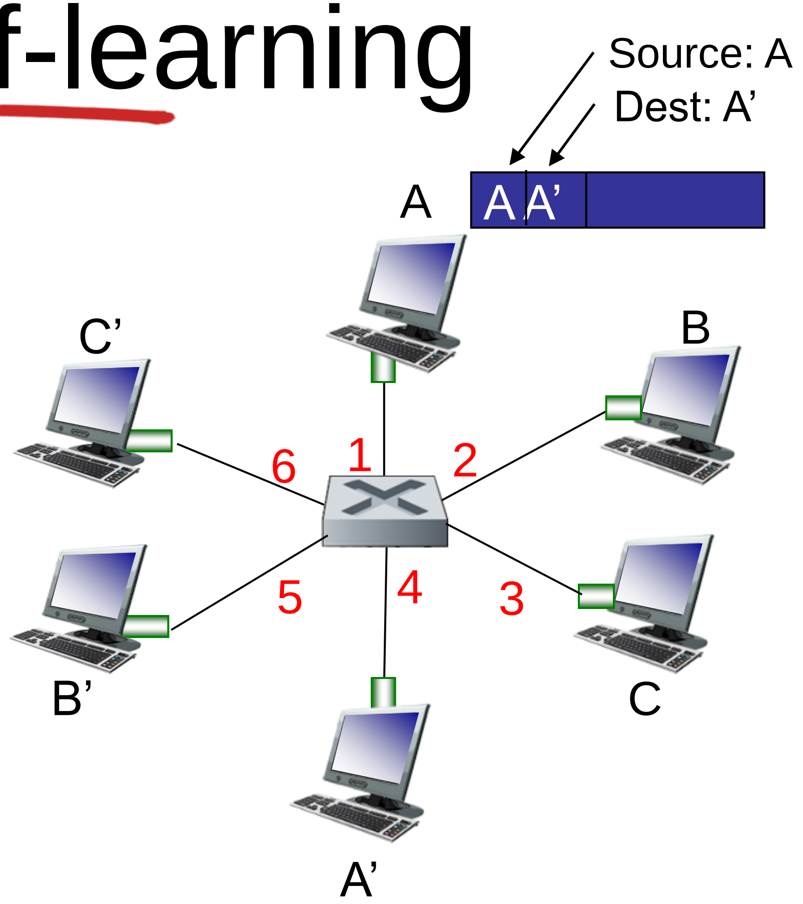
- *Q:* how does switch know that A' reachable via interface 4, B' reachable via interface 5?
- *A:* each switch has a switch table, each entry:
  - (MAC address of host, interface to reach host, time stamp)
- looks like a routing table!
- *Q:* how are entries created, maintained in switch table?
  - something like a routing protocol?



switch with six interfaces (1,2,3,4,5,6)

Switch: self-learning

- switch *learns* which hosts can be reached through which interfaces
  - when frame received, switch "learns" location of sender: incoming LAN segment
  - records sender/location pair in switch table



MAC addr	interface	TTL
A	1	60

Switch table (initially empty)

Switch: frame filtering/forwarding

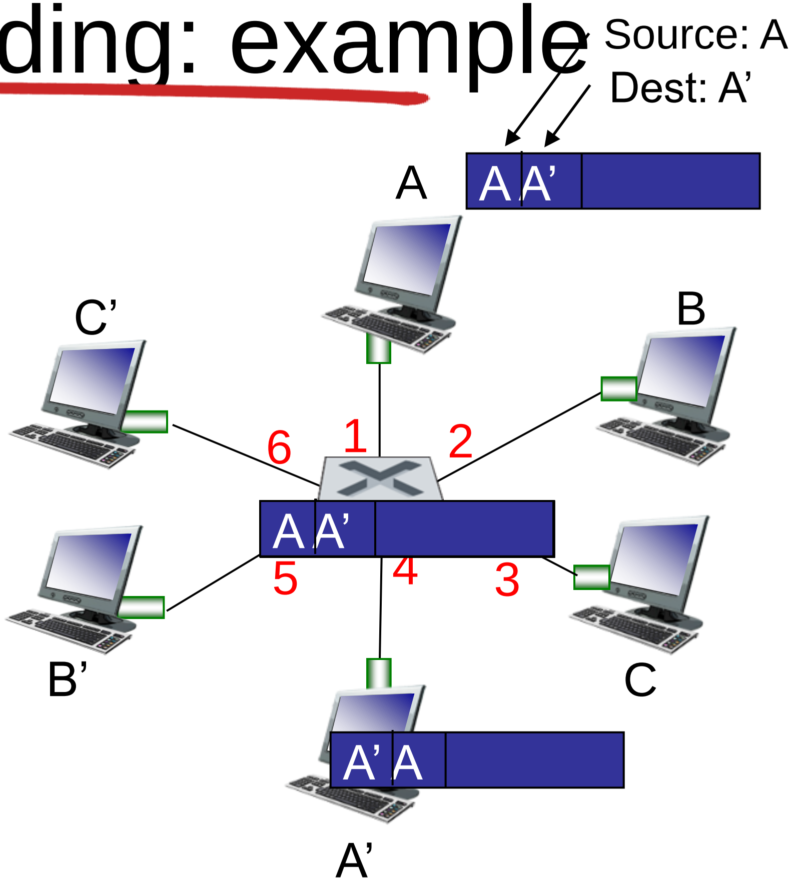
When frame received:

```
1. record link associated with sending host
2. index switch table using MAC dest address
3. if entry found for destination
      then {
     if dest on segment from which frame arrived
          then drop the frame
         else forward the frame on interface indicated
      }
     else flood
```

forward on all but the interface on which the frame arrived

Self-learning, forwarding: example

- frame destination unknown: flood
- destination A location known:
  selective send

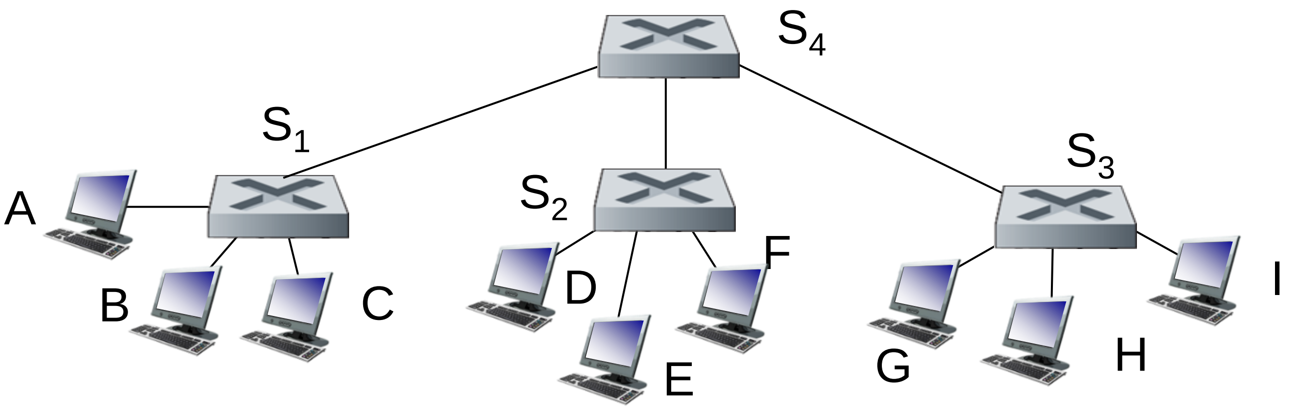


MAC addr	interface	TTL
A A'	1 4	60 60

Switch table
(initially empty)

Interconnecting switches

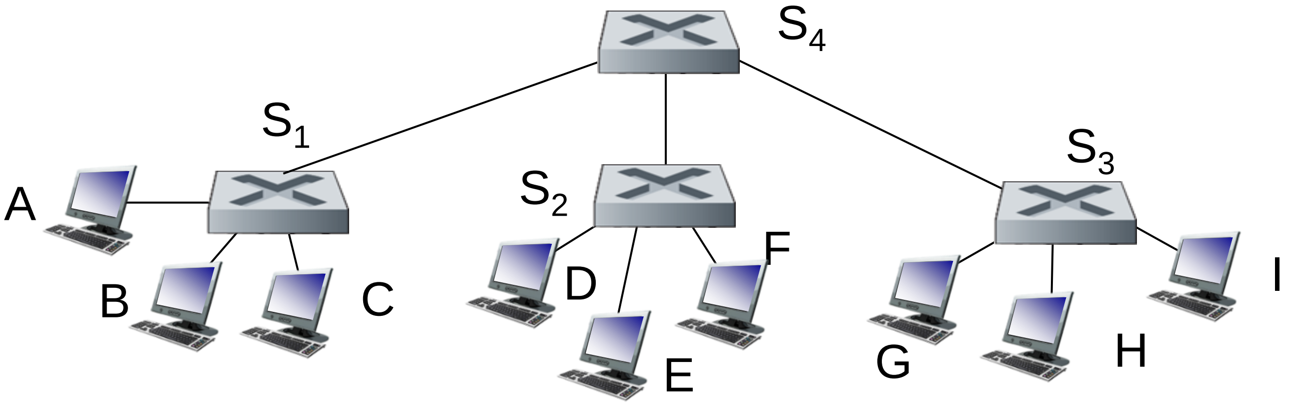
- switches can be connected together



- Q: sending from A to G - how does S_1 know to forward frame destined to F via S_4 and S_3?
- A: self learning! (works exactly the same as in single-switch case!)

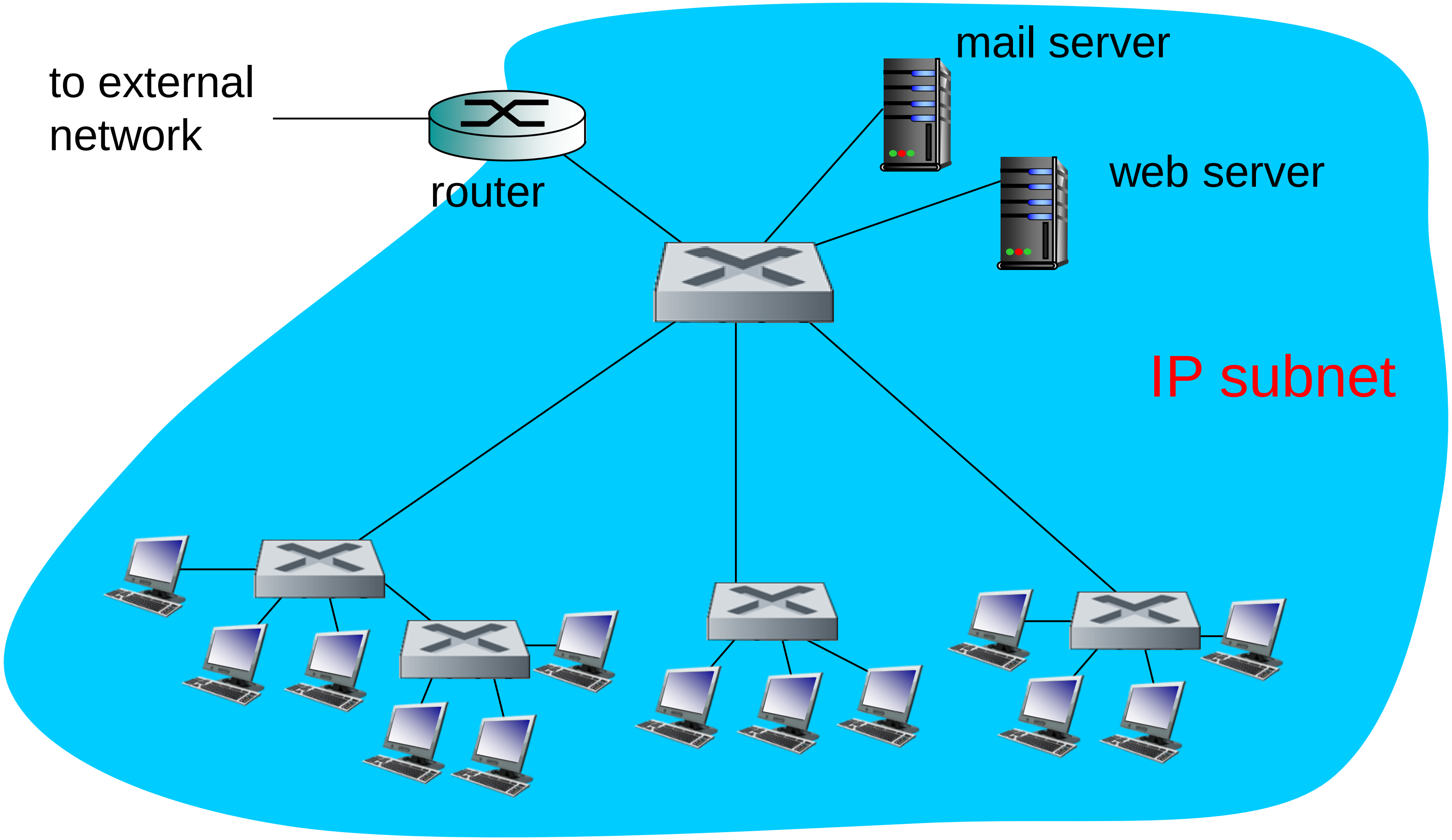
Self-learning multi-switch example

Suppose C sends frame to I, I responds to C



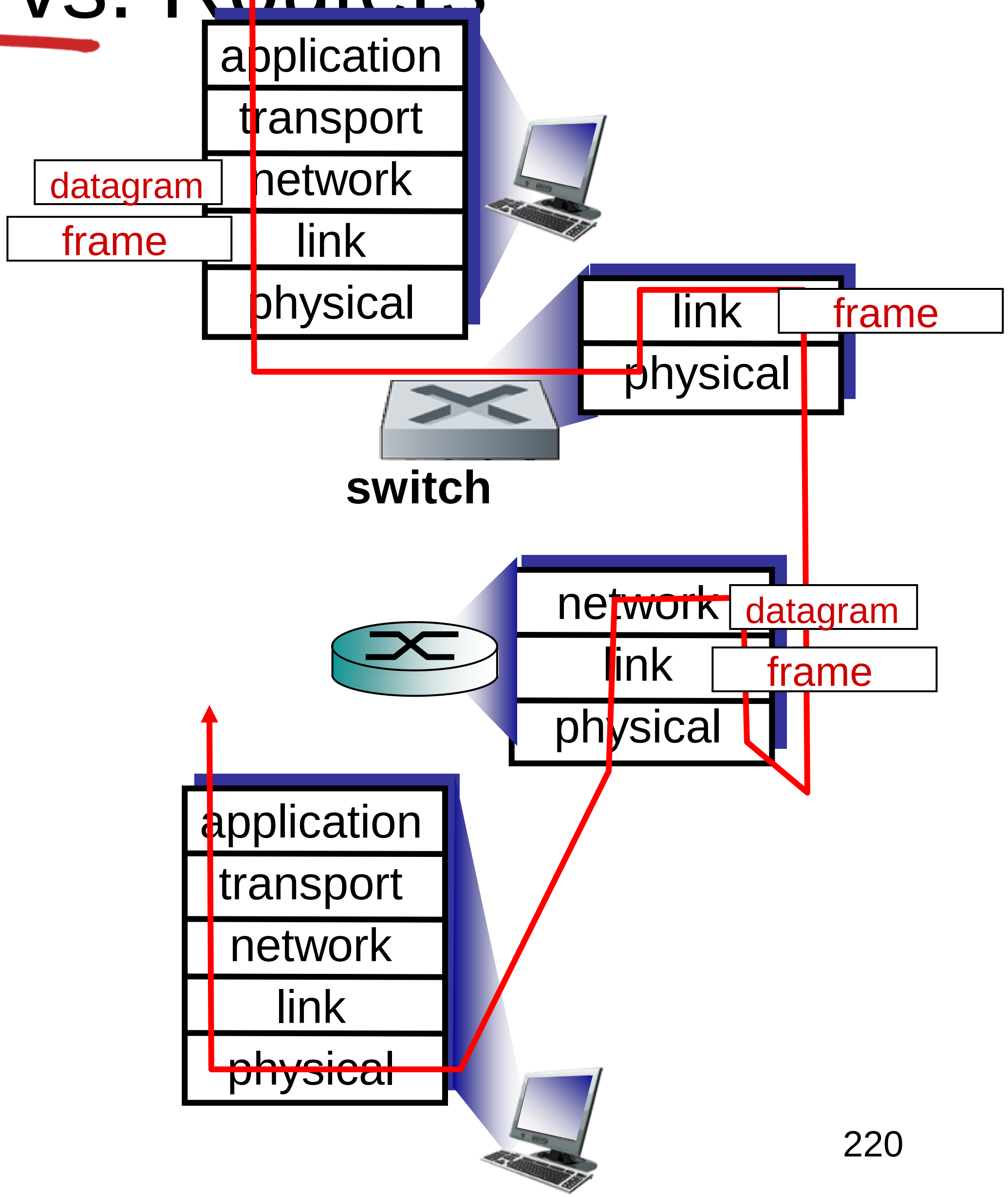
- Q: show switch tables and packet forwarding in S_1, S_2, S_3, S_4

Institutional network



Switches vs. Routers

- both store-and-forward devices
  - routers: network-layer devices (examine network-layer headers)
  - switches are link-layer devices (examine link-layer headers)
- routers maintain routing tables, implement routing algorithms
- switches maintain switch tables, implement filtering, learning algorithms



Link layer, LANs: outline

5.1 introduction, services

5.2 error detection, correction

5.3 multiple access protocols

5.4 link-layer addressing

5.5 Ethernet, LANs

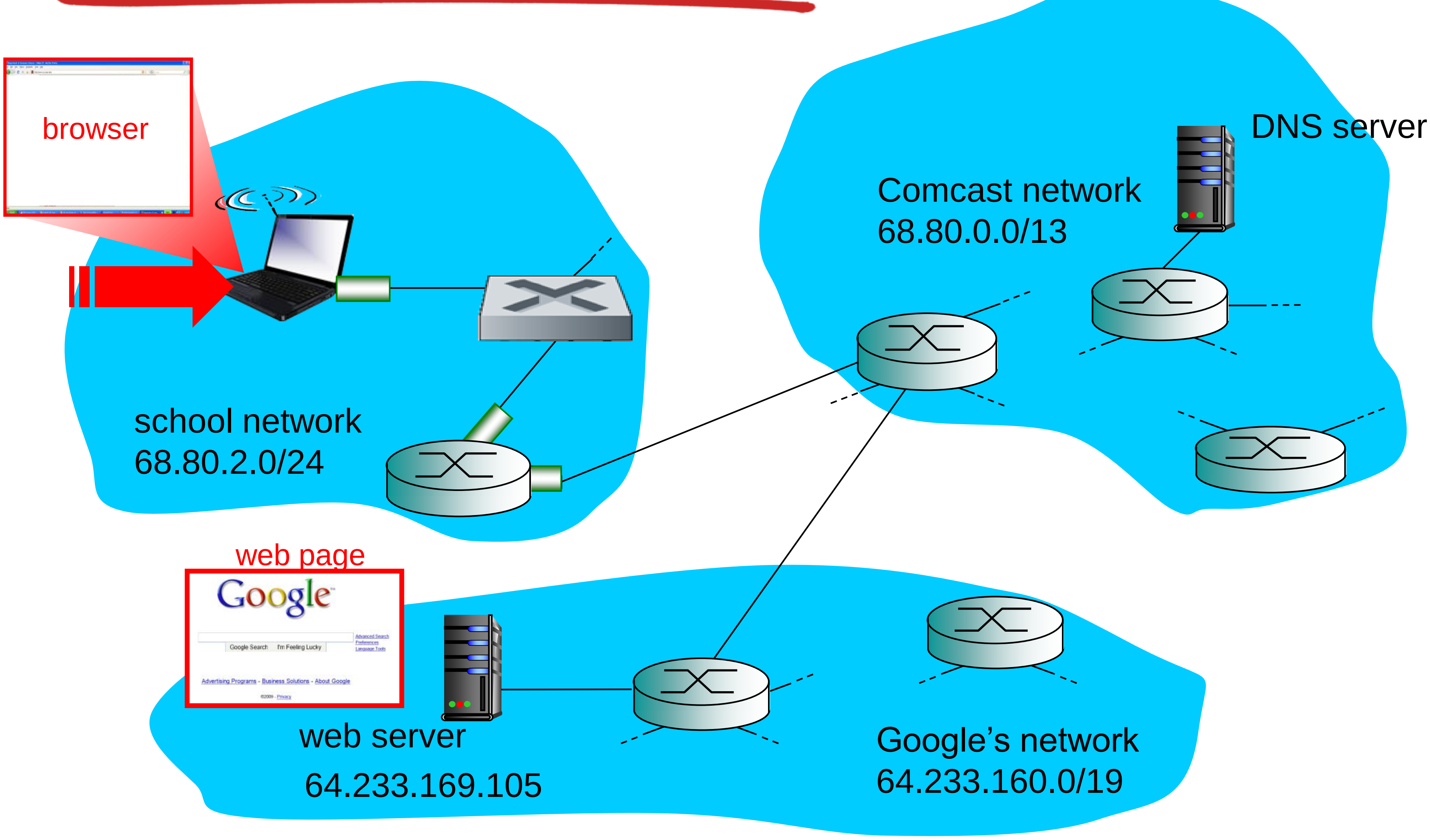
5.6 LAN switches

5.7 a day in the life of a web request

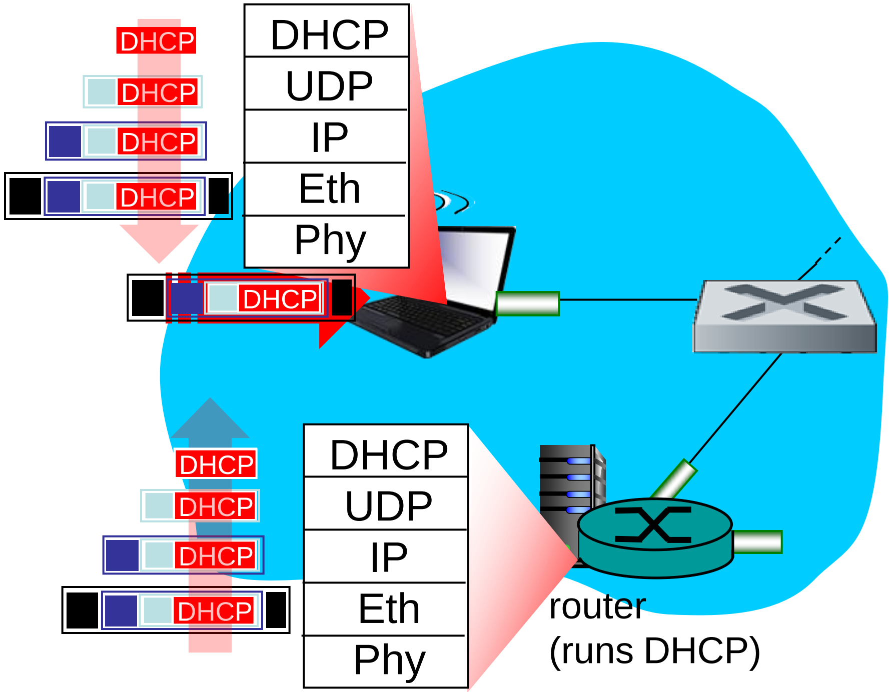
Synthesis: a day in the life of a web request

- journey down protocol stack complete!
  - application, transport, network, link
- putting-it-all-together: synthesis!
  - *goal:* identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting www page
  - *scenario:* student attaches laptop to campus network, requests/receives www.google.com

A day in the life: scenario



A day in the life… connecting to the ~~Internet~~



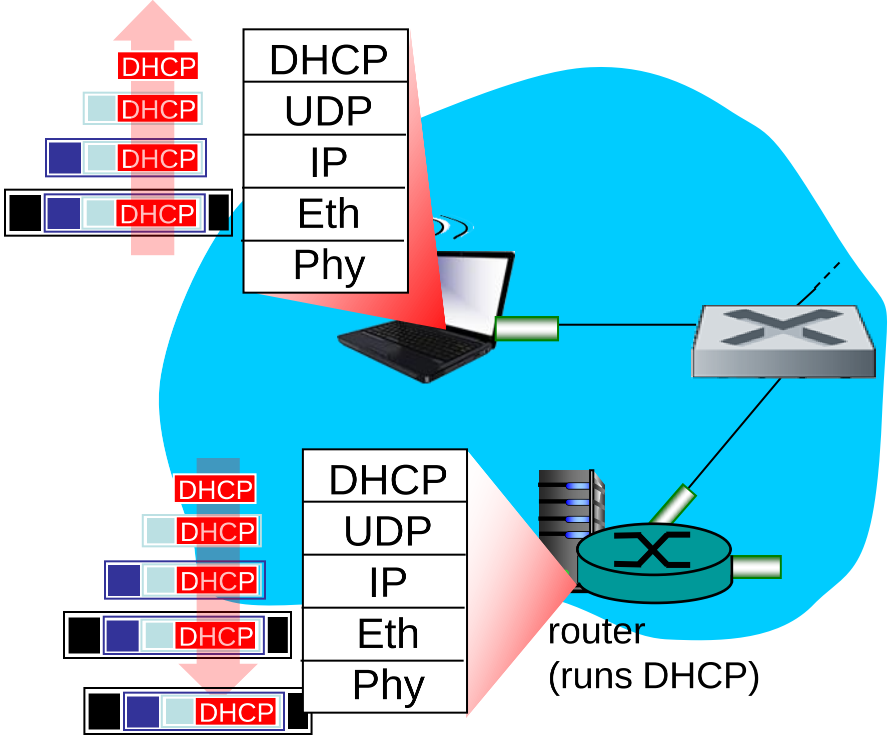
- connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use *DHCP*

- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in 802.3 Ethernet

- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server

- Ethernet demuxed to IP demuxed, UDP demuxed to DHCP

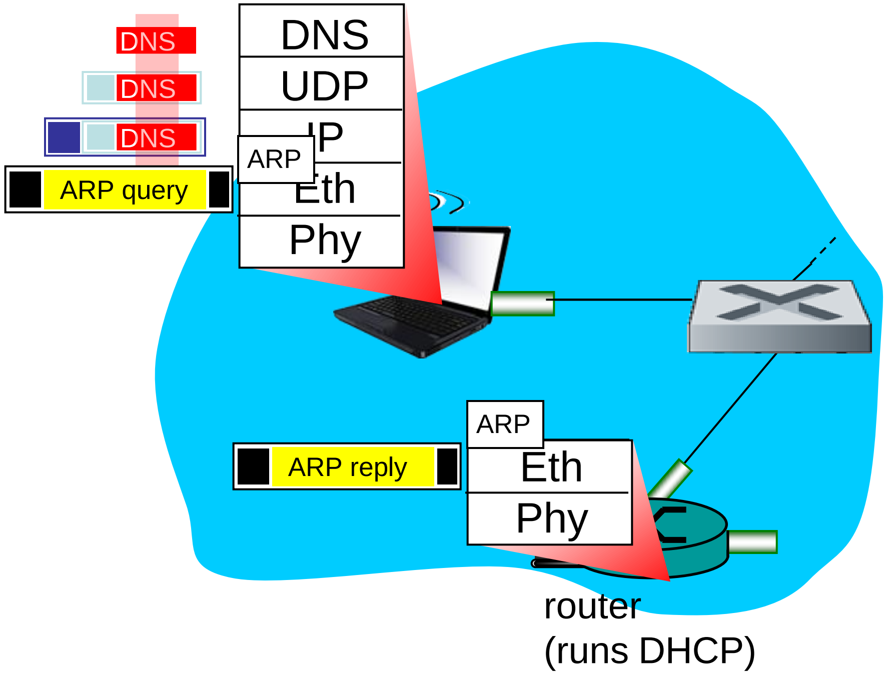
A day in the life… connecting to the Internet



- DHCP server formulates *DHCP ACK* containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulation at DHCP server, frame forwarded (switch learning) through LAN, demultiplexing at client
- DHCP client receives DHCP ACK reply

Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

A day in the life… ARP (before DNS, before HTTP)



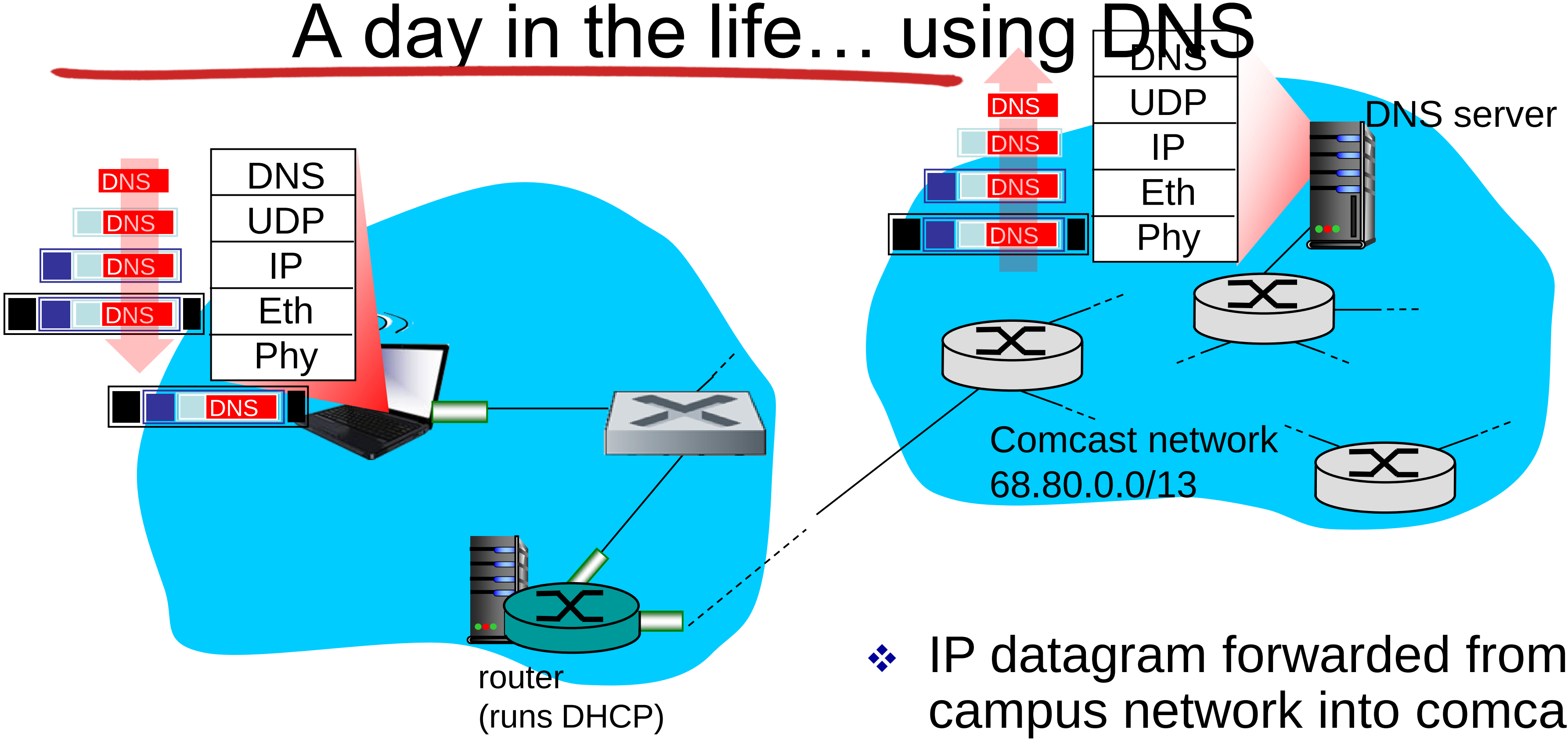
- before sending *HTTP* request, need IP address of www.google.com: *DNS*

- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. To send frame to router, need MAC address of router interface: ARP

- ARP query broadcast, received by router, which replies with ARP reply giving MAC address of router interface

- client now knows MAC address of first hop router, so can now send frame containing DNS query

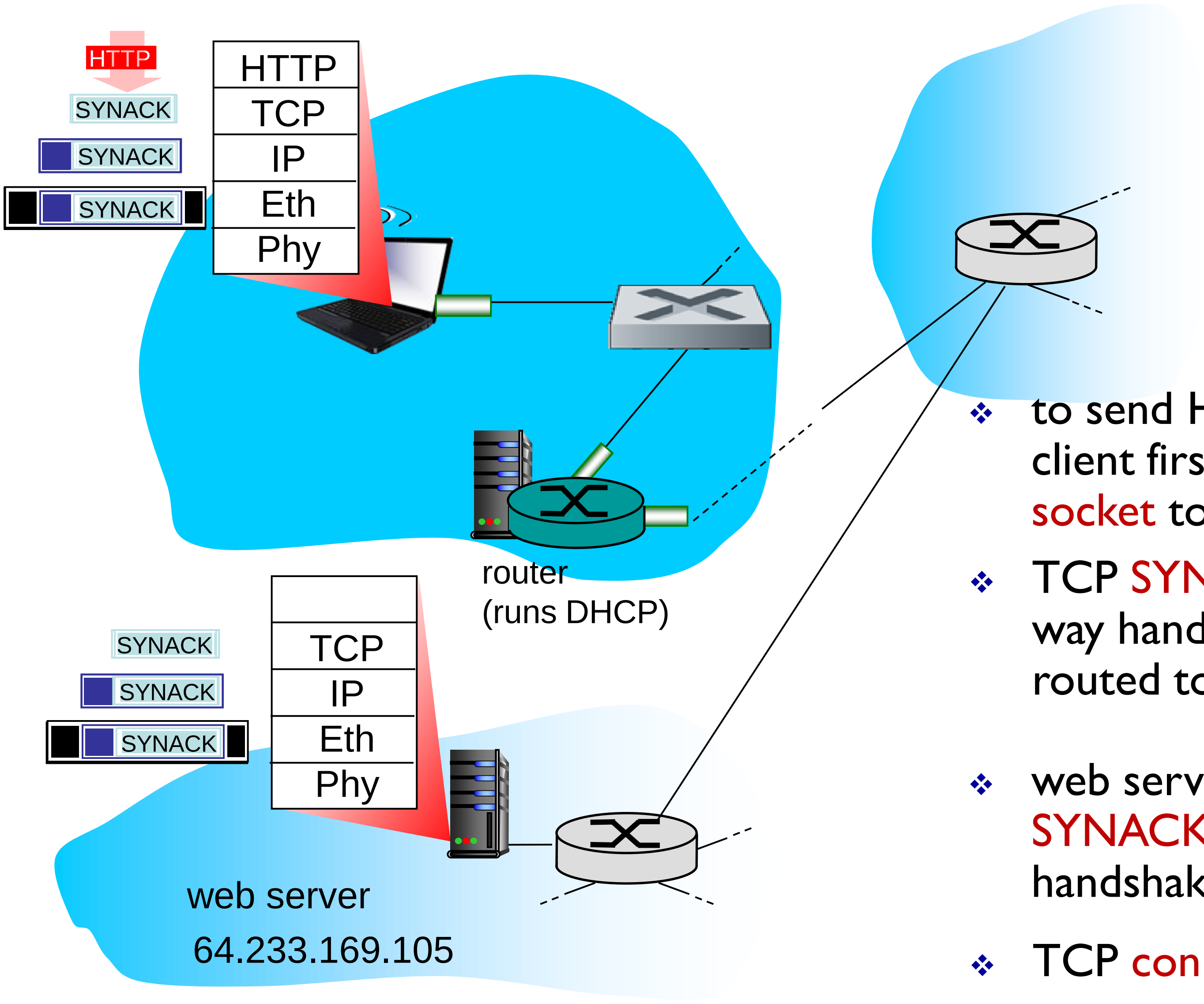
A day in the life… using DNS



- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router

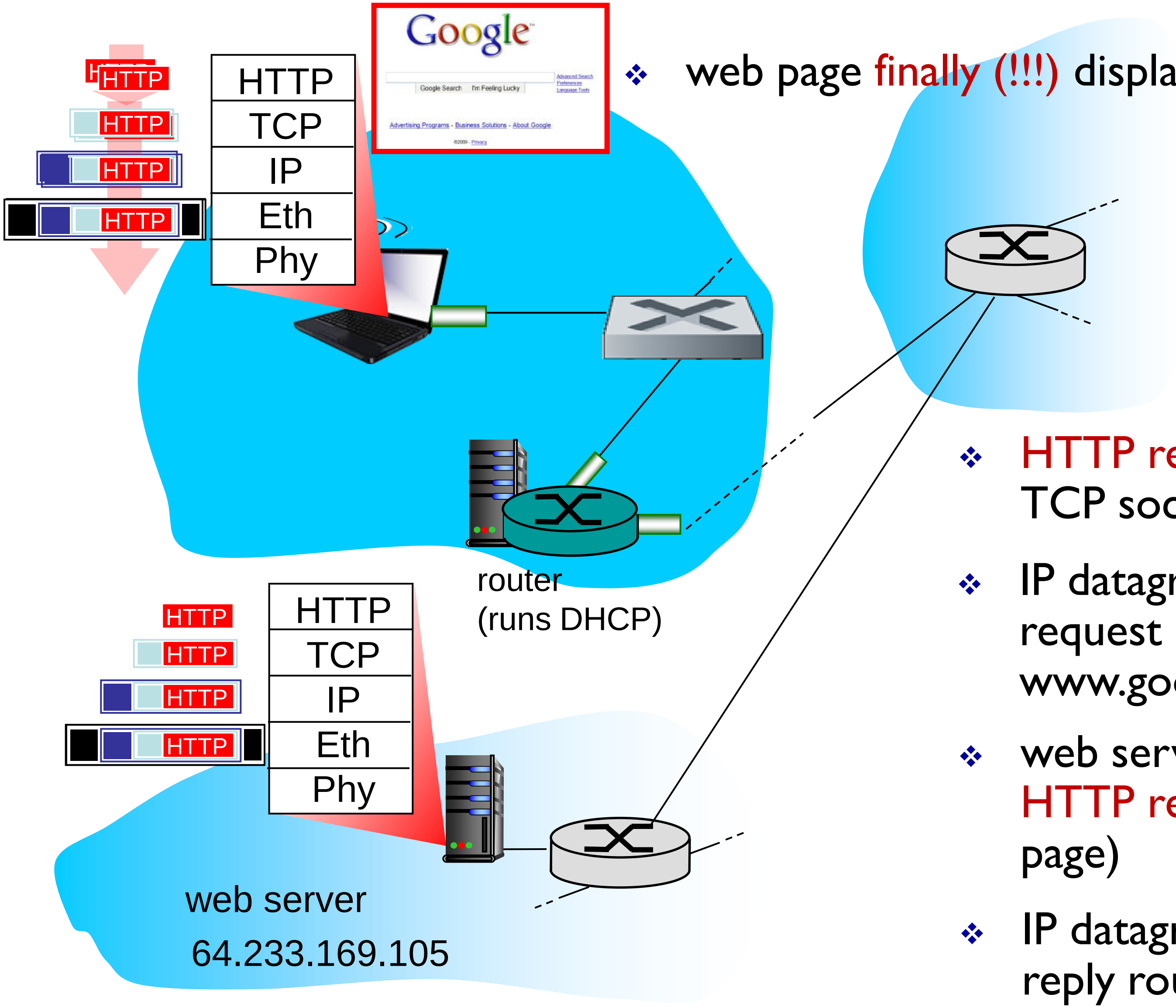
- IP datagram forwarded from campus network into comcast network, routed (tables created by RIP, OSPF, IS-IS and/or BGP routing protocols) to DNS server
- demux'ed to DNS server
- DNS server replies to client with IP address of www.google.com

A day in the life…TCP connection carrying HTTP



- to send HTTP request, client first opens TCP socket to web server
- TCP SYN segment (step 1 in 3-way handshake) inter-domain routed to web server
- web server responds with TCP SYNACK (step 2 in 3-way handshake)
- TCP connection established!

A day in the life… HTTP request/reply



- web page finally (!!!) displayed

- HTTP request sent into TCP socket
- IP datagram containing HTTP request routed to www.google.com
- web server responds with HTTP reply (containing web page)
- IP datagram containing HTTP reply routed back to client

Chapter 5: Summary

- principles behind data link layer services:
  - error detection, correction
  - sharing a broadcast channel: multiple access
  - link layer addressing
- instantiation and implementation of various link layer technologies
  - Ethernet
  - switched LANS
- synthesis: a day in the life of a web request

Chapter 5: let's take a breath

- journey down protocol stack *complete* (except PHY)
- solid understanding of networking principles, practice
- ….. could stop here …. but *lots* of interesting topics!
  - wireless
  - multimedia
  - security
  - network management

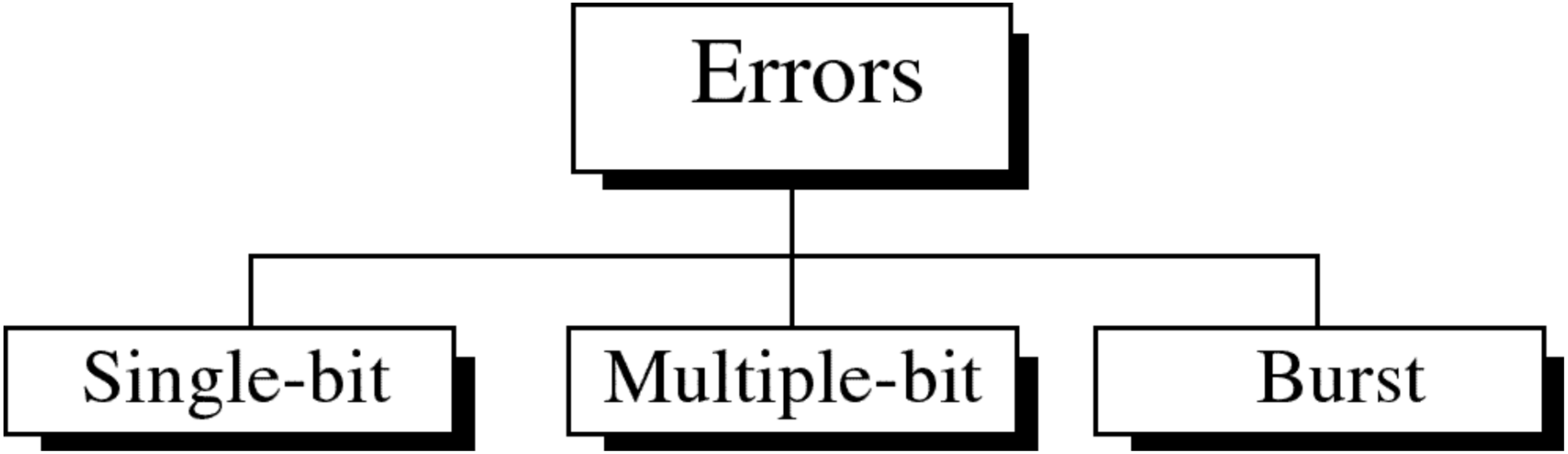
Error Detection and Correction

- **Types of Errors**
- **Detection**
- **Correction**

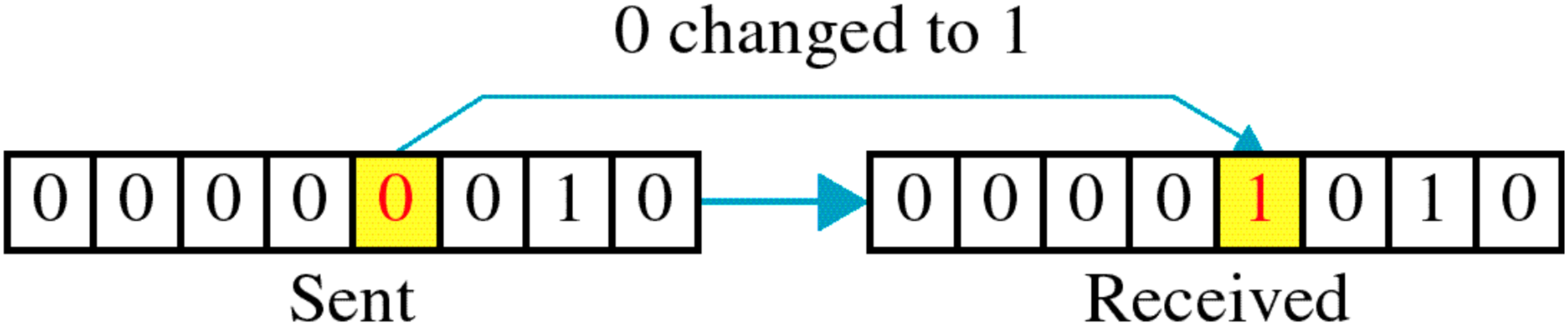
Basic concepts

- ★ Networks must be able to transfer data from one device to another with complete accuracy.
- ★ Data can be corrupted during transmission.
- ★ For reliable communication, errors must be detected and corrected.
- ★ **Error detection and correction** are implemented either at the **data link layer** or the **transport layer** of the OSI model.

Types of Errors



Single-bit error

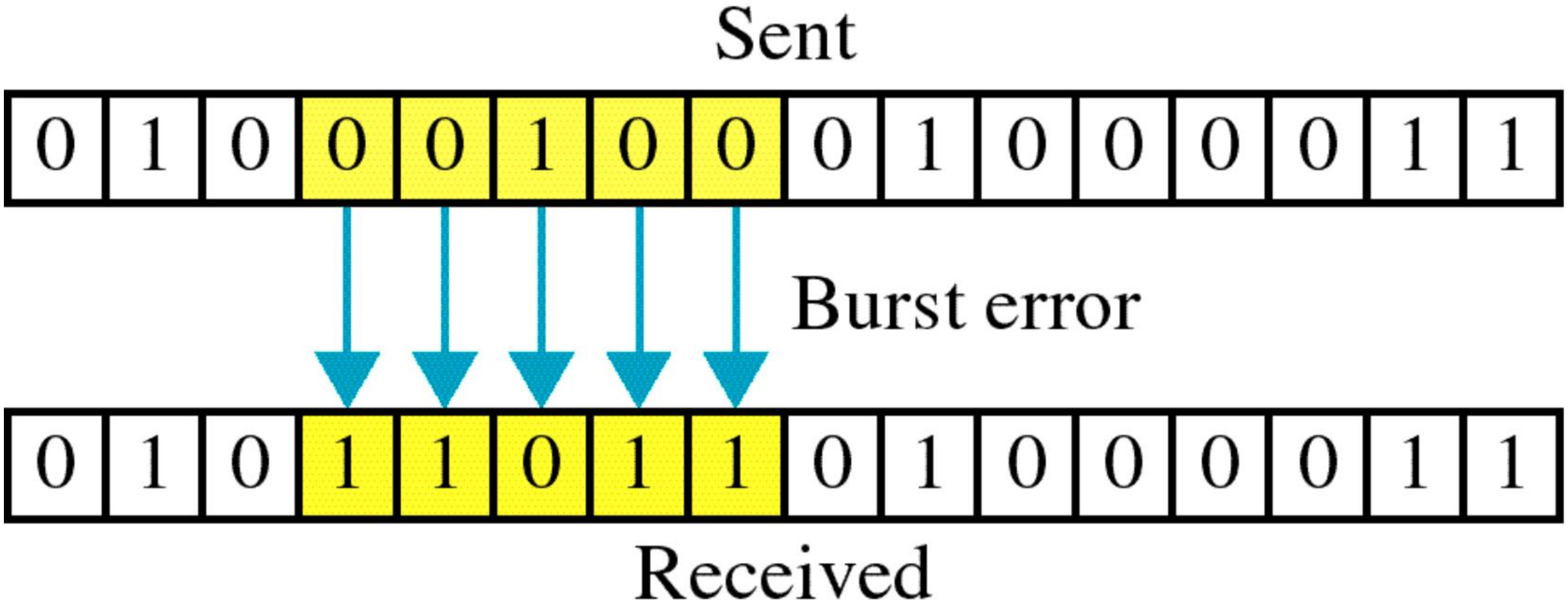


Single bit errors are the **least likely** type of errors in serial data transmission because the noise must have a very short duration which is very rare. However this kind of errors can happen in parallel transmission.

Example:

- ★If data is sent at 1Mbps then each bit lasts only 1/1,000,000 sec. or 1 μs.
- ★For a single-bit error to occur, the noise must have a duration of only 1 μs, which is very rare.

Burst error



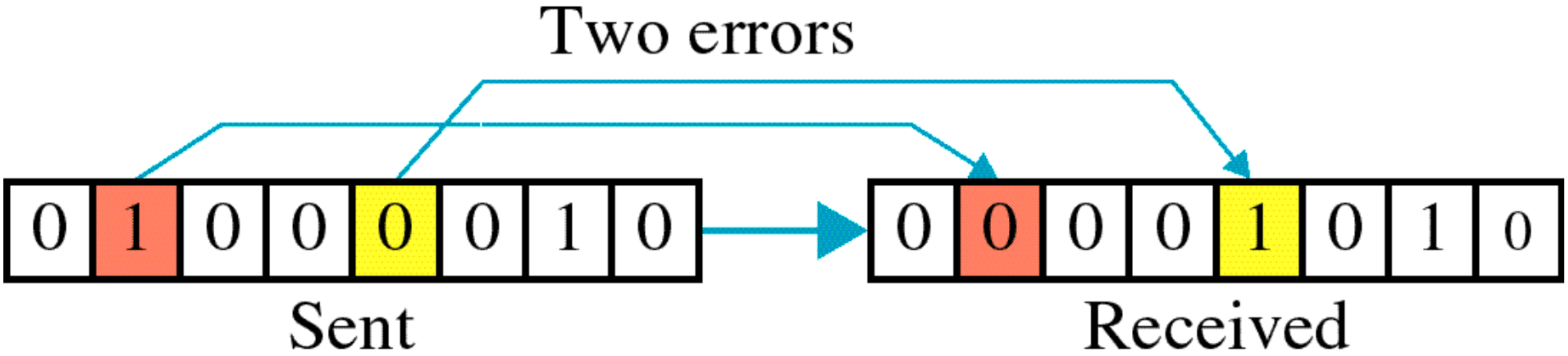
Sent

0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1

Burst error

0 1 0 1 1 0 1 1 0 1 0 0 0 0 1 1

Received



Two errors

0 1 0 0 0 0 1 0

Sent

0 0 0 0 1 0 1 0

Received

The term **burst error** means that two or more bits in the data unit have changed from 1 to 0 or from 0 to 1.

Burst errors does not necessarily **mean that the errors occur in consecutive bits**, the length of the burst is measured from the first corrupted bit to the last corrupted bit. Some bits in between may not have been corrupted.

- ★ **Burst error is most likely to happen in serial transmission** since the duration of noise is normally longer than the duration of a bit.
- ★ The number of bits affected depends on the data rate and duration of noise.

Example:

- ➔ If data is sent at rate = 1Kbps then a noise of 1/100 sec can affect 10 bits.(1/100*1000)

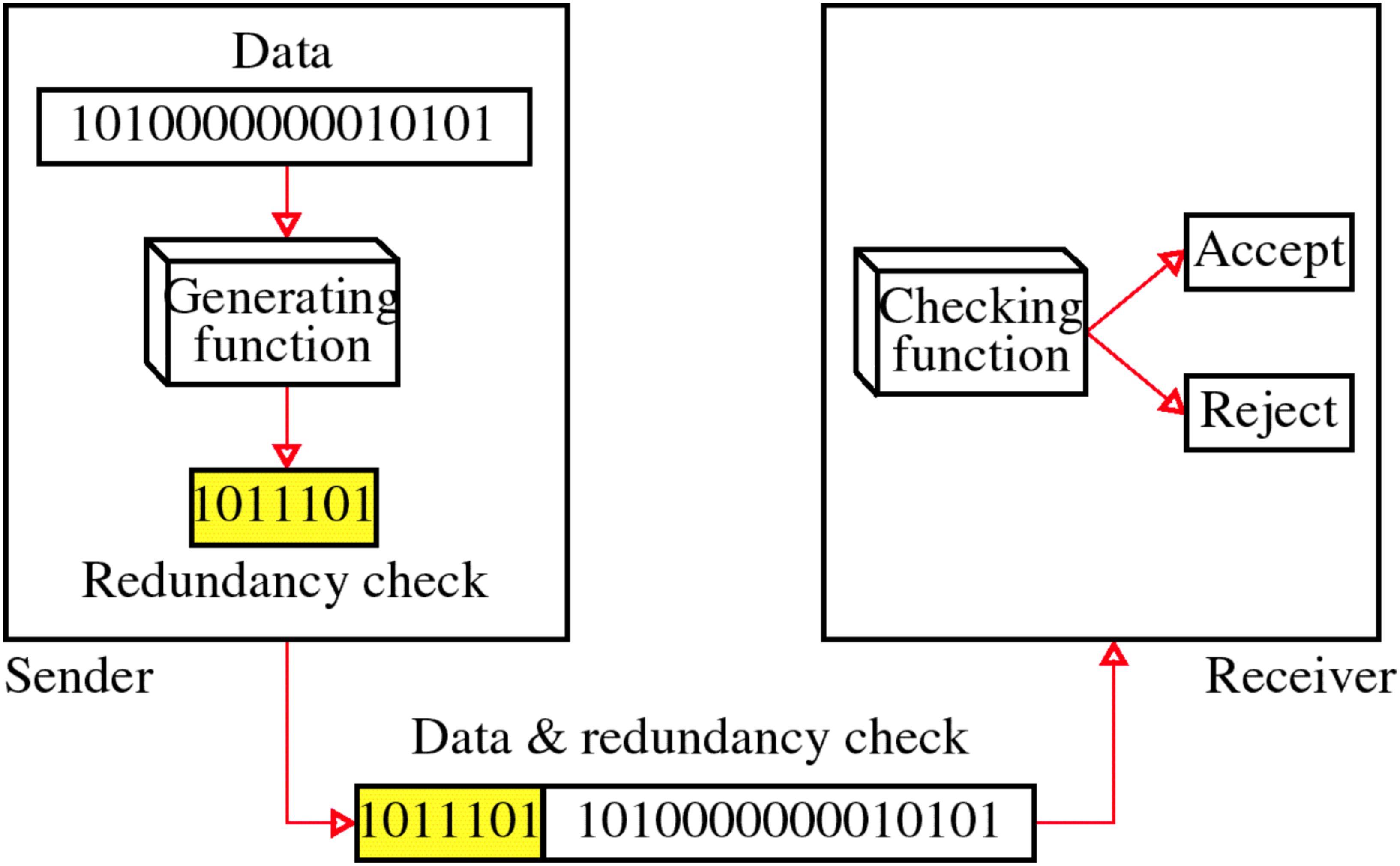
- ➔ If same data is sent at rate = 1Mbps then a noise of 1/100 sec can affect 10,000 bits.(1/100*10^6)

Error detection

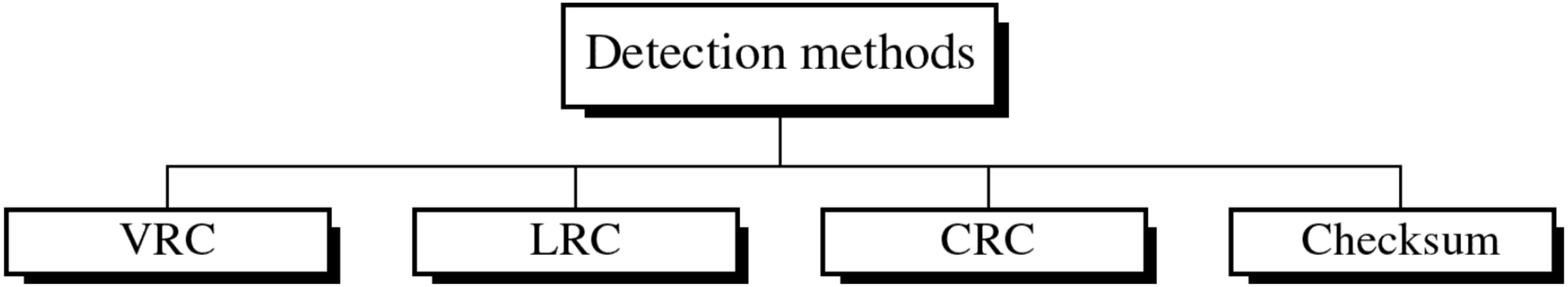
Error detection means to decide whether the received data is correct or not without having a copy of the original message.

Error detection **uses the concept of redundancy**, **which means** adding extra bits for detecting errors at the destination.

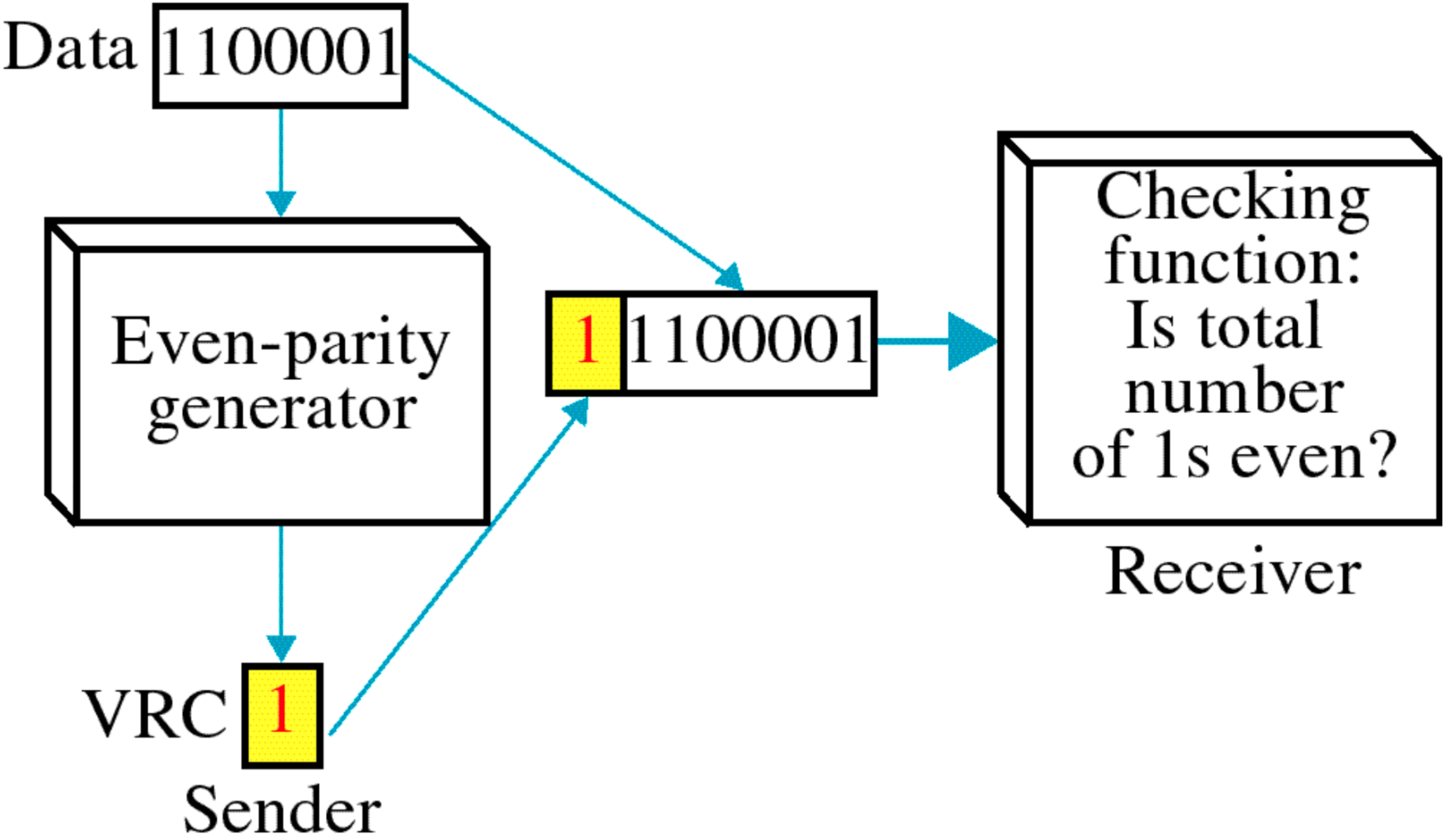
Redundancy


Data
1010000000010101
Generating function
1011101
Redundancy check
Sender
Checking function
Accept
Reject
Receiver
Data & redundancy check
1011101
1010000000010101

Four types of redundancy checks are used in data communications



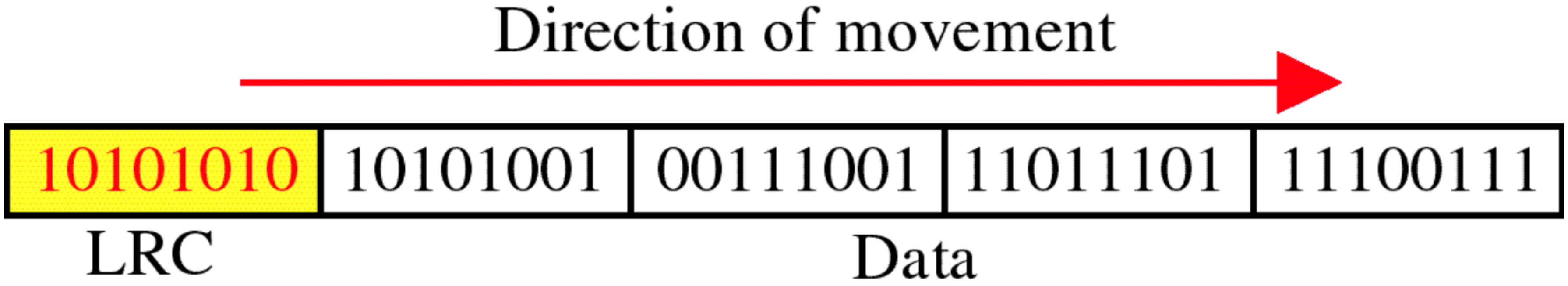
Vertical Redundancy Check
VRC



Performance

- It can detect single bit error
- It can detect burst errors only if the total number of errors is odd.

Longitudinal Redundancy Check
LRC



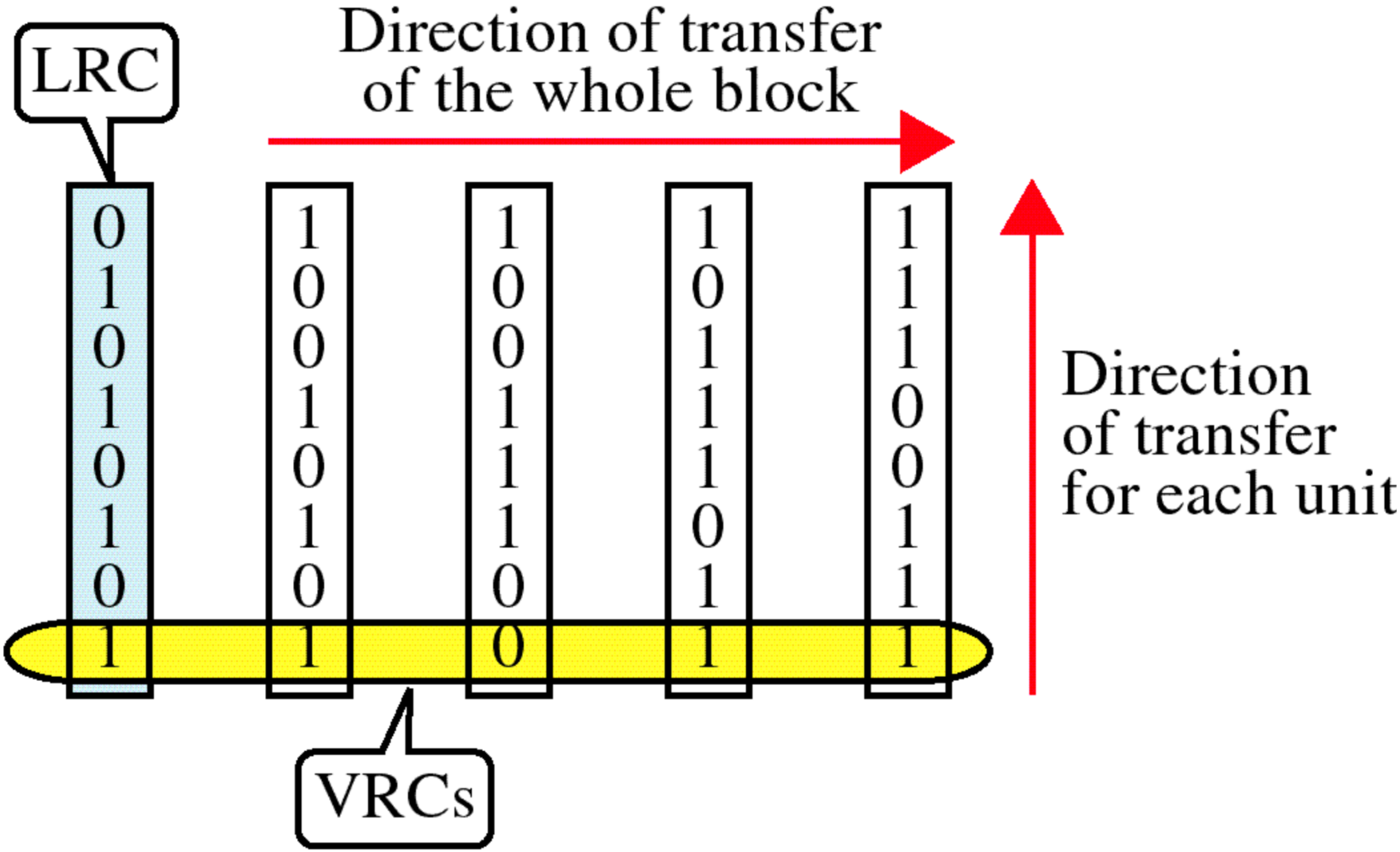
Direction of movement

LRC	Data			
10101010	10101001	00111001	11011101	11100111

Performance

- LCR increases the likelihood of detecting burst errors.
- If two bits in one data units are damaged and two bits in exactly the same positions in another data unit are also damaged, the LRC checker will not detect an error.

VRC and LRC



LRC

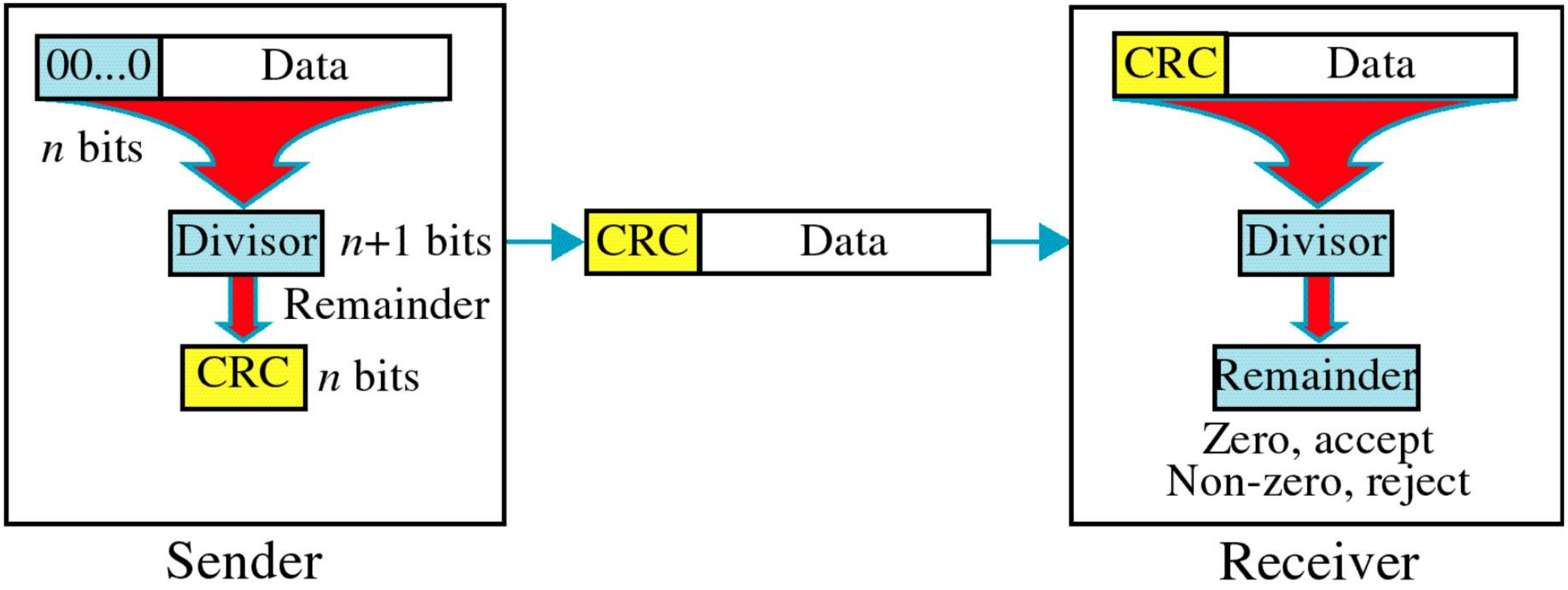
Direction of transfer of the whole block

0	1	1	1	1
1	0	0	0	1
0	0	0	1	1
1	1	1	1	0
0	0	1	1	0
1	1	1	0	1
0	0	0	1	1
1	1	0	1	1

Direction of transfer for each unit

VRCs

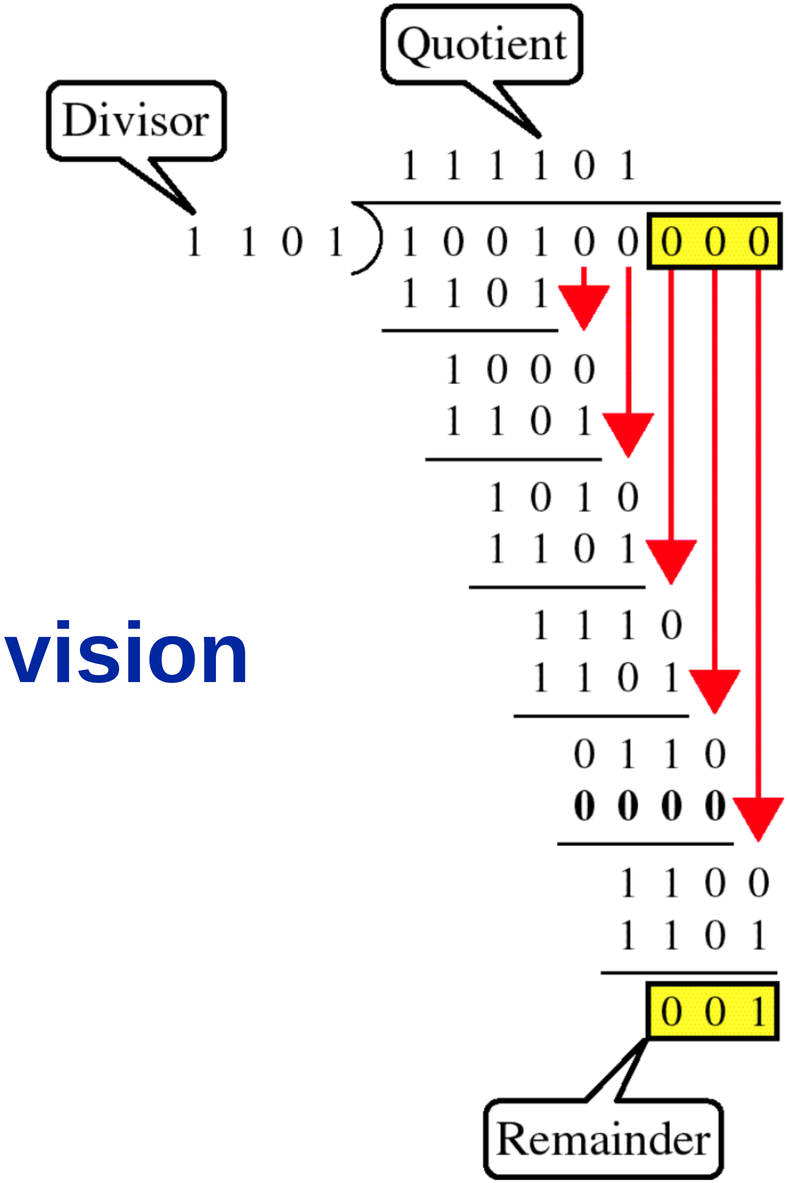
Cyclic Redundancy Check
CRC



Cyclic Redundancy Check

- Given a k-bit frame or message, the transmitter generates an n-bit sequence, known as a *frame check sequence (FCS)*, so that the resulting frame, consisting of (k+n) bits, is exactly divisible by some predetermined number.
- The receiver then divides the incoming frame by the same number and, if there is no remainder, assumes that there was no error.

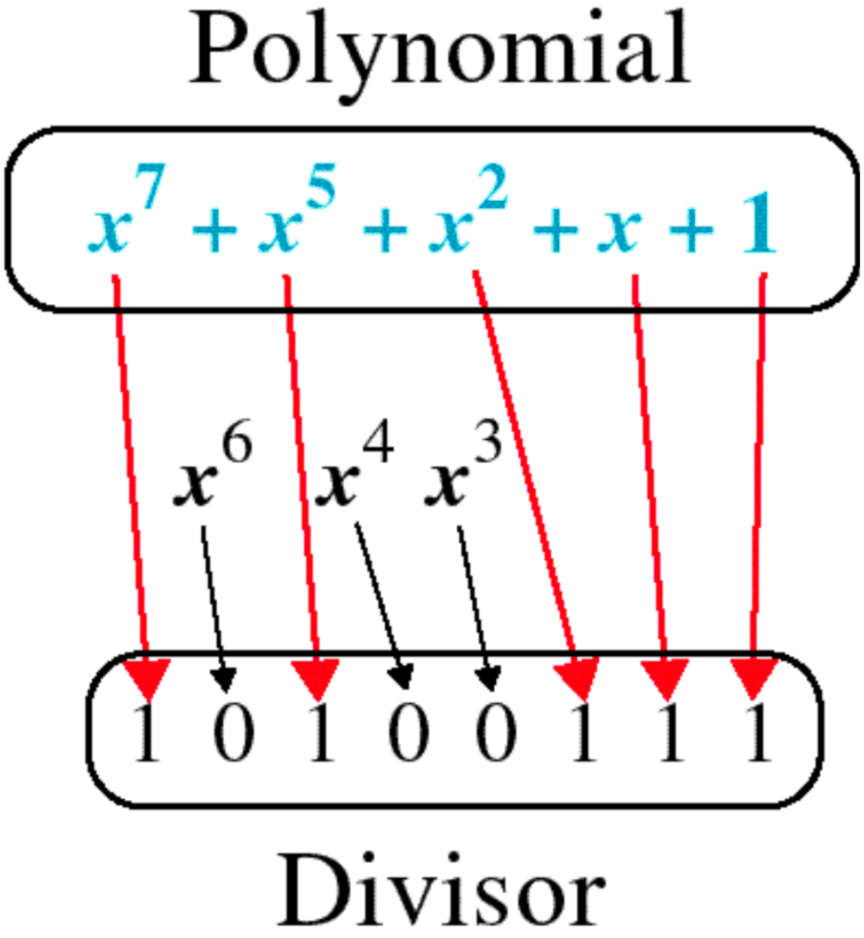
Binary Division



Polynomial

$$\boxed{x^7 + x^5 + x^2 + x + 1}$$

Polynomial and Divisor



Standard Polynomials

CRC-12

$$\boxed{x^{12} + x^{11} + x^{3} + x + 1}$$

CRC-16

$$\boxed{x^{16} + x^{15} + x^{2} + 1}$$

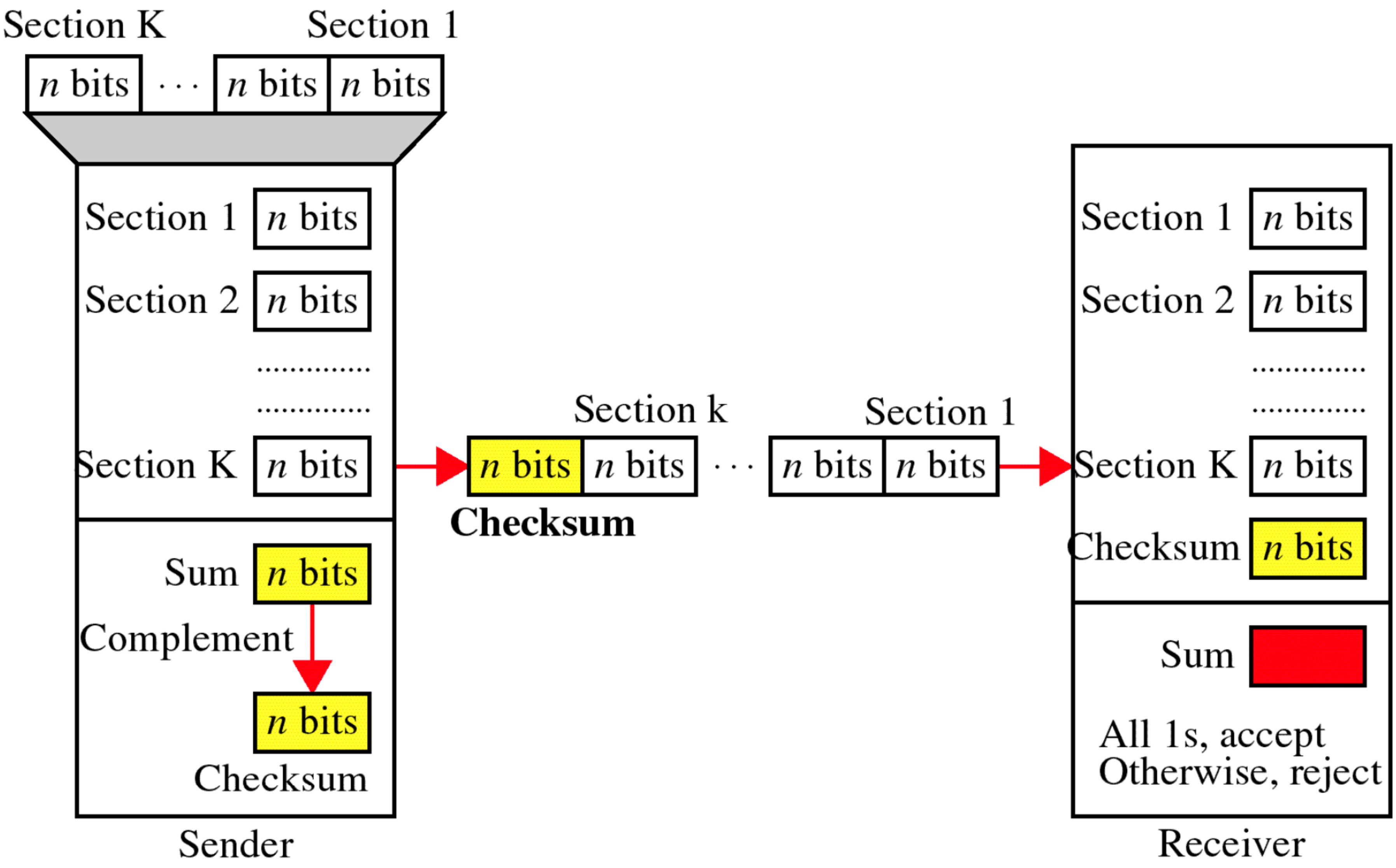
CRC-ITU

$$\boxed{x^{16} + x^{12} + x^{5} + 1}$$

CRC-32

$$\boxed{x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1}$$

Checksum


Section K
Section 1
n bits
. . .
n bits
n bits
Section 1
n bits
Section 2
n bits
....
....
Section K
n bits
Sum
n bits
Complement
n bits
Checksum
Sender
Section k
Section 1
n bits
n bits
. . .
n bits
n bits
Checksum
Section 1
n bits
Section 2
n bits
....
....
Section K
n bits
Checksum
n bits
Sum
All 1s, accept
Otherwise, reject
Receiver

At the sender

- The unit is divided into *k* sections, each of *n* bits.
- All sections are added together using one’s complement to get the sum.
- The sum is complemented and becomes the checksum.
- The checksum is sent with the data

At the receiver

- The unit is divided into *k* sections, each of *n* bits.
- All sections are added together using one's complement to get the sum.
- The sum is complemented.
- If the result is zero, the data are accepted: otherwise, they are rejected.

Performance

- ➔The checksum detects all errors involving an odd number of bits.
- ➔It detects most errors involving an even number of bits.
- ➔If one or more bits of a segment are damaged and the corresponding bit or bits of opposite value in a second segment are also damaged, the sums of those columns will not change and the receiver will not detect a problem.

Error Correction

It can be handled in two ways:

1) receiver can have the sender retransmit the entire data unit.
2) The receiver can use an error-correcting code, which automatically corrects certain errors.

Single-bit error correction

To correct an error, the receiver reverses the value of the altered bit. To do so, it must know which bit is in error.

Number of redundancy bits needed

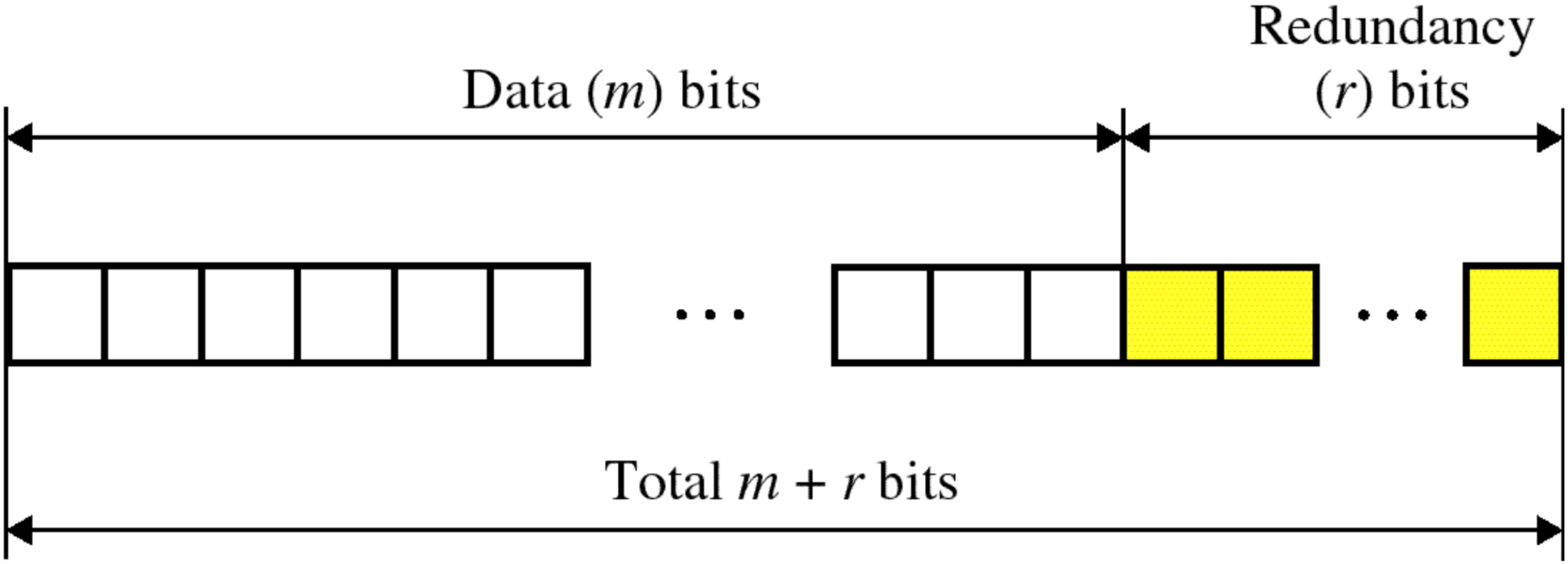
- Let data bits = *m*
- Redundancy bits = *r*

$\therefore$ Total message sent = $m+r$

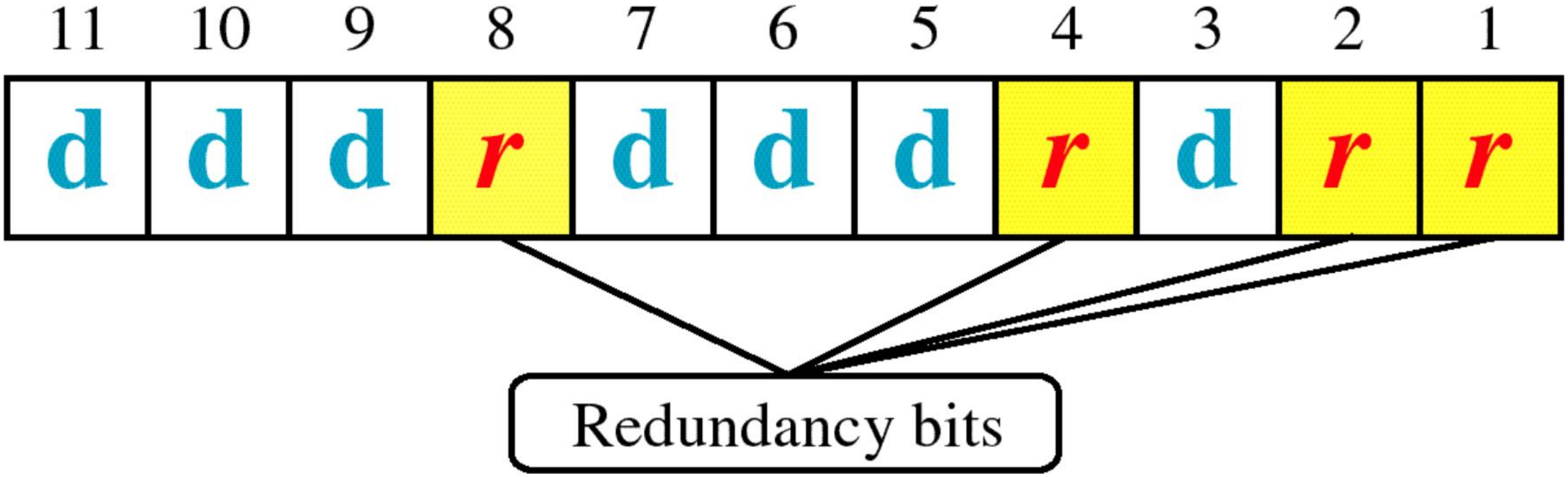
The value of r must satisfy the following relation:

$$2^r \geq m+r+1$$

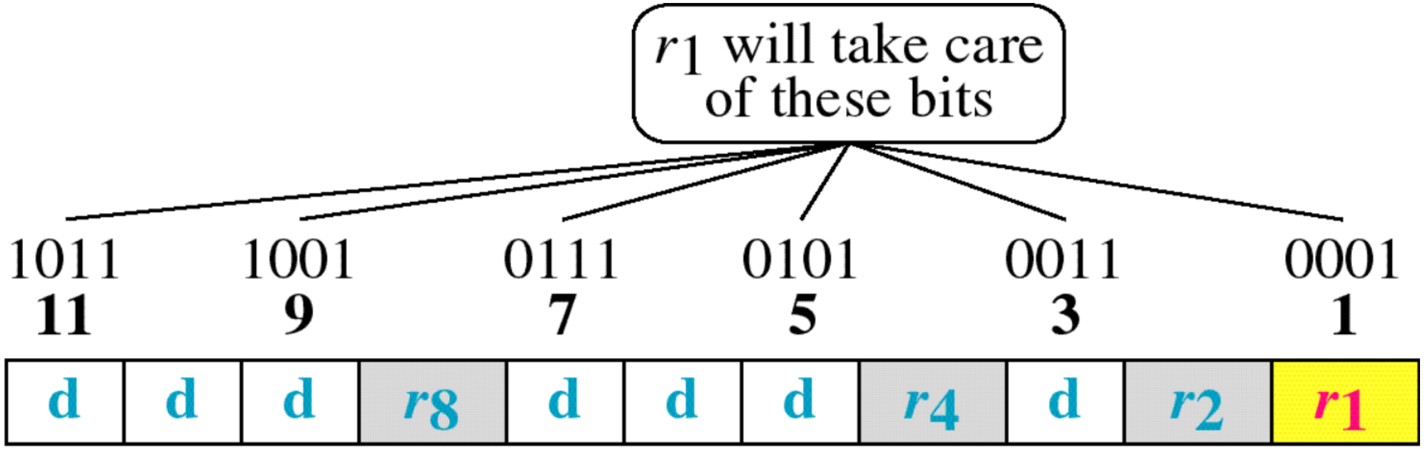
Error Correction



Hamming Code



Hamming Code

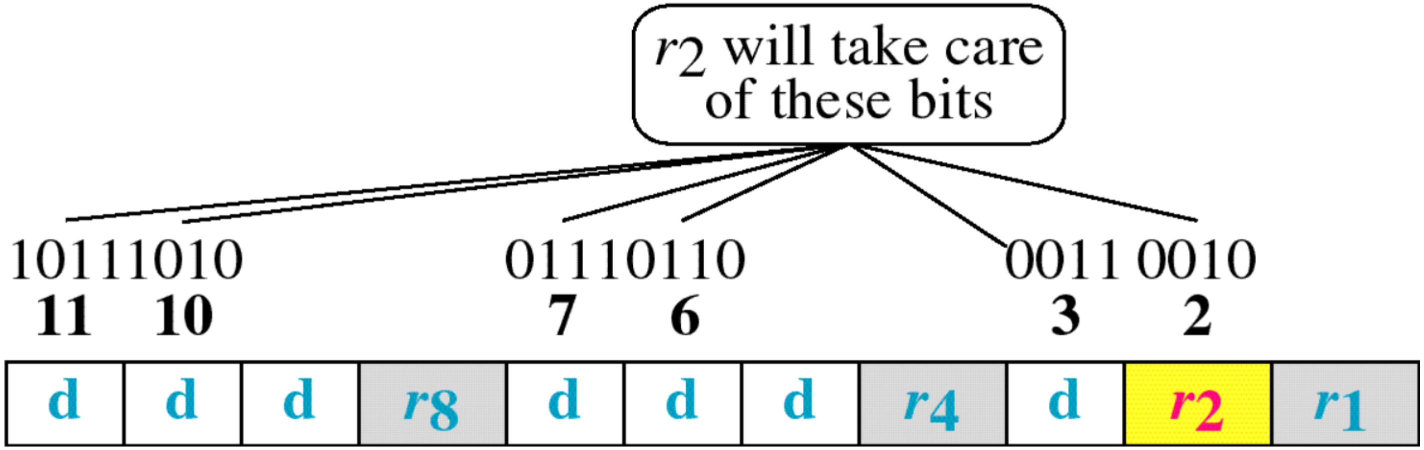


r_1 will take care of these bits

1011 1001 0111 0101 0011 0001

11 9 7 5 3 1

d d d r_8 d d d r_4 d r_2 r_1



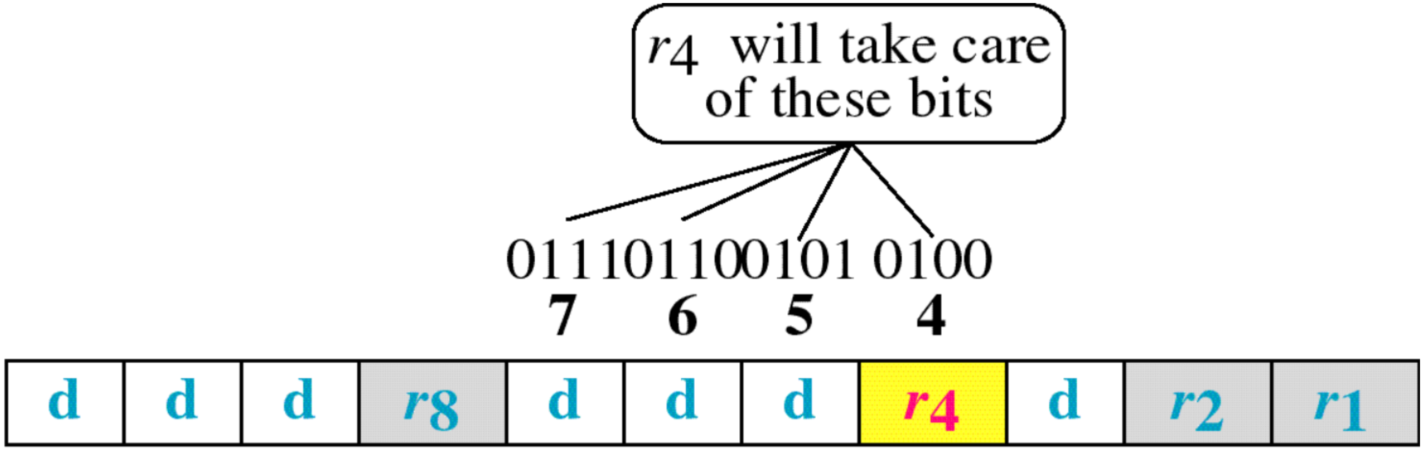
r_2 will take care of these bits

1011 1010 0111 0110 0011 0010

11 10 7 6 3 2

d d d r_8 d d d r_4 d r_2 r_1

Hamming Code

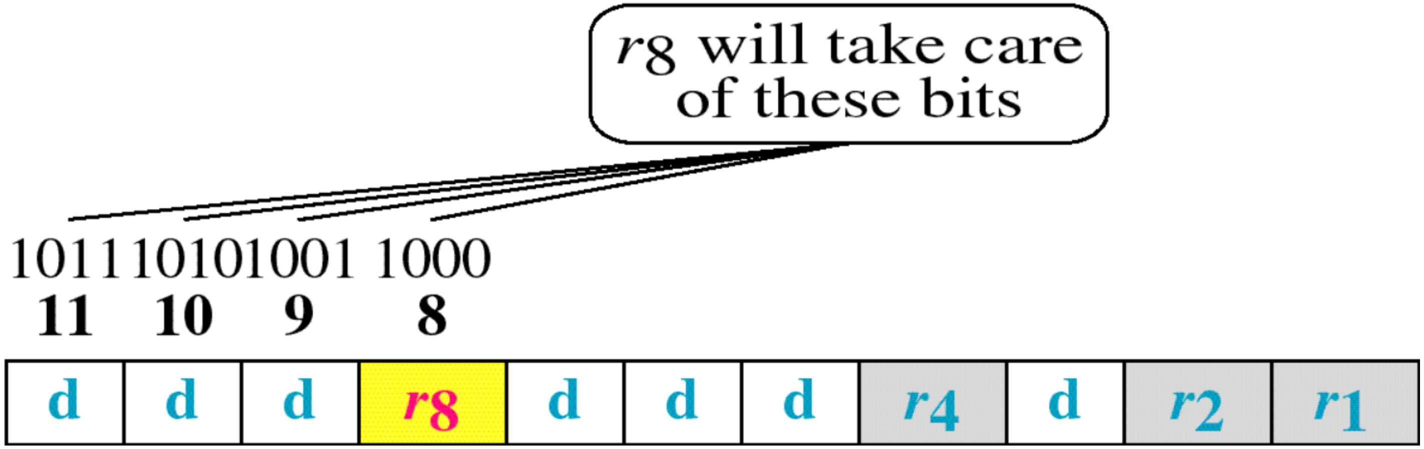


r_4 will take care of these bits

011101100101 0100

7 6 5 4

d	d	d	r_8	d	d	d	r_4	d	r_2	r_1



r_8 will take care of these bits

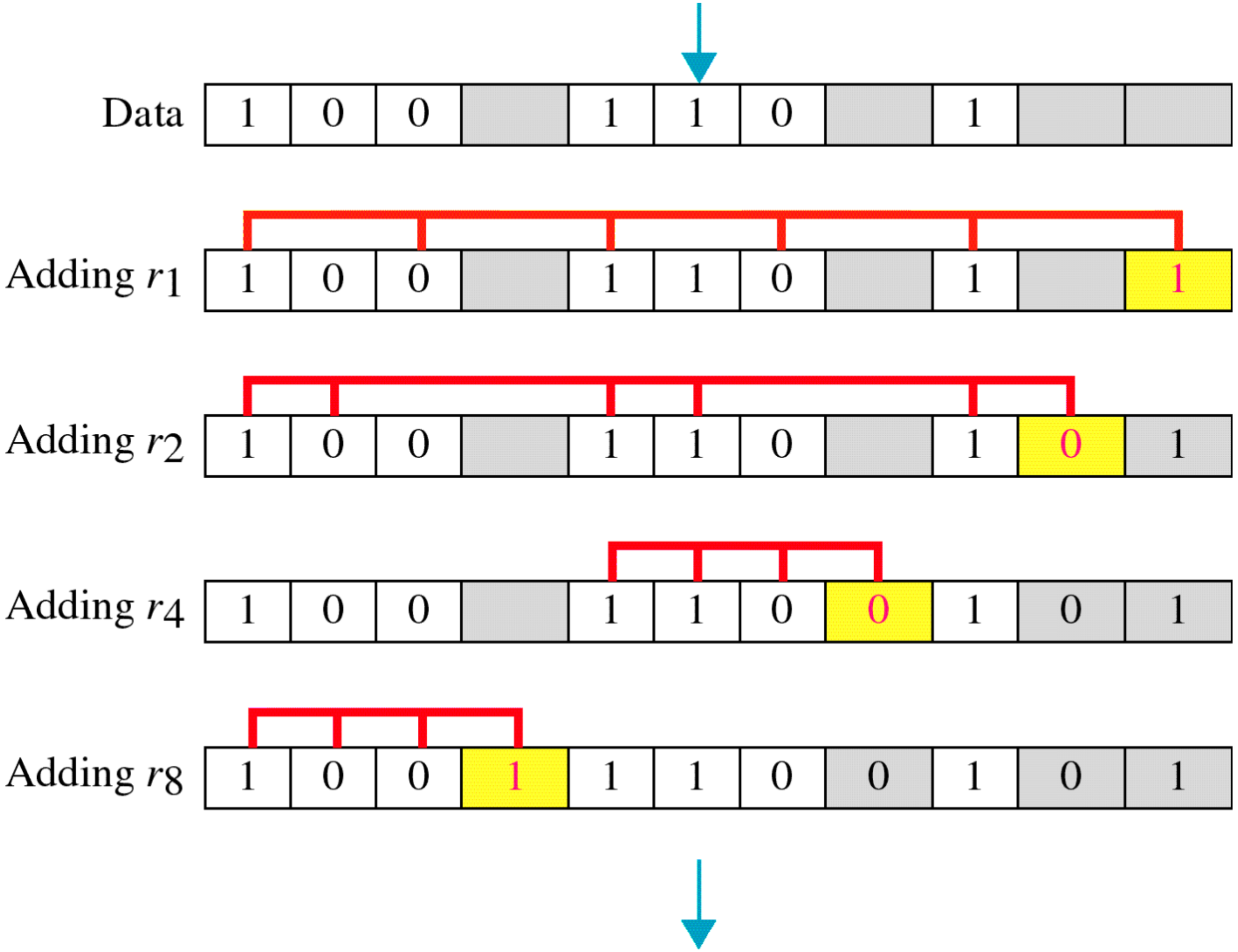
101110101001 1000

11 10 9 8

d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

Example of Hamming Code

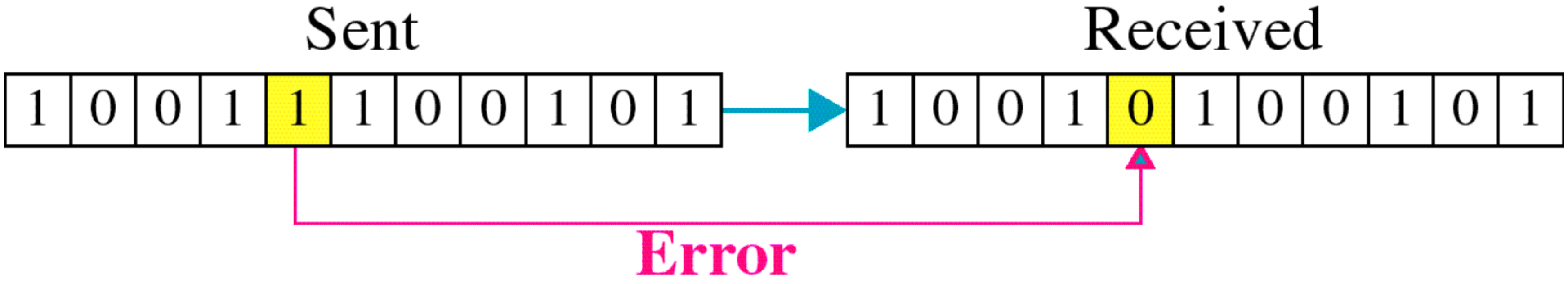
Data: 1 0 0 1 1 0 1



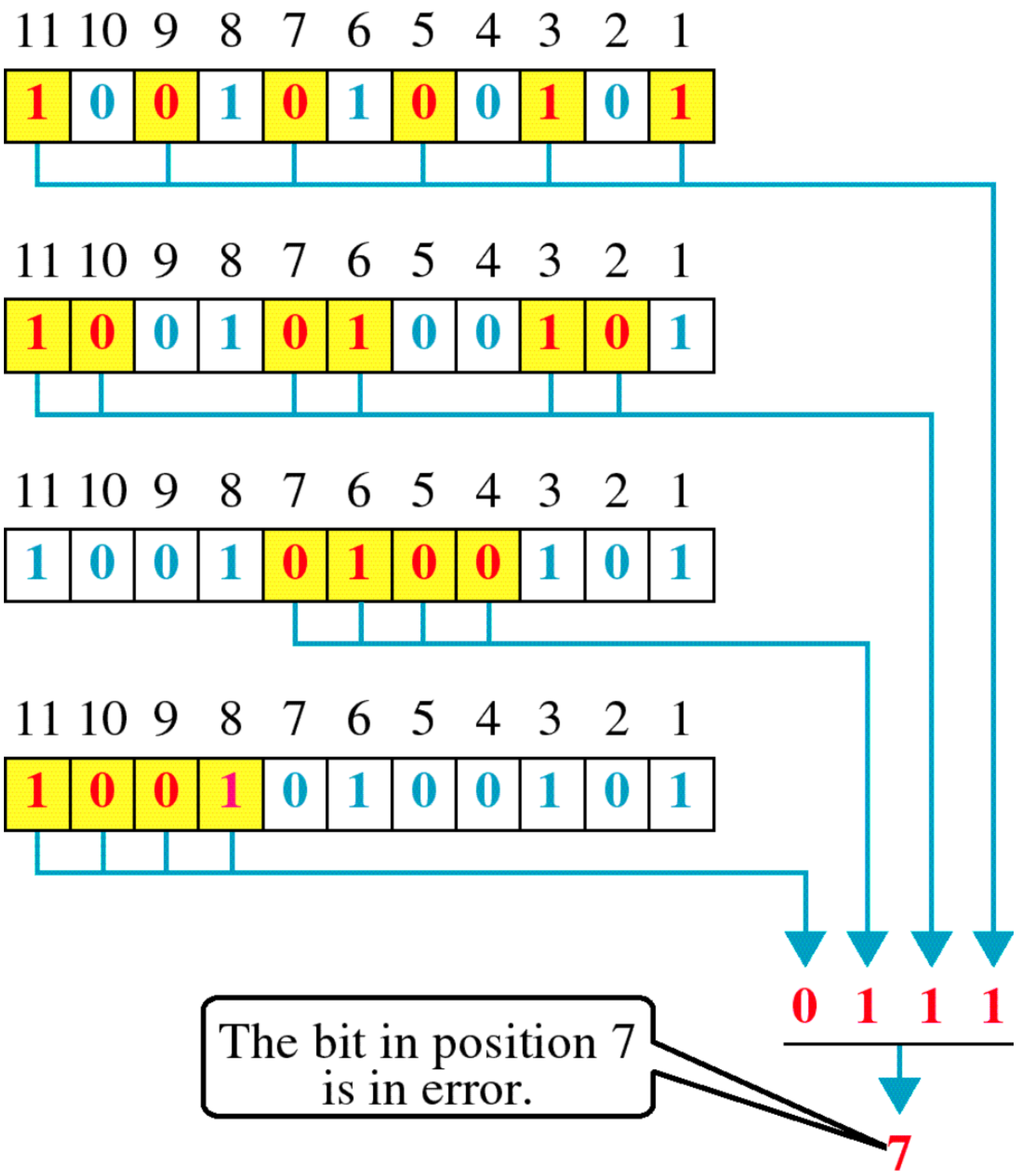
Data	1	0	0		1	1	0		1		
Adding r_1	1	0	0		1	1	0		1		1
Adding r_2	1	0	0		1	1	0		1	0	1
Adding r_4	1	0	0		1	1	0	0	1	0	1
Adding r_8	1	0	0	1	1	1	0	0	1	0	1

Code: 1 0 0 1 1 1 0 0 1 0 1

Single-bit error



Error Detection



11	10	9	8	7	6	5	4	3	2	1
1	0	0	1	0	1	0	0	1	0	1

11	10	9	8	7	6	5	4	3	2	1
1	0	0	1	0	1	0	0	1	0	1

11	10	9	8	7	6	5	4	3	2	1
1	0	0	1	0	1	0	0	1	0	1

11	10	9	8	7	6	5	4	3	2	1
1	0	0	1	0	1	0	0	1	0	1

The bit in position 7 is in error.

Data Link Control

Announcements

- Midterm: November 28, Monday, 11:40 – 13:30
  - Places:

    FENS G032 if (lastName[0] >= 'A' && lastName[0] <= 'D')

    FASS G022 if (lastName[0] >= 'E' && lastName[0] <= 'Ö')

    FASS G049 if (lastName[0] >= 'P' && lastName[0] <= 'Z')
- Exam will be closed book, closed notes
  - calculators are allowed
  - you are responsible all topics I covered in the class even if some of them are not in the book (I sometimes used other books) and not in the ppt files (I sometimes used board and showed applications on the computer)

Flow Control

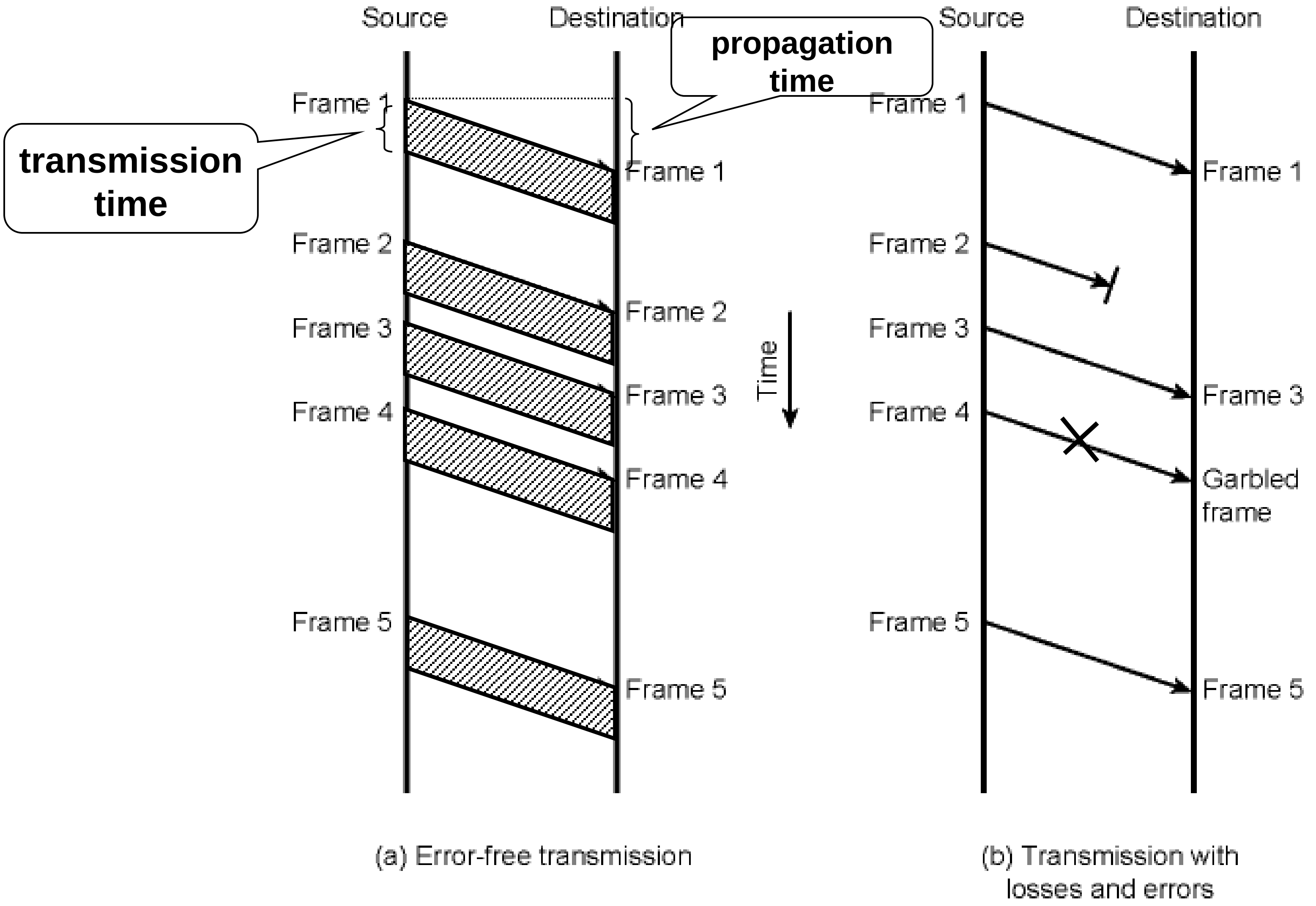
- In Data Link Layer, we deal with issues related to point to point links
  - Flow control is one of these issues

- Flow control is needed since the sending entity should not overwhelm the receiving entity
  - Recipient needs some time to process incoming packets
  - If sender sends faster than recipient

Performance Metrics and Delays (Section 5.3)

- Transmission time (delay)
  - – Time taken to emit all bits into medium
- Propagation time (delay)
  - – Time for a bit to traverse the link
- Processing time (delay)
  - – time spent at the recipient or intermediate node for processing
- Queuing time (delay)
  - – waiting time at the queue to be sent out

Model of Frame Transmission



transmission time

propagation time

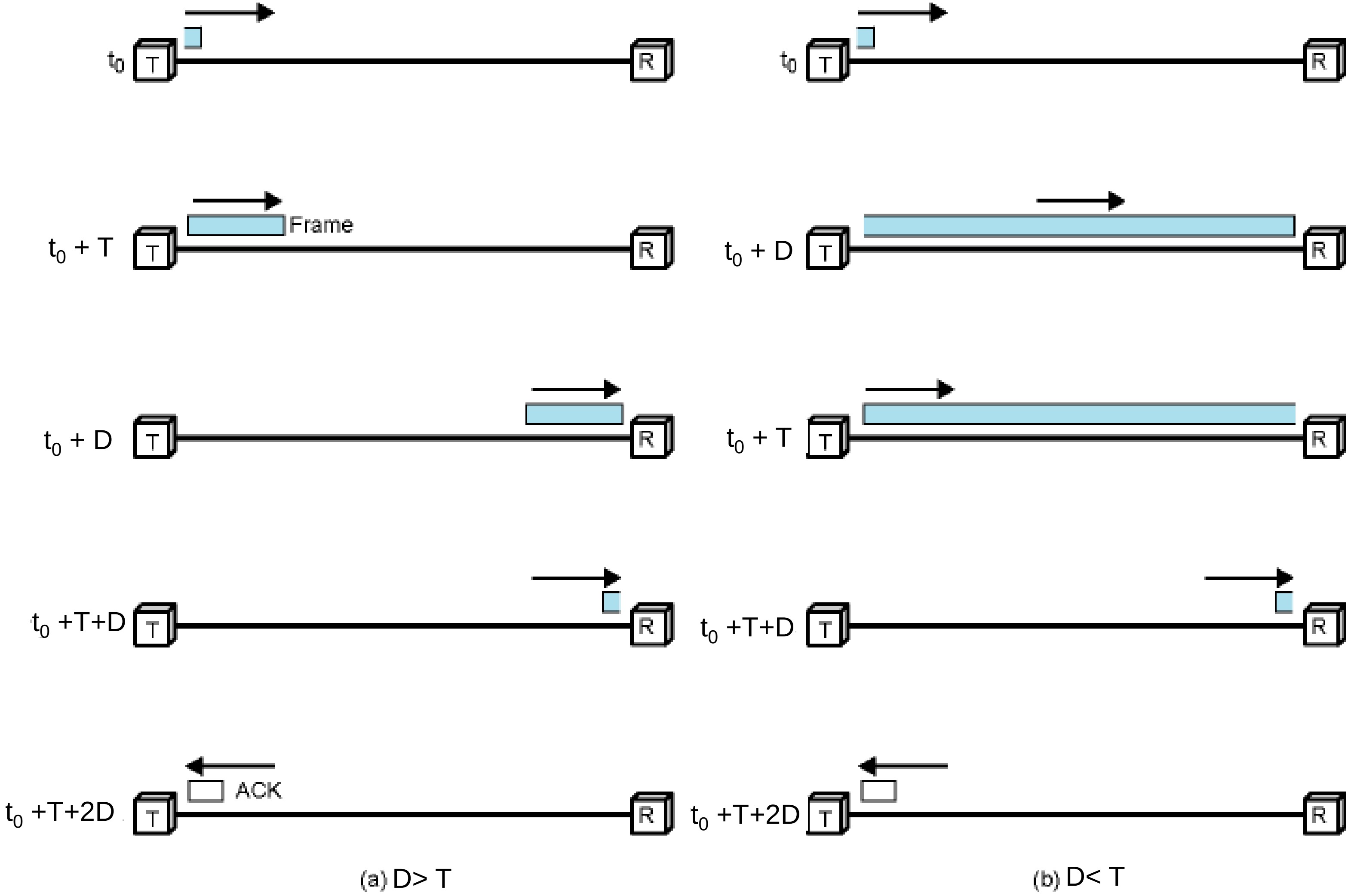
Stop and Wait Flow Control

- Source transmits frame
- Destination receives frame and replies with acknowledgement (ACK)
- Source waits for ACK before sending next frame
- Destination can stop flow by not sending ACK
- Works well for large frames
- Inefficient for smaller frames

Stop and Wait Flow Control

- However, generally large block of data split into small frames
  - Called “Fragmentation”
  - Advantages are
    - Limited buffer size at receiver
    - Errors detected sooner (when whole frame received)
      - On error, retransmission of smaller frames is needed
    - Prevents one station occupying medium for long periods
- Channel Utilization is higher when
  - the transmission time is longer than the propagation time
  - frame length is larger than the bit length of the

Stop and Wait Link Utilization - Details are on the board

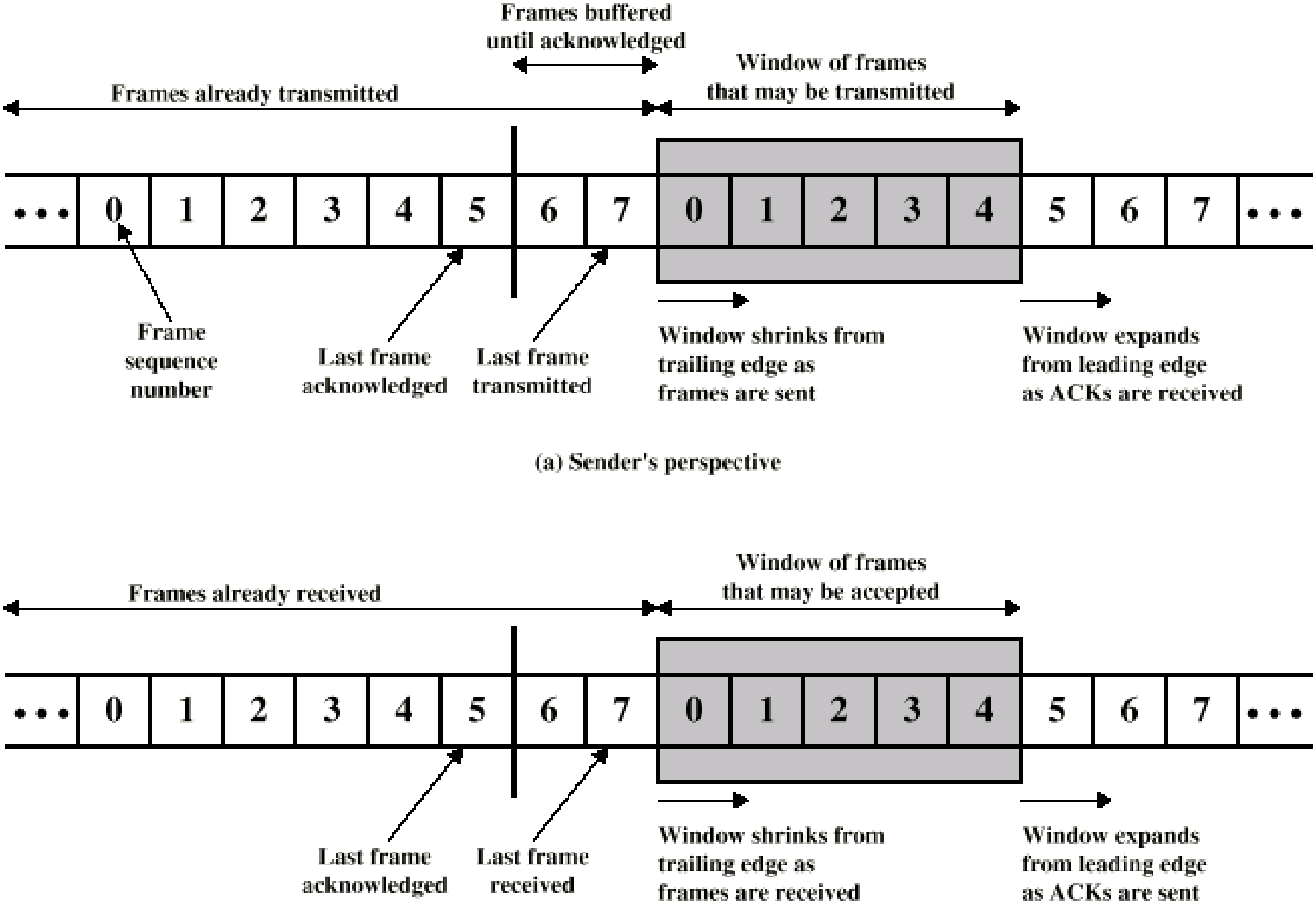


propagation time = D, transmission time = T

Sliding Window Flow Control

- The problem of “Stop and Wait” is not able to send multiple packets
- Sliding Window Protocol allows multiple frames to be in transit
- Receiver has buffer of *W* (called window size) frames
- Transmitter can send **up to *W* frames** without ACK
- Each frame is numbered
  - Sequence number bounded by size of the sequence number field (*k* bits)
  - thus frames are numbered modulo 2^k (0 … 2^k-1)
- ACK includes number of next frame expected

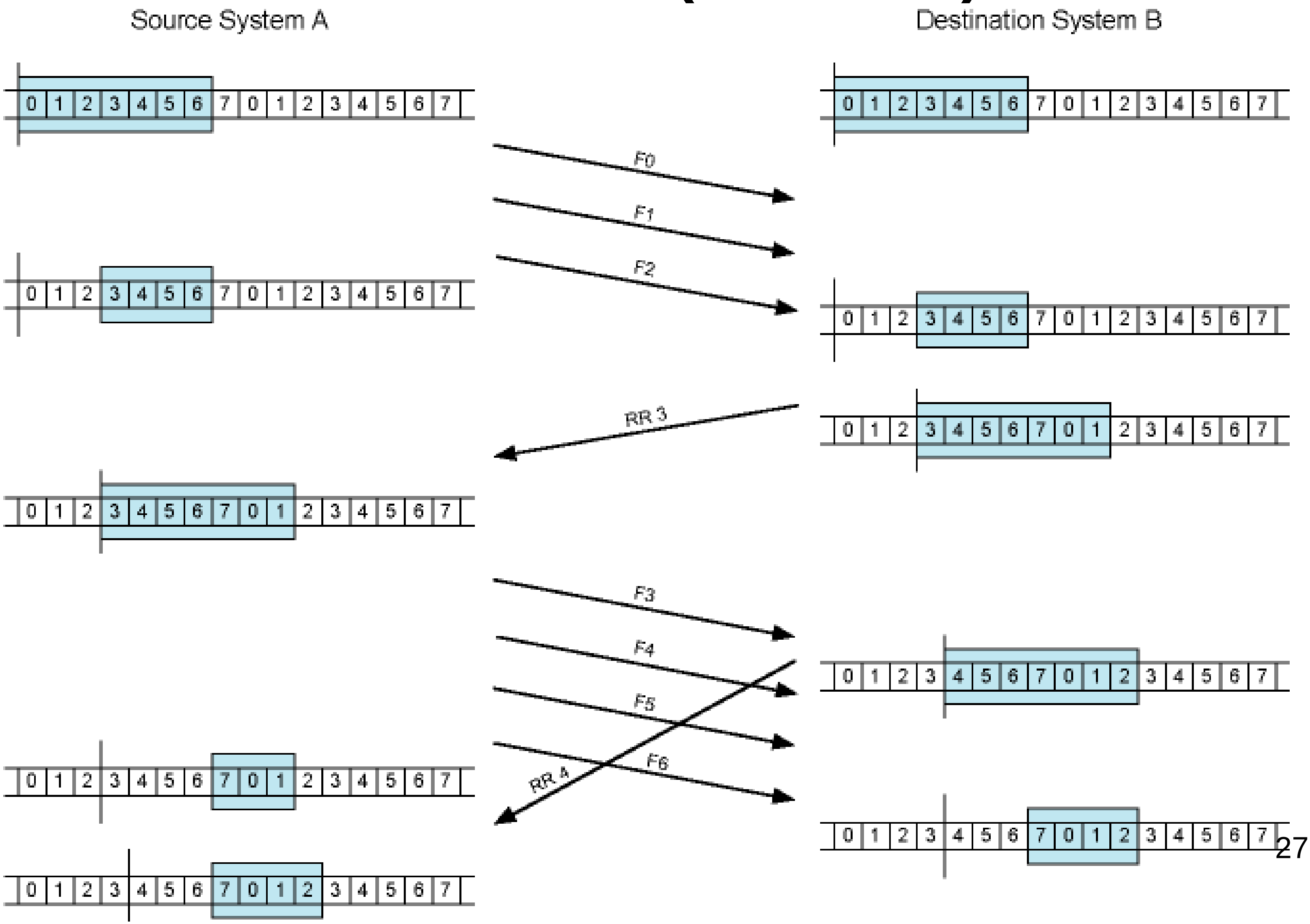
Sliding Window Flow Control (W = 7)


Frames buffered until acknowledged
Window of frames that may be transmitted
Frames already transmitted
• • •
0
1
2
3
4
5
6
7
0
1
2
3
4
5
6
7
• • •
Frame sequence number
Last frame acknowledged
Last frame transmitted
Window shrinks from trailing edge as frames are sent
Window expands from leading edge as ACKs are received
Window of frames that may be accepted
Frames already received
• • •
0
1
2
3
4
5
6
7
0
1
2
3
4
5
6
7
• • •
Last frame acknowledged
Last frame received
Window shrinks from trailing edge as frames are received
Window expands from leading edge as ACKs are sent

(a) Sender's perspective

(b) Receiver's perspective

Example of a Sliding Window Protocol (W = 7)

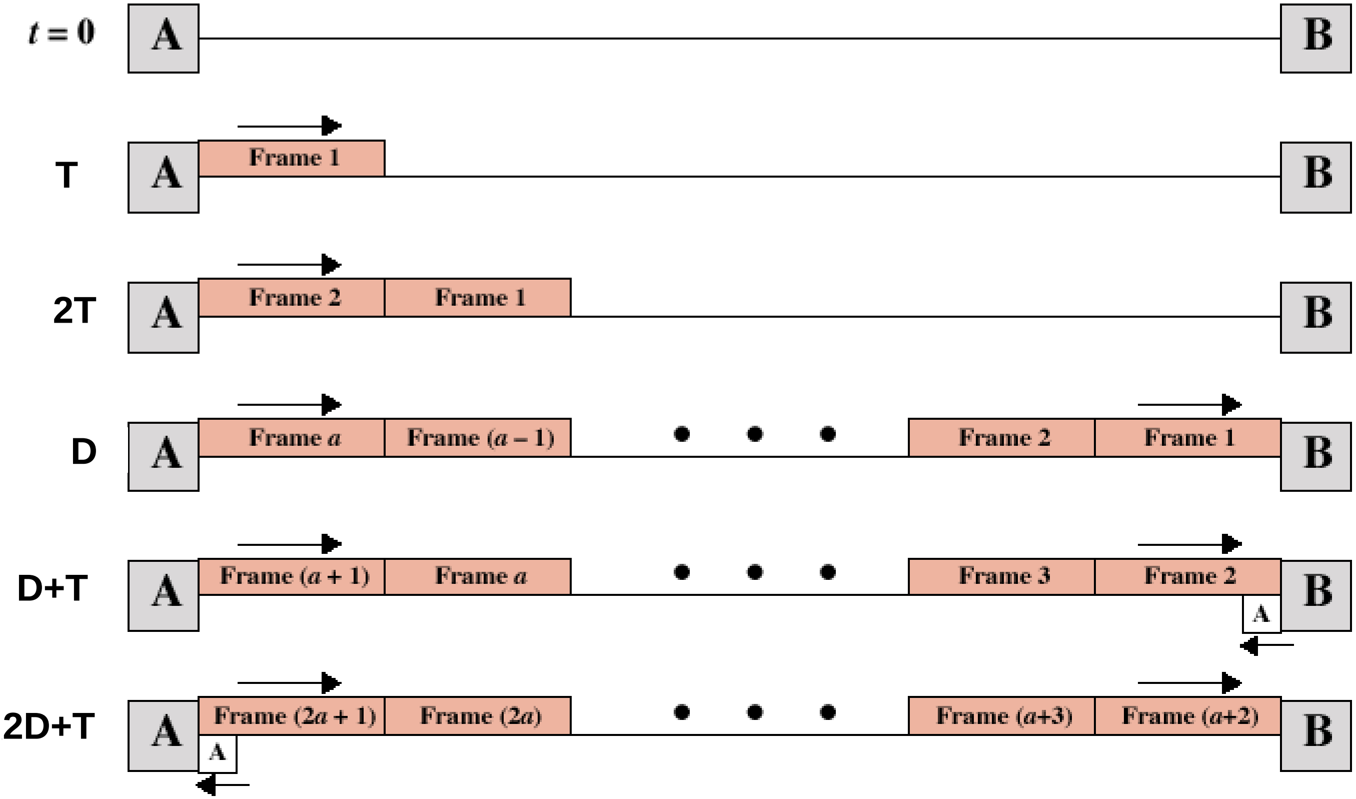


Sliding Window Enhancements in Implementation

- Receiver can acknowledge frames without permitting further transmission (*Receive Not Ready*)
  - Must send a normal acknowledgement to resume
- If the link is duplex, use *piggybacking*
  - Send data and ack together in one frame
    - frame has both data and ack fields
  - If no data to send, use acknowledgement frame
  - If data but no acknowledgement to send,

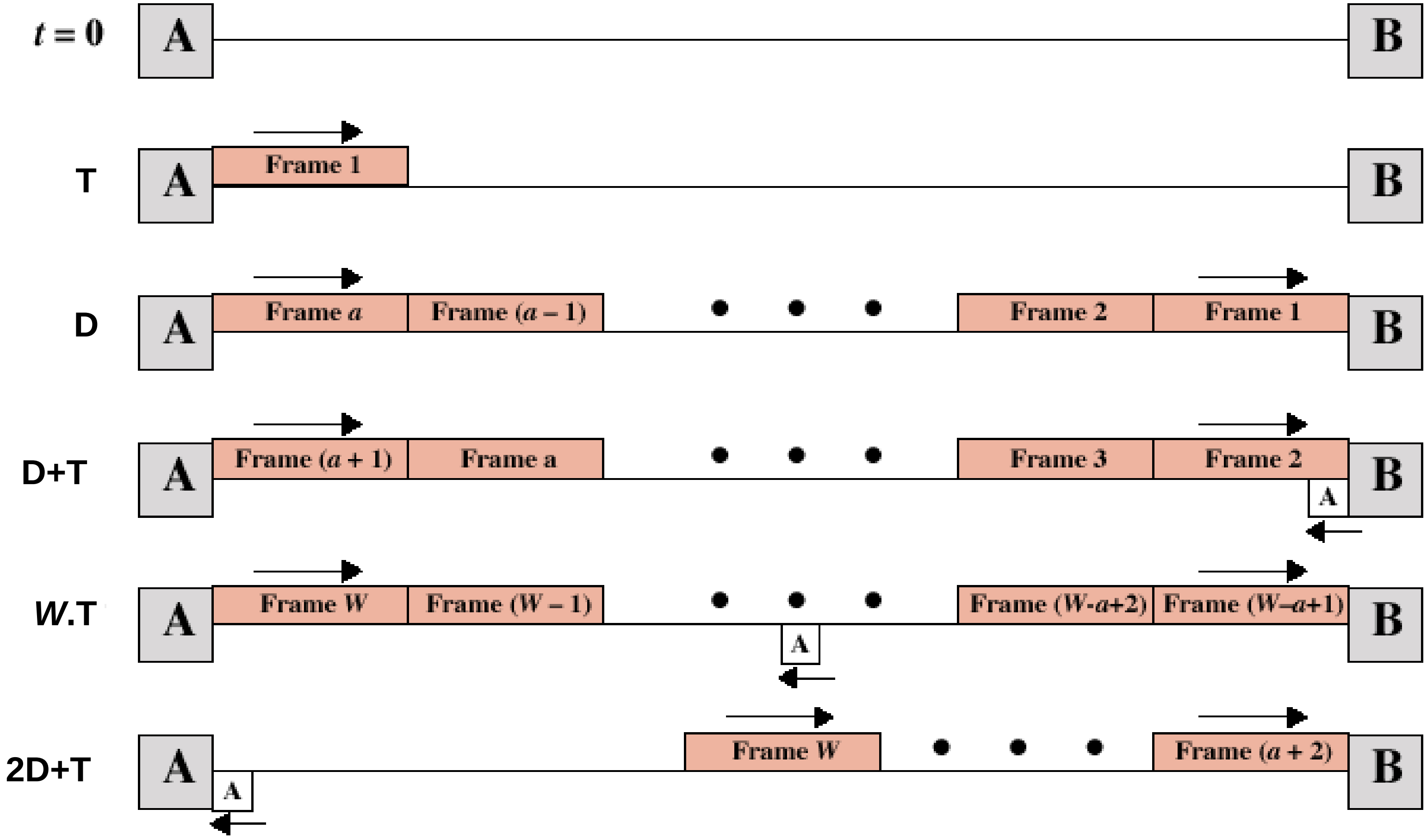
Sliding Windows Performance - 1

- two cases: W >= 2a+1 and W < 2a+1, where a=D/T
- details are on board



(a) $W \geq 2a + 1$ (W.T ≥ 2D+T)

Sliding Windows Performance - 2



(b) $W < 2a + 1$ (W.T < 2D+T)

Error Detection and Control

- So far we have seen flow control mechanisms where frames are transmitted without errors
  - – in real life any transmission facility may introduce errors
- So we have to
  - – detect errors
  - – if possible, correct errors (not in the scope of CS 408)
  - – adopt flow control algorithms such that erroneous frames are retransmitted

Types of Errors

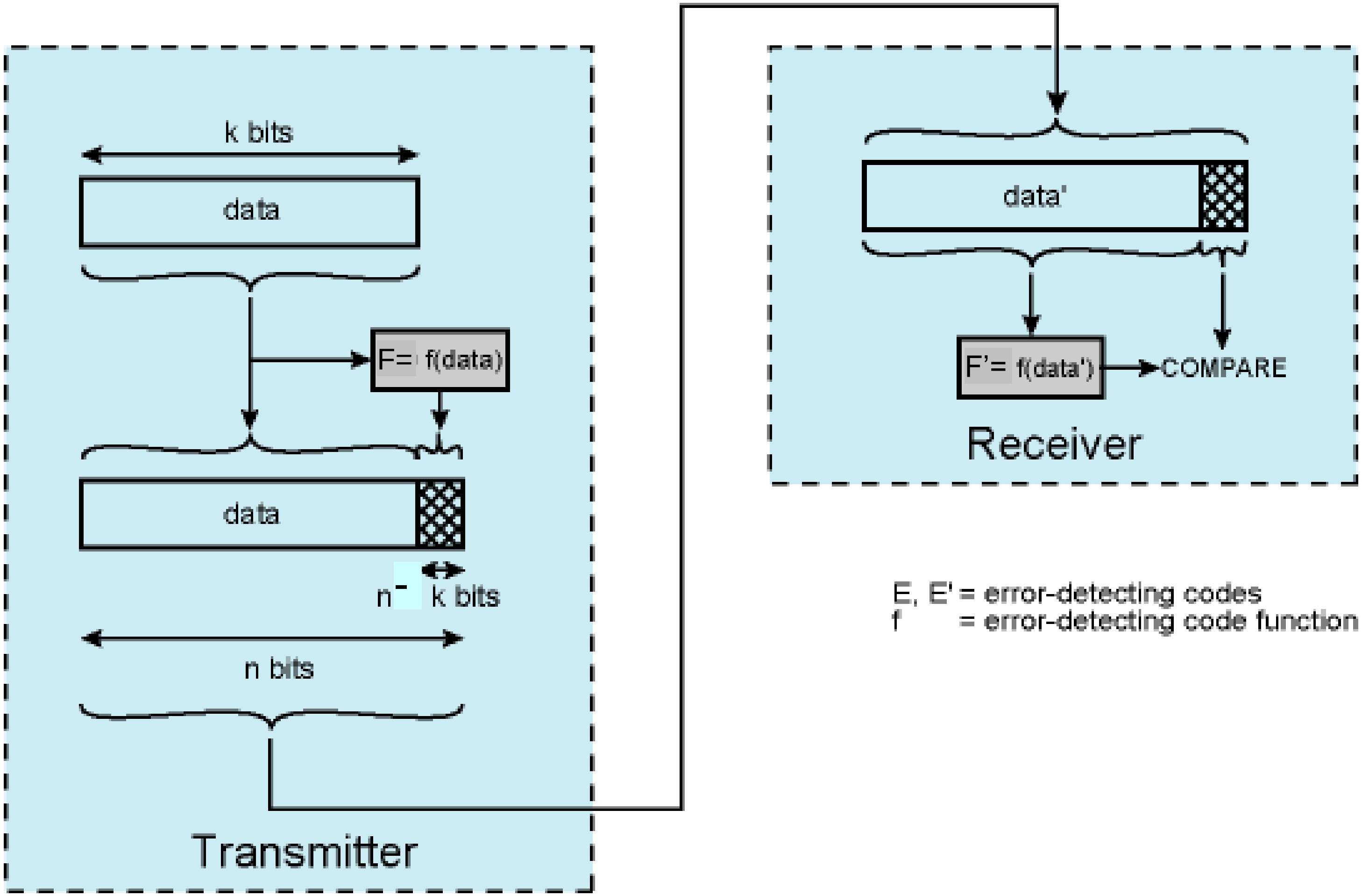
- Single bit errors
  - isolated errors
  - affects (flips) one bit, nearby bits are not altered
  - not so common in real life
- Burst errors
  - a sequence of bits are affected
  - most common case
  - a burst error of length *B* is a contiguous sequence of *B* bits in which the first and the last and <u>some</u> intermediate bits are erroneously flipped.

Error Detection

- Additional bits added by transmitter as *error detection code*
  - receiver checks this code

- Parity
  - single bit added to the end of the data
  - Value of parity bit is such that data and parity have even (even parity) or odd (odd parity) number of ones
  - Even number of bit errors goes undetected

Error Detection Process using Cyclic Redundancy Check



Cyclic Redundancy Check (CRC)

- For a data block of *k* bits, transmitter generates *n-k* bit frame check sequence (FCS) and appends it to the end of the data bits
- Transmits *n* bits, which is exactly divisible by some number (generator)
  - the length of the generator is *n*-*k*+1 and first and last bits are 1
- Receiver divides the received frame by generator
  - If no remainder, assume no error
- Division is binary division (not the same as

Cyclic Redundancy Check (CRC)

- Standard CRCs (generators are standard)
  - checks all single, double and odd number of errors
  - checks all burst errors with length less than or equal to the length of FCS (n-k)
  - checks most of the burst errors of longer length
    - for bursts of length n-k+1 (length of generator), probability of an undetected error is $1/2^{n-k-1}$
    - for longer bursts, probability of an undetected error is $1/2^{n-k}$

Error Control

- Actions to be taken against
  - Lost frames
  - Damaged frames
- Automatic repeat request (ARQ) mechanism components
  - Error detection
  - Positive acknowledgment
  - Retransmission after timeout
  - Negative acknowledgement and retransmission

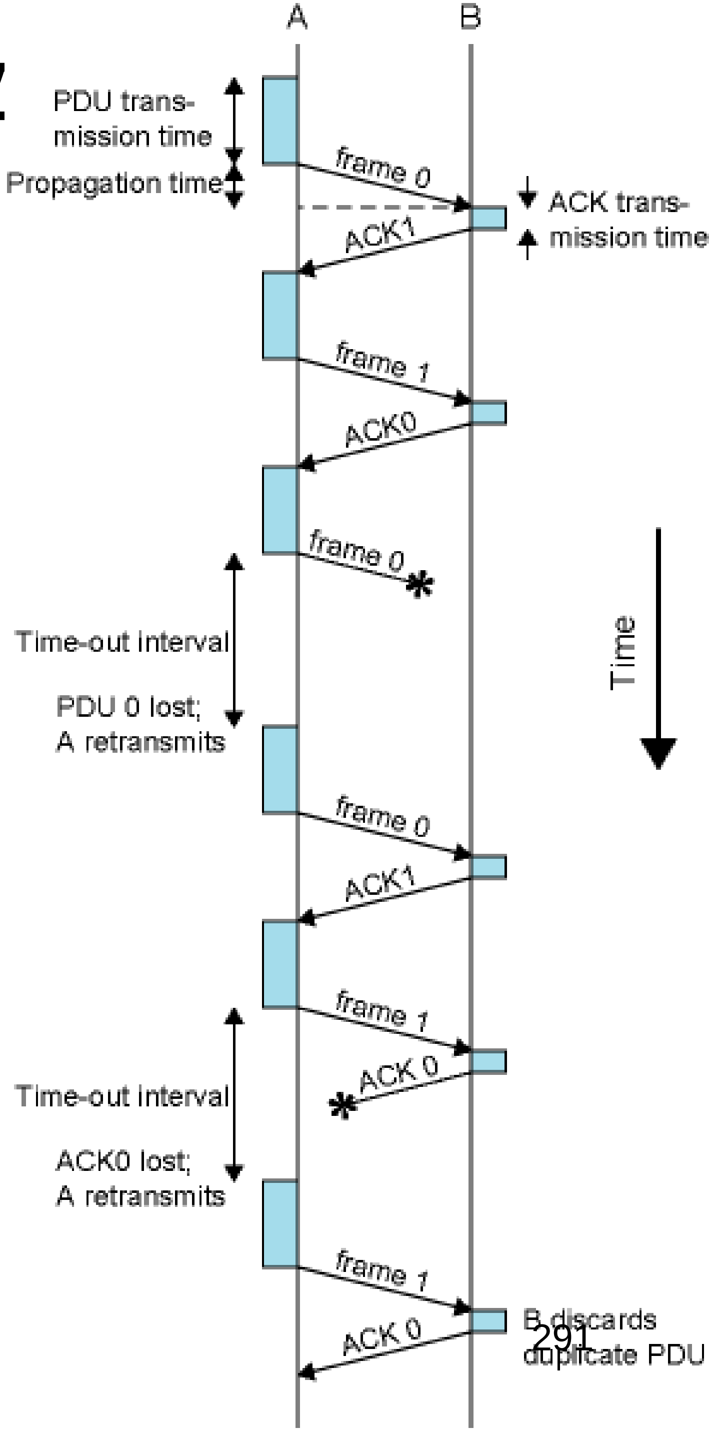
Automatic Repeat Request (ARQ)

- Stop-and-wait ARQ
- Go-back-N ARQ
- Selective-reject (selective retransmission) ARQ

Stop and Wait ARQ

- Source transmits single frame
- Wait for ACK
- If received frame is damaged, discard it
  - If transmitter receives no ACK within timeout, retransmits
- If ACK damaged,transmitter will not recognize it
  - Transmitter will retransmit after timeout
  - Receiver gets two copies of frame, but disregards one of them
  - Use ACK_0 and ACK_1

Stop-and-Wait A
Example



Stop and Wait - Pros and Cons

- Simple
- Inefficient

Go-Back-N ARQ

- Based on sliding window
- If no error, ACK as usual with next frame expected
  - ACK$_i$ means “I am ready to receive frame i” and “I received all frames between i and my previous ack”
- Sender uses window to control the number of unacknowledged frames
- If error, reply with rejection (negative ack)
  - Discard that frame and all future frames until the frame in error is received correctly
  - Transmitter must go back and retransmit that

Go-Back-N ARQ - Damaged Frame

- Receiver detects error in frame *i*
- Receiver sends “reject *i*”
- Transmitter gets “reject *i*”
- Transmitter retransmits frame *i* and all subsequent frames

Go-Back-N ARQ - Lost Frame (1)

- Frame *i* lost
- Transmitter sends frame *i*+1
- Receiver gets frame *i*+1 out of sequence
- Receiver sends “reject *i*”
- Transmitter goes back to frame *i* and retransmits it and all subsequent frames

Go-Back-N ARQ- Lost Frame (2)

- Frame *i* lost and no additional frame sent
- Receiver gets nothing and returns neither acknowledgment nor rejection
  - This is kind of a deadlock situation that needs to be resolved
- Transmitter times out and sends acknowledgment frame with P bit set to 1 (this is actually a command for ack request)
  - Receiver interprets this as an ack request command which it acknowledges with the number of the next frame it expects (*i*)
- Transmitter then retransmits frame *i*

Go-Back-N ARQ-Damaged/Lost Acknowledgment

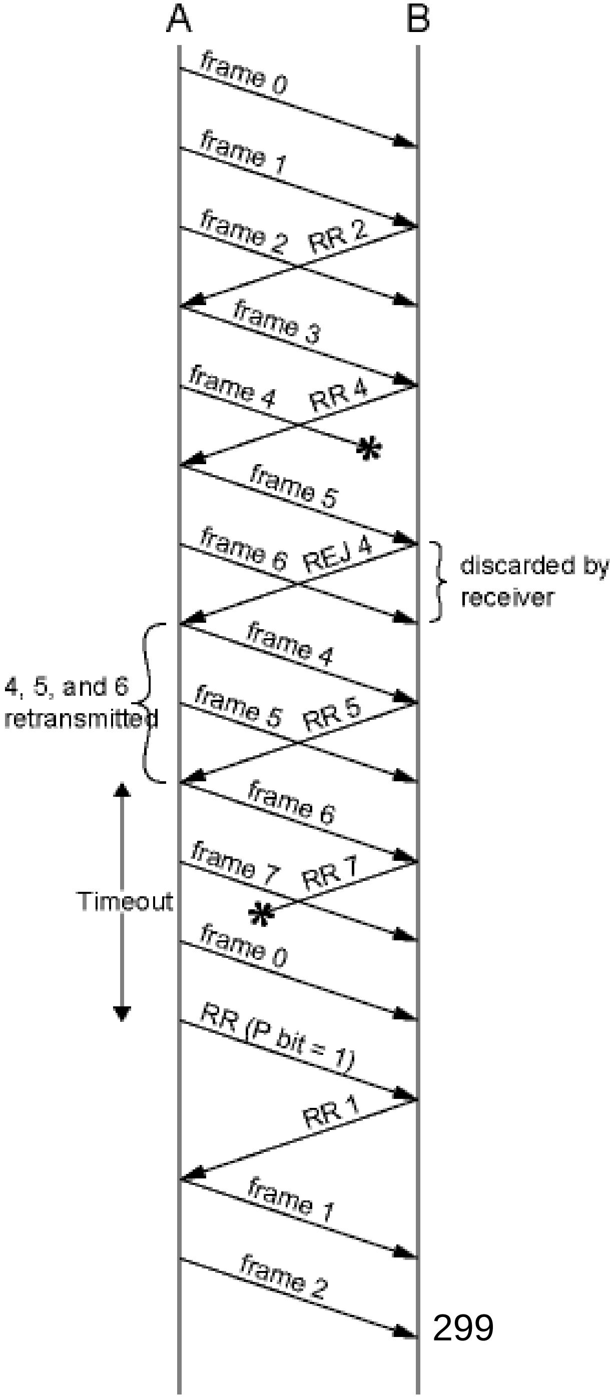
- Receiver gets frame *i* and sends acknowledgment (*i*+*1*) which is lost
- Acknowledgments are cumulative, so next acknowledgement (*i*+*n*) may arrive before transmitter times out on frame *i*

  ==> NO PROBLEM
- If transmitter times out, it sends acknowledgment request with P bit set, as before

Go-Back-N ARQ- Damaged Rejection

- As in lost frame (2)
  - – sender asks the receiver the last frame received and continue by retransmitting next frame

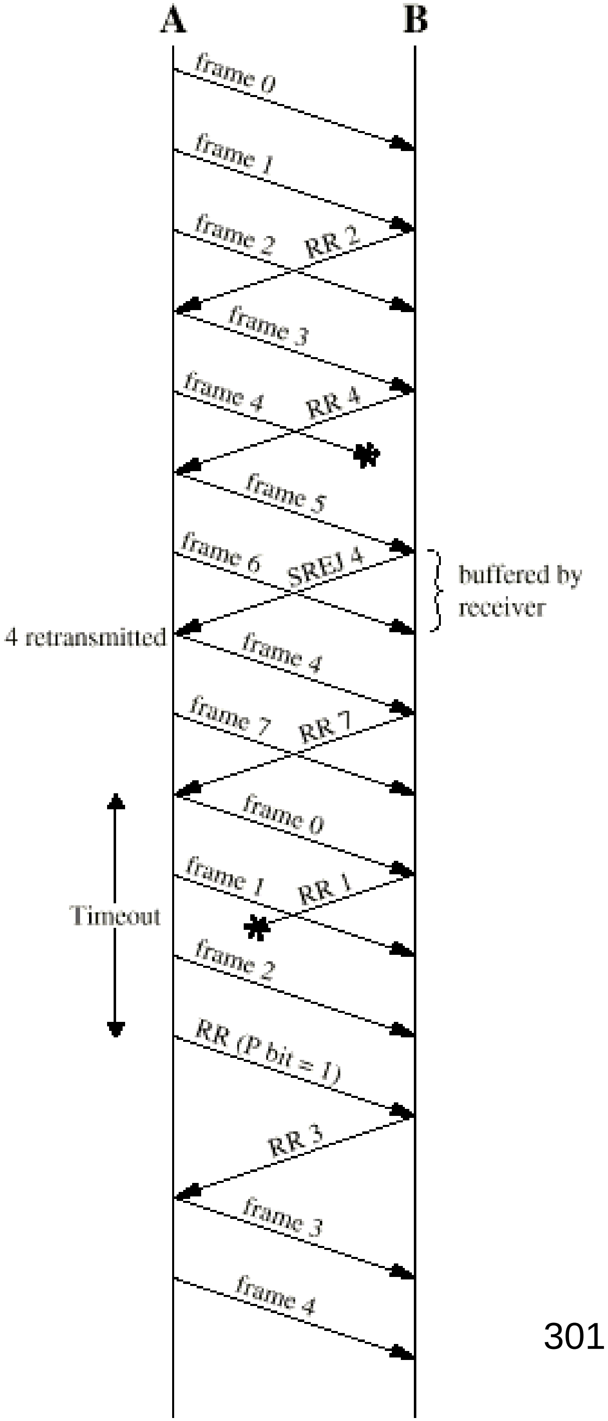
Go-Back-N AR Example



Selective Reject

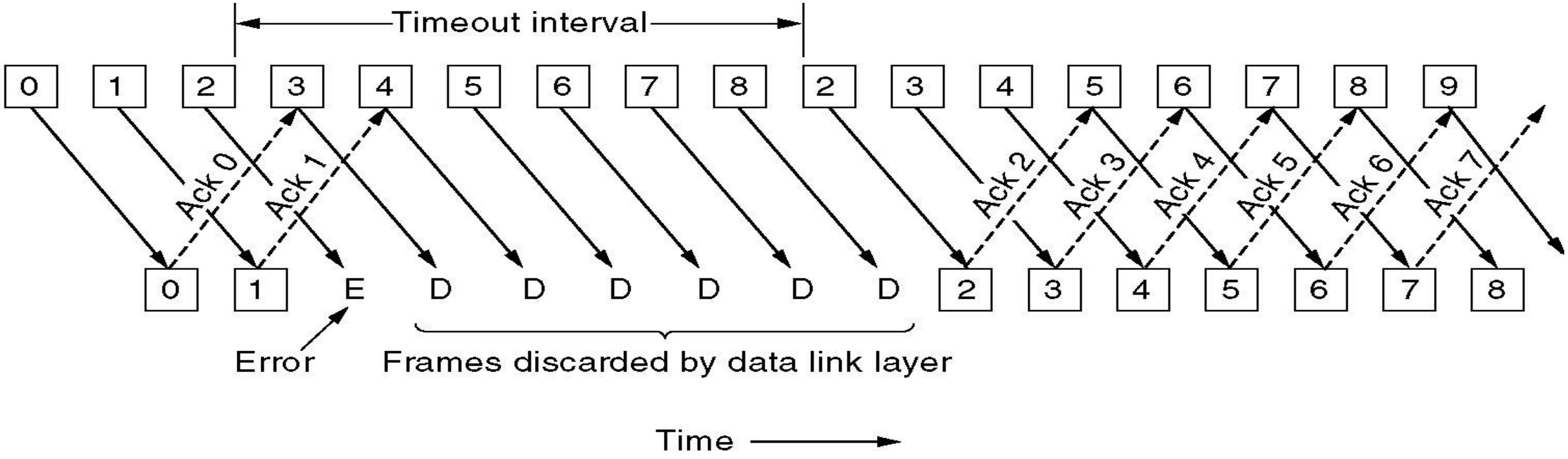
- Also called *selective retransmission*
- Only rejected frames are retransmitted
- Subsequent frames are accepted by the receiver and buffered
- Minimizes retransmissions
- Receiver must maintain large enough buffer
- Complex implementation

Selective Rej Diagram



Issues

- RR with P=1 is from HDLC standard
  - – pure protocol just have retransmissions after timeout



Issues – Window Size

- Given n-bit sequence numbers, what is Max window size?
  - go-back-n ARQ → 2^n-1
    - Why?
    - what about receiver’s window size?
      - It is 1, why?
  - selective-reject(repeat) → 2^{n-1}
    - Why?
- See the reasons on the board

Issues – Buffer Size

- Go-back-n ARQ
  - sender needs to keep a buffer equal to window size
    - for possible retransmissions
  - receiver does not need any buffer (for flow/error control)
    - why?
- Selective reject
  - sender needs to keep a buffer of window size for retransmissions
  - receiver keeps a buffer equal to window size

Issues - Performance

- Notes on board
- Appendix at the end of Chapter 14
    - selective reject ARQ is not in the book

High Level Data Link Control

- HDLC
- ISO Standard
- Basis for some other DLL protocols

HDLC Station Types

- Primary station
  - Controls operation of link
  - Frames issued are called commands
- Secondary station
  - Under control of primary station
  - Frames issued called responses
- Combined station
  - May issue commands and responses

HDLC Link Configurations

- Unbalanced
  - One primary and one or more secondary stations
  - Supports full duplex and half duplex
- Balanced
  - Two combined stations
  - Supports full duplex and half duplex

HDLC Transfer Modes (1)

- Normal Response Mode (NRM)
  - Unbalanced configuration
  - Primary initiates transfer to secondary
  - Secondary may only transmit data in response to command from primary
  - Terminal-host communication
    - Host computer as primary
    - Terminals as secondary
  - not so common nowadays

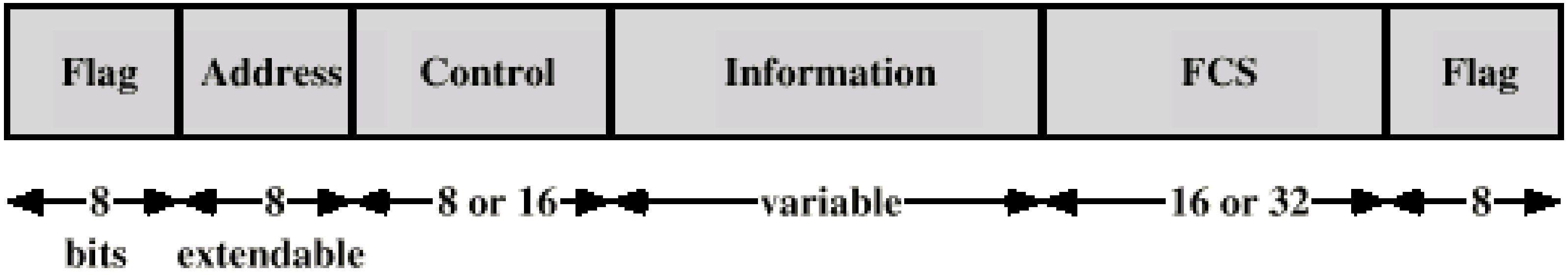
HDLC Transfer Modes (2)

- Asynchronous Balanced Mode (ABM)
  - Balanced configuration
  - Either station may initiate transmission without receiving permission
  - Most widely used

Frame Structure

- All transmissions in frames
- Single frame format for all data and control exchanges

Frame Structure Diagram



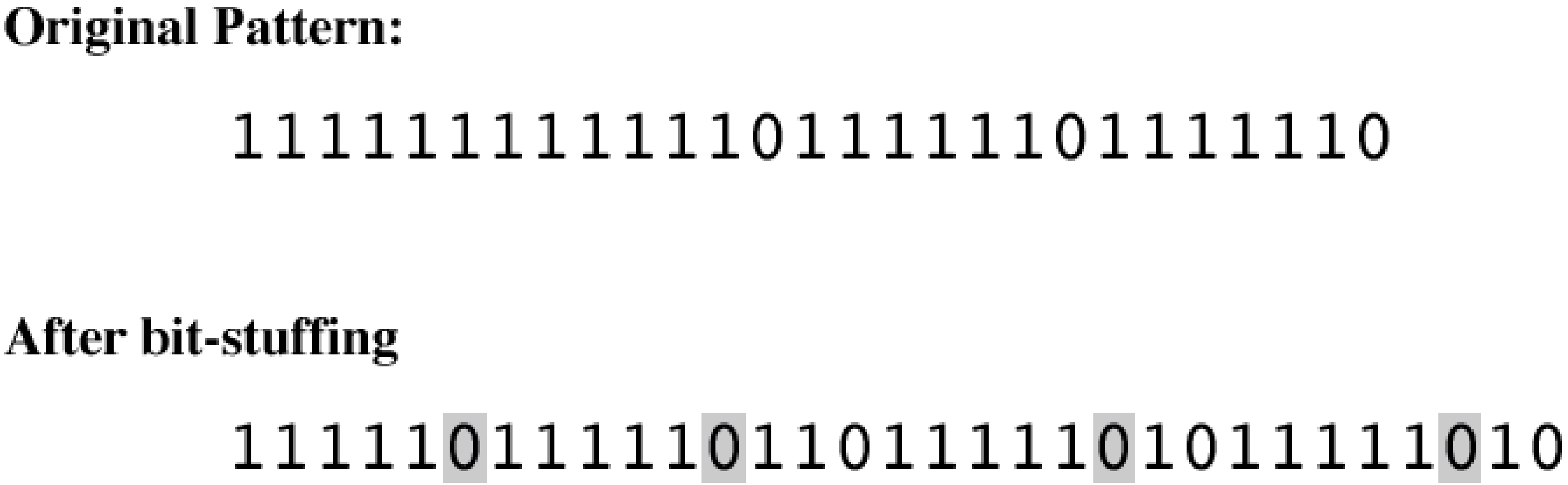
Flag | Address | Control | Information | FCS | Flag

8 bits | 8 extendable | 8 or 16 | variable | 16 or 32 | 8

Flag Fields

- Delimit frame at both ends
- 01111110
- Receiver hunts for flag sequence to synchronize
- Bit stuffing used to avoid confusion with data containing 01111110
  - 0 inserted after every sequence of five 1s
  - If receiver detects five 1s after a 0 it checks next bit
    - If 0, it is deleted

Bit Stuffing Example



Original Pattern:

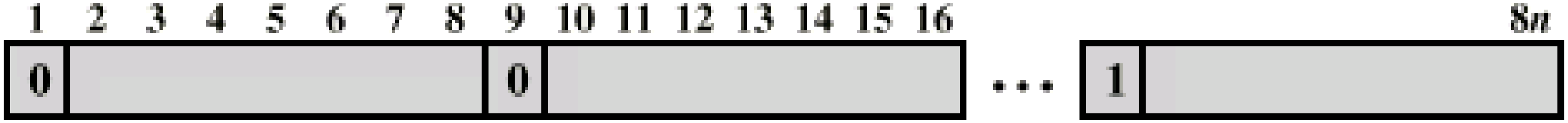
111111111111011111101111110

After bit-stuffing

1111101111101101111101011111010

Address Field

- Identifies secondary station that sent or will receive frame
- Usually 8 bits long (but 7 bits are effective)
- May be extended to multiples of 7 bits with prior agreement
  - leftmost bit of each octet indicates that it is the last octet (1) or not (0)

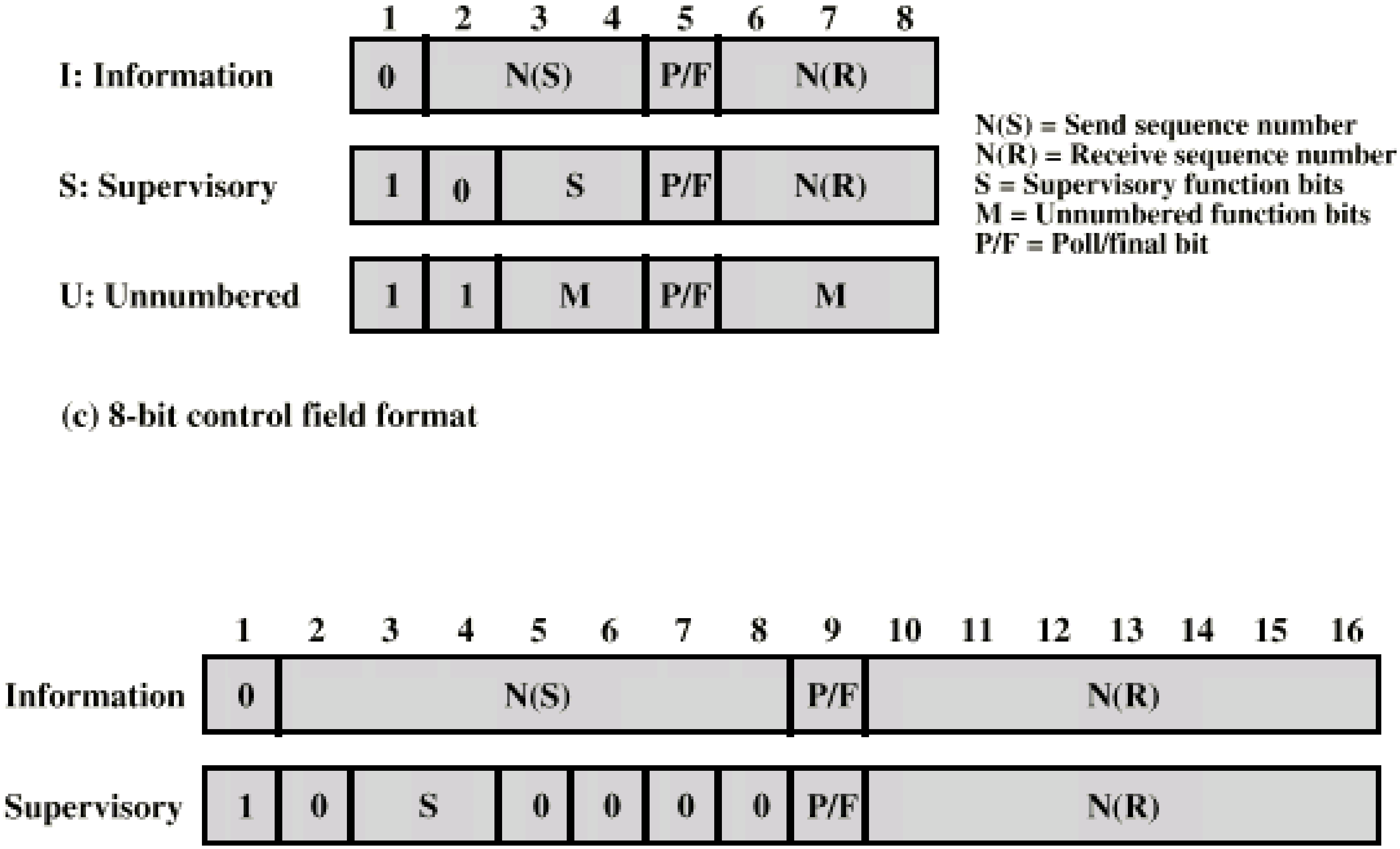


(b) Extended Address Field

Frame Types

- Information frame- data to be transmitted to user
  - Acknowledgment is piggybacked on information frames (only for positive acknowledgment)
- Supervisory frame – ARQ messages (RR/RNR/REJ/SREJ) when piggyback not used (actually only RR can be piggybacked; for the other, we need Supervisory frames)
- Unnumbered frame – supplementary link control functions. For examples,
  - setting the modes
  - disconnect

Control Field Diagram



	1	2	3	4	5	6	7	8
I: Information	0	N(S)			P/F	N(R)		
S: Supervisory	1	0	S		P/F	N(R)		
U: Unnumbered	1	1	M		P/F	M		

N(S) = Send sequence number
N(R) = Receive sequence number
S = Supervisory function bits
M = Unnumbered function bits
P/F = Poll/final bit

(c) 8-bit control field format

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Information	0	N(S)							P/F	N(R)						
Supervisory	1	0	S		0	0	0	0	P/F	N(R)						

(d) 16-bit control field format

Poll/Final Bit

- Use of this bit depends on context. A typical use is below.
- Command frame
  - P bit set to 1 to solicit (poll) supervisory frame from peer
- Response frame
  - F bit set to 1 to indicate response to soliciting command

Information Field

- Only in information and some unnumbered frames
- Must contain integral number of octets
- Variable length

Frame Check Sequence Field

- FCS
- Error detection
- 16 bit CRC
- Optional 32 bit CRC

HDLC Operation

- Exchange of information, supervisory and unnumbered frames
- Three phases
  - Initialization
  - Data transfer
  - Disconnect

Initialization

- Issue one of six *set-mode* commands
  - Signals other side that initialization is requested
  - Specifies mode (NRM, ABM, ARM)
  - Specifies 3- or 7-bit sequence numbers
- If request accepted, HDLC module on other side transmits "unnumbered acknowledged" (UA) frame
- If request rejected, "disconnected mode" (DM) sent
- All sent as unnumbered frames

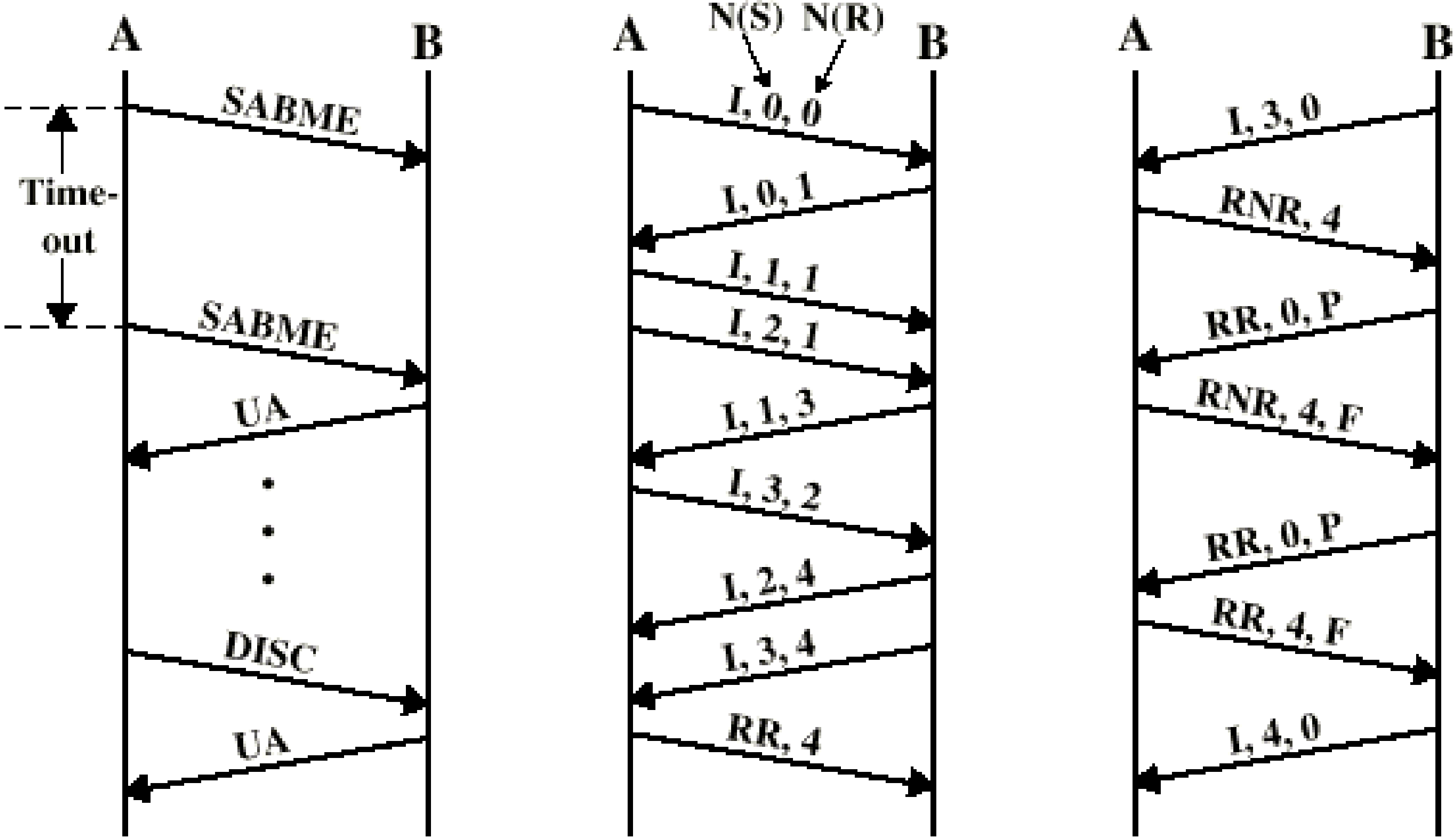
Data Transfer

- Both sides may begin to send user data in I-frames (Information Frame)
  - N(S): sequence number of outgoing I-frames
    - modulo 8 or 128, (3- or 7-bit)
  - N(R) acknowledgment for I-frames received
    - seq. number of I-frame expected next
- S-frames are also used for flow and error control
  - Receive ready (RR) frame acknowledges last I-frame received
    - Indicating next I-frame expected
    - Used when there is no reverse data
  - Receive not ready (RNR) acknowledges, but also asks peer to suspend transmission of I-frames
    - When ready, send RR to restart
  - REJ initiates go-back-N ARQ
    - Indicates last I-frame received has been rejected
    - Retransmission is requested beginning with N(R)
  - Selective reject (SREJ) requests retransmission of single frame

Disconnect

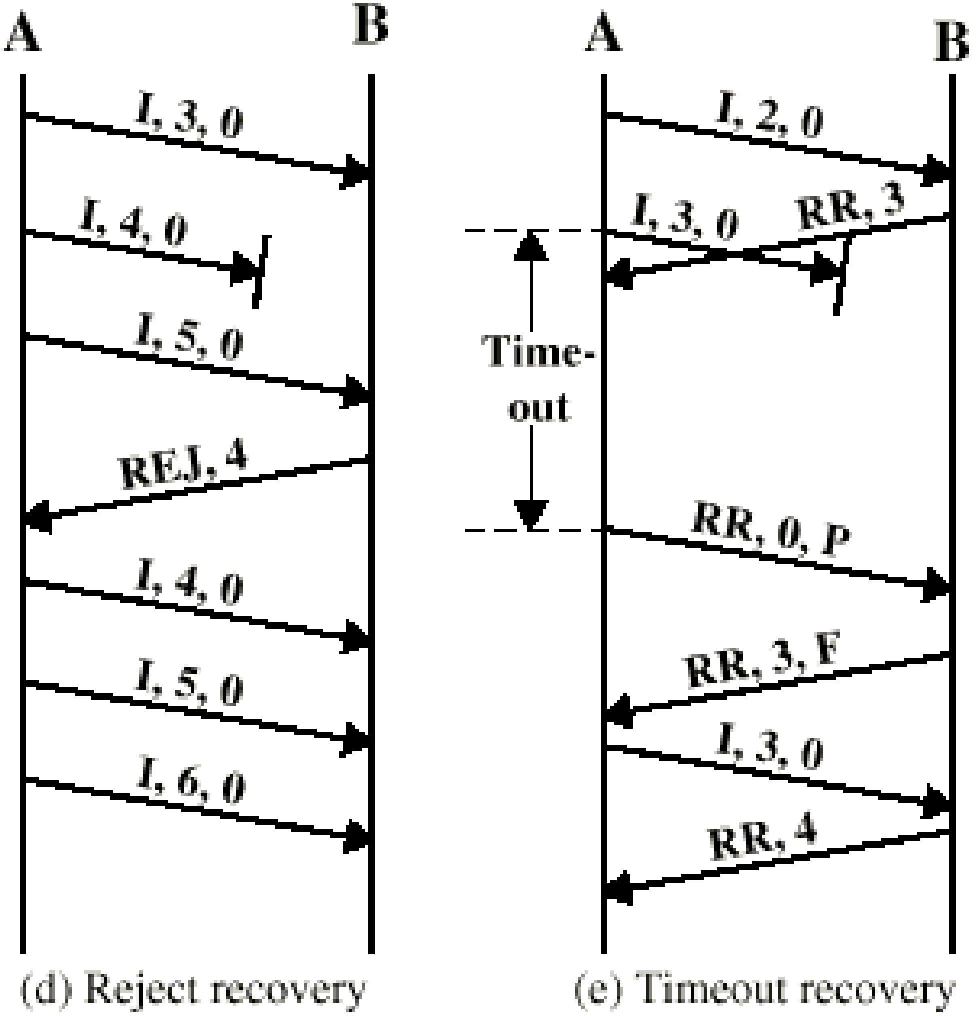
- Send disconnect (DISC) frame
- Remote entity must accept by replying with UA
  - Informs layer 3 user about the termination of connection
- These frames are unnumbered frames

Examples of Operation (1)



(a) Link setup and disconnect (b) Two-way data exchange (c) Busy condition

Examples of Operation (2)



(d) Reject recovery

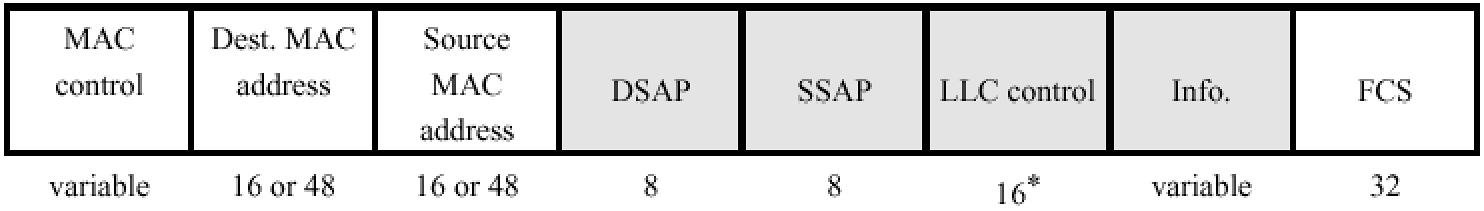
(e) Timeout recovery

Other DLC Protocols (LAPB,LAPD)

- Link Access Procedure, Balanced (LAPB)
  - Part of X.25 (ITU-T)
  - Subset of HDLC - ABM (Async. Balanced Mode)
  - Point to point link between user and packet switching network node
  - HDLC frame format
- Link Access Procedure, D-Channel (LAPD)
  - Part of ISDN (ITU-T)
  - ABM
  - Always 7-bit sequence numbers (no 3-bit)

Other DLC Protocols (LLC)

- Logical Link Control (LLC)
  - IEEE 802
  - For LANs (Local Area Networks)
  - Link control split between medium access control layer (MAC) and LLC (on top of MAC)
  - Different frame format
    - Two addresses needed (sender and receiver) – actually at MAC layer
    - Sender and receiver SAP addresses
    - Control field is same as HDLC (16-bit version for I and S frames; 8-bit for U frames)
  - No primary and secondary - all stations are peers
  - Error detection at MAC layer
    - 32 bit CRC



MAC control	Dest. MAC address	Source MAC address	DSAP	SSAP	LLC control	Info.	FCS
variable	16 or 48	16 or 48	8	8	16*	variable	32

Other DLC Protocols (LLC)

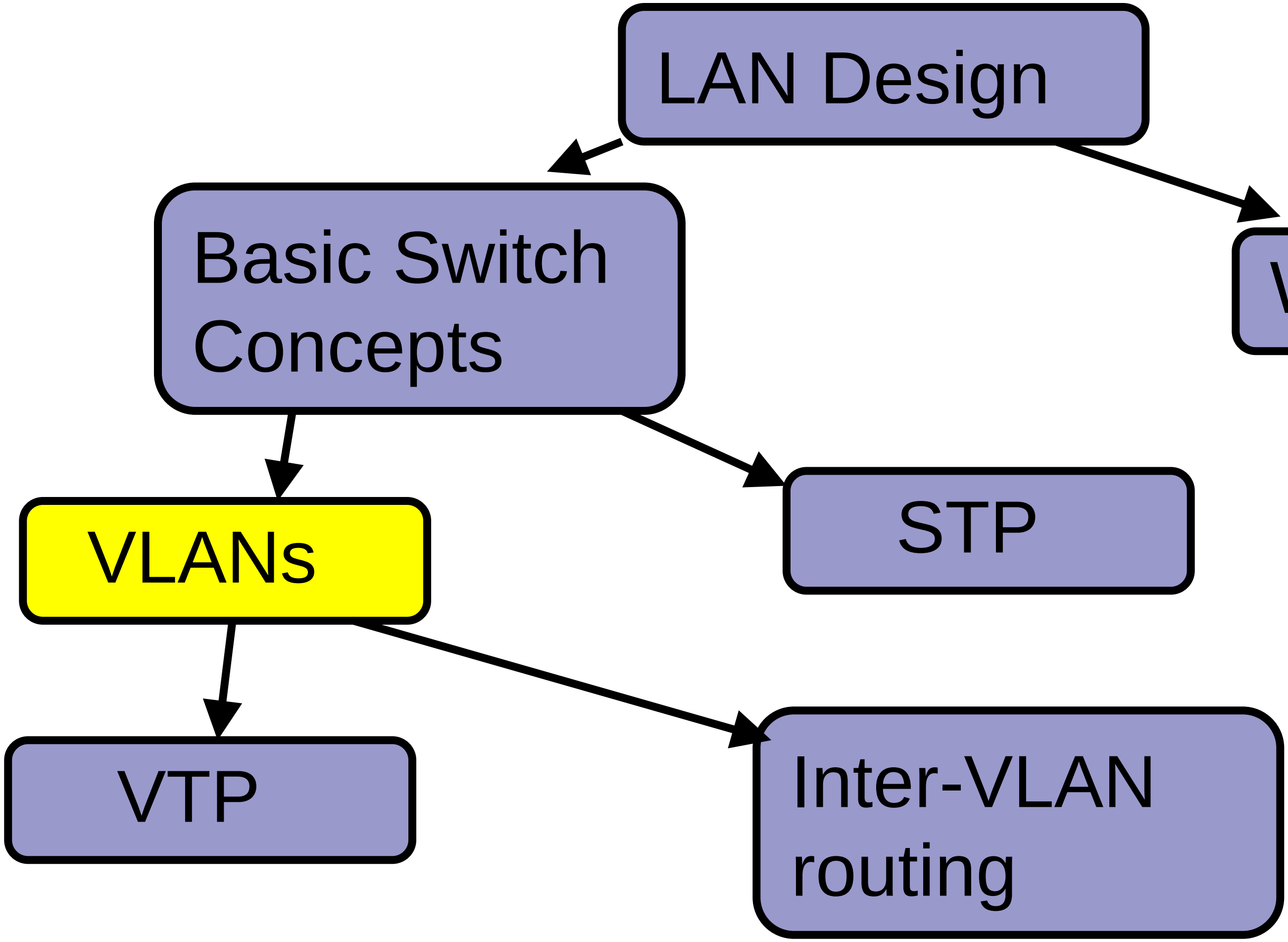
- LLC Services
  - 3 alternatives
  - Connection Mode Services
    - Similar to HDLC ABM
  - Unacknowledged connectionless services
    - no connection setup
    - No flow-control, no error control, no acks (thus not reliable)
    - good to be used with TCP/IP. Why?
  - Acknowledged Connectionless Service
    - No connection setup
    - reliable communication

VLANs

Topics

- The role of VLANs in a network
- Trunking VLANs
- Configure VLANs on switches
- Troubleshoot common VLAN problems

Semester 3

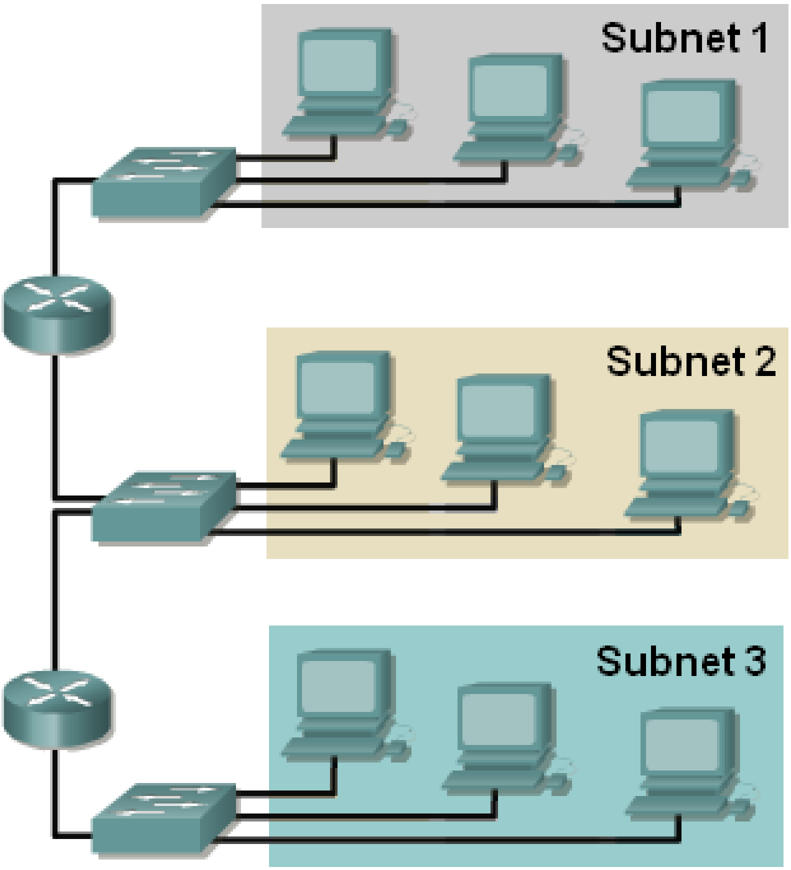

LAN Design
Basic Switch Concepts
Wireless
VLANs
STP
VTP
Inter-VLAN routing

Some requirements of LANs

- Need to split up broadcast domains to make good use of bandwidth
- People in the same department may need to be grouped together for access to servers
- Security: restrict access by certain users to some areas of the LAN
- Provide a way for different areas of the LAN to communicate with each other

Solution using routers

- Divide the LAN into subnets
- Use routers to link the subnets



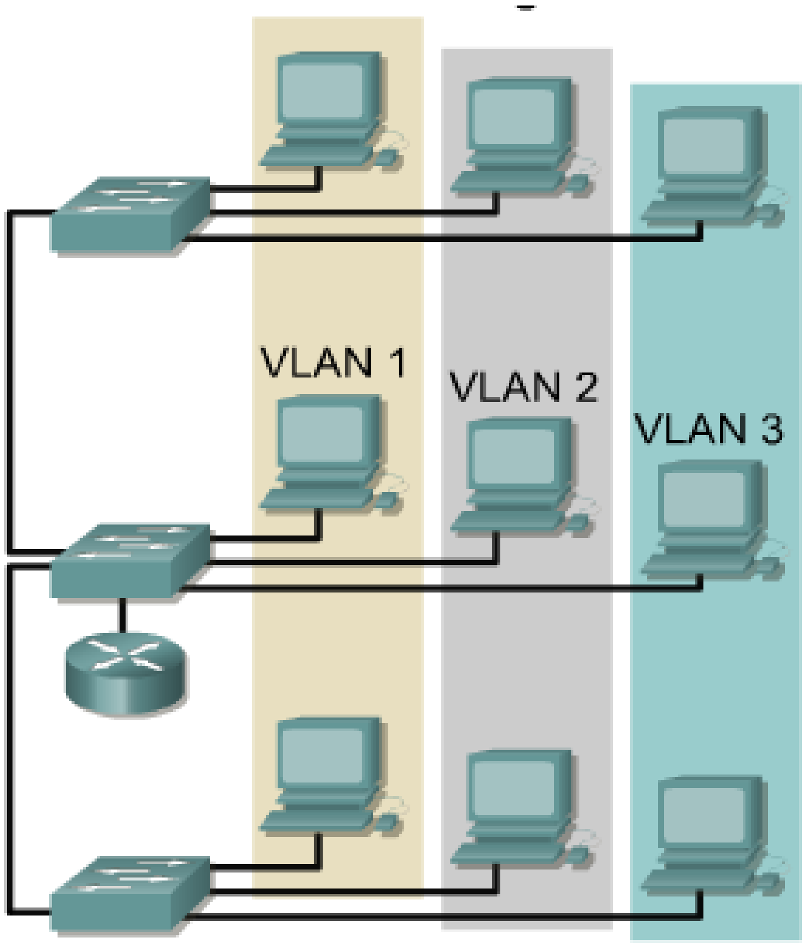
Solution using routers

BUT

- Routers are expensive
- Routers are slower than switches
- Subnets are restricted to limited physical areas
- Subnets are inflexible

Solution using VLANs

- VLAN membership can be by function and not by location
- VLANs managed by switches
- Router needed for communication between VLANs



VLANs

- All hosts in a VLAN have addresses in the same subnet. A VLAN is a subnet.
- Broadcasts are kept within the VLAN. A VLAN is a broadcast domain.
- The switch has a separate MAC address table for each VLAN. Traffic for each VLAN is kept separate from other VLANs.
- Layer 2 switches cannot route between VLANs.

VLAN numbers

- VLAN 1: default Ethernet LAN, all ports start in this VLAN.
- VLANs 1002 – 1005 automatically created for Token Ring and FDDI
- Numbers 2 to 1001 can be used for new VLANs
- Up to 255 VLANs on Catalyst 2960 switch
- Extended range 1006 – 4094 possible but fewer features

VLAN information

- VLAN information is stored in the VLAN database.
- vlan.dat in the flash memory of the switch.

Port based

- Each switch port intended for an end device is configured to belong to a VLAN.
- Any device connecting to that port belongs to the port’s VLAN.
- There are other ways of assigning VLANs but this is now the normal way.
- Ports that link switches can be configured to carry traffic for all VLANs (trunking)

Types of VLAN

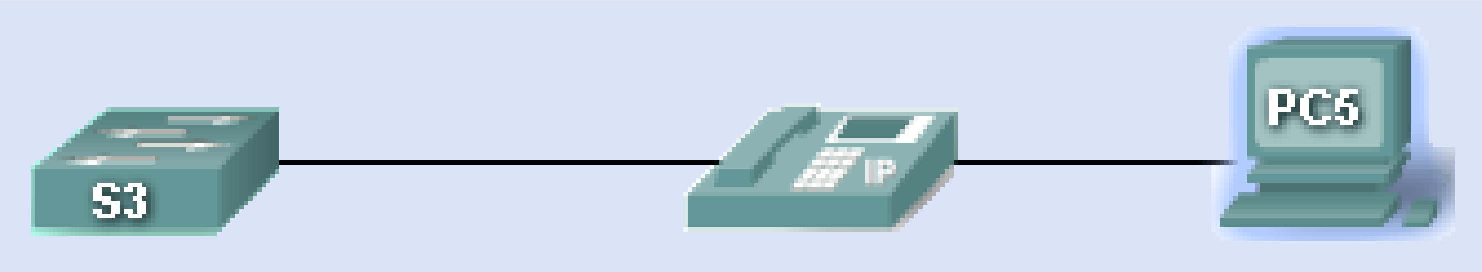
- Data or user VLAN
- Voice VLAN
- Management VLAN
- Native VLAN
- Default VLAN

Data VLAN

- Carry files, e-mails, shared application traffic, most user traffic.
- Separate VLAN for each group of users.

Voice VLAN

- Use with IP phone.
- Phone acts as a switch too.
- Voice traffic is tagged, given priority.
- Data not tagged, no priority.



Management VLAN

- Has the switch IP address.
- Used for telnet/SSH or web access for management purposes.
- Better not to use VLAN 1 for security reasons.

Native VLAN

- For backward compatibility with older systems.
- Relevant to trunk ports.
- Trunk ports carry traffic from multiple VLANs.
- VLAN is identified by a “tag” in the frame.
- Native VLAN does not have a tag.

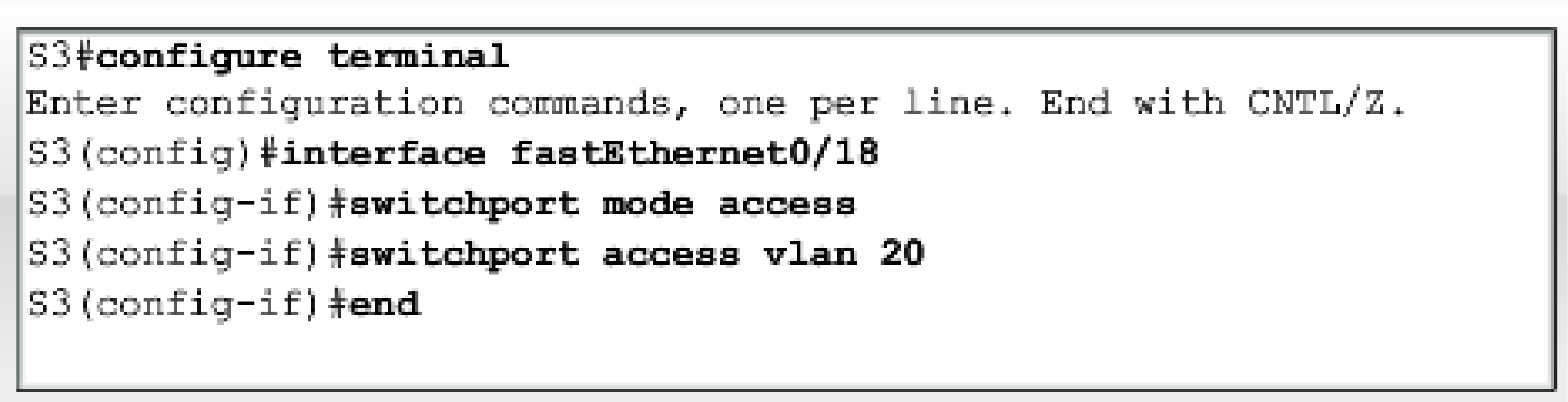
Default VLAN

- VLAN 1 on Cisco switches.
- Carries CDP and STP (spanning tree protocol) traffic.
- Initially all ports are in this VLAN.
- Do not use it for data, voice or management traffic for security reasons.

Static VLAN

- The normal type. Port configured to be on a VLAN. Connected device is on this VLAN.
- VLAN can be created using CLI command, given number and name.
- VLAN can be learned from another switch.
- If a port is put on a VLAN and the VLAN does not exist, then the VLAN is created.

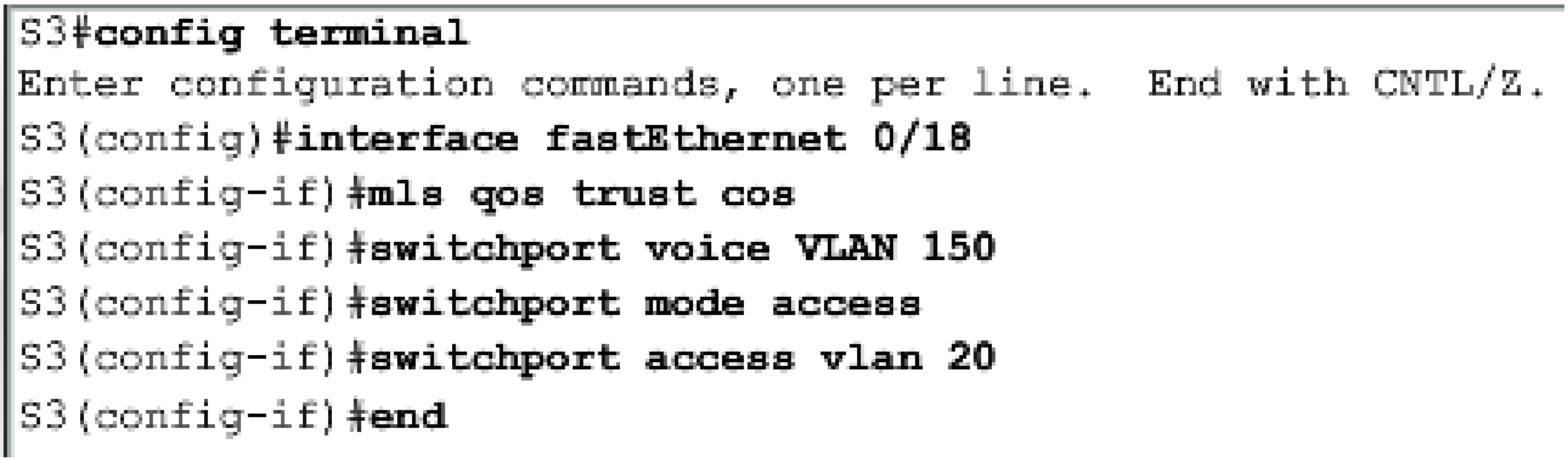
Static VLAN (Port-centric)



```
S3#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
S3(config)#interface fastEthernet0/18
S3(config-if)#switchport mode access
S3(config-if)#switchport access vlan 20
S3(config-if)#end
```

- If VLAN 20 did not exist before – then it does now.

Voice VLAN



```
S3#config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
S3(config)#interface fastEthernet 0/18
S3(config-if)#mls qos trust cos
S3(config-if)#switchport voice VLAN 150
S3(config-if)#switchport mode access
S3(config-if)#switchport access vlan 20
S3(config-if)#end
```

- Configured for voice VLAN and data VLAN.

Dynamic VLAN

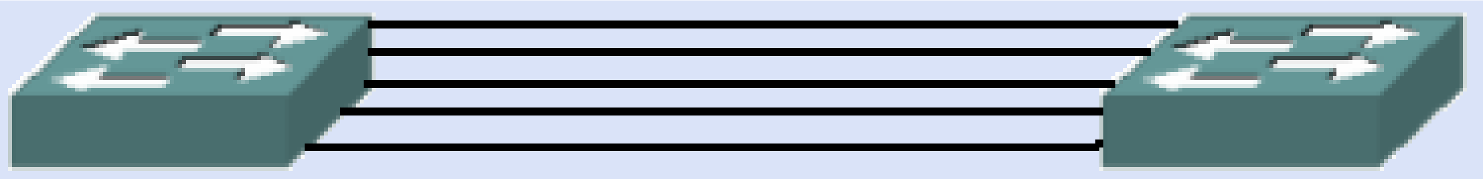
- Not widely used.
- Use a VLAN Membership Policy Server (VMPS).
- Assign a device to a VLAN based on its MAC address.
- Connect device, server assigns VLAN.
- Useful if you want to move devices around.

Traffic between VLANs

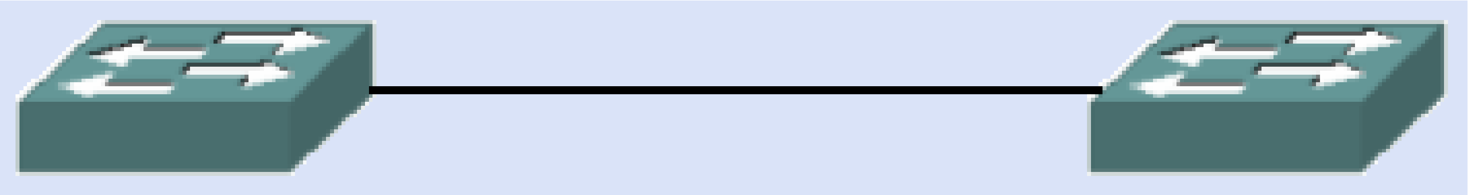
- Layer 2 switch keeps VLANs separate.
- Router can route between VLANs. It needs to provide a default gateway for each VLAN as VLANs are separate subnets.
- Layer 3 switch has a switch virtual interface (SVI) configured for each VLAN. These act like router interfaces to route between VLANs.

Trunking

- Both switches have the same 5 VLANs.
- Do you have a link for each VLAN?

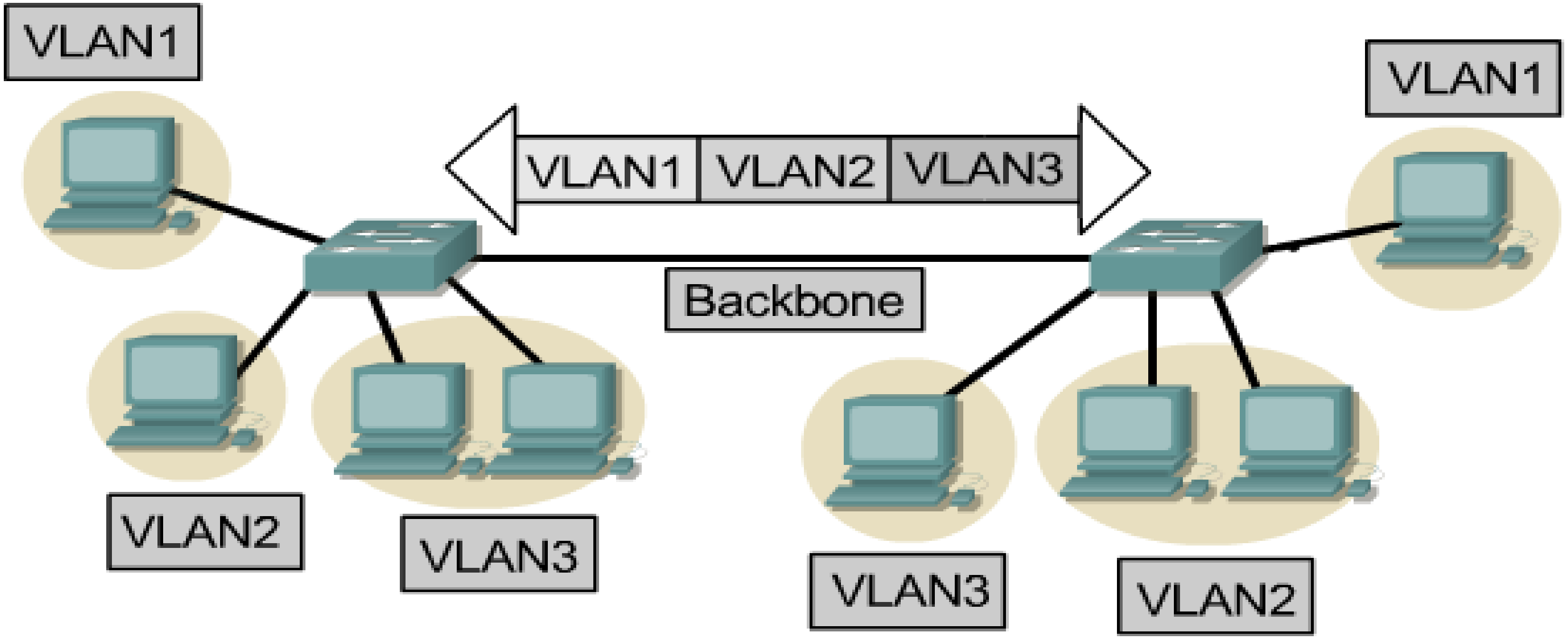


- More efficient for them to share a link.



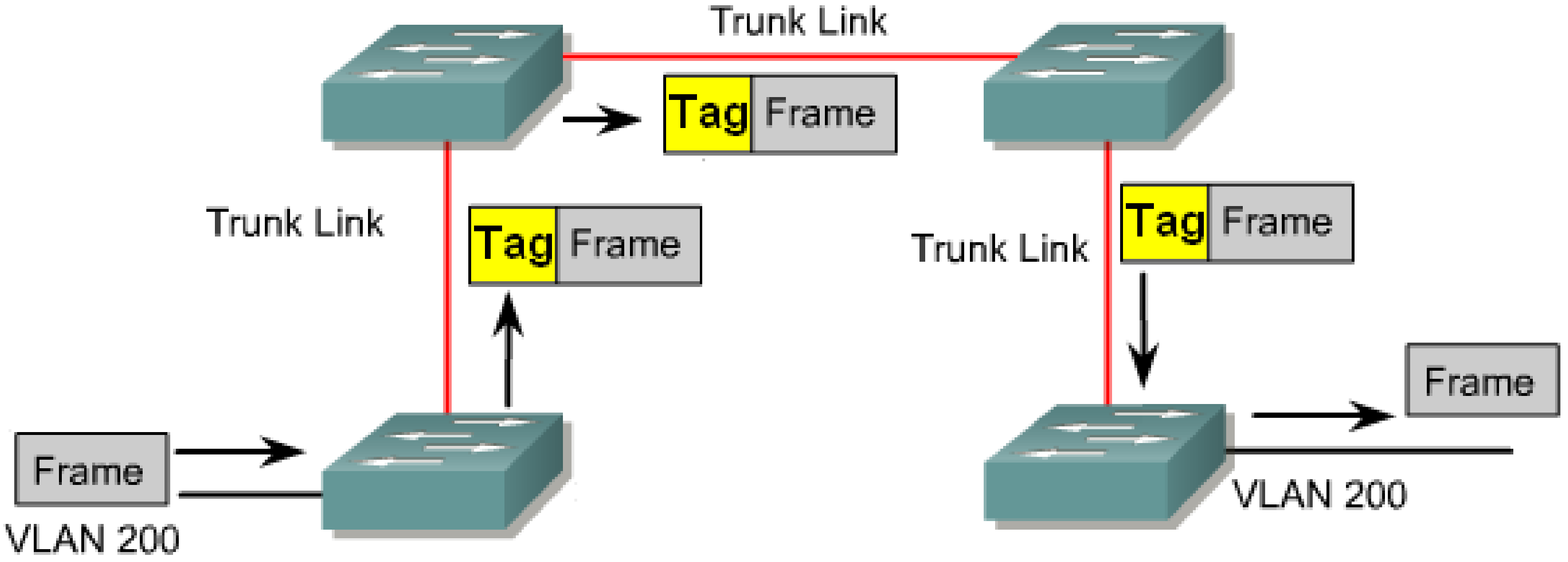
Trunking

- Traffic for all the VLANs travels between the switches on a shared trunk or backbone



Tag to identify VLAN

- Tag is added to the frame when it goes on to the trunk
- Tag is removed when it leaves the trunk



Frame tagging IEEE 802.1Q

Dest Add	Source Add	Type/Len	Data	FCS

Normal frame

Dest Add	Source Add	Tag	Type/Len	Data	FCS

Add 4-byte tag, recalculate FCS

Tag protocol ID 0x8100	Priority	CFI for token ring	VLAN ID 1 - 4096

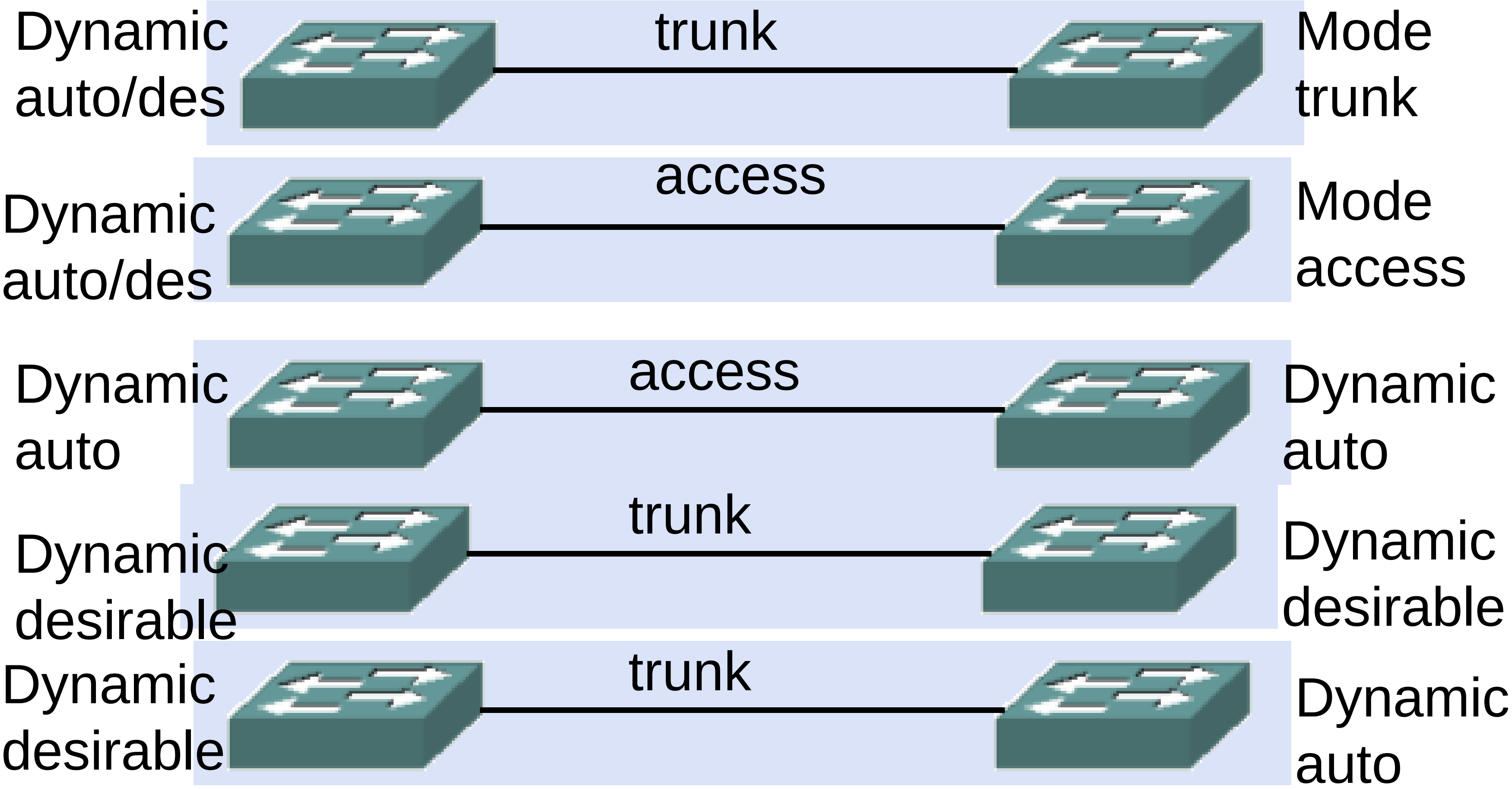
Native VLAN

- Untagged frames received on a trunk port are forwarded on to the native VLAN.
- Frame received from the native VLAN should be untagged.
- Switch will drop tagged frames received from the native VLAN. This can happen if non-Cisco devices are connected.

Configure trunk port

- Make a port into a trunk port and tell it which VLAN is native.
- SW1(config)#**int fa0/1**
- SW1(config-if)**switchport mode trunk**
- SW1(config-if)**switchport trunk native vlan 99**
- By default native VLAN is 1.

Dynamic trunking protocol



Dynamic auto/des — trunk — Mode trunk

Dynamic auto/des — access — Mode access

Dynamic auto — access — Dynamic auto

Dynamic desirable — trunk — Dynamic desirable

Dynamic desirable — trunk — Dynamic auto

Create a VLAN

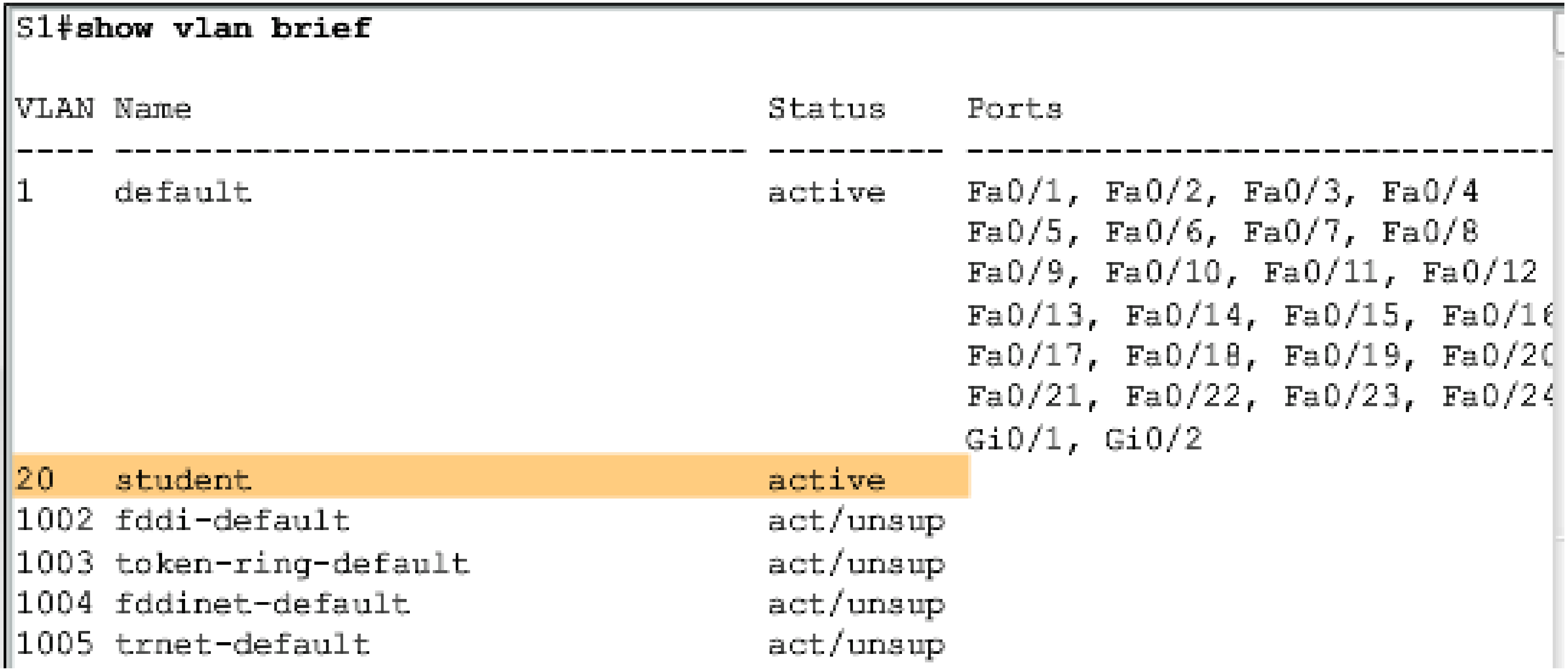
- SW1(config)#**vlan 20**
- SW1(config-vlan)#**name Finance**
- SW1(config-vlan)#**end**
- VLAN will be saved in VLAN database rather than running config.
- If you do not give it a name then it will be called vlan0020.

Assign port to VLAN

- SW1(config)#**int fa 0/14**
- SW1(config-if)#**switchport mode access**
- SW1(config-if)#**switchport access vlan 20**
- SW1(config-if)#**end**

show vlan brief

- List of VLANs with ports



```
S1#show vlan brief

VLAN Name                             Status    Ports
---- -------------------------------- --------- -----------------------------
1    default                          active    Fa0/1, Fa0/2, Fa0/3, Fa0/4
                                                Fa0/5, Fa0/6, Fa0/7, Fa0/8
                                                Fa0/9, Fa0/10, Fa0/11, Fa0/12
                                                Fa0/13, Fa0/14, Fa0/15, Fa0/1
                                                Fa0/17, Fa0/18, Fa0/19, Fa0/2
                                                Fa0/21, Fa0/22, Fa0/23, Fa0/2
                                                Gi0/1, Gi0/2
20   student                          active
1002 fddi-default                     act/unsup
1003 token-ring-default               act/unsup
1004 fddinet-default                  act/unsup
1005 trnet-default                    act/unsup
```

Show commands

- show vlan brief (list of VLANs and ports)
- show vlan summary
- show interfaces vlan (up/down, traffic etc)
- Show interfaces fa0/14 switchport (access mode, trunking)

Remove port from VLAN

- SW1(config)#**int fa 0/14**
- SW1(config-if)#**no switchport access vlan**
- SW1(config-if)#**end**
- The port goes back to VLAN 1.
- If you assign a port to a new VLAN, it is automatically removed from its existing VLAN.

Delete a VLAN

- SW1(config)#**no vlan 20**
- SW1(config)#**end**
- VLAN 20 is deleted.
- Any ports still on VLAN 20 will be inactive – not on any VLAN. They need to be reassigned.

Delete VLAN database

- Erasing the startup configuration does not get rid of VLANs because they are saved in a separate file.
- SW1#**delete flash:vlan.dat**
- Switch goes back to the default with all ports in VLAN 1.
- You cannot delete VLAN 1.

UNIT-3

The Network Layer

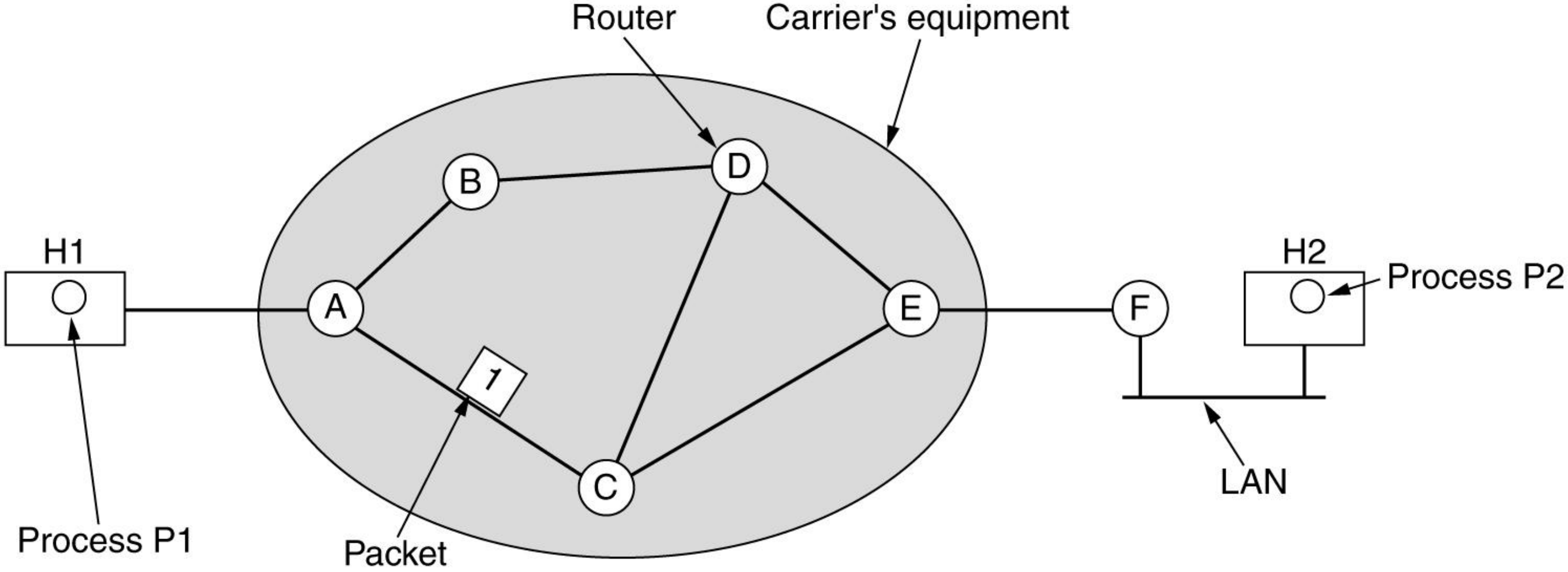
Design Issues & Routing Algorithms

Network Layer Design Isues

- Store-and-Forward Packet Switching
- Services Provided to the Transport Layer
- Implementation of Connectionless Service
- Implementation of Connection-Oriented Service
- Comparison of Virtual-Circuit and Datagram Subnets

Store-and-Forward Packet Switching

The environment of the network laver

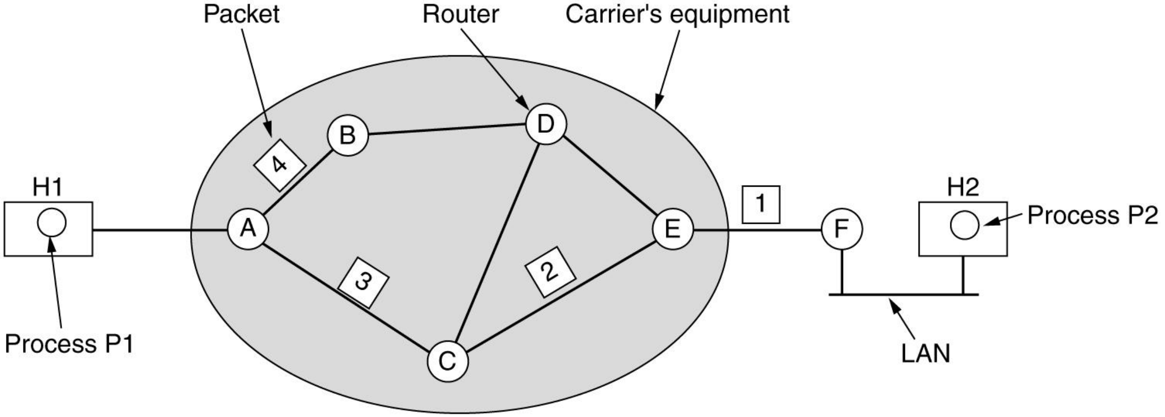


Services Provided to the Transport Layer

1. The services should be independent of the router technology
2. The transport layer should be shielded from the number, type and topology of the routers present
3. The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs

Implementation of Connectionless Service

Rou



A's table

initially

Dest.	Line
A	–
B	B
C	C
D	B
E	C
F	C

later

A	–
B	B
C	C
D	B
E	B
F	B

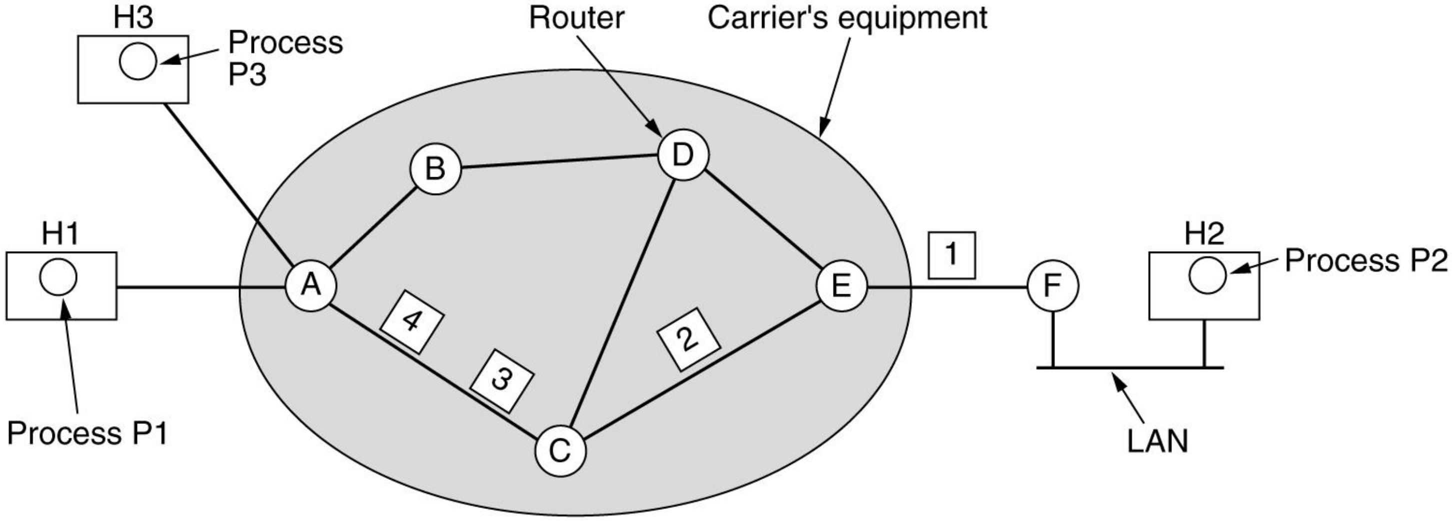
C's table

A	A
B	A
C	–
D	D
E	E
F	E

E's table

A	C
B	D
C	C
D	D
E	–
F	F

Implementation of Connection-Oriented Service



A's table

In		Out	
H1	1	C	1
H3	1	C	2

C's table

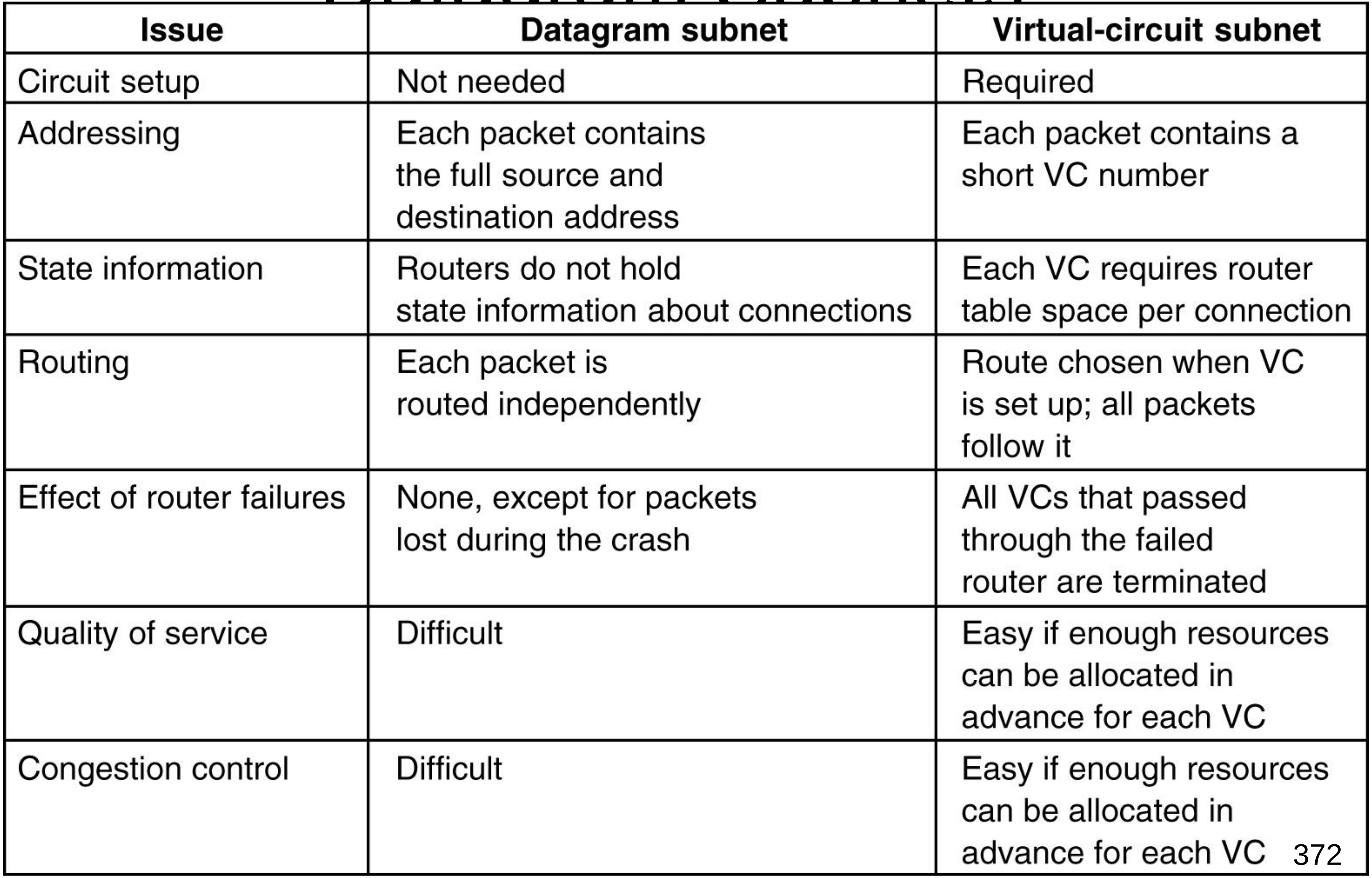
A	1	E	1
A	2	E	2

E's table

C	1	F	1
C	2	F	2

Routing within a virtual-circuit subnet.

Comparison of Virtual-Circuit and Datagram Subnets



Issue	Datagram subnet	Virtual-circuit subnet
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

Routing Algorithms

- The Optimality Principle
- Shortest Path Routing
- Flooding
- Distance Vector Routing
- Link State Routing
- Hierarchical Routing
- Broadcast Routing
- Multicast Routing
- Routing for Mobile Hosts
- Routing in Ad Hoc Networks

Desirable Properties (Elaborate)

Correctness

Simplicity

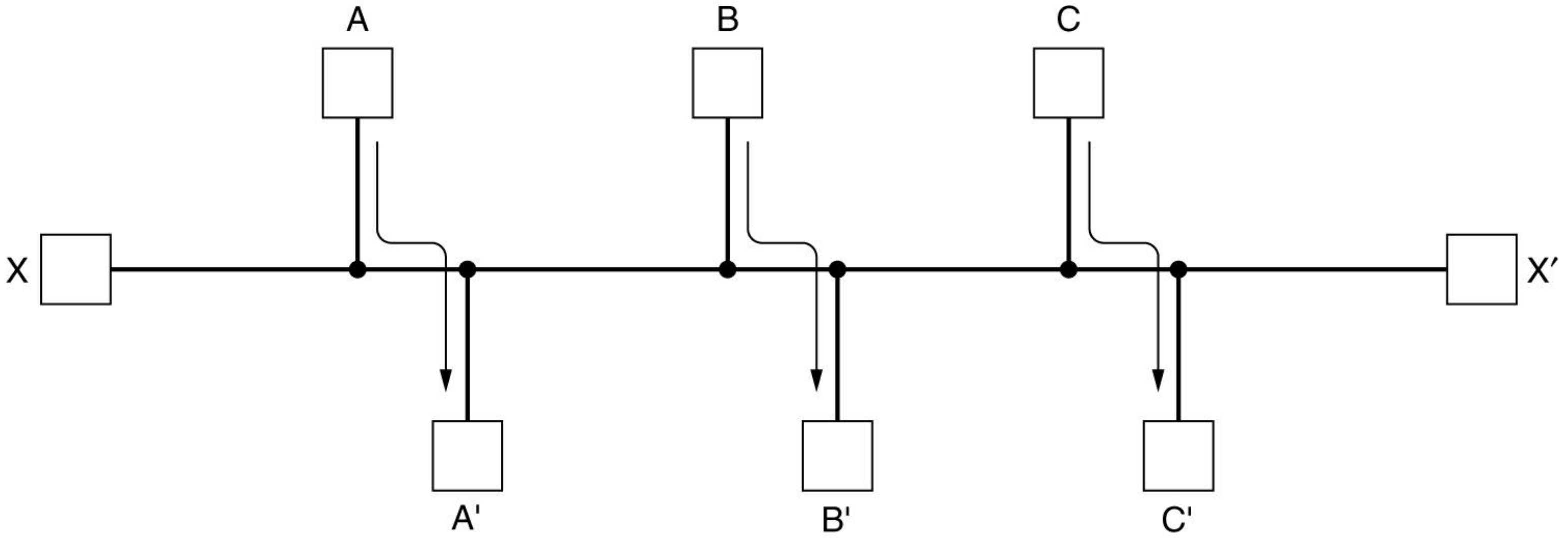
Robustness – System will be in place for years with small failures

Stability – Fast convergence

Fairness,

Efficiency.

Routing Algorithms (2)



A – A', B – B', C – C', can fill the channel, then X-X' doesn't get a chance

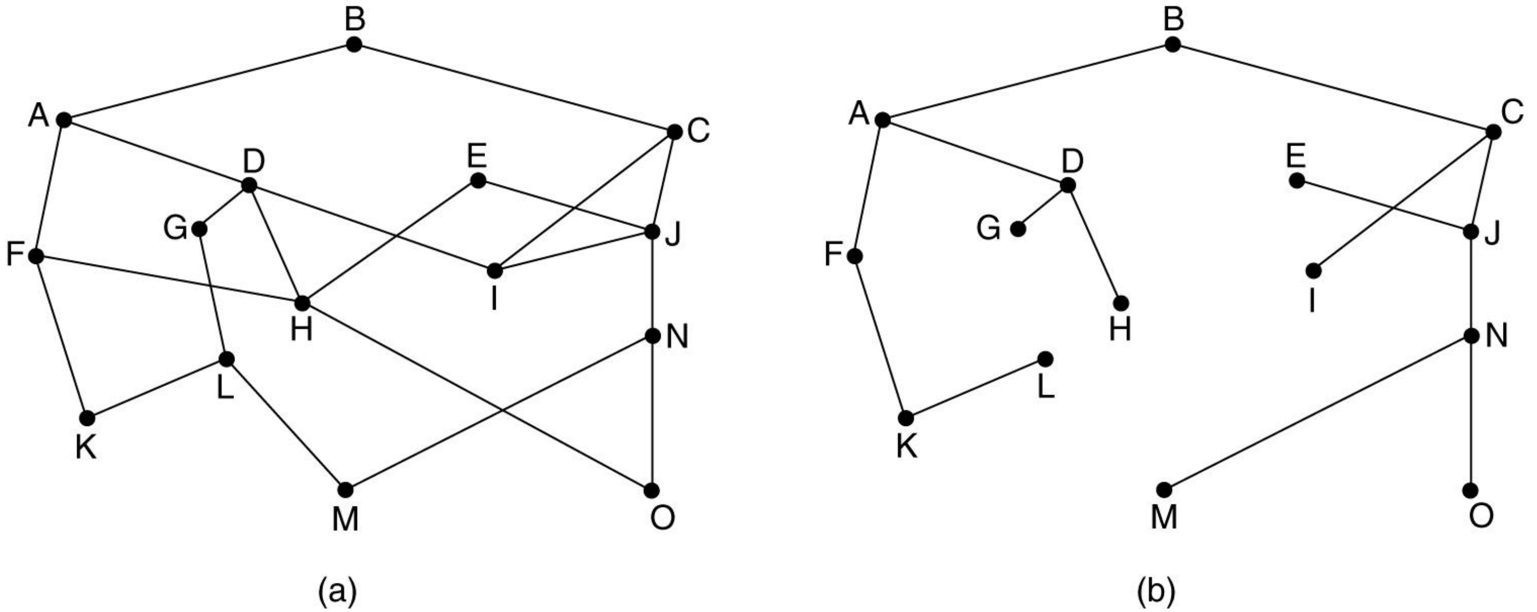
Conflict between fairness and optimality.

Minimizing the mean packet delay is an obvious candidate to send traffic through the network

Elaborate

- Adaptive/Non-adaptive routing

The Optimality Principle



Optimality Principle – If router J is on the optimal path from router I to router K then the optimal path from J to K also falls along the same route.

(a) A subnet. (b) A sink tree for router B.

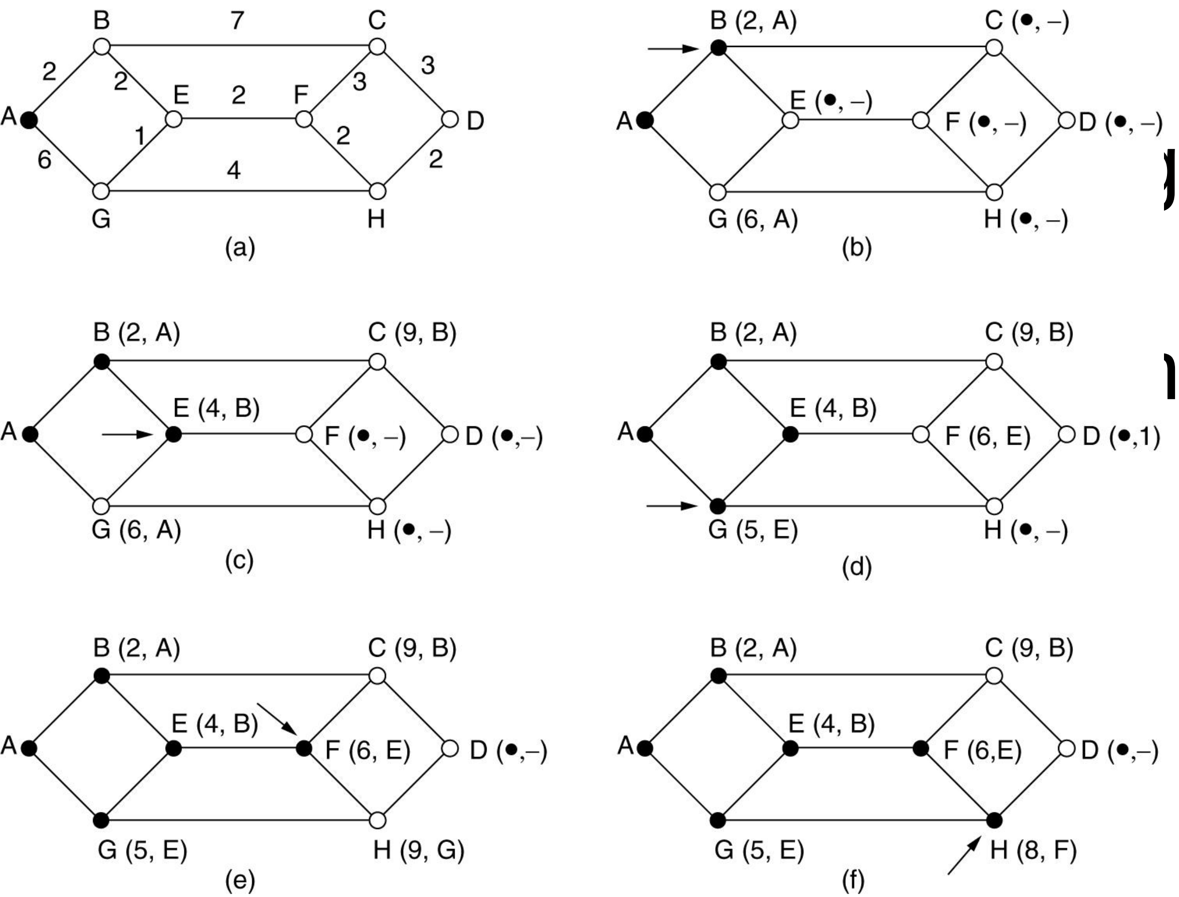
Shortest Path Routing

The

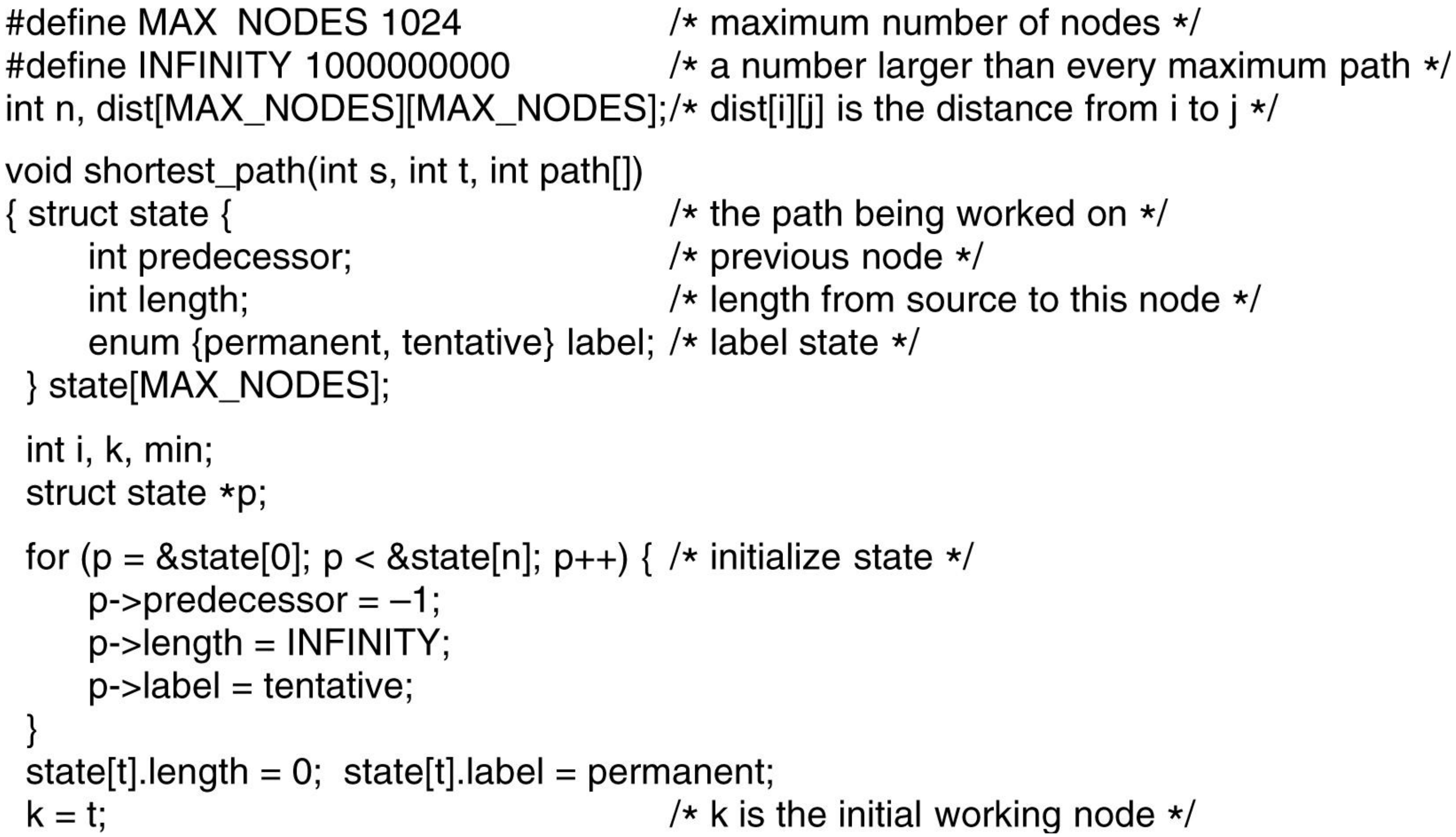
the

Th

node.



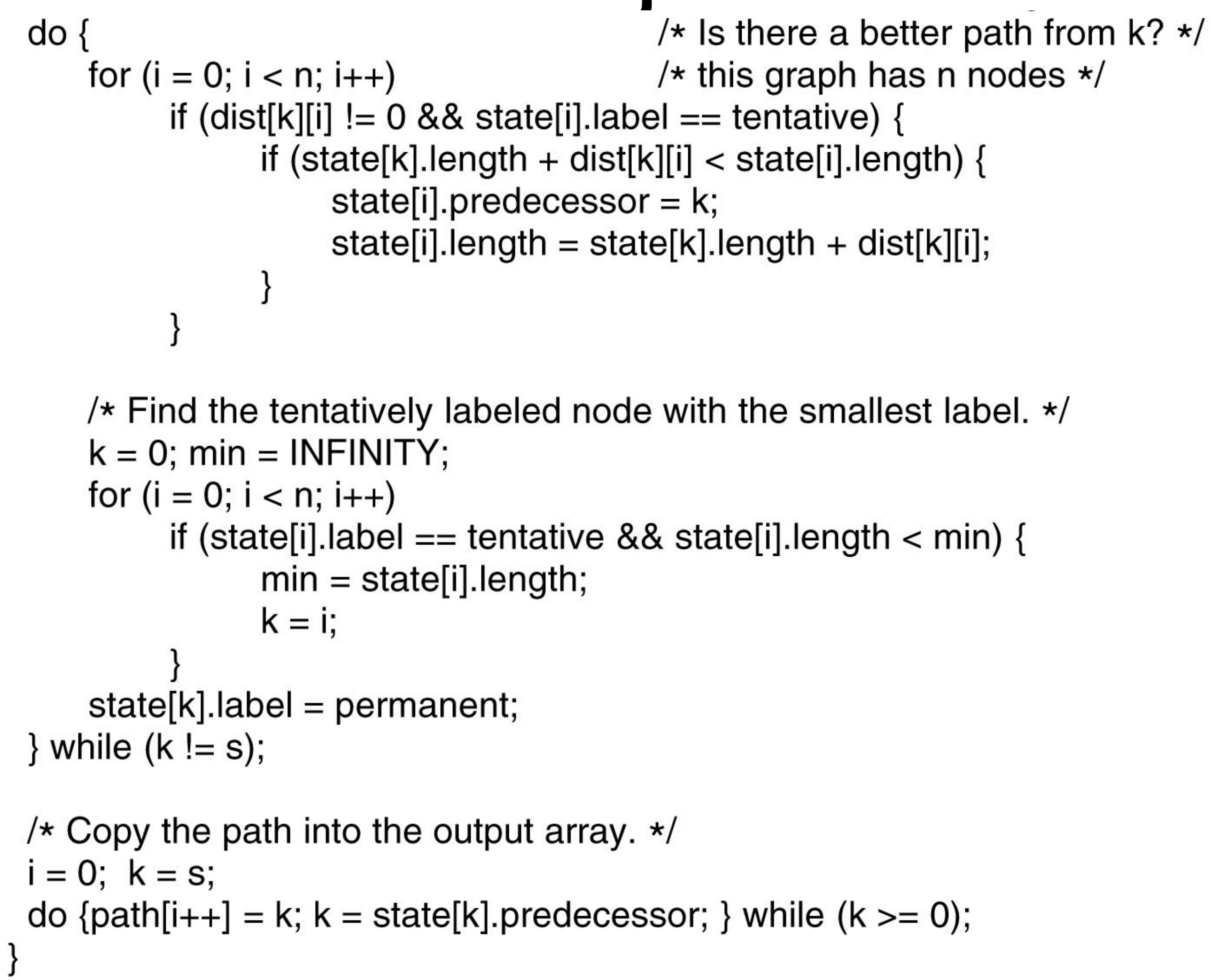
Dijkstra



```
#define MAX  NODES 1024                  /* maximum number of nodes */
#define INFINITY 1000000000              /* a number larger than every maximum path */
int n, dist[MAX_NODES][MAX_NODES];/* dist[i][j] is the distance from i to j */

void shortest_path(int s, int t, int path[])
{ struct state {                          /* the path being worked on */
      int predecessor;                    /* previous node */
      int length;                         /* length from source to this node */
      enum {permanent, tentative} label; /* label state */
  } state[MAX_NODES];

  int i, k, min;
  struct state *p;

  for (p = &state[0]; p < &state[n]; p++) { /* initialize state */
      p->predecessor = –1;
      p->length = INFINITY;
      p->label = tentative;
  }
  state[t].length = 0;  state[t].label = permanent;
  k = t;                                  /* k is the initial working node */
```

Dijkstra's algorithm to compute the shortest path through a graph

Dijkstra



```
    do {                                          /* Is there a better path from k? */
        for (i = 0; i < n; i++)                   /* this graph has n nodes */
             if (dist[k][i] != 0 && state[i].label == tentative) {
                   if (state[k].length + dist[k][i] < state[i].length) {
                         state[i].predecessor = k;
                         state[i].length = state[k].length + dist[k][i];
                   }
             }

        /* Find the tentatively labeled node with the smallest label. */
        k = 0; min = INFINITY;
        for (i = 0; i < n; i++)
             if (state[i].label == tentative && state[i].length < min) {
                   min = state[i].length;
                   k = i;
             }
        state[k].label = permanent;
    } while (k != s);

    /* Copy the path into the output array. */
    i = 0;  k = s;
    do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

Dijkstra's algorithm to compute the shortest path through a graph.

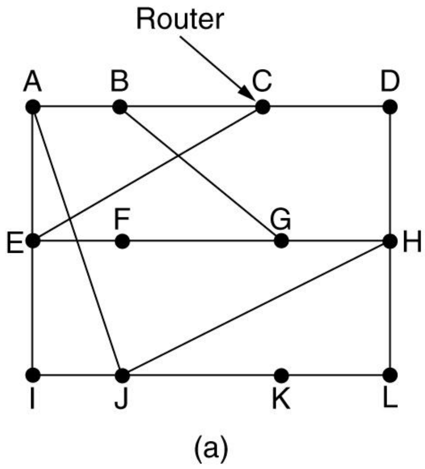
Flooding

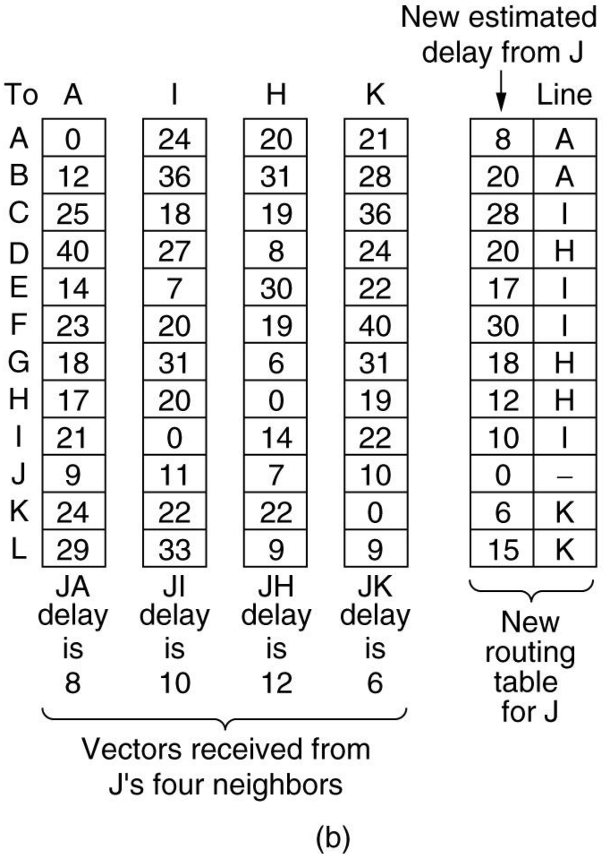
Robust but costly.

TTL and keep track…

- •Used in military application
- •Wireless Networks
- •Distributed Database
- •Metrics against which other routing algorithms are compared.

Distance Vector Routing





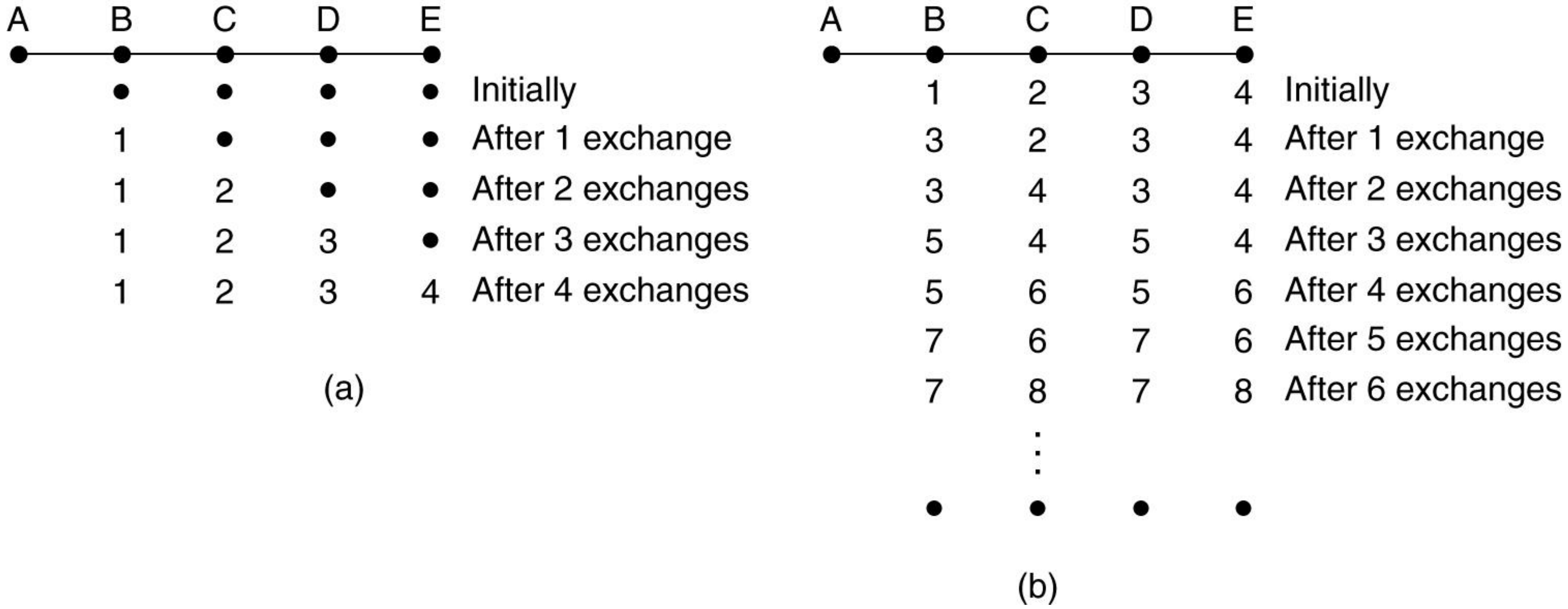
To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

Vectors received from J's four neighbors

(b)

(a) A subnet. (b) Input from A, I, H, K, and the new routing table for J

Distance Vector Routing (2)



A	B	C	D	E	
	•	•	•	•	Initially
	1	•	•	•	After 1 exchange
	1	2	•	•	After 2 exchanges
	1	2	3	•	After 3 exchanges
	1	2	3	4	After 4 exchanges

(a)

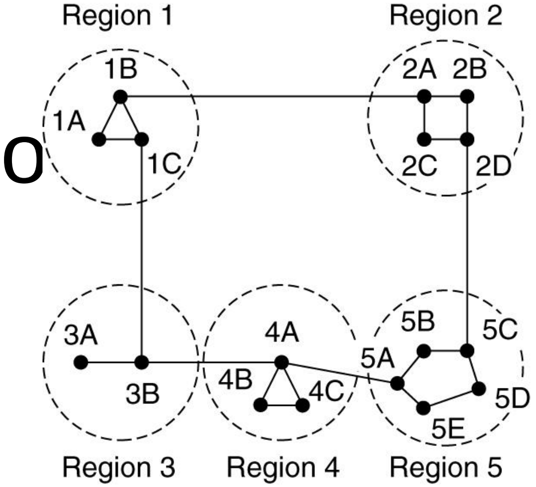
A	B	C	D	E	
	1	2	3	4	Initially
	3	2	3	4	After 1 exchange
	3	4	3	4	After 2 exchanges
	5	4	5	4	After 3 exchanges
	5	6	5	6	After 4 exchanges
	7	6	7	6	After 5 exchanges
	7	8	7	8	After 6 exchanges
		⋮			
	•	•	•	•	

(b)

The count-to-infinity problem.

Hierarchical Routing

Pro



(a)

Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

ed

Hierarchical Routing (2)

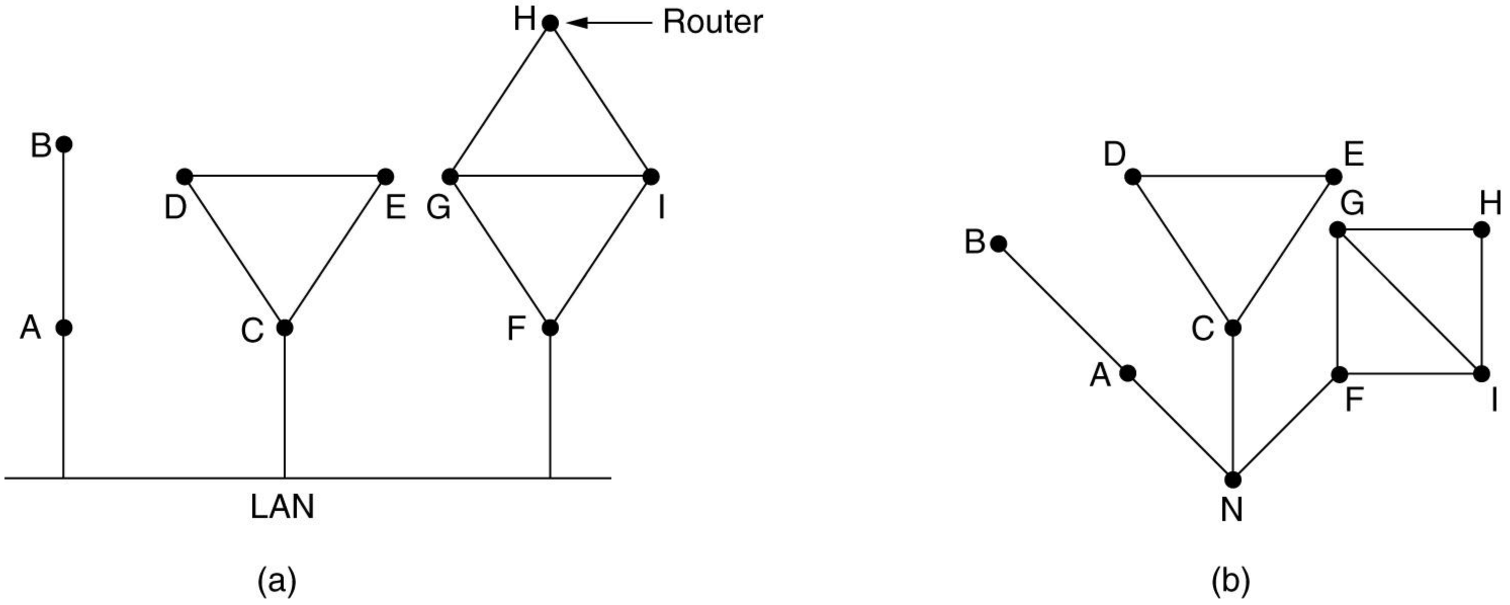
- How many levels of hierarchy?
- 720 routers.
- 720 routers in 24 regions.
- Three levels of hierarchy – 8 clusters each containing 9 regions of 10 clusters.

Link State Routing

Each router must do the following:

1. Discover its neighbors, learn their network address.
2. Measure the delay or cost to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to all other routers.
5. Compute the shortest path to every other router.

Learning about the Neighbors



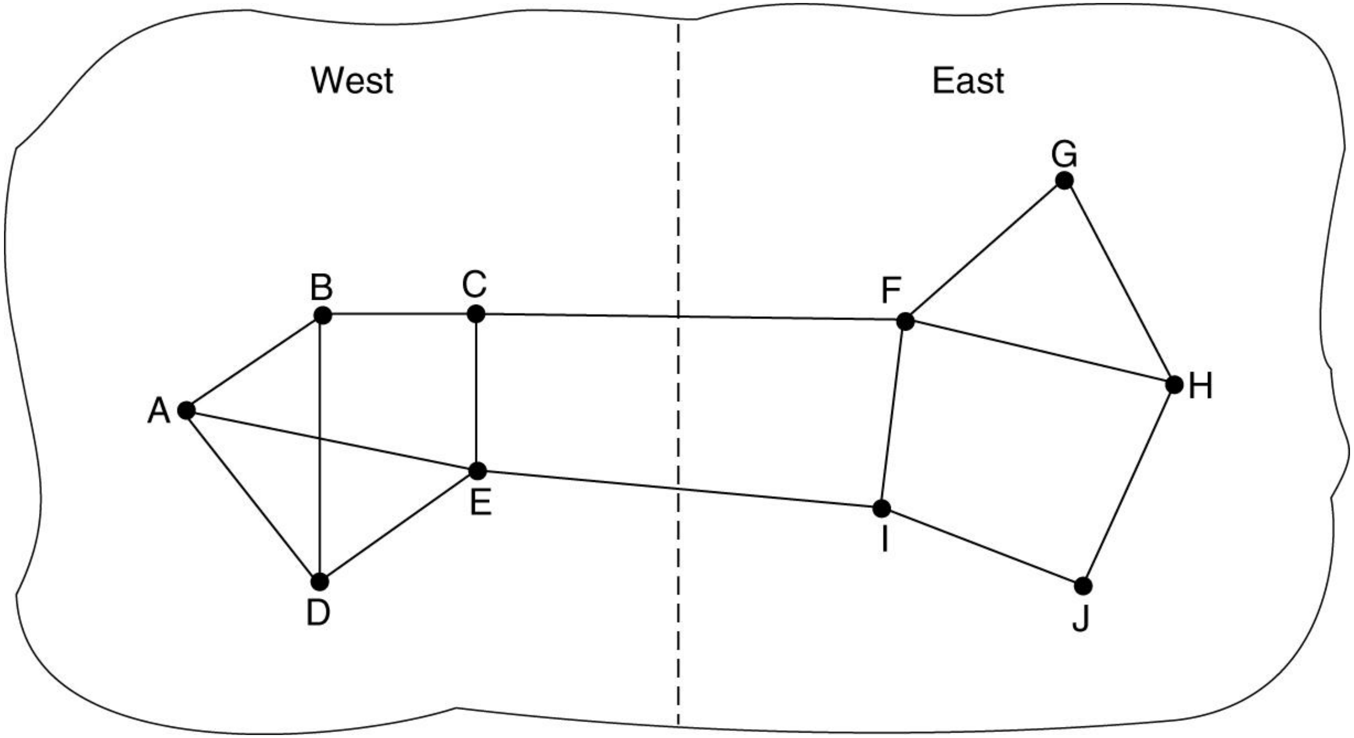
(a) Nine routers and a LAN. (b) A graph model of (a). All routers

Setting Link Cost

- Bandwidth
- Delay – measured by sending special ECHO

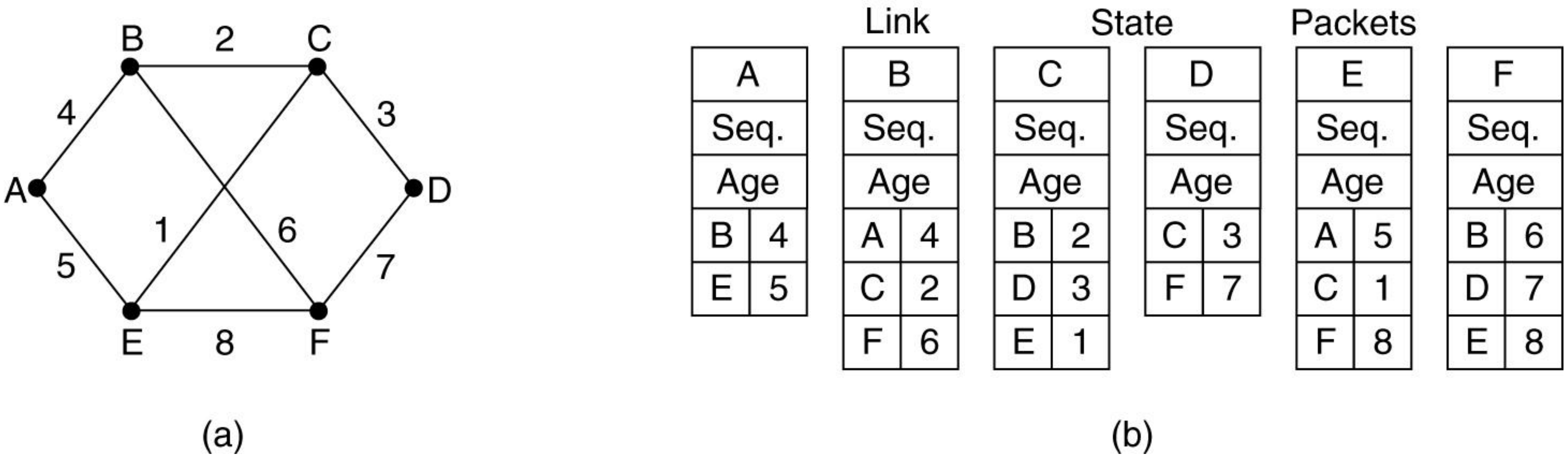
- Geographically spread out links

Measuring Line Cost



- A subnet in which the East and West parts are connected by 2 lines.
- Including queuing delay may lead to a lot of

Building Link State Packets



Link State Packets

A	
Seq.	
Age	
B	4
E	5

B	
Seq.	
Age	
A	4
C	2
F	6

C	
Seq.	
Age	
B	2
D	3
E	1

D	
Seq.	
Age	
C	3
F	7

E	
Seq.	
Age	
A	5
C	1
F	8

F	
Seq.	
Age	
B	6
D	7
E	8

(a) A subnet. (b) The link state packets for this subnet.

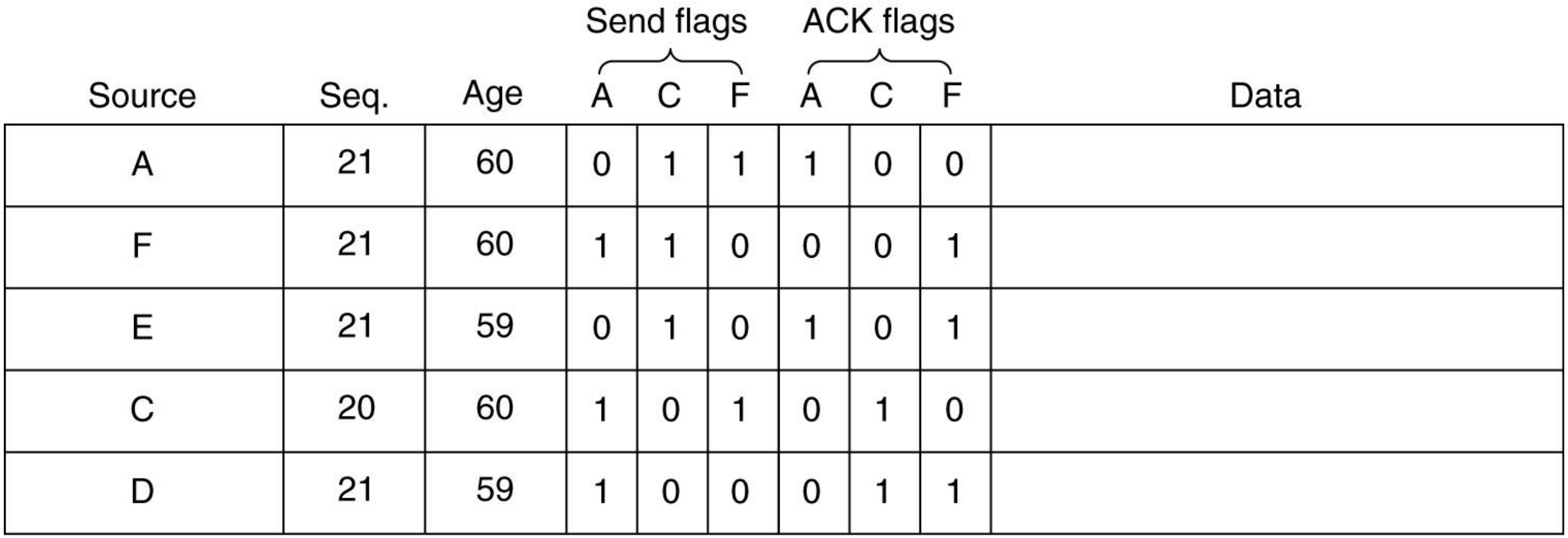
Few Problems

Algorithm – Sequence number less means obsolete

- If sequence numbers wrap around, confusion will reign
- Router crashes, sequence number is lost
- Sequence number gets corrupted

•Aging and then dropping the packet.

Distributing the Link State Packets



			Send flags			ACK flags			
Source	Seq.	Age	A	C	F	A	C	F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

- The packet buffer for router B in the previous slide (Fig. 5-13).
- E has arrived twice.

OSPF (Open Shortest Path First) IS-IS (intermediate System-Intermediate System)

- Refreshed every 60 seconds.
- Hardware problem, router getting corrupt, etc.

Broadcast Routing

Multidimensional Routing

•Each packet contains a list of destinations.

•On arrival of a packet, router checks the set of destinations, and sends copies of packet along outgoing links to those destinations.

Flooding

•Flood with a sequence number per source.

Spanning Tree

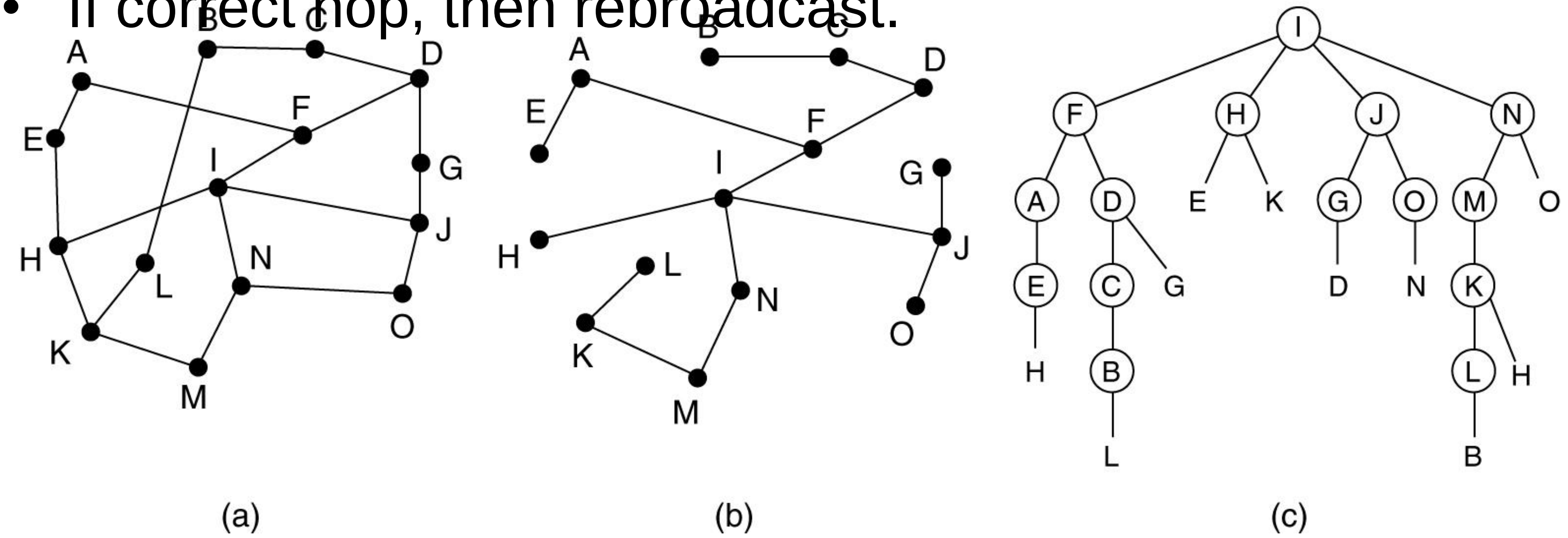
•Build spanning tree (such as, a sink tree).

•Forward packet along all links of spanning tree except the one from which packet is received.

Reverse Path Forwarding

Broadcast – Reverse Path Forwarding

- Broadcast.
- Check if the packet has arrived following the correct hop or not.
- If correct hop, then rebroadcast.

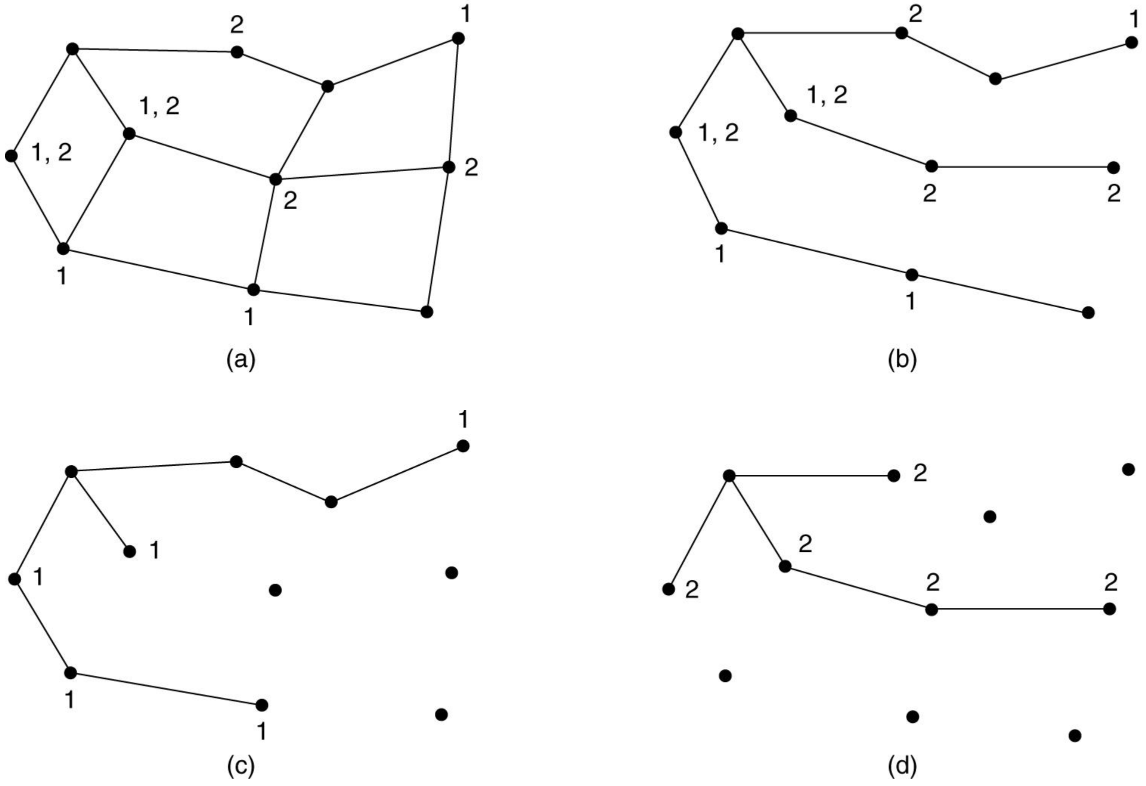


Reverse path forwarding.

(a) A subnet. (b) a Sink tree.

(c) The tree built by reverse path forwarding.

Multicast Routing



(a) A network. (b) A spanning tree for the leftmost router.
(c) A multicast tree for group 1. (d) A multicast tree for group 2.

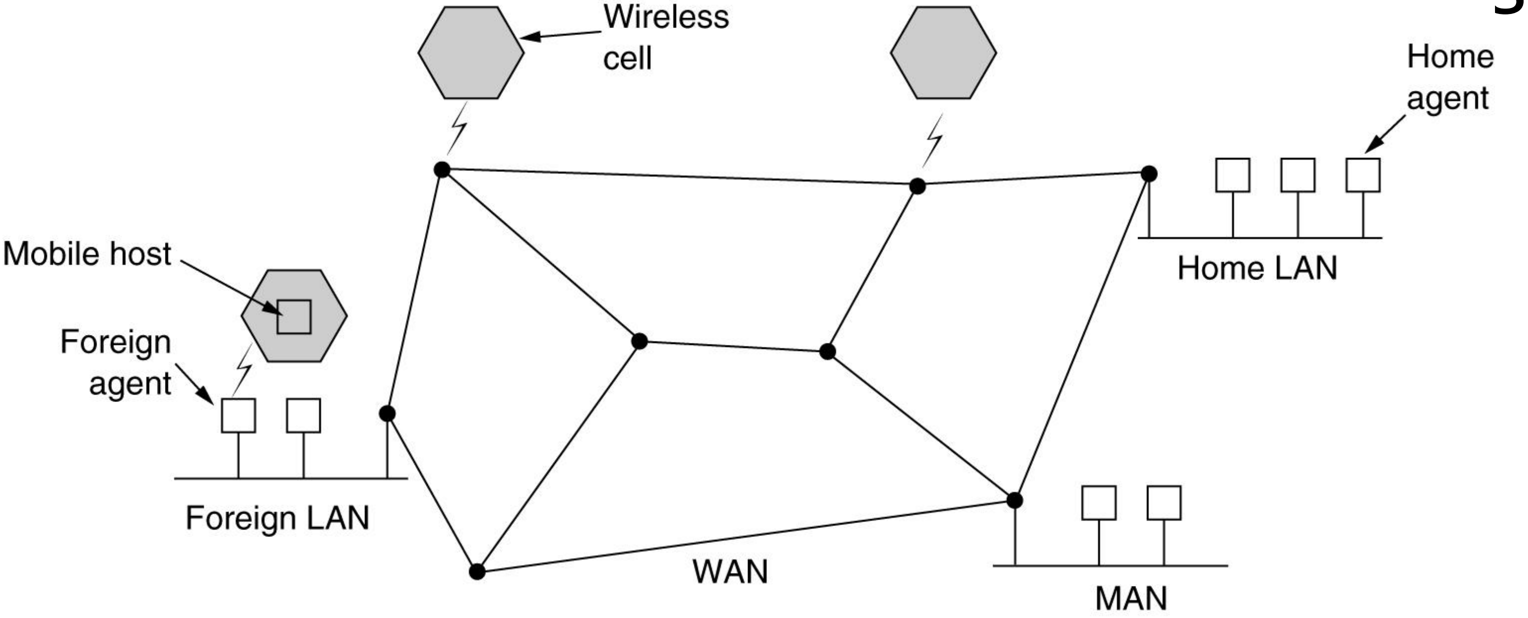
Typically done in Multi-state routing

Anycast Routing

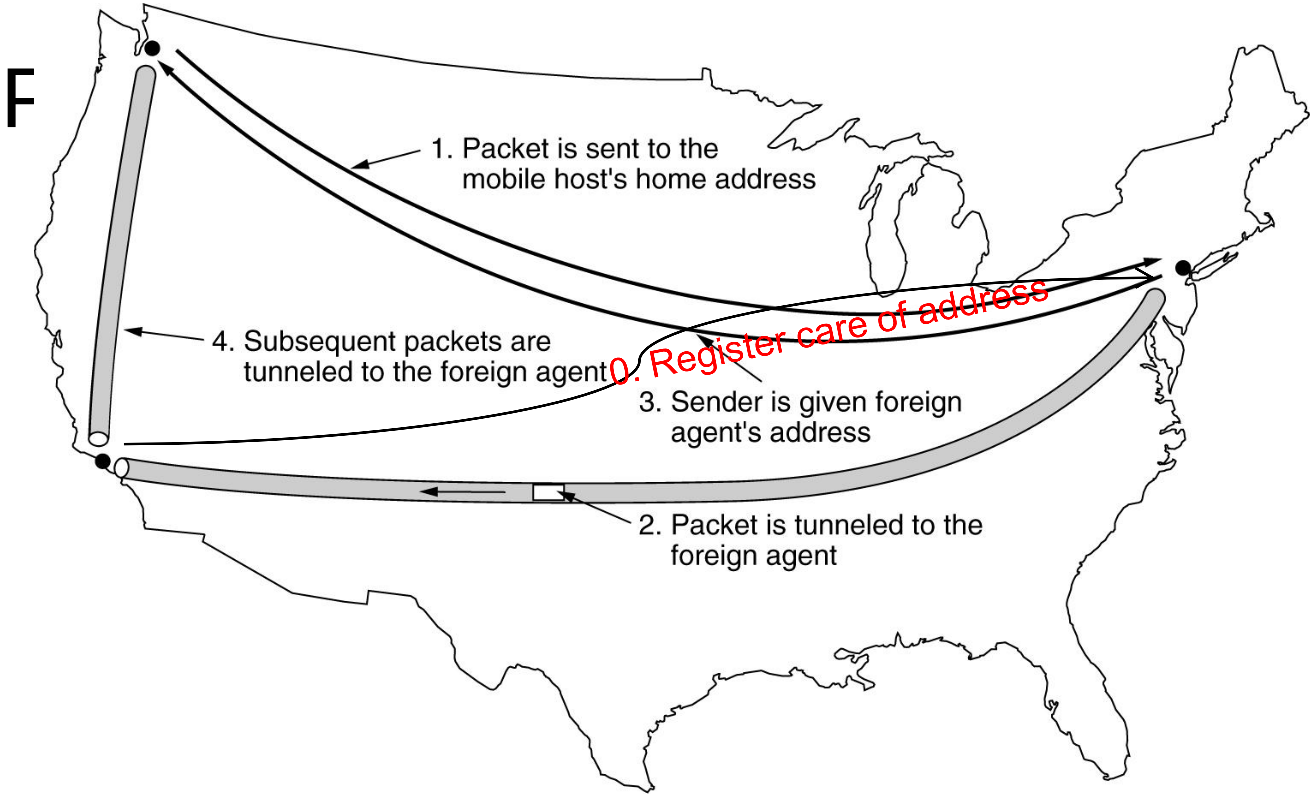
- Reaching any one of the servers in the group
- DNS server

Routing for Mobile Hosts

A WAN to which LANs, MANs, and wireless



Routing for Mobile Hosts (2)

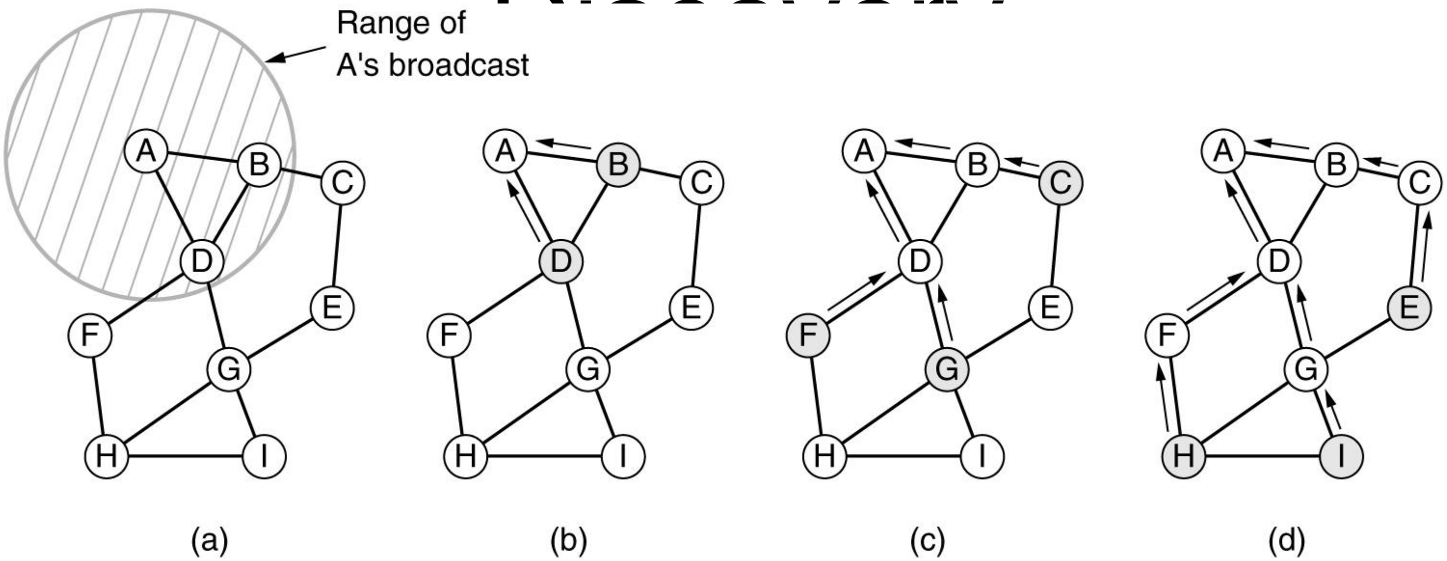

F
1. Packet is sent to the mobile host's home address
0. Register care of address
4. Subsequent packets are tunneled to the foreign agent
3. Sender is given foreign agent's address
2. Packet is tunneled to the foreign agent

Routing in Ad Hoc Networks

Possibilities when the routers are mobile:

1. Military vehicles on battlefield.
   - No infrastructure.
2. A fleet of ships at sea.
   - All moving all the time
3. Emergency works at earthquake .
   - The infrastructure destroyed.
4. A gathering of people with notebook computers.
   - In an area lacking 802.11.

Ad Hoc Networks: Route Discovery

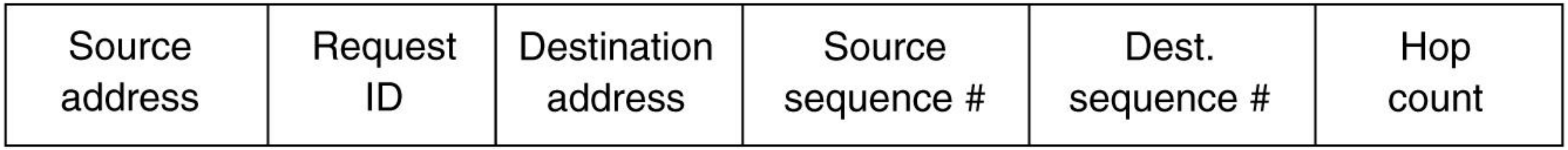


(a) Range of A's broadcast.

(b) After B and D have received A's broadcast.

(c) After C, F, and G have received A's broadcast.

Route Discovery (2)



Source address	Request ID	Destination address	Source sequence #	Dest. sequence #	Hop count

Format of a ROUTE REQUEST packet.

Route Discovery (3)

- The (Source Address, Request ID) pair is looked up in a local history table

- Receiver looks up the destination in its route table. If a fresh route is known, then a ROUTE REPLY is sent.

- Destination sequence number is higher than the Destination sequence in the Route Discovery Packet

- Increments Hop count and rebroadcasts ROUTE REPLY

- Stores the data in a new entry in its reverse route table.

Route Discovery (4)

Source address	Destination address	Destination sequence #	Hop count	Lifetime

Format of a ROUTE REPLY packet.

Route Discovery (5)

IN response

•Source addr., destination addr. and Hop Count copied but Dest. Seq. number taken from its counter.

•Hopcount is set to 0, Lifetime field controls how long the route is valid.

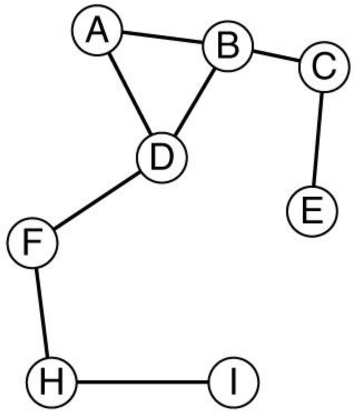
At each intermediate node:

1.No route to I is known,
2.Sequence number of I in the ROUTE REPLY packet is greater than the value in the routing table
3.The sequence numbers are equal but the new route is shorter
4.Hop Count incremented
5.In large network, discovery increases with Time to Live incrementally being increased from 1, 2, 3, …

Ad Hoc Networks: Route Maintenance

Dest.	Next hop	Distance	Active neighbors	Other fields
A	A	1	F, G	
B	B	1	F, G	
C	B	2	F	
E	G	2		
F	F	1	A, B	
G	G	1	A, B	
H	F	2	A, B	
I	G	2	A, B	

(a)



(b)

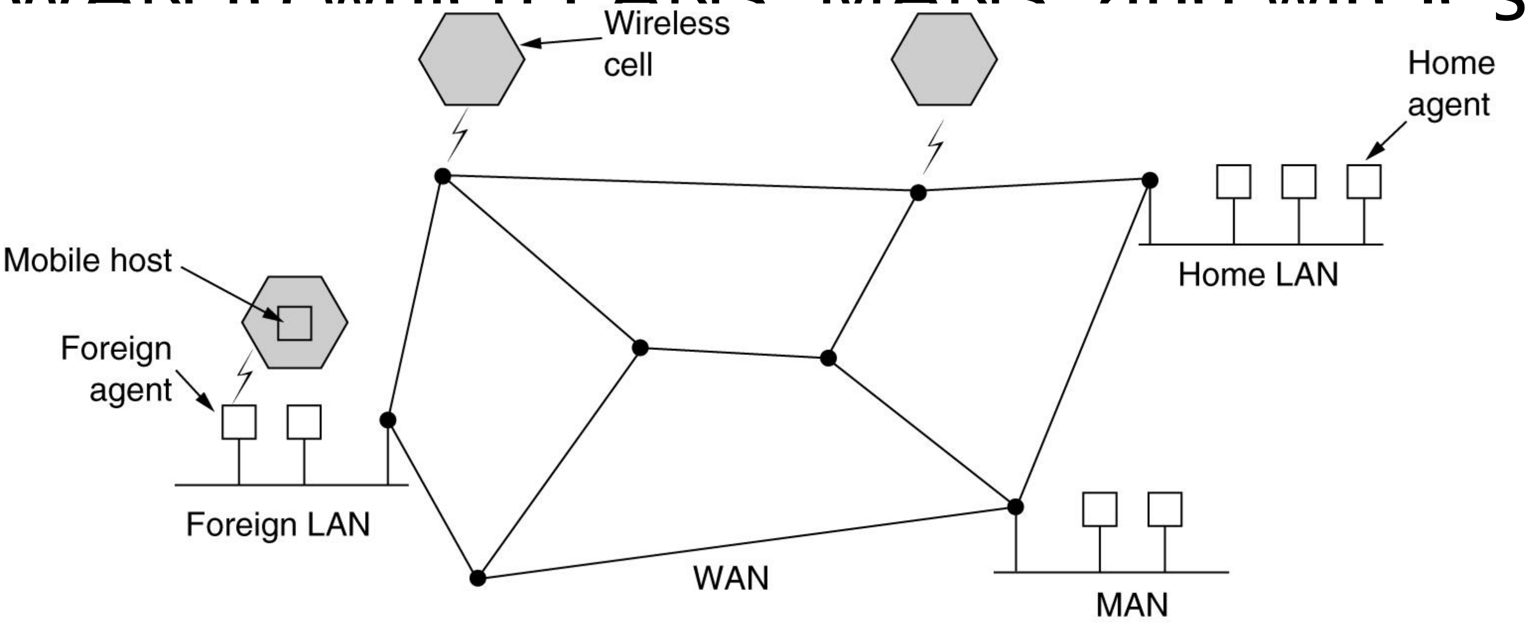
Active Neighbours that have fed in those destinations to A in last t seconds

(a) D's routing table before G goes down.

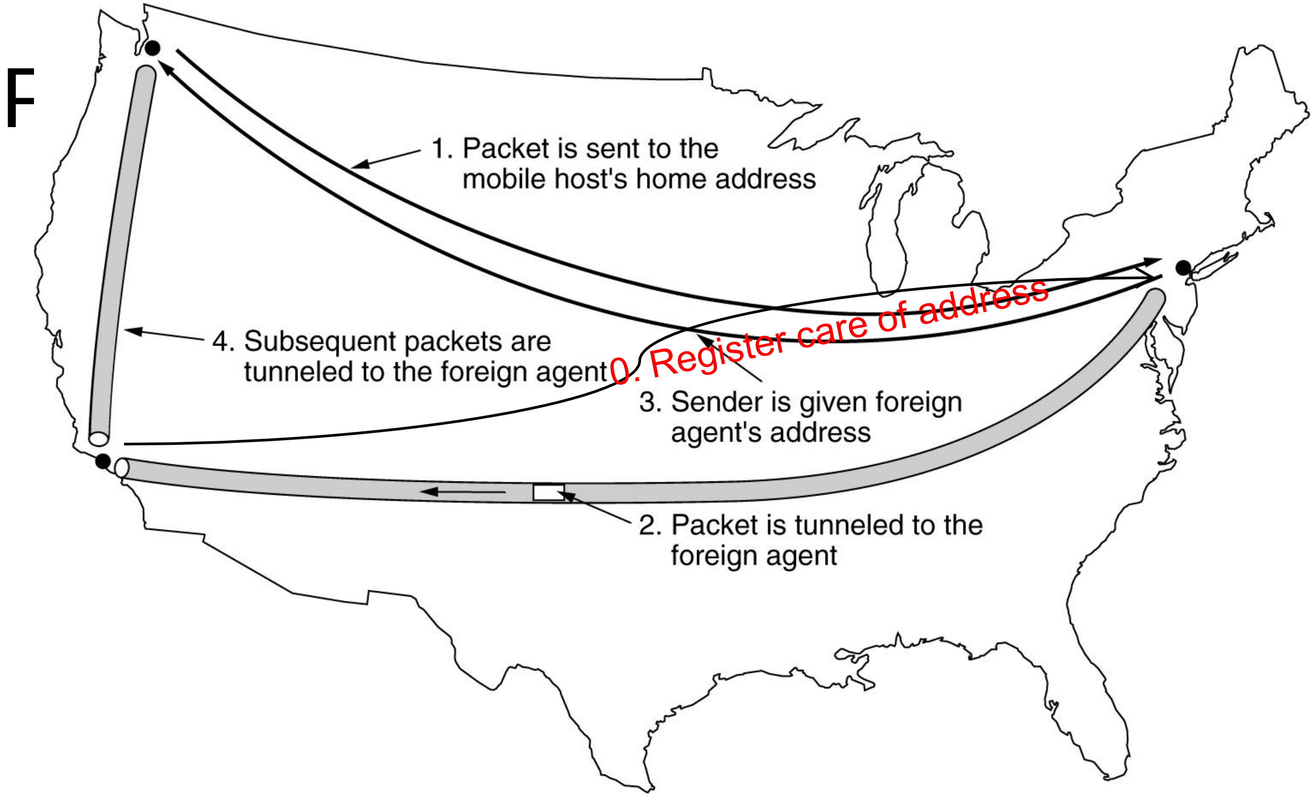
(b) The graph after G has gone down.

Routing for Mobile Hosts

A WAN to which LANs, MANs, and wireless



Routing for Mobile Hosts (2)


F
1. Packet is sent to the mobile host's home address
0. Register care of address
4. Subsequent packets are tunneled to the foreign agent
3. Sender is given foreign agent's address
2. Packet is tunneled to the foreign agent

Routing in Ad Hoc Networks

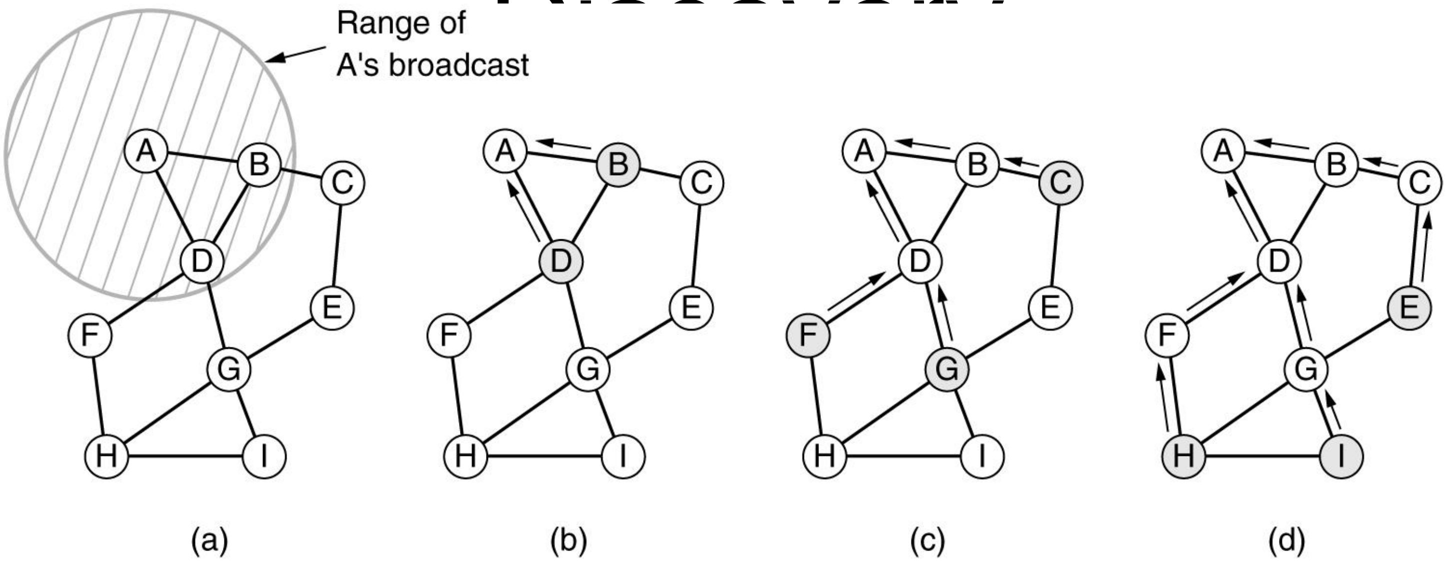
Possibilities when the routers are mobile:

1. Military vehicles on battlefield.
   - No infrastructure.
2. A fleet of ships at sea.
   - All moving all the time
3. Emergency works at earthquake .
   - The infrastructure destroyed.
4. A gathering of people with notebook computers.
   - In an area lacking 802.11.

AODV

Adhoc On-demand Distance Vector

Ad Hoc Networks: Route Discovery

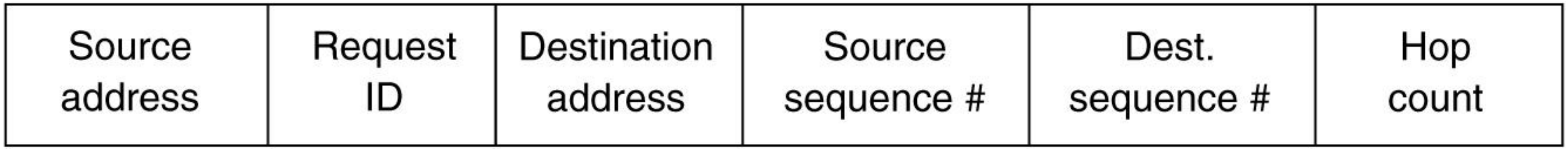


(a) Range of A's broadcast. (Maintains a distance vector routing table)

(b) After B and D have received A's broadcast.

(c) After C, F, and G have received A's broadcast.

Route Discovery (2)



Source address	Request ID	Destination address	Source sequence #	Dest. sequence #	Hop count

Format of a ROUTE REQUEST packet.

Route Discovery (3)

- The (Source Address, Request ID) pair is looked up in a local history table
- Receiver looks up the destination in its route table. If a fresh route is known, then a ROUTE REPLY is sent.
- Destination sequence number is higher than the Destination sequence in the Route Discovery Packet
- Increments Hop count and rebroadcasts ROUTE REPLY
- Stores the data in a new entry in its reverse route table.

Route Discovery (4)

Source address	Destination address	Destination sequence #	Hop count	Lifetime

Format of a ROUTE REPLY packet.

Route Discovery (5)

IN response

•Source addr., destination addr. and Hop Count copied but Dest. Seq. number taken from its counter.

•Hopcount is set to 0, Lifetime field controls how long the route is valid.

At each intermediate node:

1.No route to I is known,

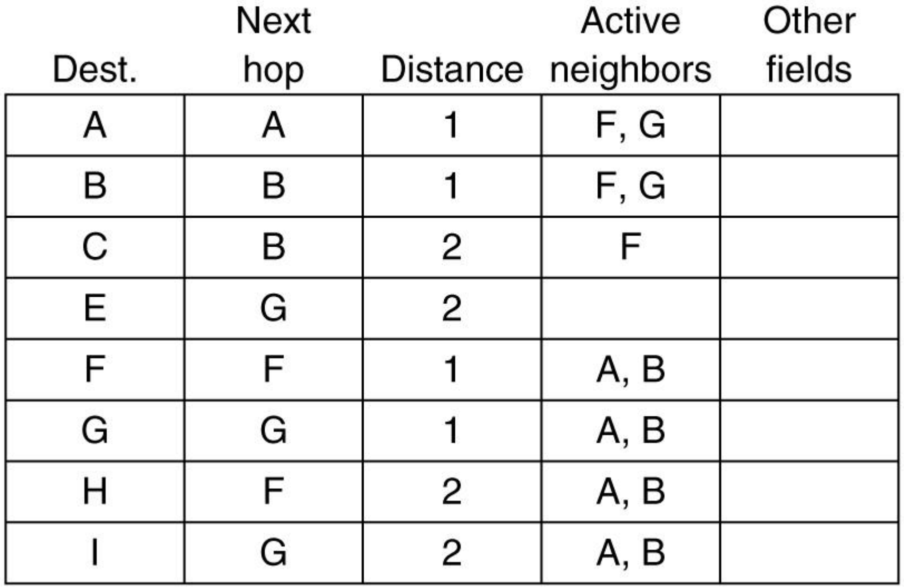
2.Sequence number of I in the ROUTE REPLY packet is greater than the value in the routing table

3.The sequence numbers are equal but the new route is shorter

4.Hop Count incremented

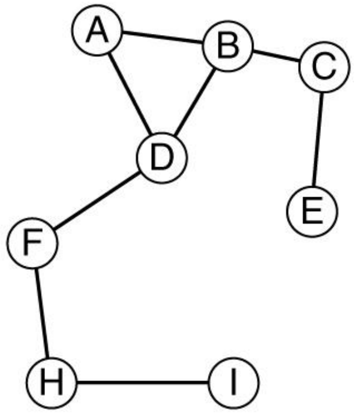
5.In large network, discovery increases with Time to Live incrementally being increased from 1, 2, 3, …

Ad Hoc Networks: Route Maintenance



Dest.	Next hop	Distance	Active neighbors	Other fields
A	A	1	F, G	
B	B	1	F, G	
C	B	2	F	
E	G	2		
F	F	1	A, B	
G	G	1	A, B	
H	F	2	A, B	
I	G	2	A, B	

(a)



(b)

Active Neighbours that have fed in those destinations to A in last t seconds

(a) D's routing table before G goes down.

(b) The graph after G has gone down.

Node Lookup in Peer-to-Peer Networks

P2P: Large connection of computers, without central control where typically each node has some information of interest.

- No central control for routing
- No central data repository

Two basic questions:

1. How to make data at each node available?
2. How to find required information?

The questions are interrelated, but will be looked at separately.

Assumption

- Each record (data to be shared) can be identified by a ASCII string such as the filename.

Over the past 3-4 years, there has been several proposals for P2P architectures we shall look at *Chord*.

Basics of Chord

- Uses a hash function such as SHA-1.

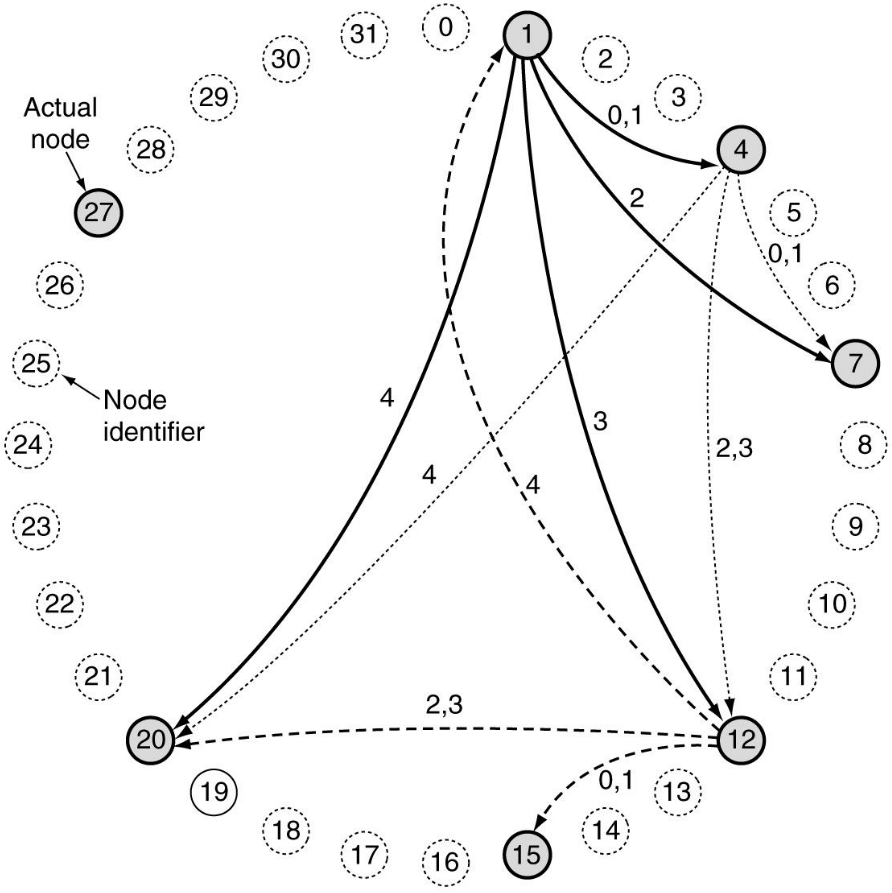
- SHA-1 converts a variable length input into a highly random 160 bit value

- Using SHA-1, Chord hashes:

node IP addresses node identifiers (160 bits)

names of records keys (160 bits)

Storing Records



Start	IP addr of successor
2	4
3	4
5	7
9	12
17	20

Node 1's finger table

Start	IP addr of successor
5	7
6	7
8	12
12	12
20	20

Node 4's finger table

Start	IP addr of successor
13	15
14	15
16	20
20	20
28	1

Node 12's finger table

(a) A set of 32 node identifiers arranged in a circle. The shaded ones correspond to actual machines. The arcs show the fingers from nodes 1, 4, and 12. The labels on the

Storing records

- *successor (k)* is the first *real node* after k.

- To store data *name,* a node N creates a tuple (name, N's IP address) and stores the tuple at *successor(hash(name)).* The original data remain at N, just the tuple is stored at *successor(hash(name)).*

- If *hash(name)* = 22, then the tuple is stored at node 27.

- To find information *name,* a node does *key* = *hash(name),* then gets the record tuple from *successor(key).*

- Simple? Mostly, except for implementing *successors(key)* efficiently.

Finding records

Each node needs to store the IP addresses of its successor.

Initially, the network start out with just a few nodes:

1. All nodes know each other.
2. They can easily arrange themselves into a the Chord ring.
3. *successor(k)* can be computed.

When a node tries to join:

1. It calculates its node ID say *p*.
2. Then asks any node already in the ring to find successor(p).
3. Asks *successor(p)* for *successor(p)*'s predecessor and inserts itself between them.

Any node in the ring can find *successor(k)* by propagating the query around the ring starting with its successor.

Finger table

- Even if both successor and predecessor pointers are used, a sequential search will take time on average *O(n/2)* [*n* is the number of nodes].

- Chord reduces this search time using a finger table at each node.

- The finger table contains up to *m* entries where each entry *i* consists of IP address of *successor(start[i])*

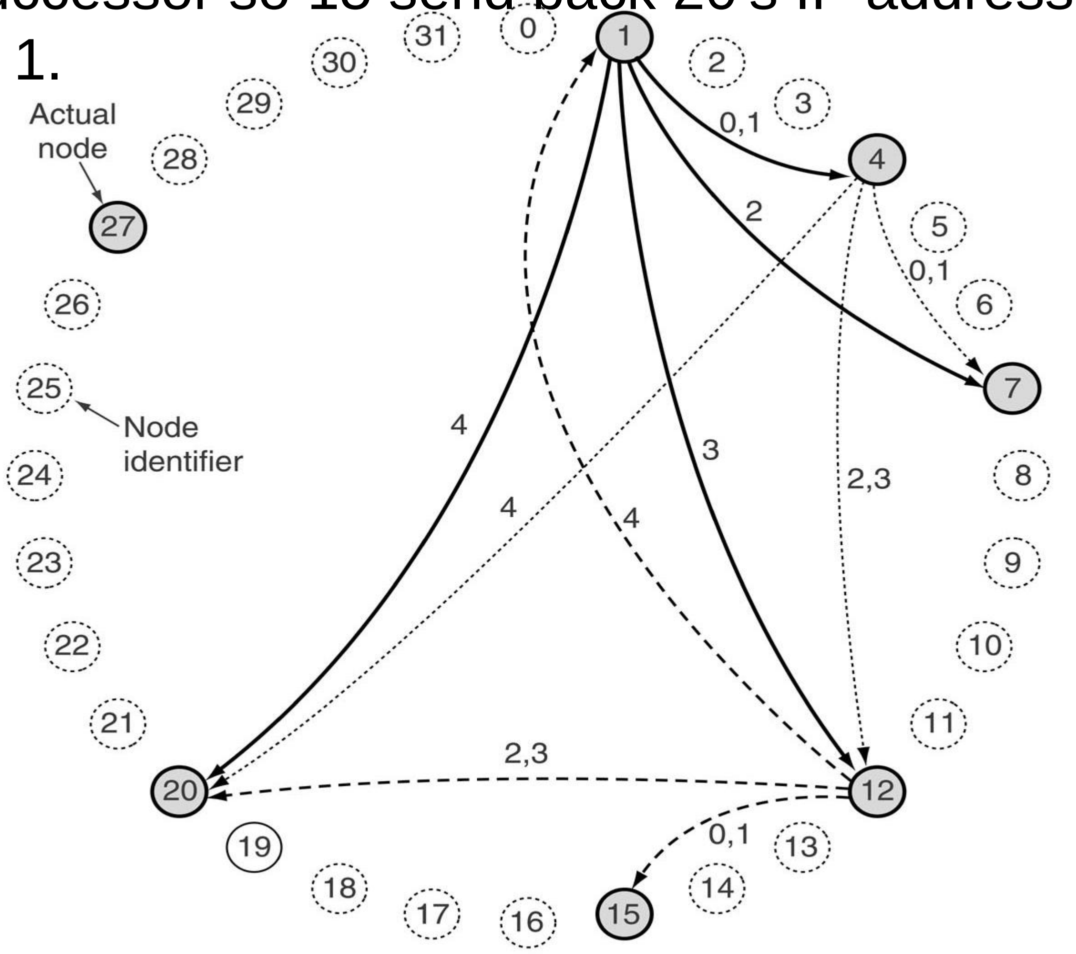
- *Start[i] = k + 2^i (modulo 2^m)*

- To find a record for key *k,* a node can directly jump to the closest predecessor of *k*.

- Average time can be reduced to O(log n).

Looking up key 16 at node 1

1. Nearest pred. 9 so query sent to 12
2. At 12 nearest pred. of 16 is 14 so query sent to 15
3. 15 knows that 16 is between itself and its successor so 15 send back 20's IP address to 1.



Start	IP addr of successor
2	4
3	4
5	7
9	12
17	20

Node 1's finger table

Start	IP addr of successor
5	7
6	7
8	12
12	12
20	20

Node 4's finger table

Start	IP addr of successor
13	15
14	15
16	20
20	20
28	1

Node 12's finger table

Maintaining finger table

- Maintaining the finger table does not come for free.

- Every time a new node is added a few successors and predecessor entries will change.

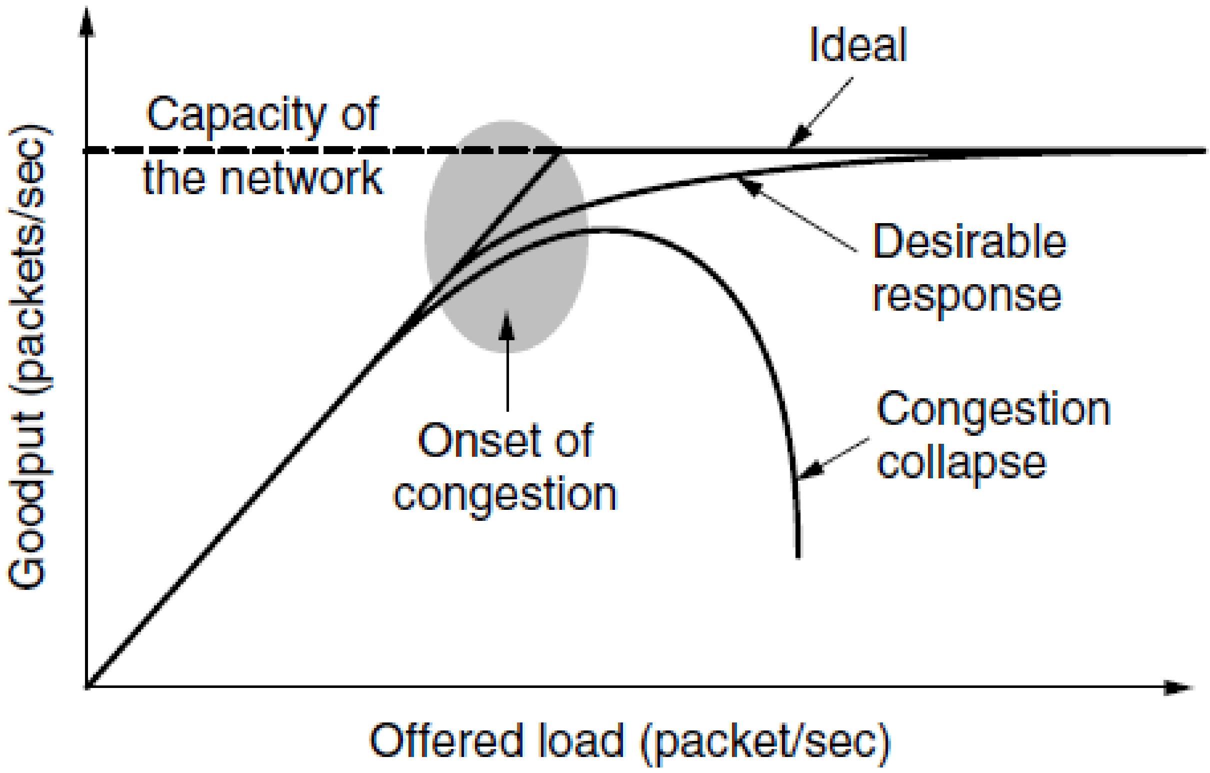
The Network Layer

Congestion Control Algorithms & Quality-of-Service

Congestion Control Algorithms

- Approaches to Congestion Control
- Traffic-Aware Routing
- Admission Control
- Traffic Throttling
- Load Shedding

Congestion



When too much traffic is offered, congestion sets in and performance degrades sharply.

General Principles of Congestion Control

1. Monitor the system
   - detect when and where congestion occurs.
2. Pass information to where action can be taken.
3. Adjust system operation to correct the problem.

4. Difference between Congestion control and flow control – Elaborate

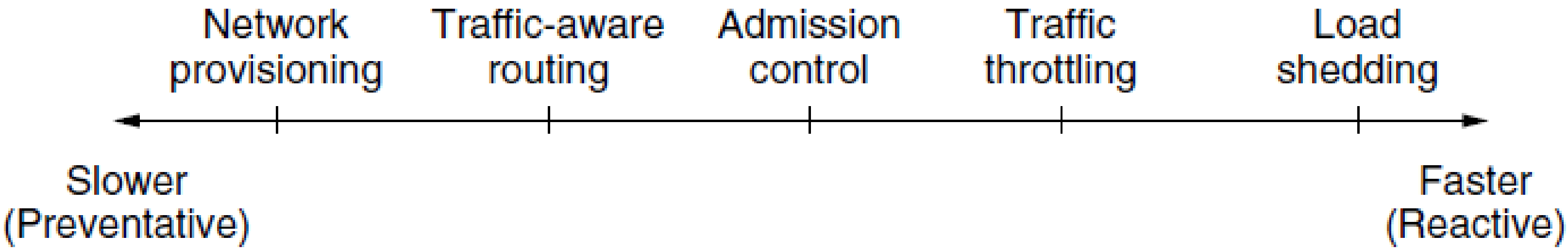
5. Problem of infinite buffer – Elab

Approaches to Congestion Control

Two solutions possible:

1)Increase resources

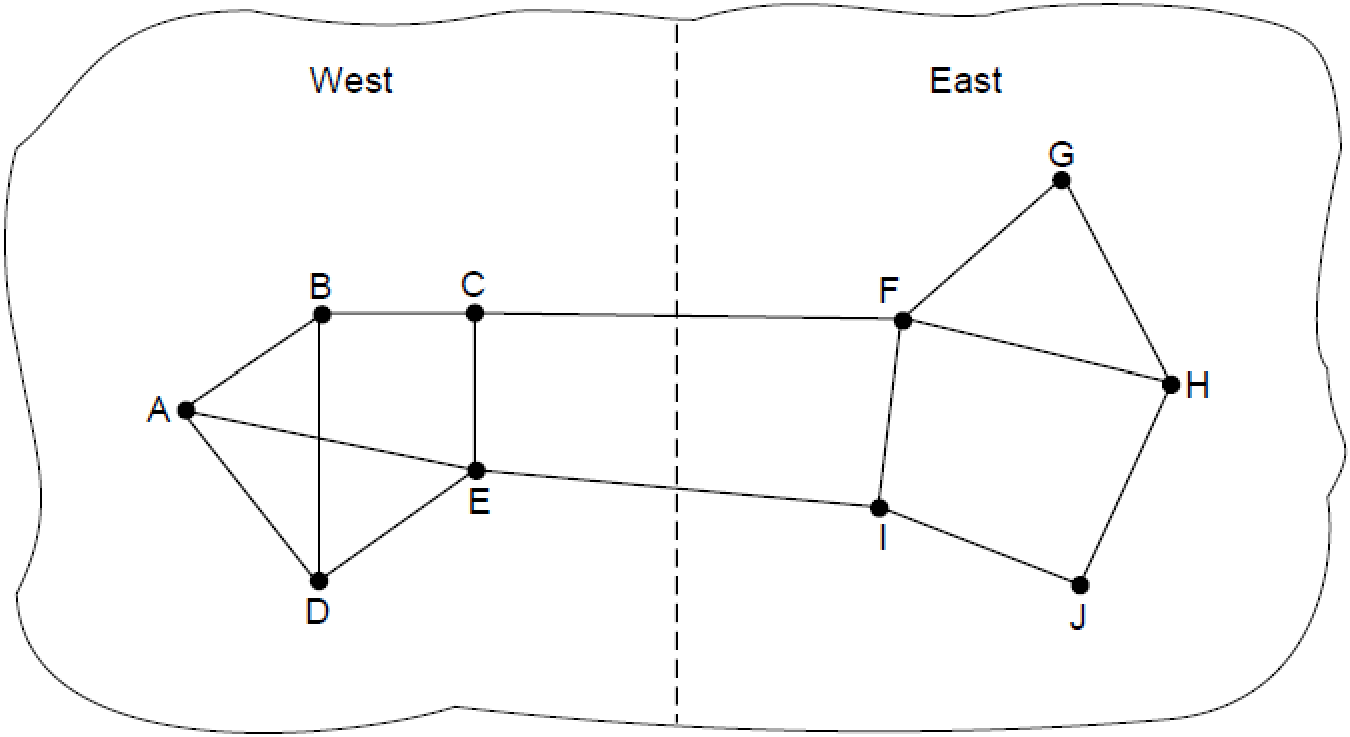
2)Decrease load



Timescales of approaches to congestion control.

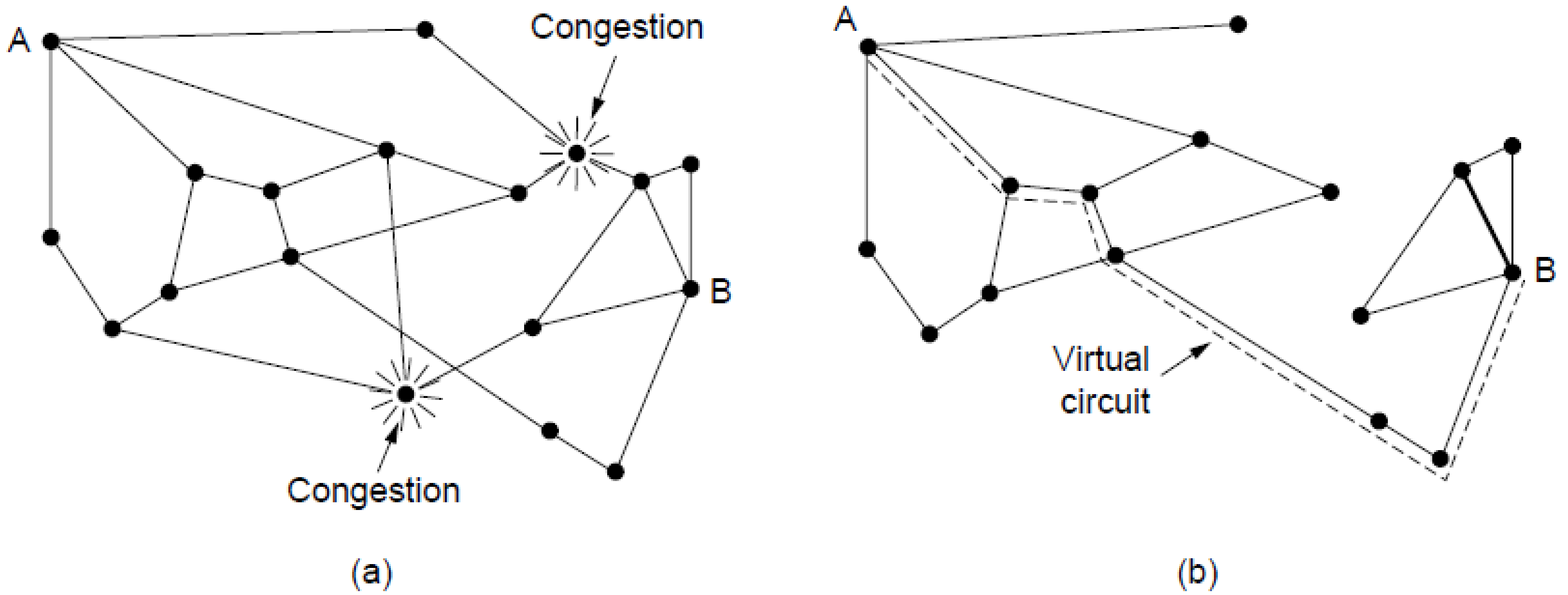
Admission control in virtual networks – no

Traffic-Aware Routing



A network in which the East and West parts are connected by two links.

Admission Control



(a)A congested network. (b) The portion of the network that is not congested. A virtual circuit from A to B is also shown.

Problem is in virtual circuits – there may be

Traffic Throttling:Congestion Detection

- Routers must determine when congestion is approaching, ideally before it has arrived.
- Each router can continuously monitor the resources it is using.
- 3 possibilities:

  1. utilization of the output links
  2. buffering of queued packets inside the router (most useful)
  3. no. of packets that are lost due to insufficient buffering

EWMA (Exponentially Weighted Moving Average)

- $d_{new} = \alpha d_{old} + (1 - \alpha)s$,

  where, the constant α determines how fast the router forgets recent history.

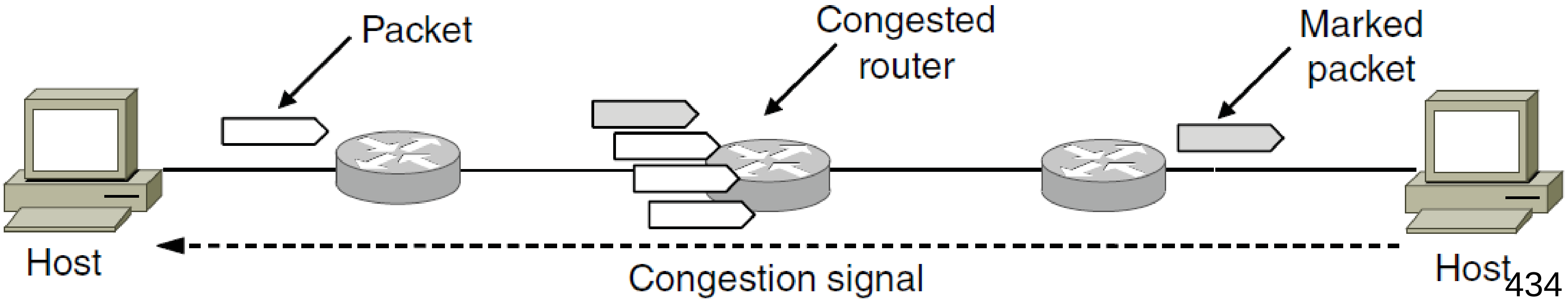
- Whenever d moves above the threshold, the router notes

Traffic Throttling: Feedback

- Routers must deliver timely feedback to the senders that are causing the congestion.
- The router must identify the appropriate senders.
- It must then warn them carefully, without sending many more packets into the already congested network.

- Many feedback mechanisms:

Mechanism 1: Explicit Congestion Notification (ECN)

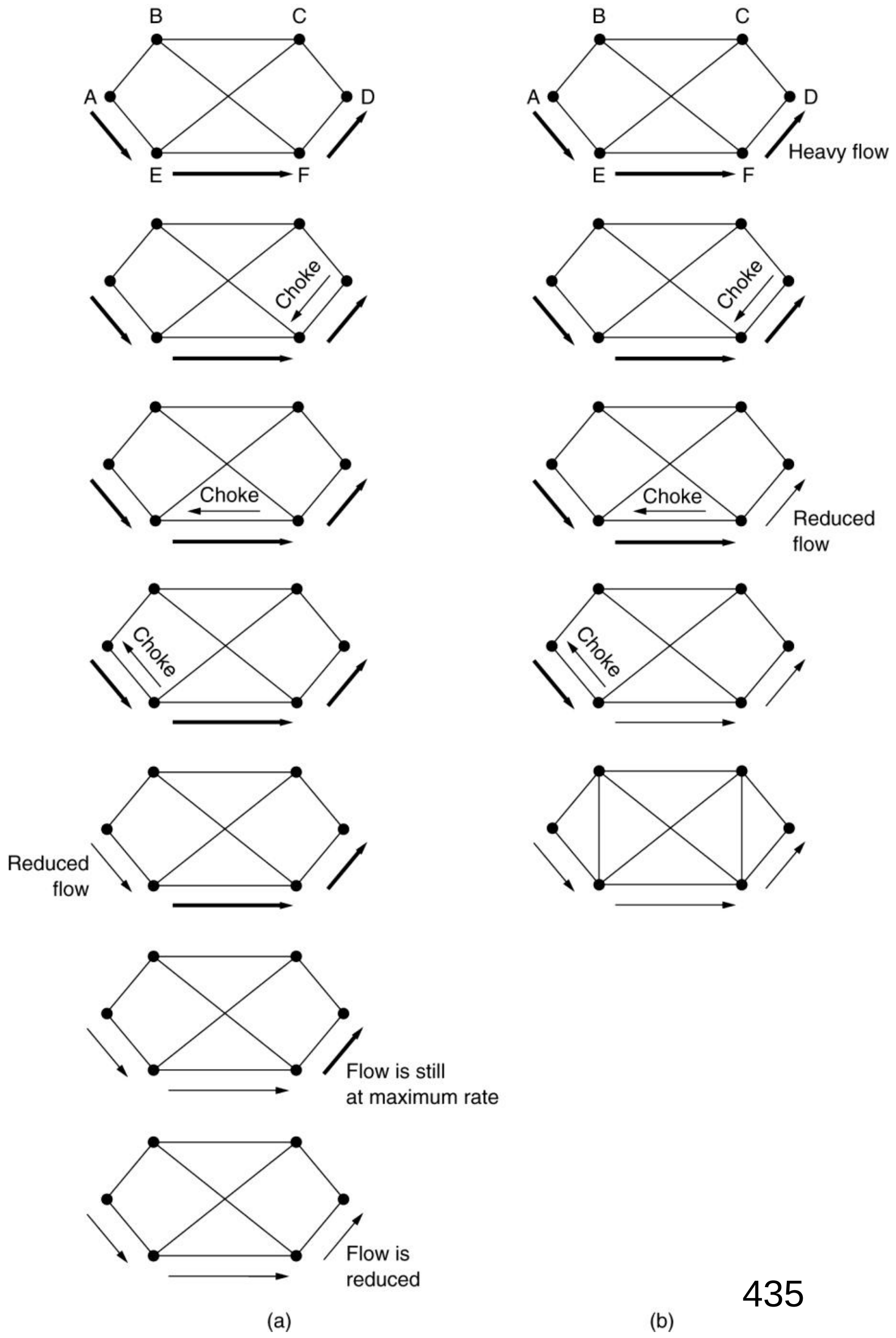


Mechanisms 2 & 3

Direct Choke Packets,
Hop-by-Hop
Backpressure

M-2: A choke packet that affects only the source.

M-3: A choke packet that affects each hop it passes through.



Mechanism 4: Load Shedding

- Performed when all other strategies fail.
- Cause blackout in some areas to save the entire network from failing.
- Intelligent packet drop policy desired.
- Which packets to discard may depend on application

  Multimedia – old packets (full frame not to be discarded)

  Text – Recent Packets
- Packet's importance can be marked in the beginning (application layer), then decision on which packets to discard can be taken

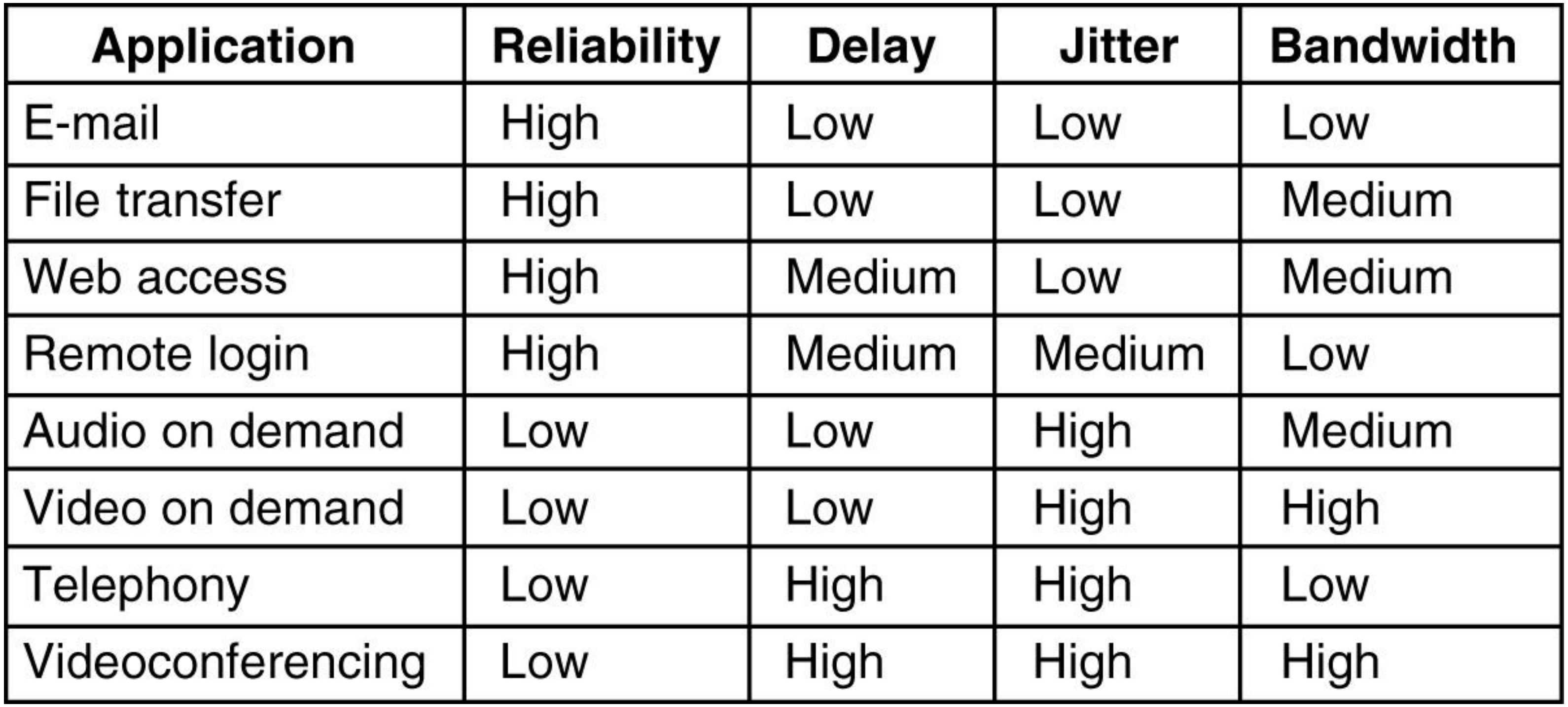
Mechanism 5: Random Early Detection

- Discard packets before all the buffer space is really exhausted.
- To determine when to start discarding, routers maintain a running average of their queue lengths.
- When average queue length exceeds a threshold, the link is said to be congested – small fraction of packets dropped at random.
- The affected sender will notice the loss when there is no acknowledgement – transport protocol slowed down.

Quality of Service

- Requirements
  - Minimum throughput and maximum latency
- Techniques for Achieving Good Quality of Service
- Integrated Services
- Differentiated Services
- Label Switching and MPLS

Requirements

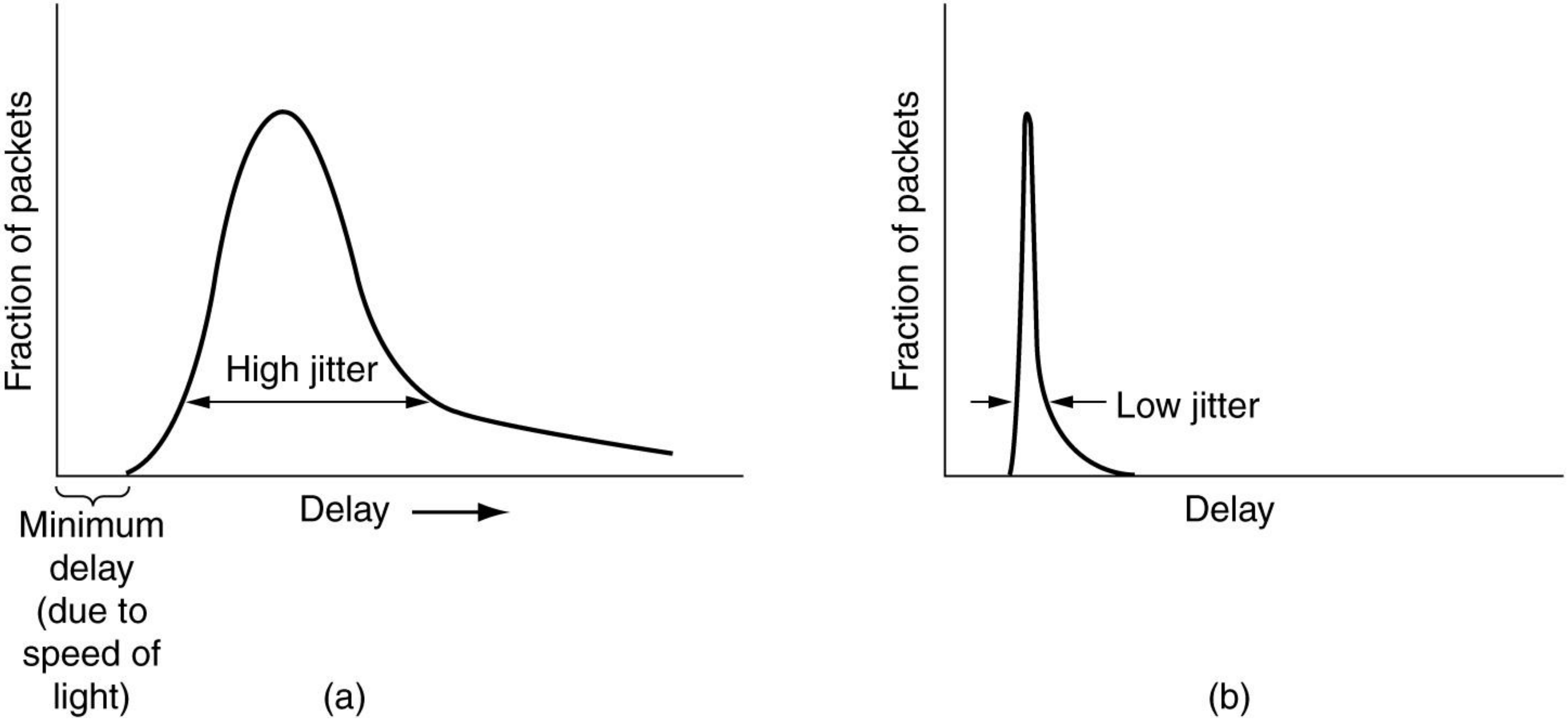


Application	Reliability	Delay	Jitter	Bandwidth
E-mail	High	Low	Low	Low
File transfer	High	Low	Low	Medium
Web access	High	Medium	Low	Medium
Remote login	High	Medium	Medium	Low
Audio on demand	Low	Low	High	Medium
Video on demand	Low	Low	High	High
Telephony	Low	High	High	Low
Videoconferencing	Low	High	High	High

Categories of QoS and Examples

1. Constant bit rate
   - Telephony
2. Real-time variable bit rate
   - Compressed videoconferencing
3. Non-real-time variable bit rate
   - Watching a movie on demand
4. Available bit rate
   - File transfer

Jitter Control

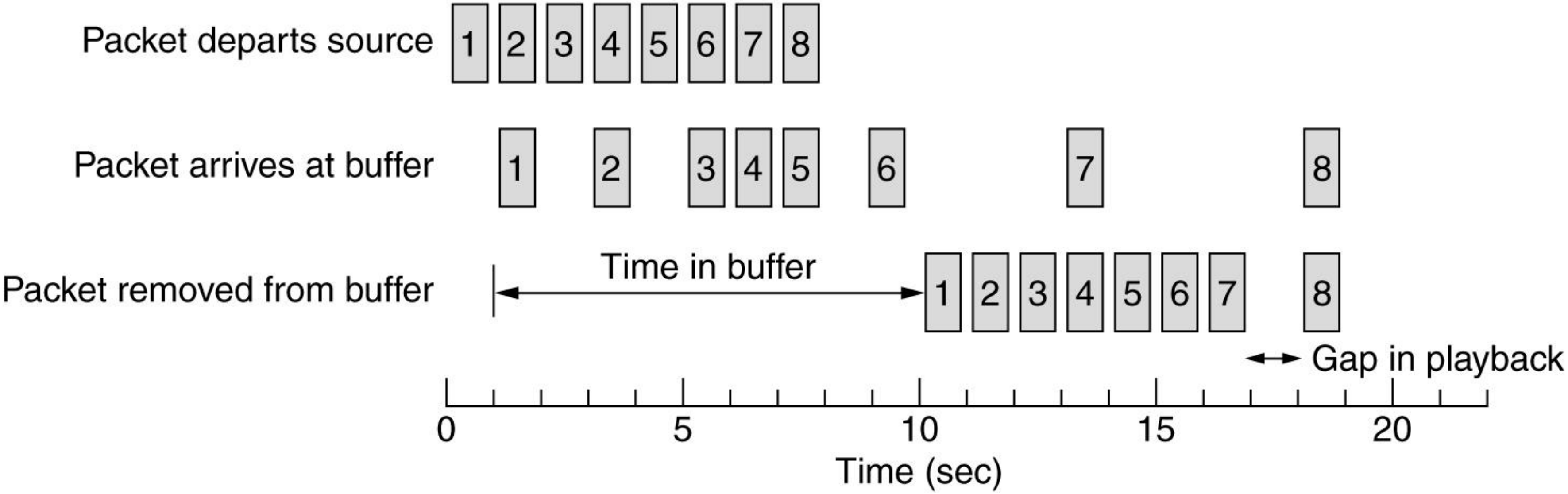


Schedule Delay according to Deadline Miss

(a) High jitter. (b) Low jitter

Buffering

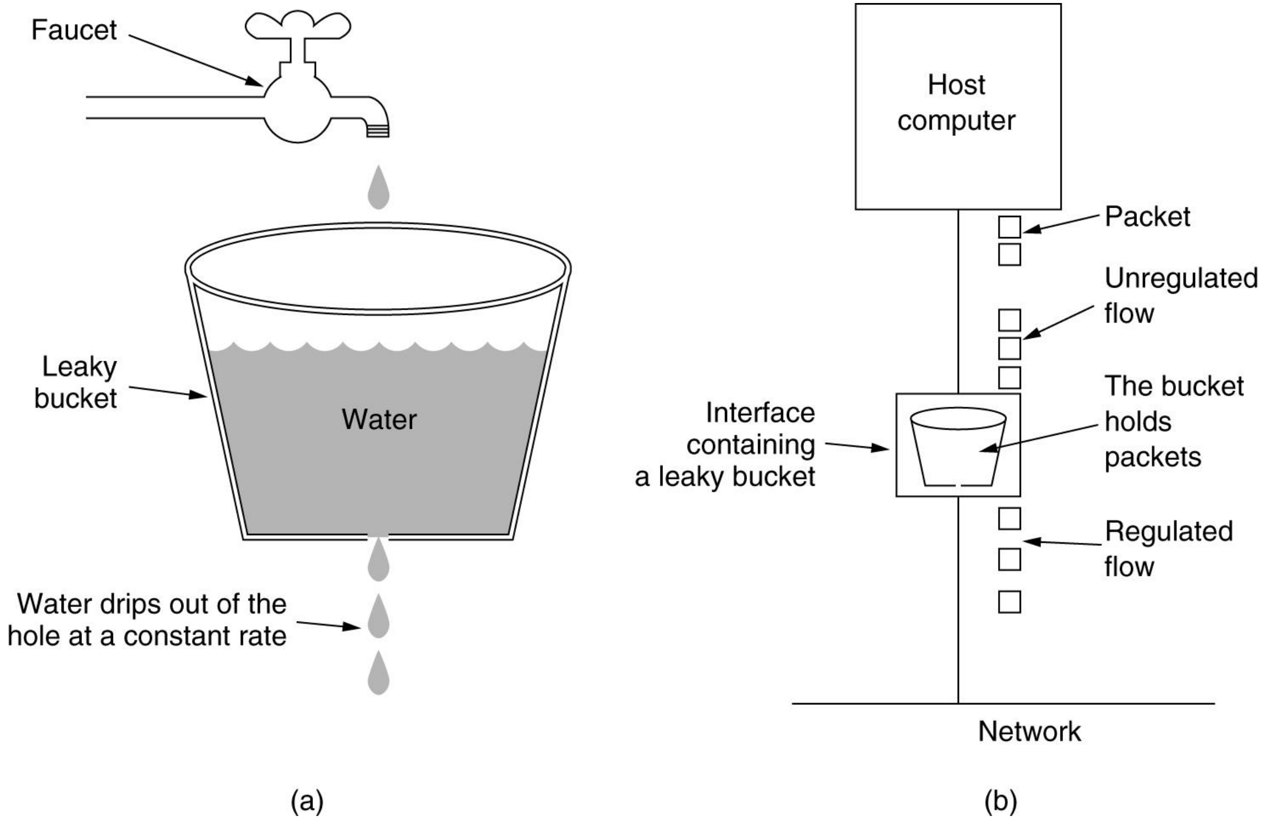
Smoothing the output stream by buffering



Traffic Shaping

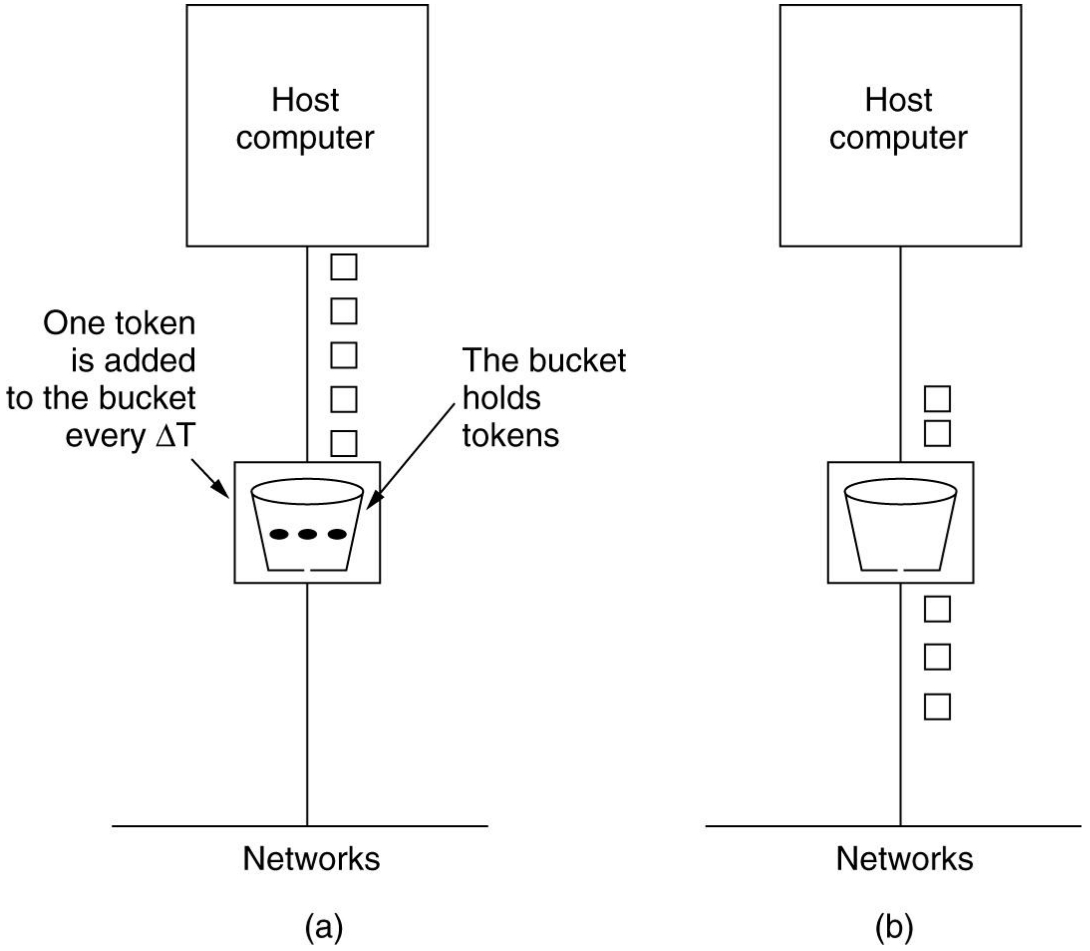
- Traffic in data networks is **bursty** – typically arrives at non-uniform rates as the traffic rate varies.
- **Traffic shaping** is a technique for regulating the average rate and burstiness of a flow of data that enters the network.
- When a flow is set up, the user and the network agree on a certain traffic pattern (shape).
- Sometimes this agreement is called an **SLA** (**Service Level Agreement**).

The Leaky Bucket Algorithm



(a) A leaky bucket with water. (b) a leaky bucket with packets.

The Token Bucket Algorithm



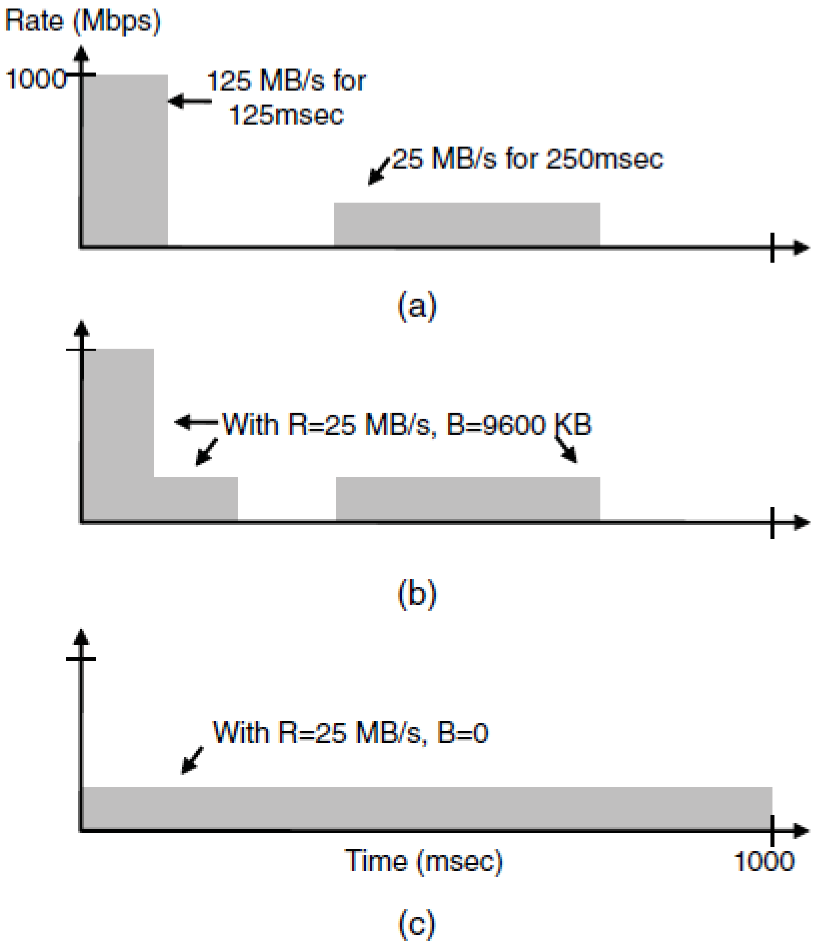
(a) Before. (b) After.

Token Bucket Algorithm (2)

- Burst length – *S* sec.
- Maximum output rate – *M* bytes/sec
- Token bucket capacity – *B* bytes
- Token arrival rate – *R* bytes/sec

- An output burst contains a maximum of **(*B* + *RS*)** bytes.
- The number of bytes in a maximum speed burst of length ***S*** seconds is ***MS***.
- Hence, we have: ***B* + *RS* = *MS***
- This equation can be solved to get ***S* = *B* /(*M* − *R*)**

Traffic Shaping (2)



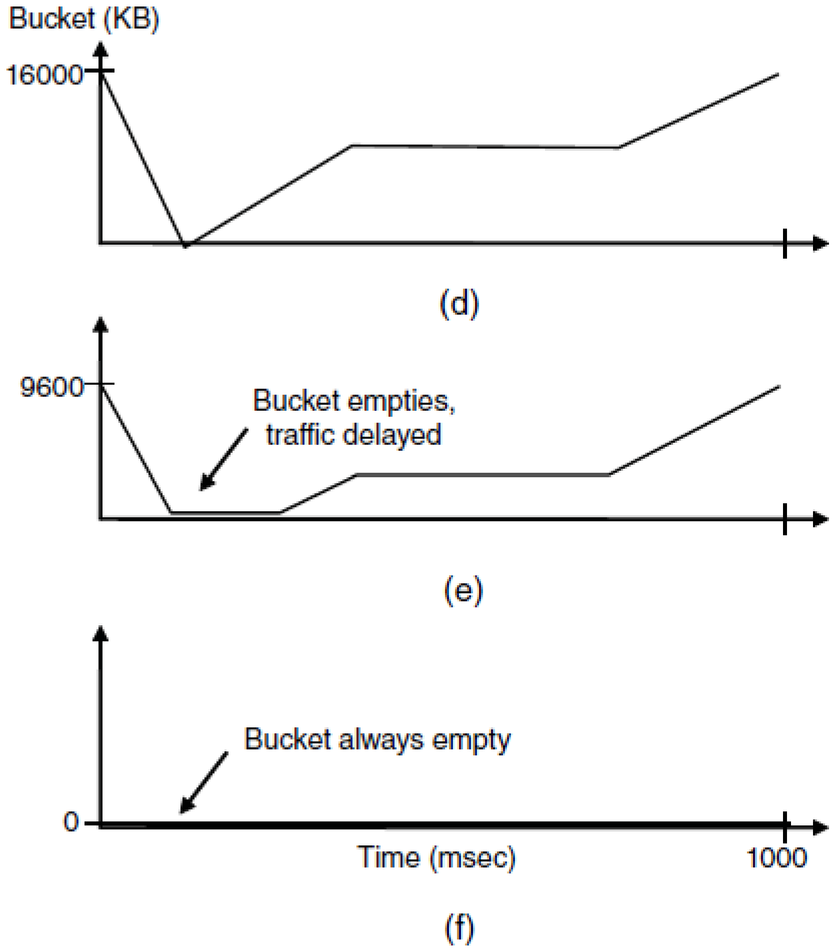
(a)

(b)

(c)

(a) Traffic from a host. Output shaped by a token bucket of rate 200 Mbps and capacity (b) 9600 KB, (c) 0 KB.

Traffic Shaping (3)



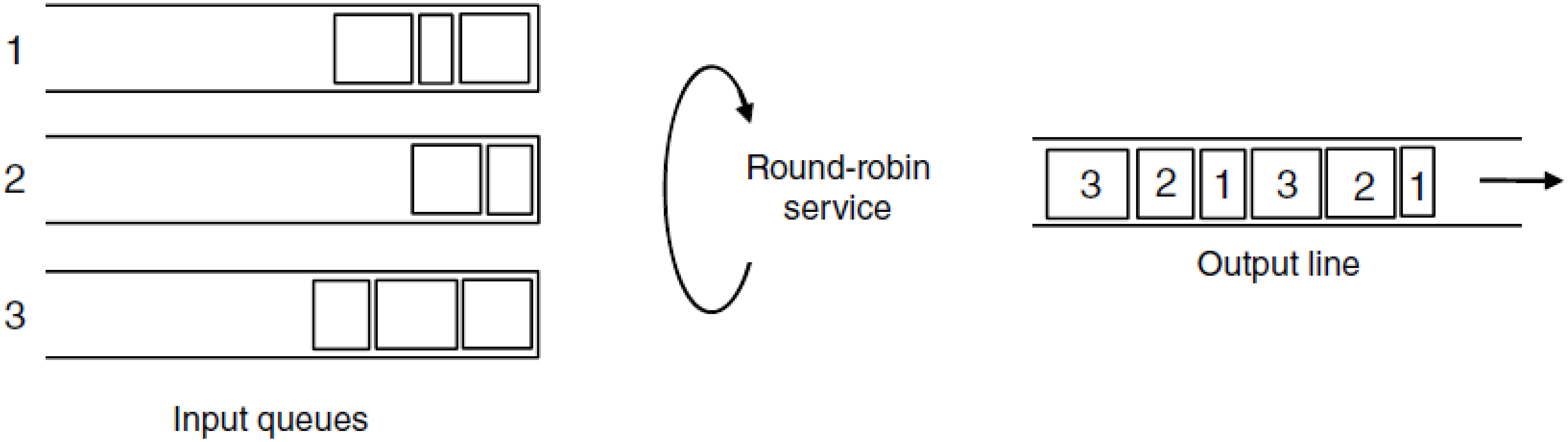
Token bucket level for shaping with rate 200 Mbps and capacity (d) 16000 KB, (e) 9600

Packet Scheduling

Kinds of resources that can potentially be reserved for different flows:

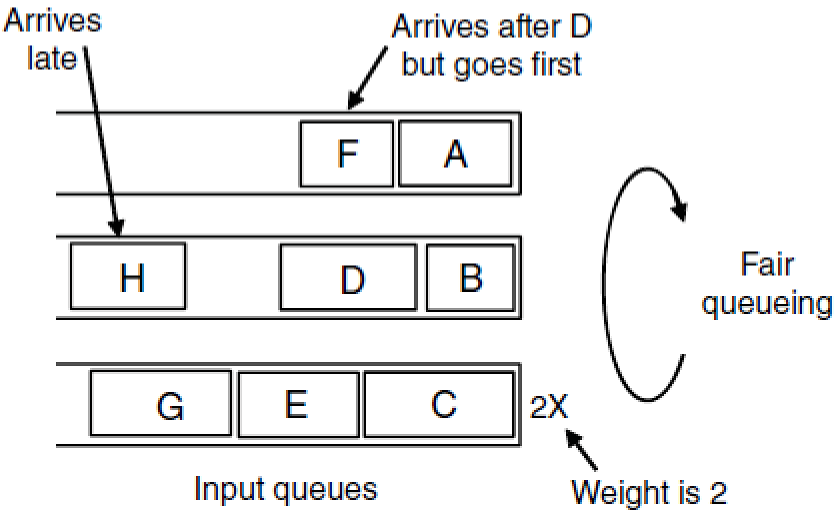
1. Bandwidth.
2. Buffer space.
3. CPU cycles.

Packet Scheduling (2)

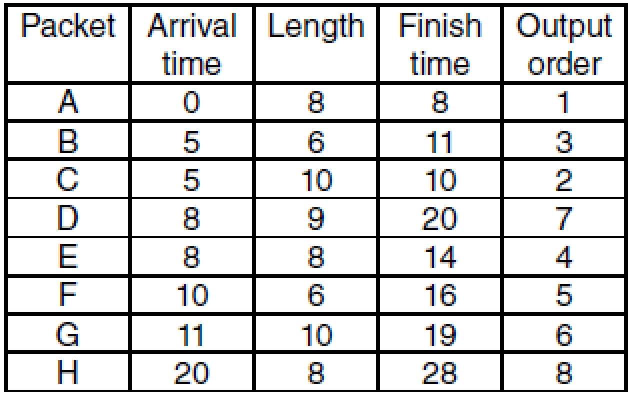


Round-robin Fair Queuing

Packet Scheduling (3)



(a)



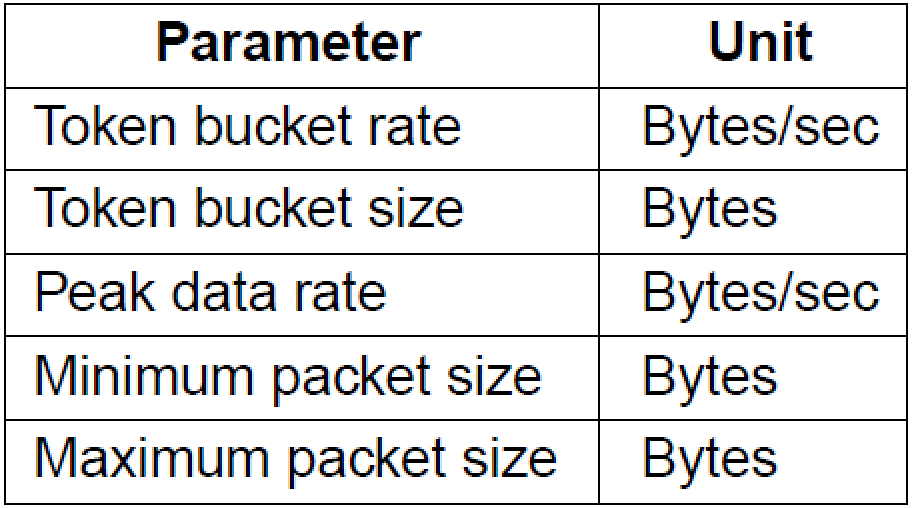
Packet	Arrival time	Length	Finish time	Output order
A	0	8	8	1
B	5	6	11	3
C	5	10	10	2
D	8	9	20	7
E	8	8	14	4
F	10	6	16	5
G	11	10	19	6
H	20	8	28	8

(b)

(a)Weighted Fair Queueing.

(b)Finishing times for the packets.

Admission Control (1)



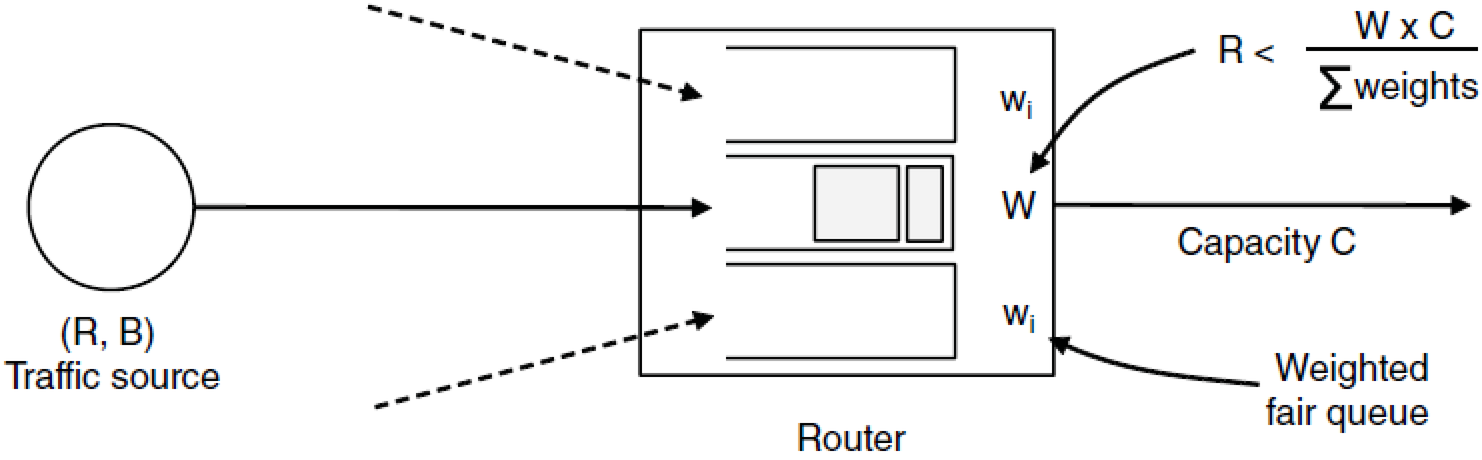
Parameter	Unit
Token bucket rate	Bytes/sec
Token bucket size	Bytes
Peak data rate	Bytes/sec
Minimum packet size	Bytes
Maximum packet size	Bytes

An example flow specification

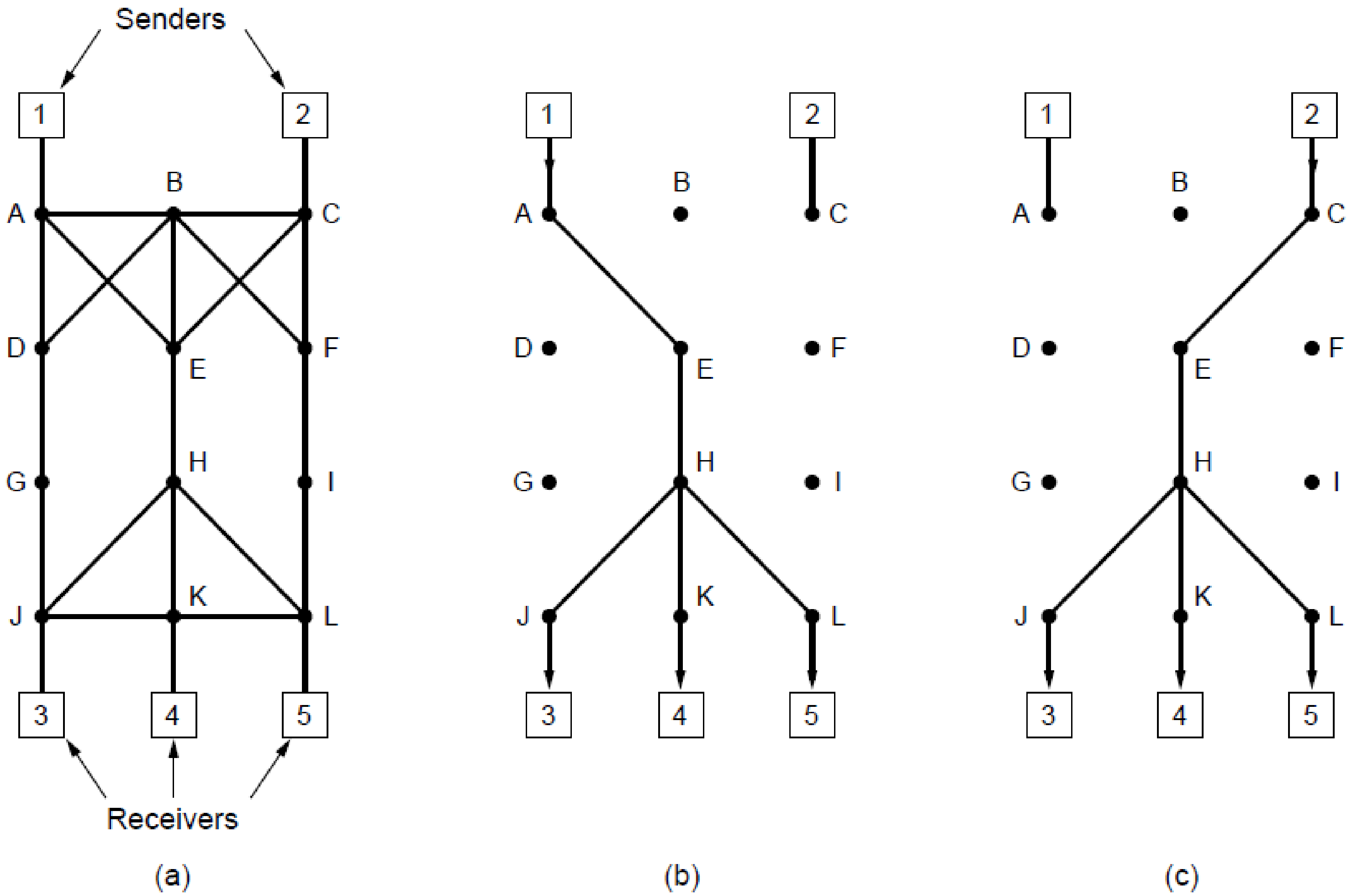
- T = 1/\mu X 1/(1-\lambda/\mu) -- \lambda = 0.95Mpackets/sec
- \mu = 1Mb packets/sec

Admission Control (2)

Bandwidth and delay guarantees with token



Integrated Services:
RSVP—The Resource reSerVation Protocol



(a) A network. (b) The multicast spanning tree for host 1

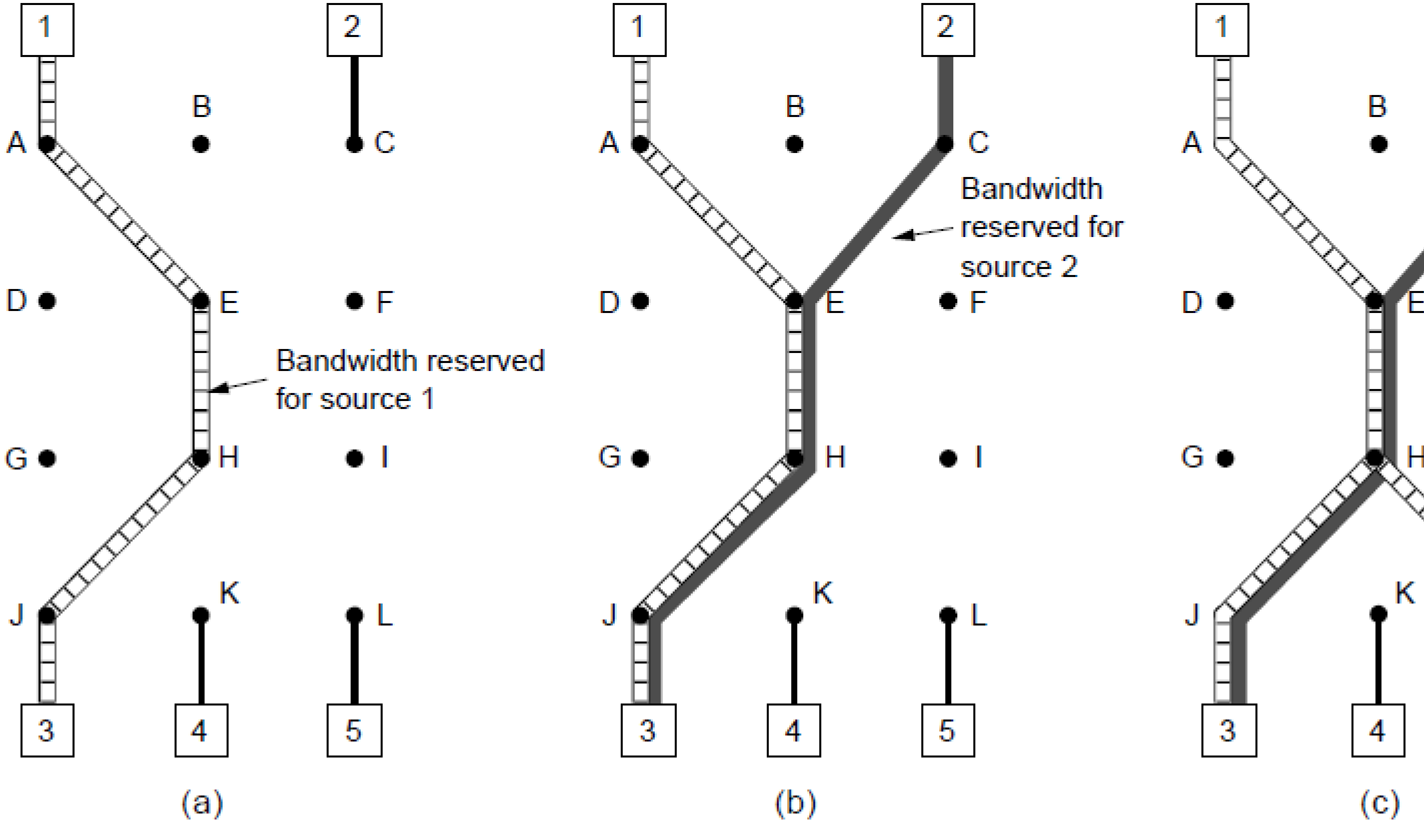
Hosts 1 and 2 are multicast sender

3,4, 5 are multicast receiver

Host 3 reserves for Host 1 and the Host 2

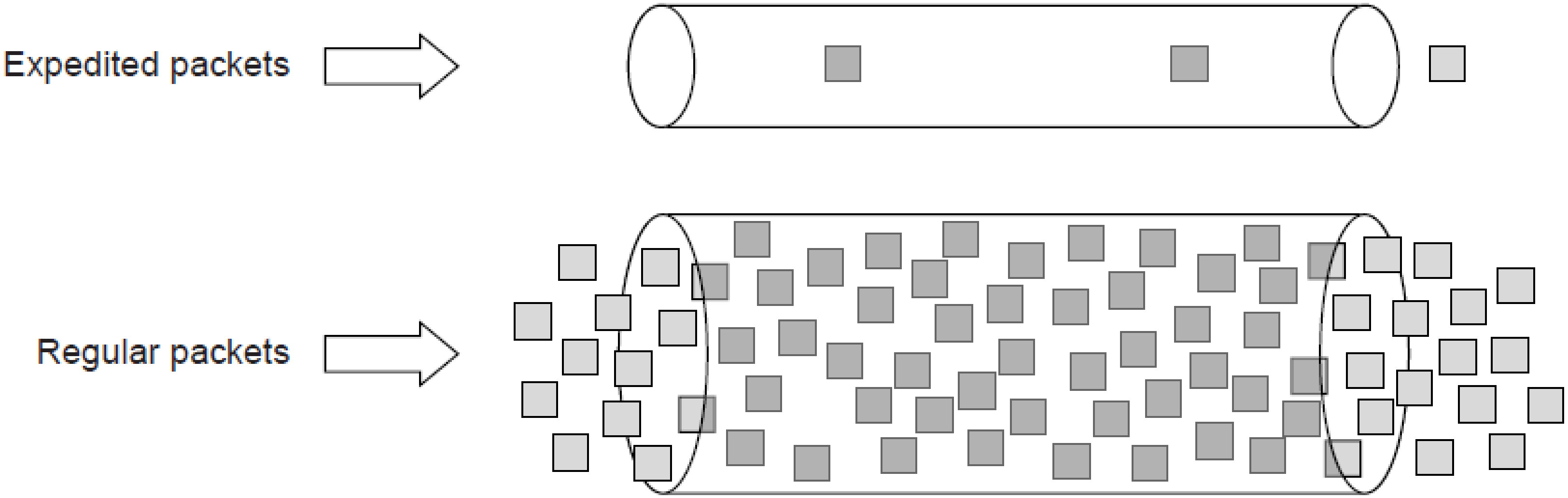
Host 5 reserves Host 1 (so the common path is utilized). However depending on need (Host 5 may be a bigger TV) – provision is made for the greediest part

RSVP (2)


1
2
B
A
C
D
E
F
Bandwidth reserved for source 1
G
H
I
J
K
L
3
4
5
(a)
1
2
B
A
C
Bandwidth reserved for source 2
D
E
F
G
H
I
J
K
L
3
4
5
(b)
1
2
B
A
C
D
E
F
G
H
I
J
K
L
3
4
5
(c)

(a) Host 3 requests a channel to host 1. (b) Host 3 then requests a second channel, to host 2. (c) Host 5 requests a channel to host 1.

Differentiated Services: Expedited Forwarding



Expedited packets experience a traffic-free network

Class-Based Service

Per Hop Behaviors

Traffic within a class are given preferential treatment

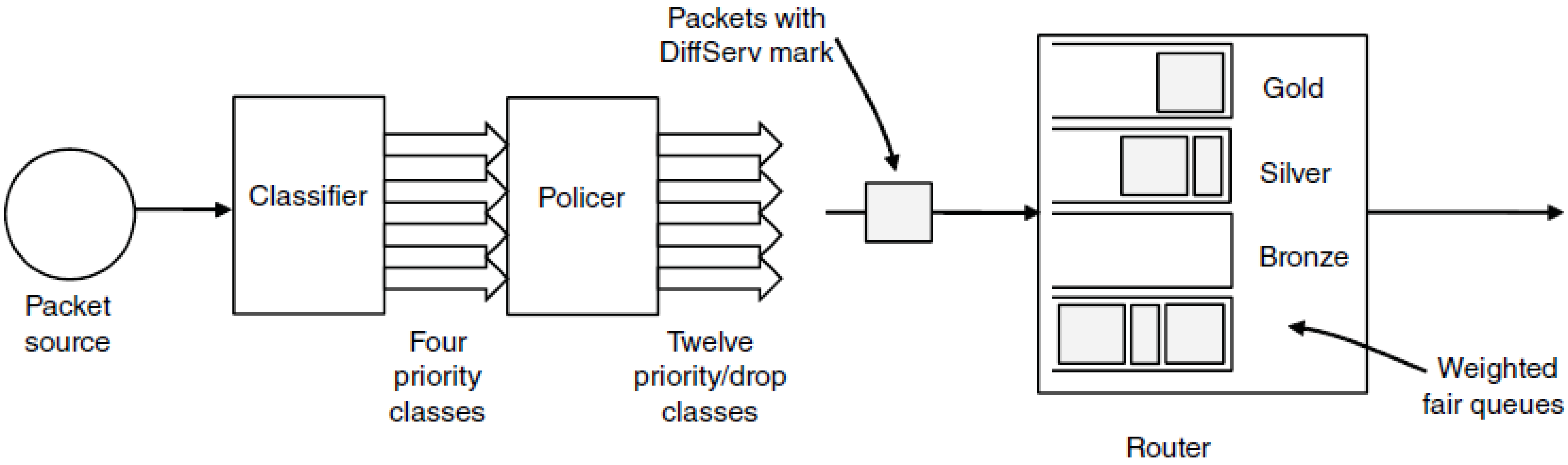
Expedited Forwarding

Packets marked – Regular or Expedited

Assured Forwarding

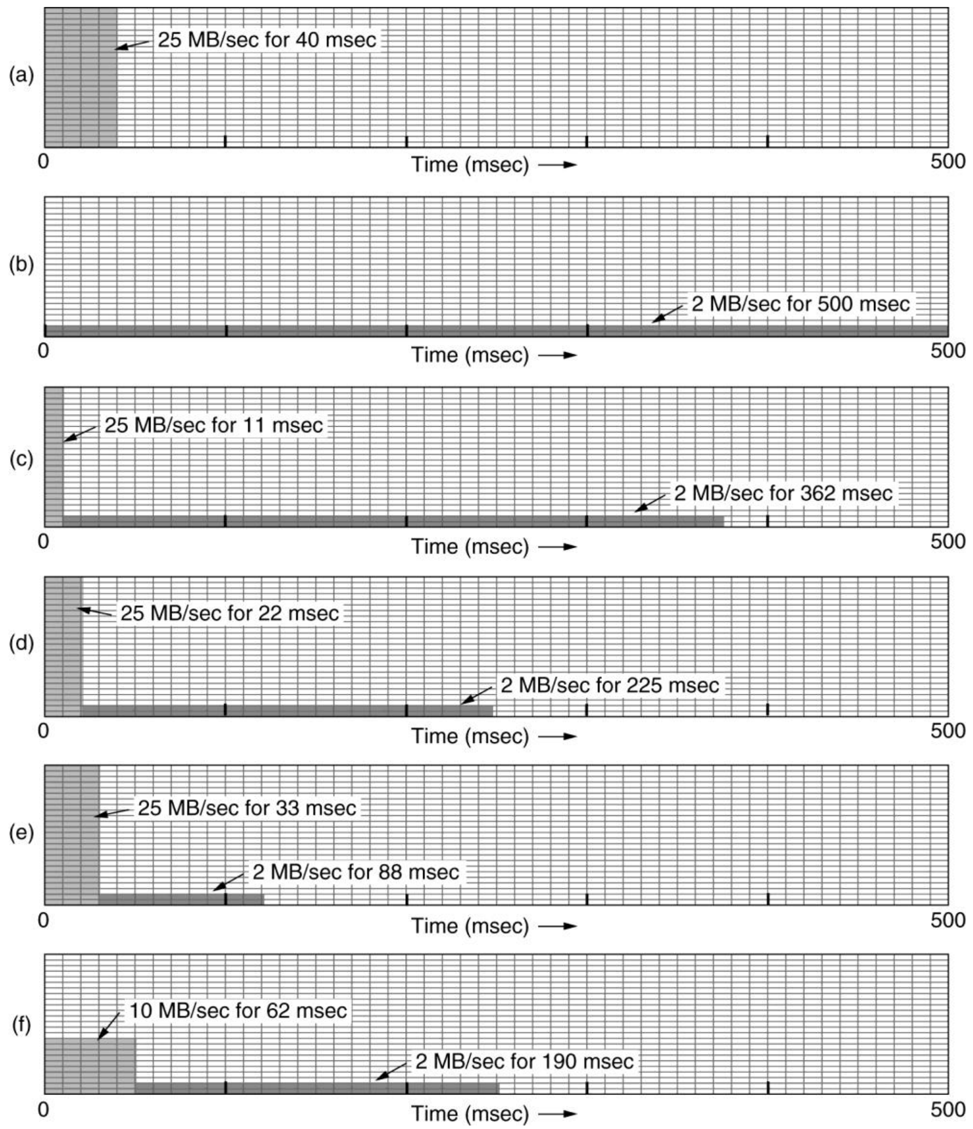
Gold, Silver, Bronze, common

Differentiated Services: Assured Forwarding



A possible implementation of assured forwarding, weighted fair scheduling, RED

The Leaky Bucket Algorithm



(a) Input to a leaky bucket. (b) Output from a leaky bucket. Output from a token bucket with capacities of (c) 250 KB, (d) 500 KB, (e) 750 KB, (f) Output from a 500KB token bucket feeding a 10-MB/sec leaky bucket.

Network Layer

Routing

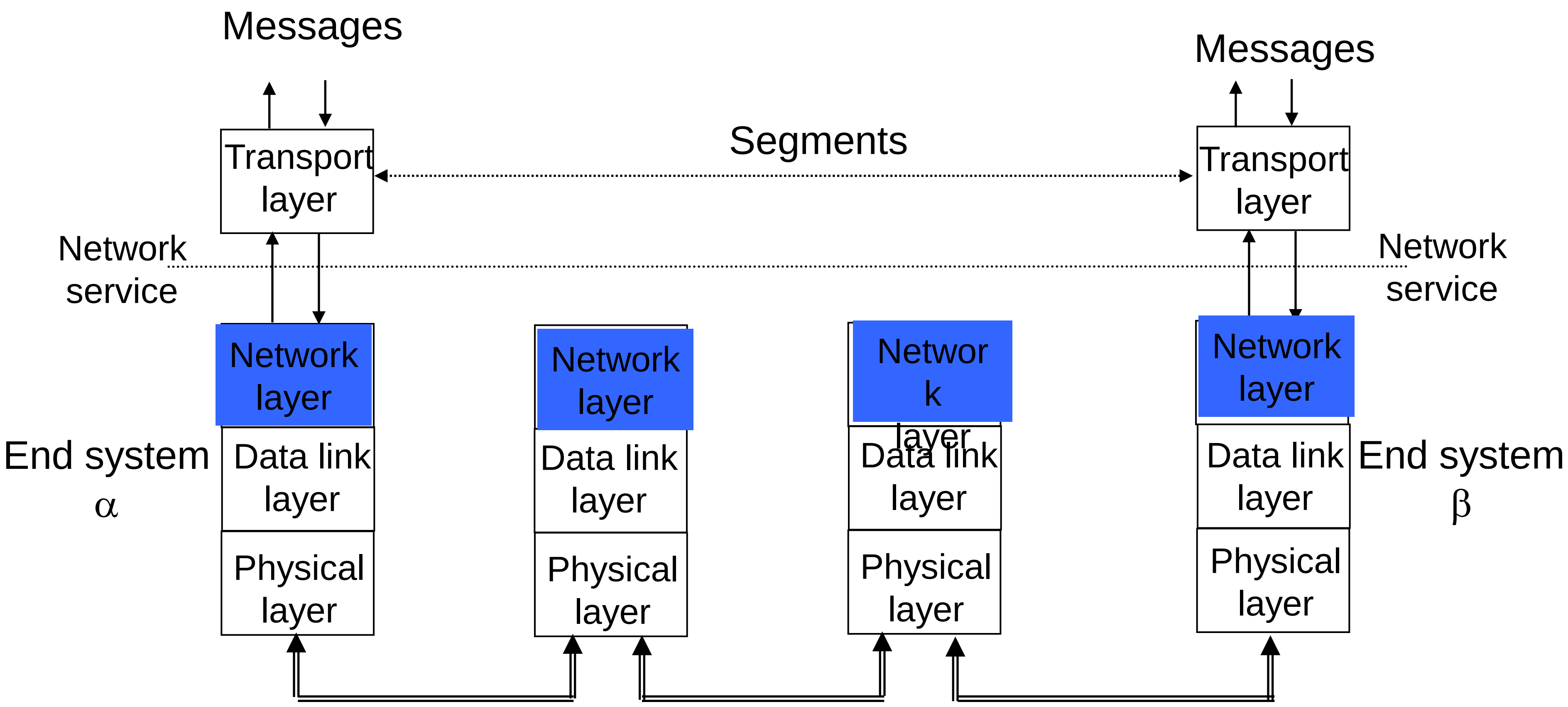
Network Layer

- Concerned with getting packets from source to destination.
- The network layer must know the topology of the subnet and choose appropriate paths through it.
- When source and destination are in *different networks*, the network layer **(IP)** must deal with these differences.

* **Key issue:** *what service does the network layer provide to the transport layer* (connection-oriented or connectionless).

Network Layer Design Goals

1. The services provided by the network layer should be independent of the subnet topology.
2. The Transport Layer should be shielded from the number, type and topology of the subnets present.
3. The network addresses available to the Transport Layer should use a uniform numbering plan (even across LANs and WANs).



Leon-Garcia & Widjaja: *Communication Networks*

Figure 7.2

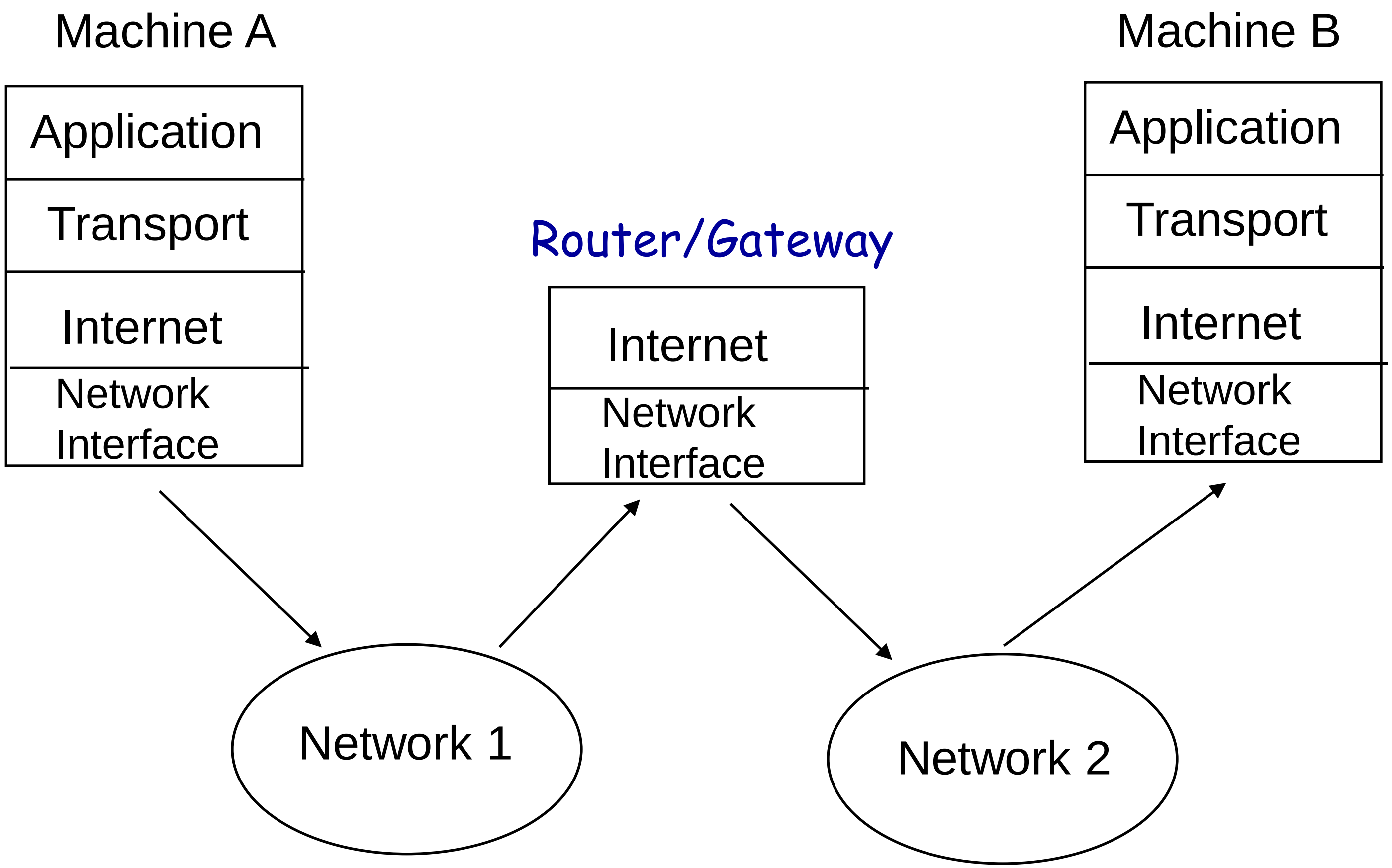
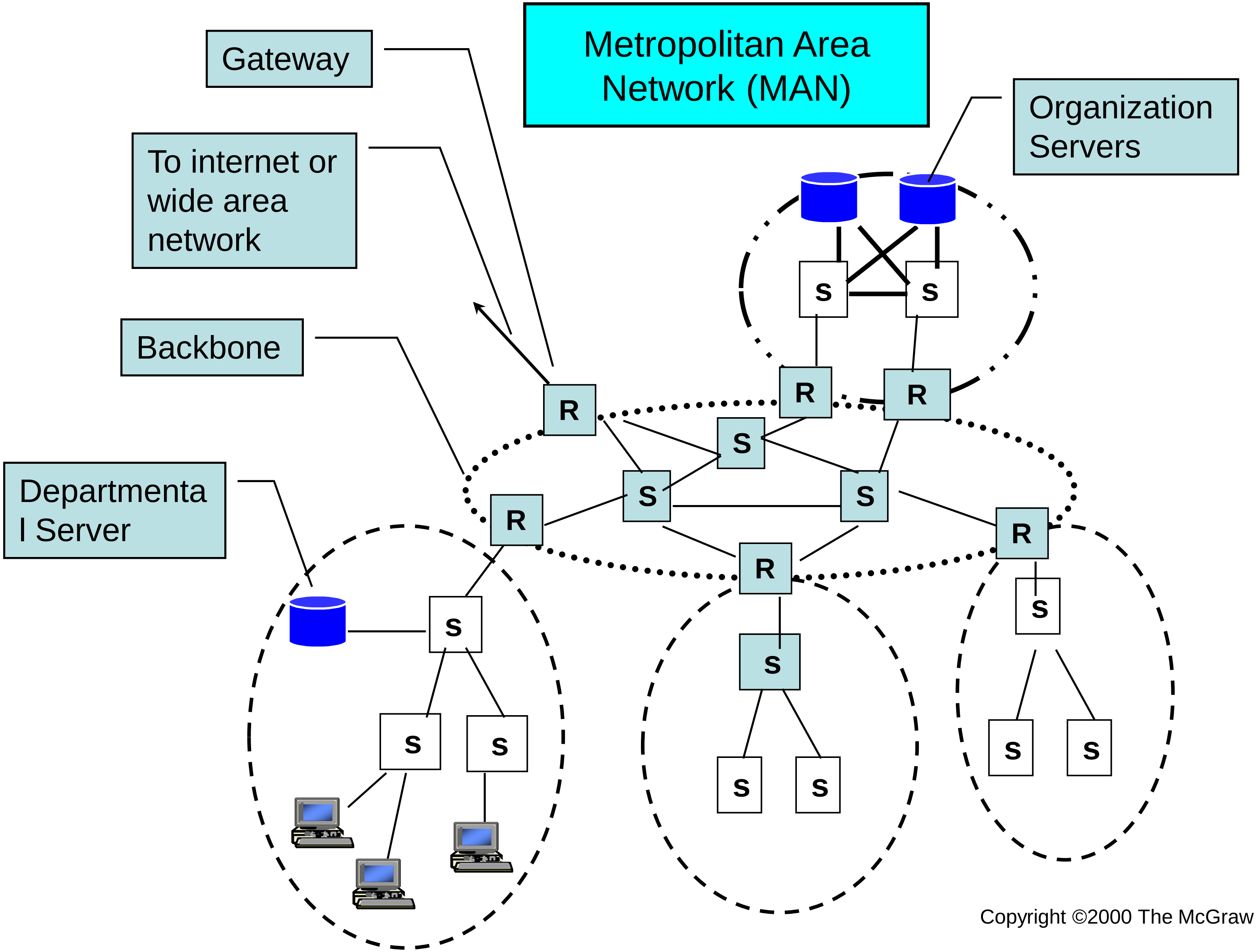

Machine A
Application
Transport
Internet
Network Interface
Router/Gateway
Internet
Network Interface
Machine B
Application
Transport
Internet
Network Interface
Network 1
Network 2

Figure 8.3

Metropolitan Area Network (MAN)



Gateway

To internet or wide area network

Backbone

Departmenta l Server

Organization Servers

Leon-Garcia & Widjaja: *Communication Networks*

Figure 7.6

Wide Area Network (WAN)

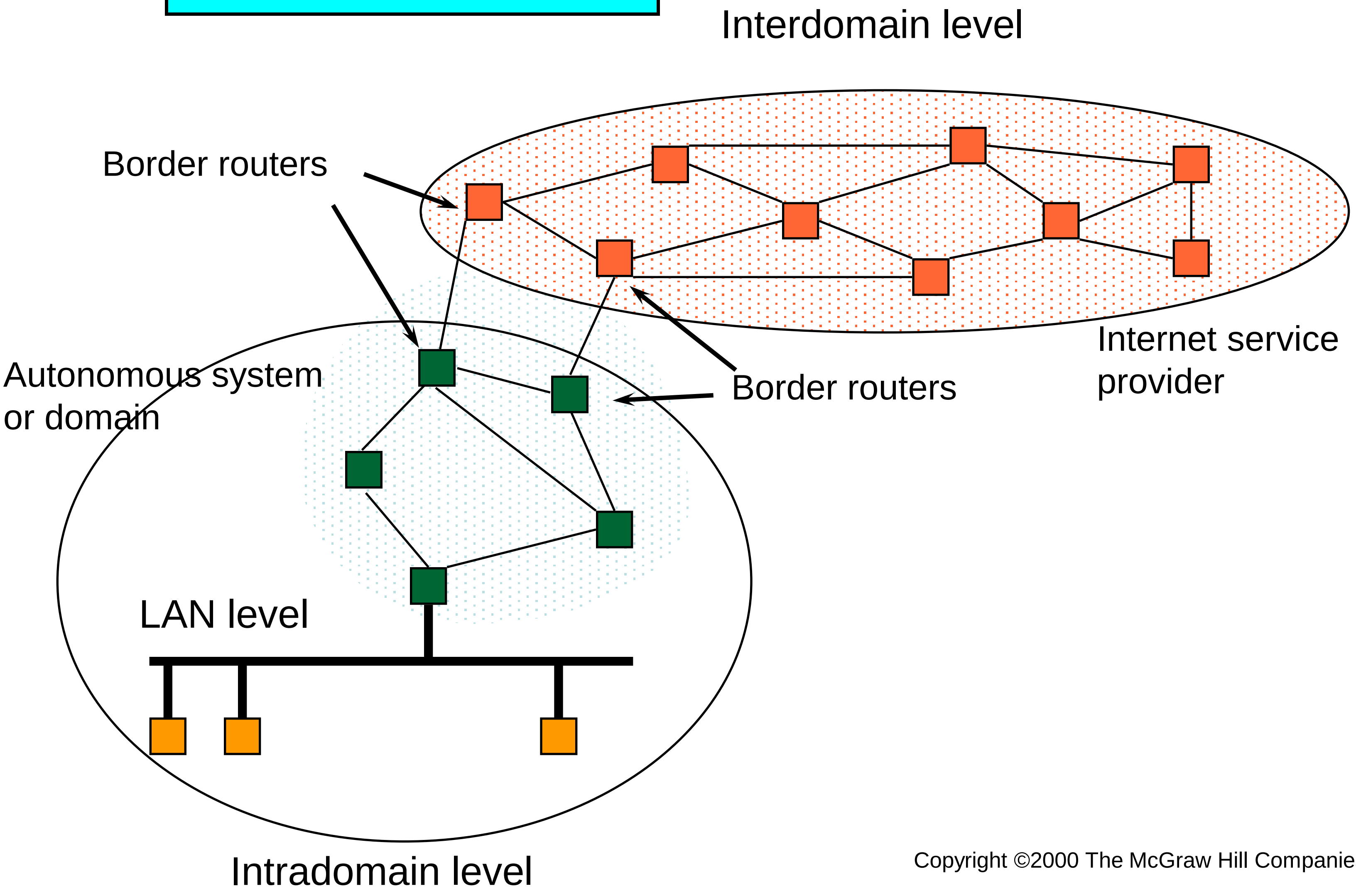


Figure 7.7

National ISPs

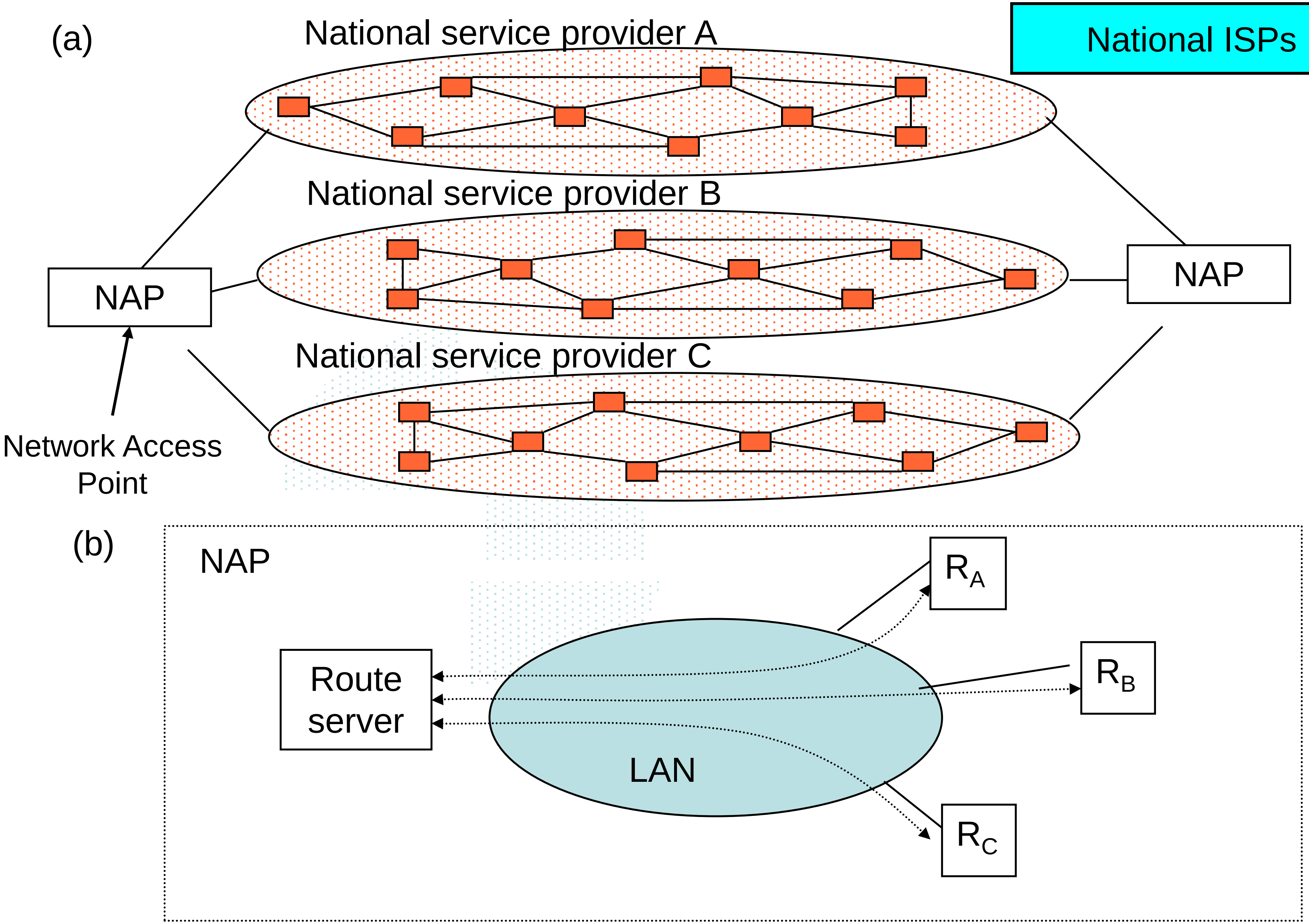
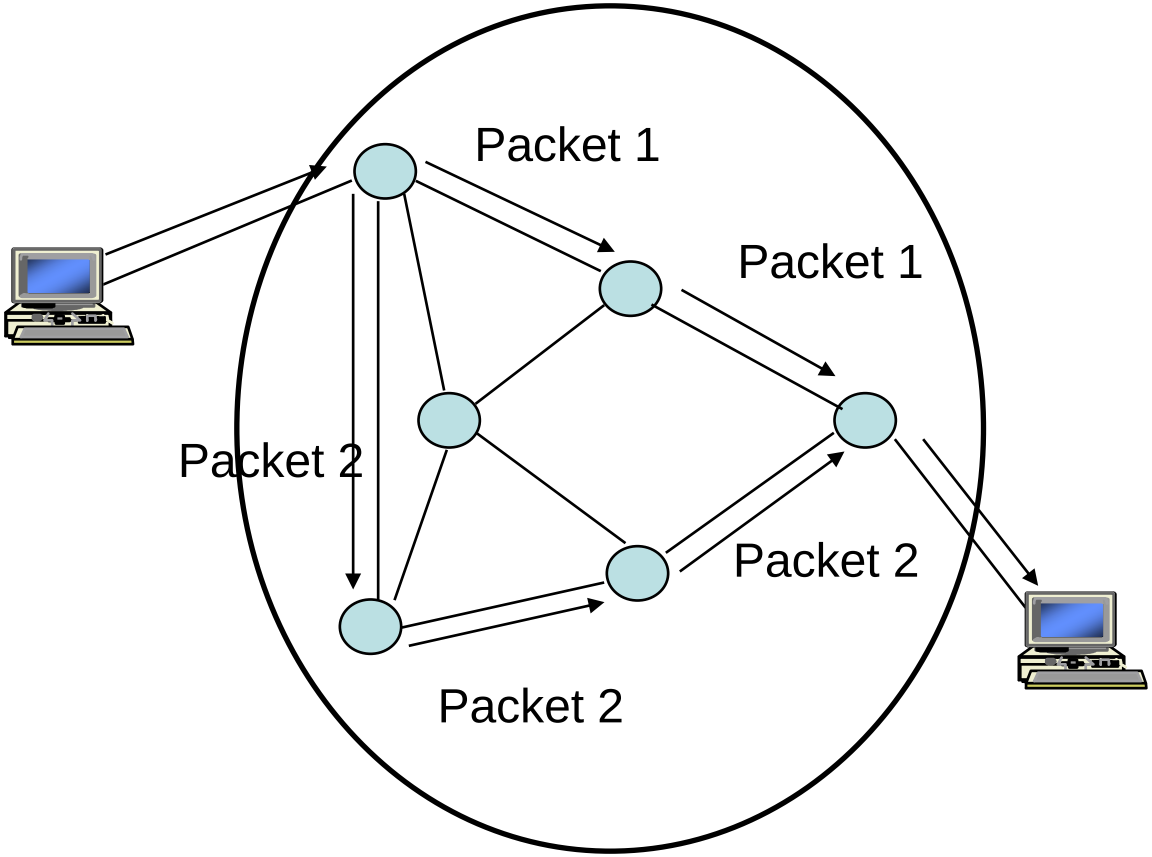

(a)
National service provider A
National service provider B
National service provider C
NAP
NAP
Network Access Point
(b)
NAP
R_A
R_B
Route server
LAN
R_C

Figure 7.8

Datagram Packet Switching



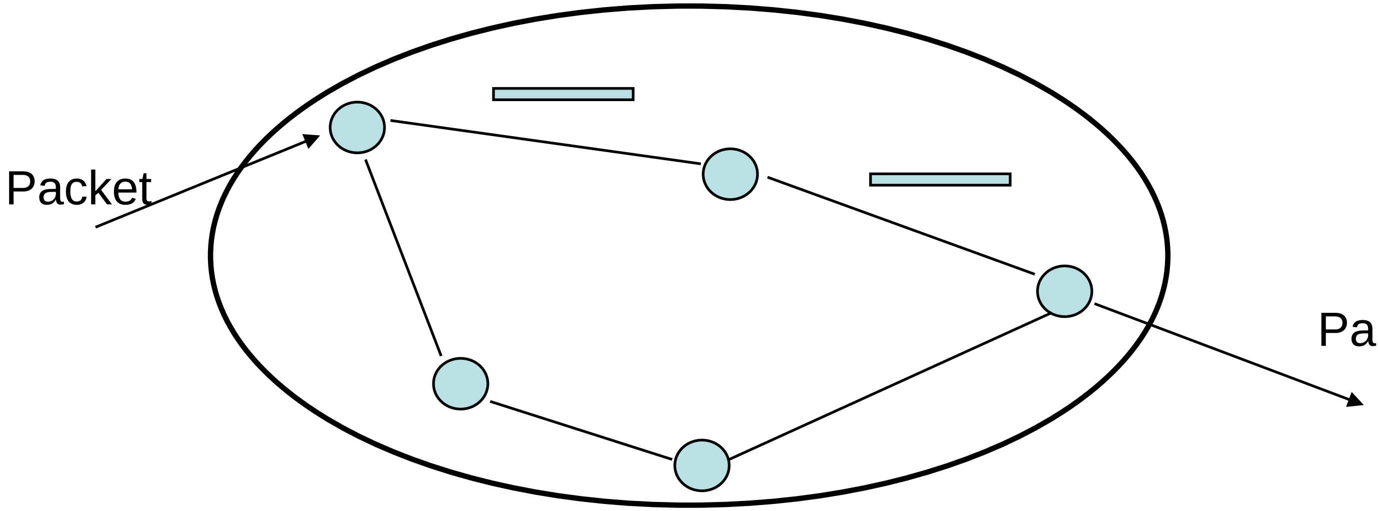
Copyright ©2000 The McGraw Hill Companies   Leon-Garcia & Widjaja: *Communication Networks*   Figure 7.15

Routing Table
in Datagram Network

Destination address	Output port
0785	7
1345	12
1566	6
2458	12

Figure 7.16

Virtual Circuit Packet Switching



Leon-Garcia & Widjaja: *Communication Networks* Figure 7.17

Routing Table
in Virtual Circuit Network

Entry for packets with identifier 15 →

Identifier	Output port	Next identifier
12	13	44
15	15	23
27	13	16
58	7	34

Figure 7.21

Routing

Routing algorithm:: that part of the Network Layer responsible for deciding on which output line to transmit an incoming packet.

- Remember: For virtual circuit subnets the routing decision is made ONLY at set up.

Algorithm properties:: correctness, simplicity, robustness, stability, fairness, optimality, and scalability.

Routing Classification

Adaptive Routing

- based on current measurements

  of traffic and/or topology.

1. centralized
2. isolated
3. distributed

Non-Adaptive Routing

- routing computed in advance and off-line

1. flooding
2. static routing using shortest path algorithms

Flooding

- *Pure flooding* :: every incoming packet to a node is sent out on every outgoing line.
  - Obvious adjustment – do not send out on arriving link (assuming full-duplex links).
  - The routing algorithm can use a hop counter (e.g., TTL) to dampen the flooding.
  - *Selective flooding* :: only send on those lines going “approximately” in the right direction.

Shortest Path Routing

1. Bellman-Ford Algorithm [Distance Vector]
2. Dijkstra's Algorithm [Link State]

What does it mean to be the shortest (or optimal) route?

Choices:

a. Minimize the number of hops along the path.

b. Minimize mean packet delay.

c. Maximize the network throughput.

Metrics

- Set all link costs to 1.
  - Shortest hop routing.
  - Disregards delay.
- Measure the number of packets queued.
  - Did not work well.
- Timestamp ArrivalTime and DepartTime* and use link-level ACK to compute:

  Delay = (DepartTime - ArrivalTime) + TransmissionTime + Latency

* Reset after retransmission

Metrics

  - Unstable under heavy link load.
  - Difficulty with granularity of the links.
- Revised ARPANET routing metric:
  - Compress dynamic range of the metric
  - Account for link type
  - Smooth variation of metric with time:
    - Delay transformed into link utilization
    - Utilization was averaged with last reported utilization.
    - Hard limit set on how much the metric could change per measurement cyle.

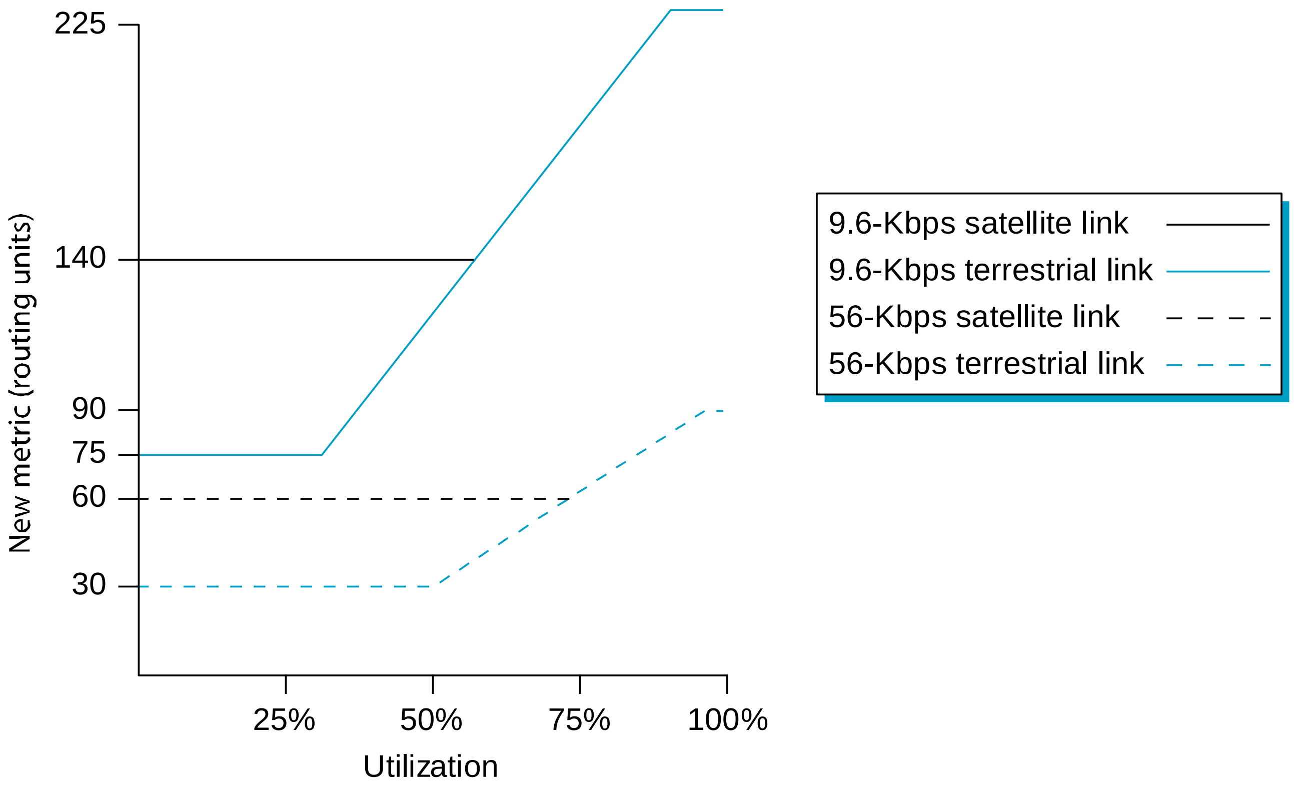


Figure 4.22 Revised ARPANET routing metric versus link utilization

P&D slide

Dijkstra’s Shortest Path Algorithm

Initially mark all nodes (except source) with infinite distance.

working node = source node

Sink node = destination node

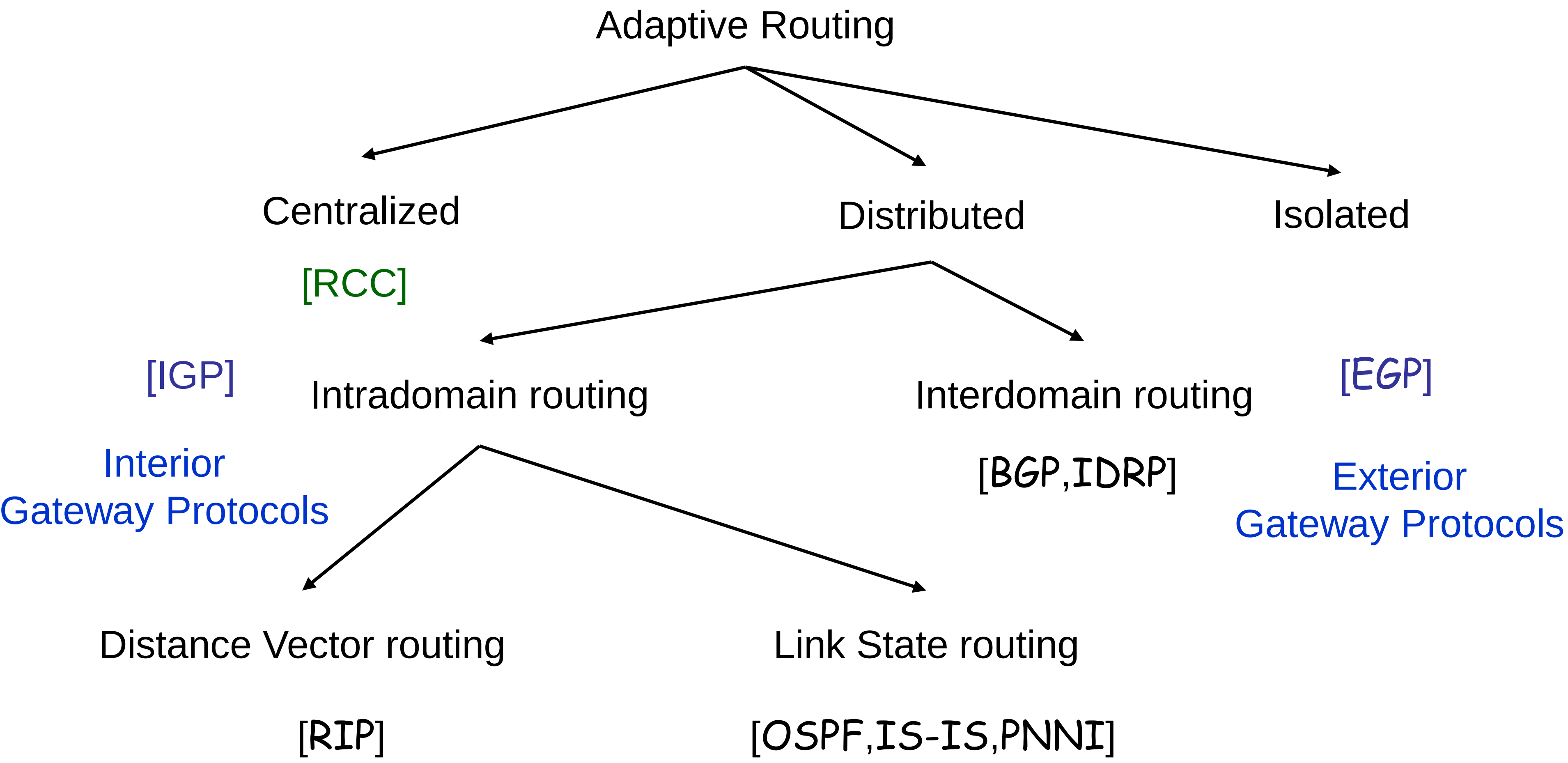
While the working node is not equal to the sink

1. Mark the working node as permanent.
2. Examine all adjacent nodes in turn

   If the sum of label on working node plus distance from working node to adjacent node is less than current labeled distance on the adjacent node, this implies a shorter path. Relabel the distance on the adjacent node and label it with the node from which the probe was made.

3. Examine all tentative nodes (not just

Internetwork Routing [Halsall]



Adaptive Routing

Centralized

[RCC]

Distributed

Isolated

[IGP]

Intradomain routing

Interior
Gateway Protocols

Interdomain routing

[EGP]

[BGP,IDRP]

Exterior
Gateway Protocols

Distance Vector routing

[RIP]

Link State routing

[OSPF,IS-IS,PNNI]

Adaptive Routing

Basic functions:

1. Measurement of pertinent network data.
2. Forwarding of information to where the routing computation will be done.
3. Compute the routing tables.
4. Convert the routing table information into a routing decision and then dispatch the data packet.

Adaptive Routing

Design Issues:

1. How much overhead is incurred due to gathering the routing information and sending ***routing packets?***
2. What is the time frame (i.e, the frequency) for sending ***routing packets*** in support of adaptive routing?
3. What is the complexity of the routing strategy?

Distance Vector Routing

- Historically known as the ***old*** ARPANET routing algorithm {or known as *Bellman-Ford algorithm*}.

Basic idea: each network node maintains a Distance Vector table containing the ***distance*** between itself and **ALL** possible destination nodes.

- Distances are based on a chosen metric and are computed using information from the **neighbors'** distance vectors.

Metric: usually hops or delay

Distance Vector Routing

Information kept by DV router

1. each router has an ID
2. associated with each link connected to a router, there is a link cost (static or dynamic) **the metric issue!**

Distance Vector Table Initialization

Distance to itself = 0

Distance to ALL other routers = infinity number

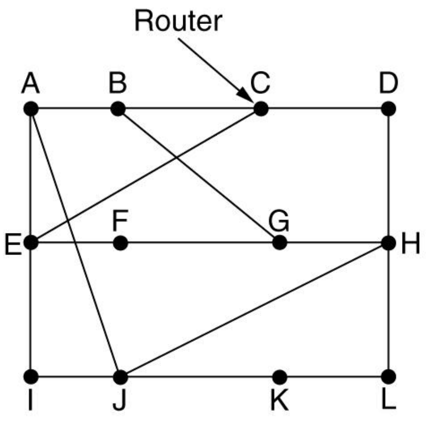
Distance Vector Algorithm

[Perlman]

1. Router transmits its *distance vector* to each of its neighbors.
2. Each router receives and saves the most recently received *distance vector* from each of its neighbors.
3. A router **recalculates** its distance vector when:
   a. It receives a *distance vector* from a neighbor containing different information than before.
   b. It discovers that a link to a neighbor has gone down (i.e., a topology change).

- The DV calculation is based on minimizing the cost to each destination.

Distance Vector Routing



(a)

To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

Vectors received from J's four neighbors

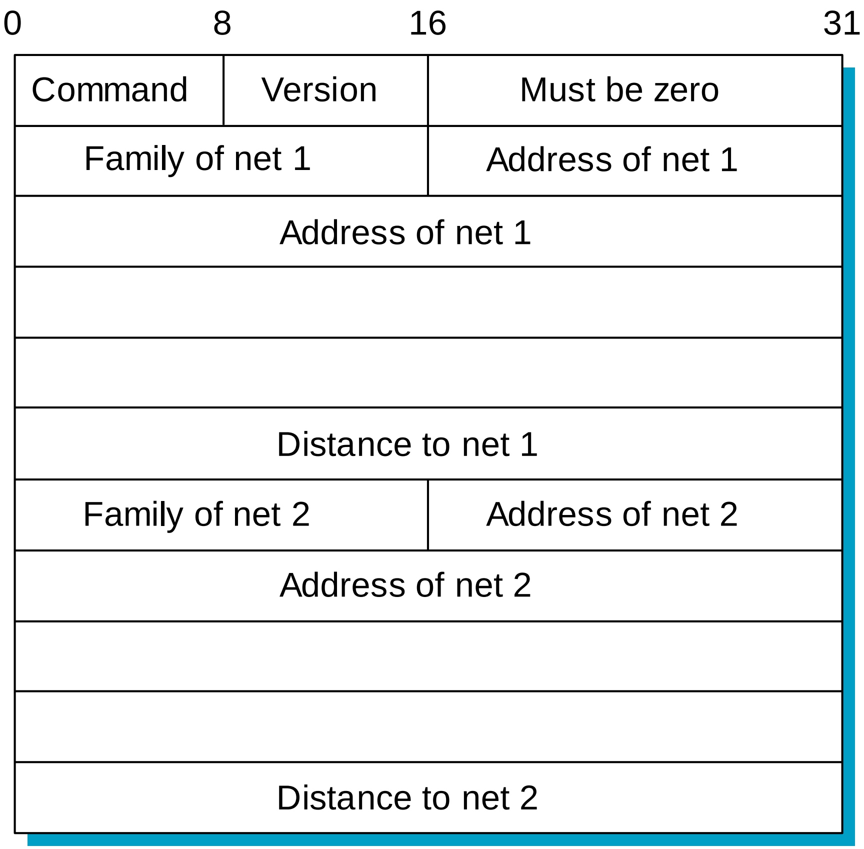
(b)

Figure 5-9.(a) A subnet. (b) Input from A, I, H, K, and the new routing table for J.

Routing Information Protocol (RIP)

- RIP had widespread use because it was distributed with BSD Unix in *"routed", a router management daemon.*
- **RIP** is the most used Distance Vector protocol.
- RFC1058 in June 1988.
- Sends packets every 30 seconds or faster.
- Runs over UDP.
- Metric = hop count
- BIG problem is max. hop count =16

  ➔ RIP limited to running on small networks!!
- Upgraded to RIPv2



0 8 16 31

Command	Version	Must be zero
Family of net 1		Address of net 1
Address of net 1		
Distance to net 1		
Family of net 2		Address of net 2
Address of net 2		
Distance to net 2		

(network_address, distance) pairs

Figure 4.17 RIP Packet Format

P&D slide

Link State Algorithm

1. Each router is responsible for meeting its neighbors and learning their names.
2. Each router constructs a link state packet **(LSP)** which consists of a list of names and cost to reach each of its neighbors.
3. The **LSP** is transmitted to ***ALL other routers.*** Each router stores the most recently generated **LSP** from each other router.
4. Each router uses complete information on the network topology to compute the *shortest path route* to each destination node.

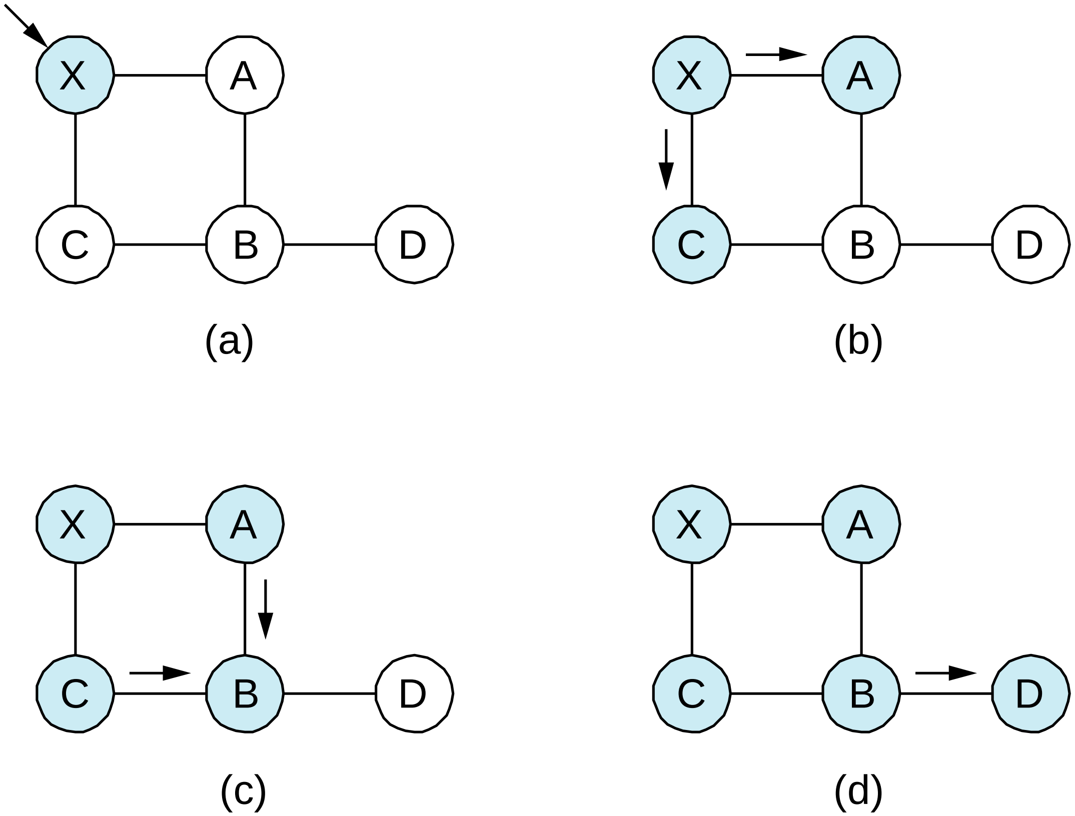


Figure 4.18 Reliable LSP Flooding

P&D slide

Reliable Flooding

- The process of making sure all the nodes participating in the routing protocol get a copy of the link-state information from all the other nodes.
- LSP contains:
  - Sending router’s node ID
  - List connected neighbors with the associated link cost to each neighbor
  - Sequence number
  - Time-to-live

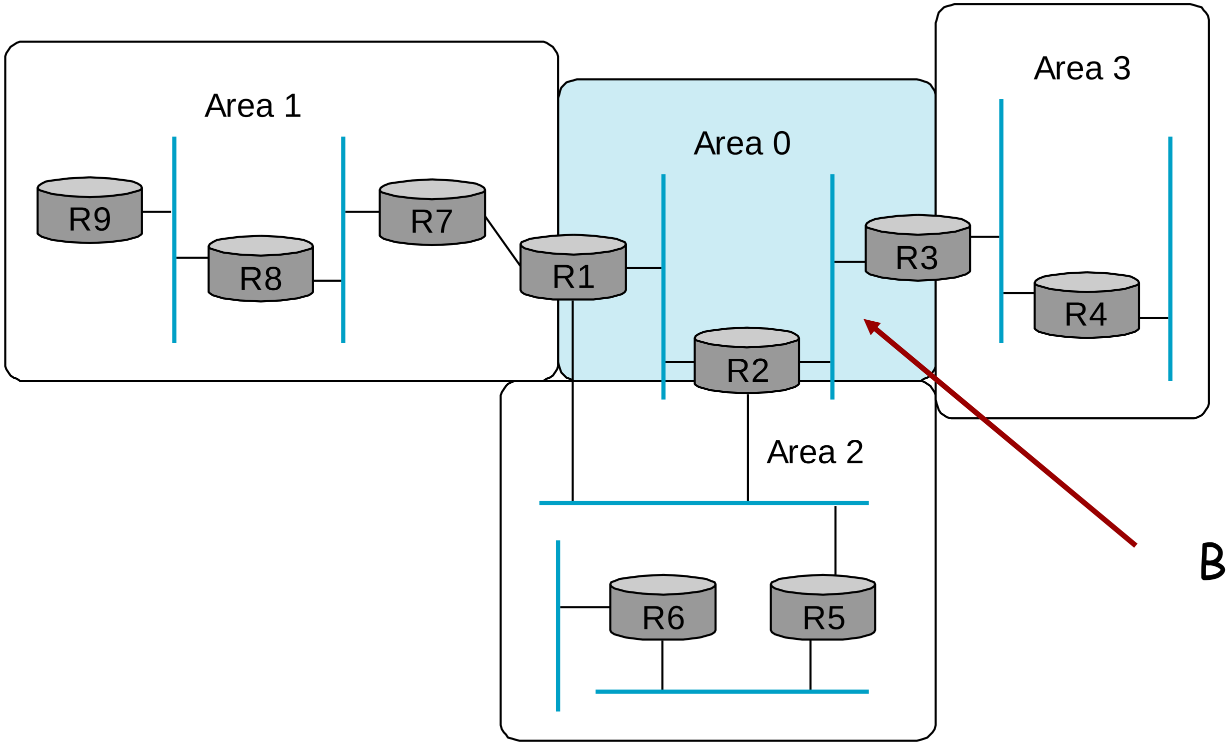
Reliable Flooding

- First two items enable route calculation
- Last two items make process reliable
  - – ACKs and checking for duplicates is needed.
- Periodic Hello packets used to determine the demise of a negihbor
- The sequence numbers are not expected to wrap around.
  - – => field needs to be large (64 bits)

Open Shortest Path First (OSPF)

- Provides for authentication of routing messages.
  - 8-byte password designed to avoid misconfiguration.
- Provides additional hierarchy
  - Domains are partitioned into *areas.*
  - This reduces the amount of information transmitted in packet.
- Provides load-balancing via multiple routes.

Open Shortest Path First (OSPF)



Backbone area

Figure 4.32 A Domain divided into Areas

P&D slide

Open Shortest Path First (OSPF)

- OSPF runs *on top of* IP, i.e., an OSPF packet is transmitted with IP data packet header.
- Uses Level 1 and Level 2 routers
- Has: backbone routers, area border routers, and AS boundary routers
- LSPs referred to as **LSAs (Link State Advertisements)**
- Complex algorithm due to **five** distinct LSA types.

OSPF Terminology

Internal router :: a level 1 router.

Backbone router :: a level 2 router.

Area border router (ABR) :: a backbone router that attaches to more than one area.

AS border router :: (an interdomain router), namely, a router that attaches to routers from other ASs across AS boundaries.

OSPF LSA Types

1. Router link advertisement [Hello message]
2. Network link advertisement
3. Network summary link advertisement
4. AS border router’s summary link advertisement
5. AS external link advertisement

<table>
<tr><td colspan="4">LS Age</td><td colspan="2">Options</td><td colspan="2">Type=1</td></tr>
<tr><td colspan="8">Link-state ID</td></tr>
<tr><td colspan="8">Advertising router</td></tr>
<tr><td colspan="8">LS sequence number</td></tr>
<tr><td colspan="4">LS checksum</td><td colspan="4">Length</td></tr>
<tr><td>0</td><td>Flags</td><td colspan="2">0</td><td colspan="4">Number of links</td></tr>
<tr><td colspan="8">Link ID</td></tr>
<tr><td colspan="8">Link data</td></tr>
<tr><td colspan="2">Link type</td><td colspan="2">Num_TOS</td><td colspan="4">Metric</td></tr>
<tr><td colspan="8">Optional TOS information</td></tr>
<tr><td colspan="8">More links</td></tr>
</table>

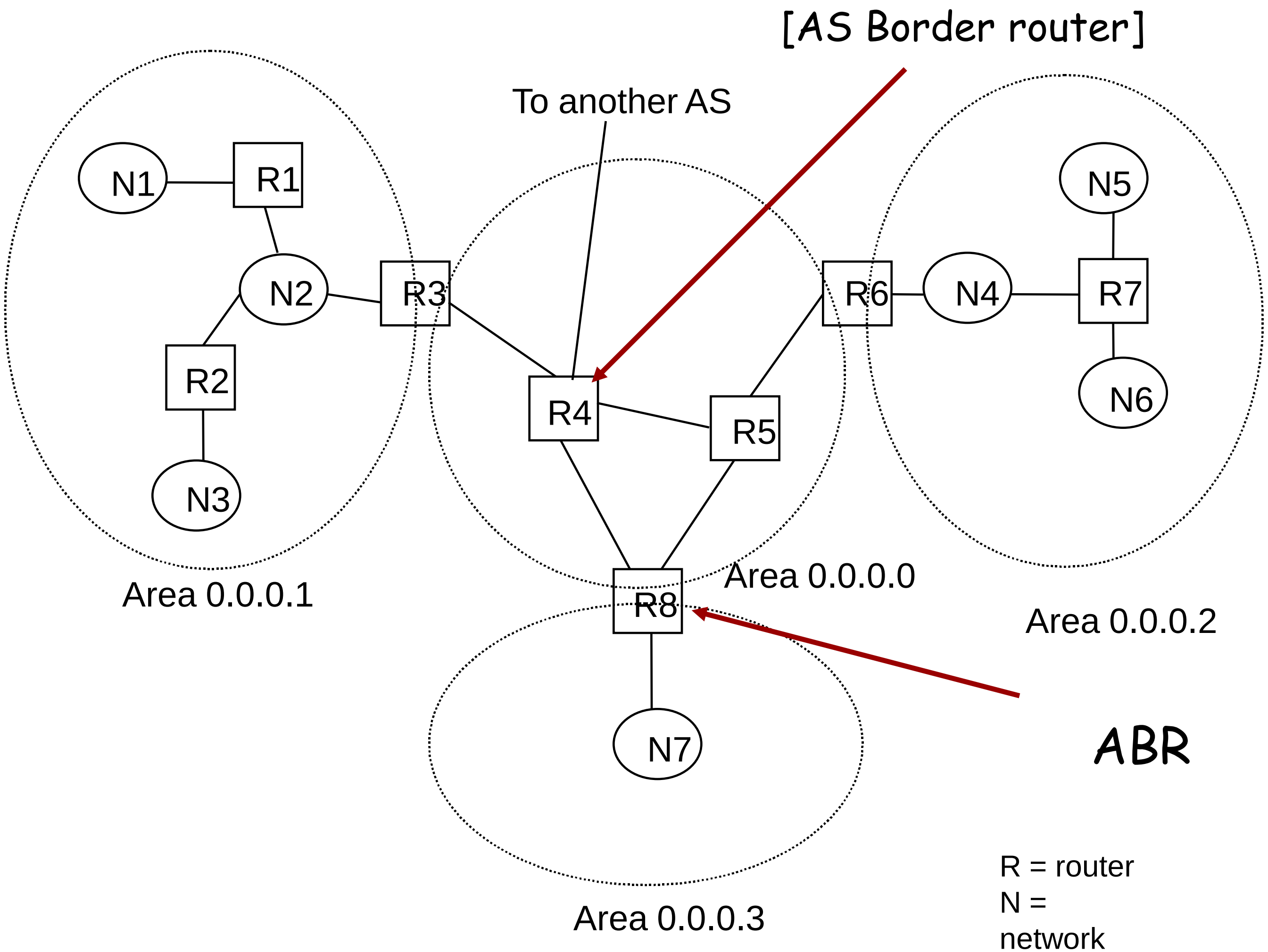
Indicates LSA type

Indicates link cost

Figure 4.21 OSF Type 1 Link-State Advertisement

P&D slide

OSPF Areas



R = router
N = network

Leon-Garcia & Widjaja: *Communication Networks*

Figure 8.33

OSPF

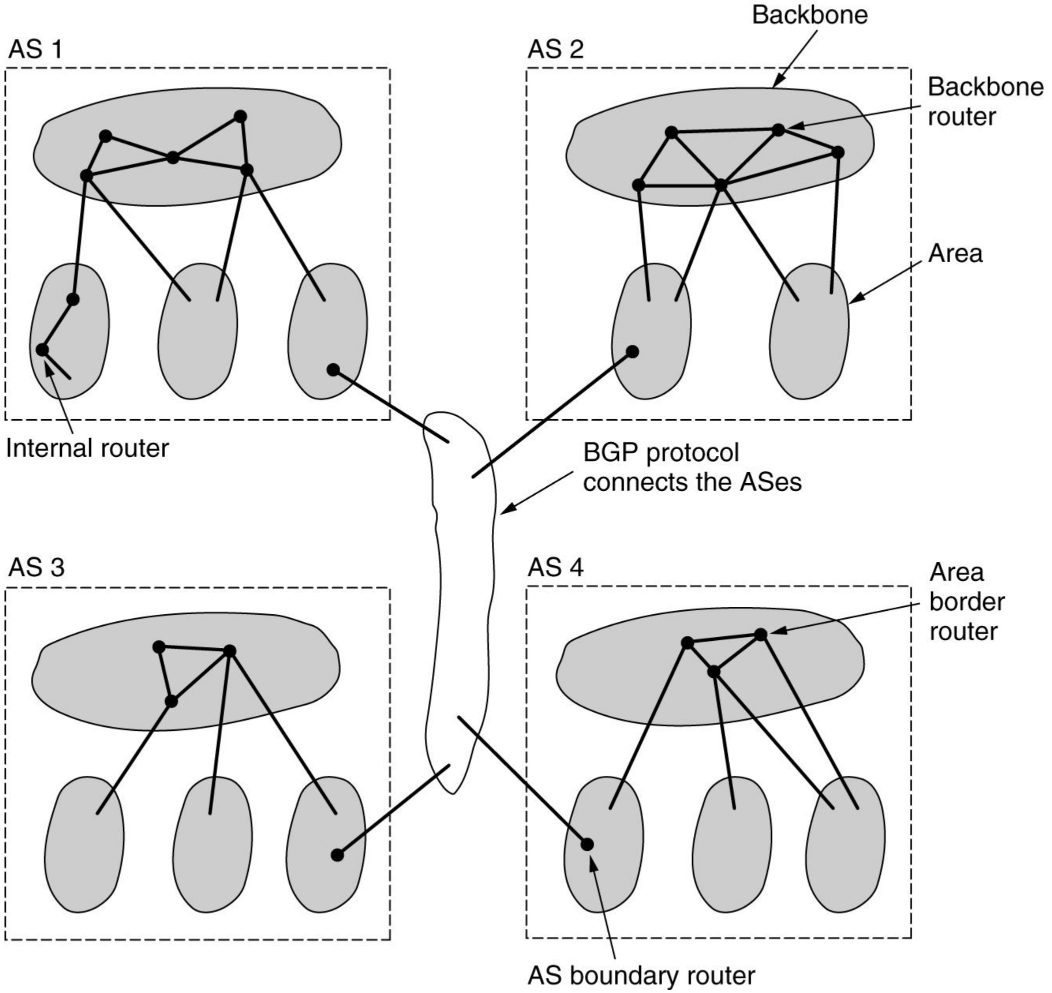


Figure 5-65.The relation between ASes, backbones, and areas in OSPF.

Border Gateway Protocol (BGP)

- The replacement for EGP is BGP. Current version is BGP-4.
- BGP assumes the Internet is an arbitrary interconnected set of AS's.
- In *interdomain* routing the goal is to find ANY path to the intended destination that is loop-free. The protocols are more concerned with **reachability** than optimality.

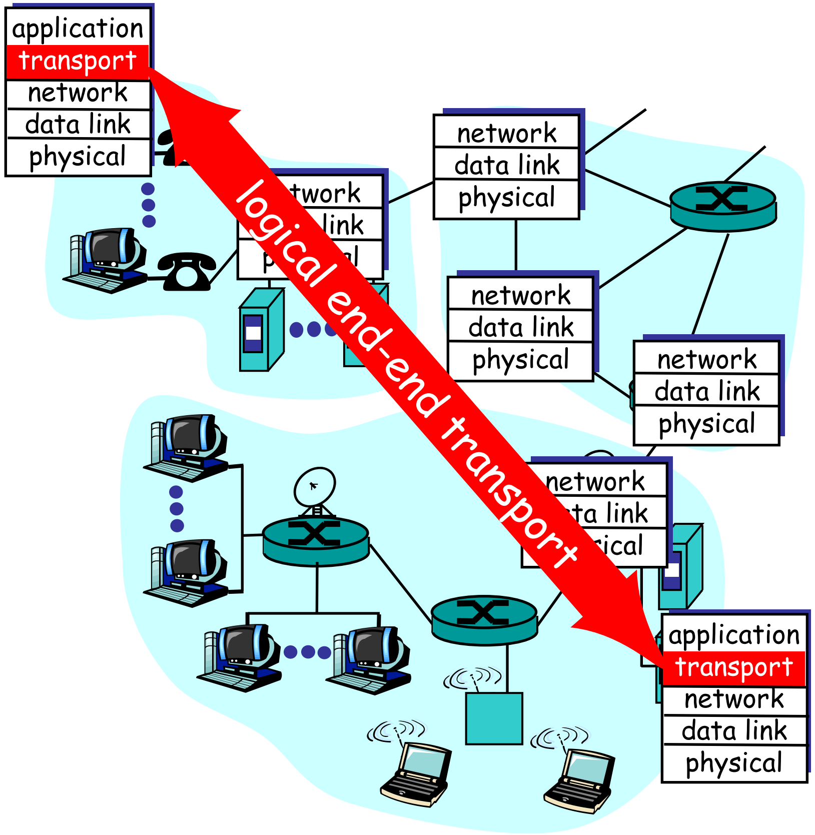
UNIT-4

The Transport layer

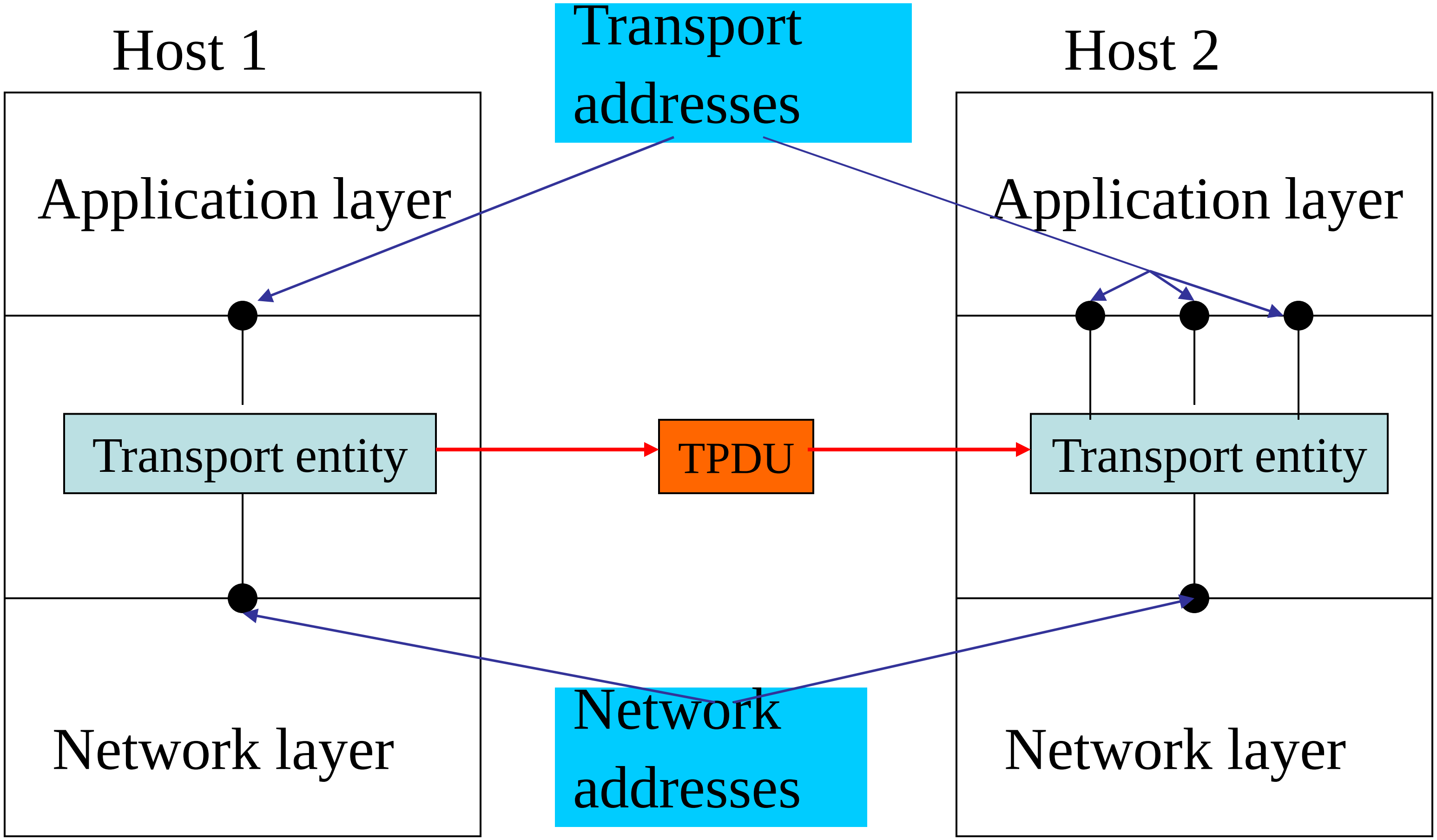
Transport Layer

- Services
- Elements of transport protocol
- Simple transport protocol
- UDP
- Remote Procedure Call (see Distributed Systems)
- TCP

Layer overview



Layer overview



Services

- To upper layer
  - efficient, reliable, cost-effective service
  - 2 kinds
    - Connection oriented
    - Connectionless

Services

- needed from network layer
  - packet transport between hosts
  - relationship network <> transport
    - Hosts <> processes
    - Transport service
      - independent network
      - more reliable
    - Network
      - run by carrier
      - part of communication subnet for WANs

Simple service: primitives

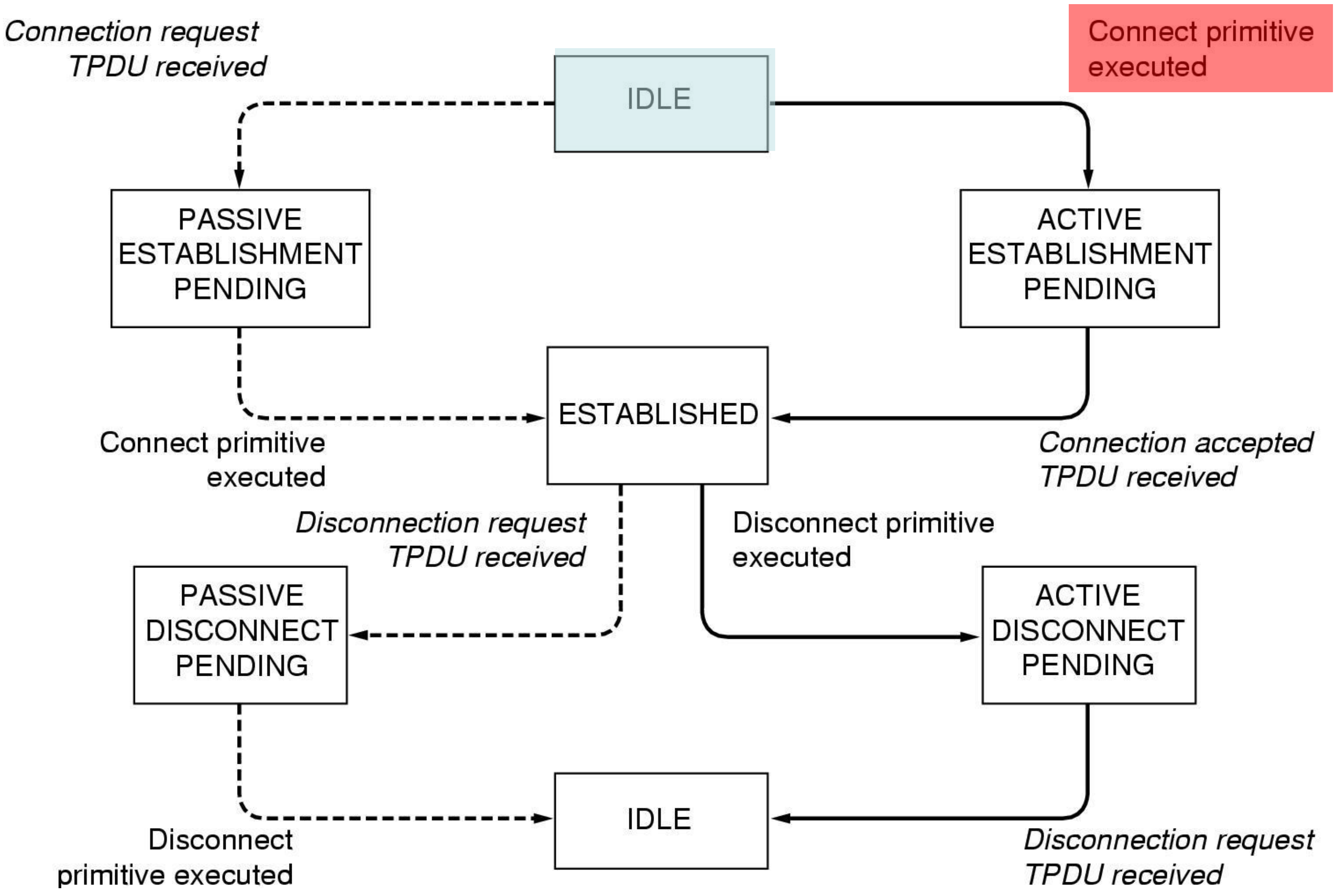
- Simple primitives:
  - connect
  - send
  - receive
  - disconnect
- How to handle incoming connection request in server process?

  ➔Wait for connection request from client!
  - listen

Simple service: primitives

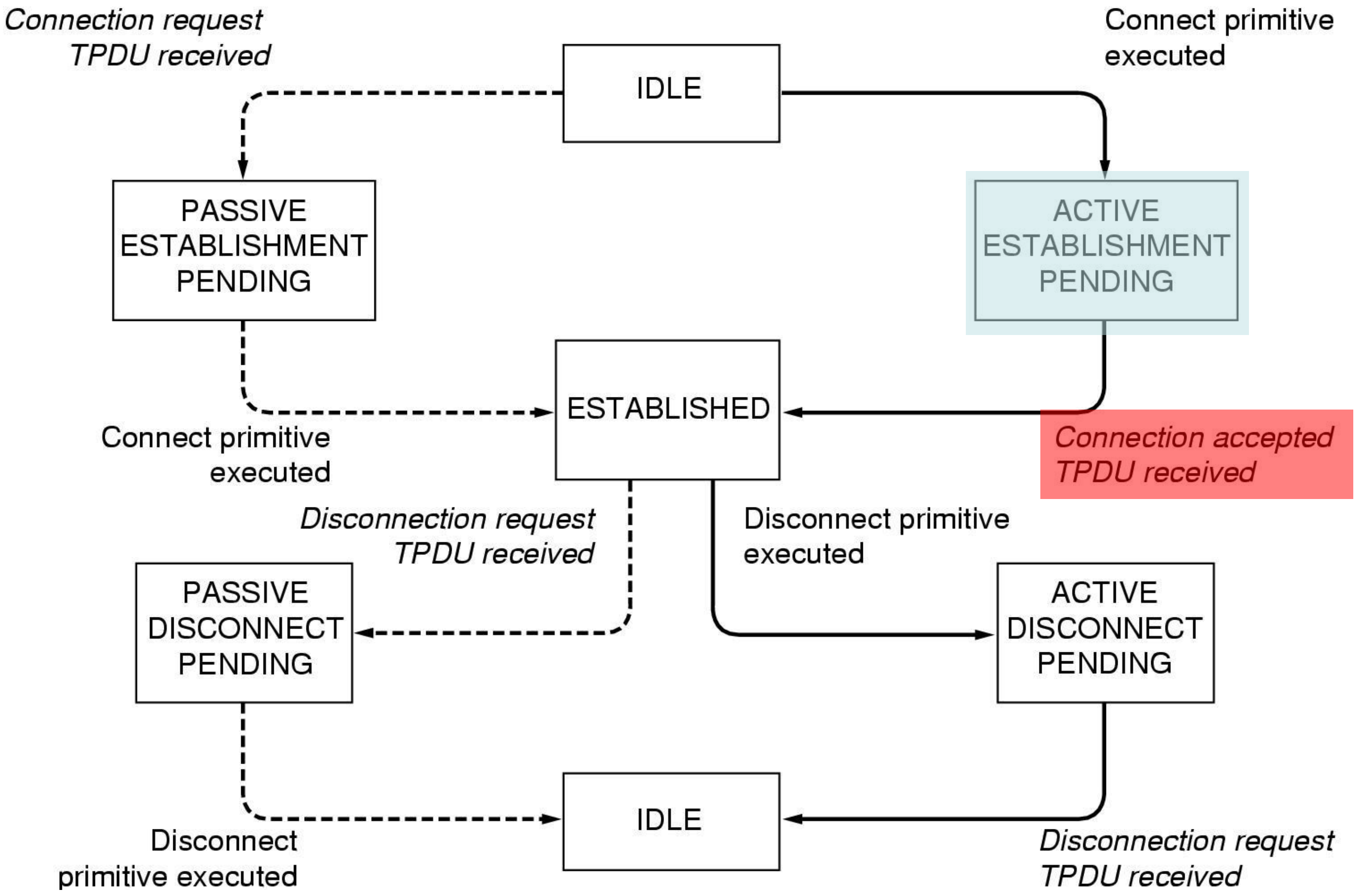
listen	No TPDU
connect	Connection Request TPDU
send	Data TPDU
receive	No TPDU
disconnect	Disconnect TPDU

Simple service: state diagram

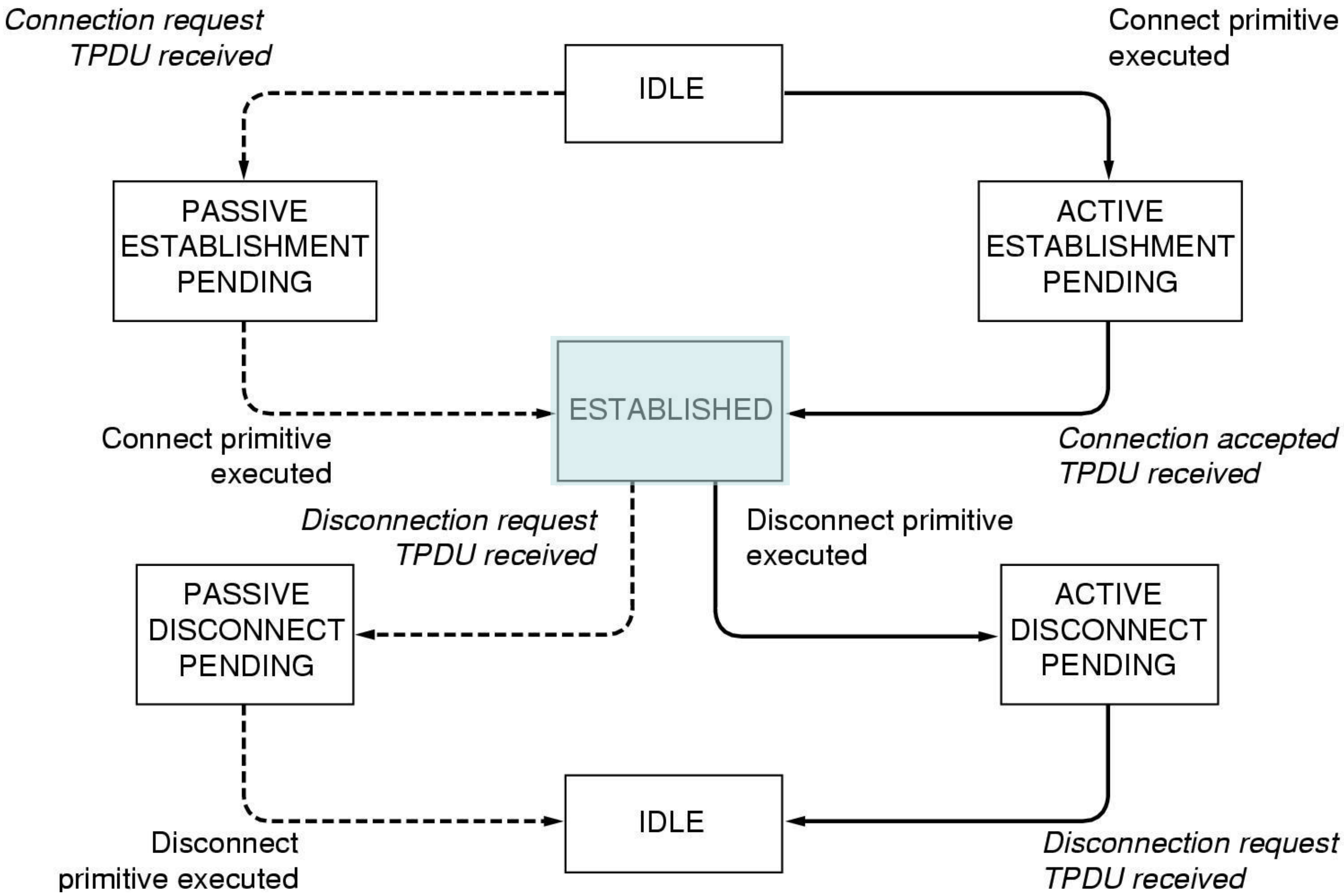


Connection request TPDU received
IDLE
Connect primitive executed
PASSIVE ESTABLISHMENT PENDING
ACTIVE ESTABLISHMENT PENDING
ESTABLISHED
Connect primitive executed
Connection accepted TPDU received
Disconnection request TPDU received
Disconnect primitive executed
PASSIVE DISCONNECT PENDING
ACTIVE DISCONNECT PENDING
IDLE
Disconnect primitive executed
Disconnection request TPDU received

Simple service: state diagram


Connection request TPDU received
IDLE
Connect primitive executed
PASSIVE ESTABLISHMENT PENDING
ACTIVE ESTABLISHMENT PENDING
ESTABLISHED
Connect primitive executed
Connection accepted TPDU received
Disconnection request TPDU received
Disconnect primitive executed
PASSIVE DISCONNECT PENDING
ACTIVE DISCONNECT PENDING
IDLE
Disconnect primitive executed
Disconnection request TPDU received

Simple service: state diagram



Berkeley service primitives

- Used in Berkeley UNIX for TCP
- Addressing primitives:

socket

bind

- Server primitives:

listen

accept

send + receive

close

- Client primitives:

connect

send + receive

close

Berkeley service primitives

socket	create new communication end point
bind	attach a local address to a socket
listen	announce willingness to accept connections; give queue size
accept	block caller until a connection request arrives
connect	actively attempt to establish a connection
send	send some data over the connection
receive	receive some data from the connection
close	release the connection

Transport Layer

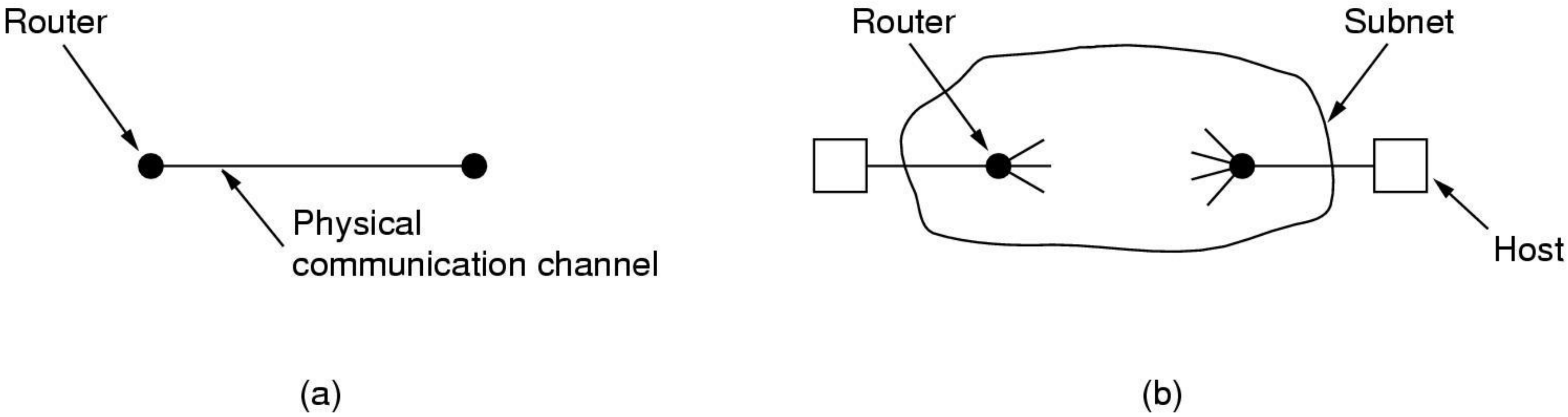
- Services
- Elements of transport protocol
- Simple transport protocol
- UDP
- Remote Procedure Call (see Distributed Systems)
- TCP

Elements of transport protocols *(etp)*

- Transport <> Data Link
- Addressing
- Establishing a connection
- Releasing a connection
- Flow control and buffering
- Multiplexing
- Crash recovery

etp: Transport <> data link

- Physical channel <> subnet



Explicit addressing

Connection establishment

Potential existence of storage capacity in subnet

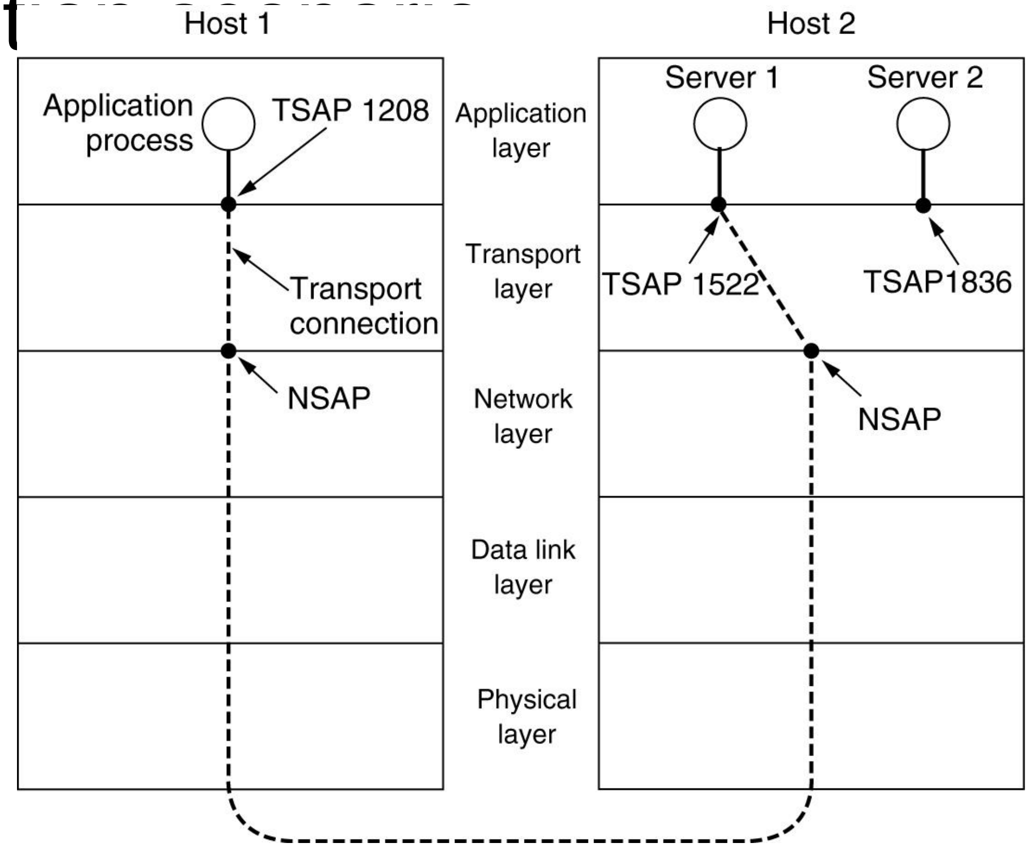
Dynamically varying number of connections

etp: Addressing

- TSAP = transport service access point
  - Internet: IP address + local port
  - ATM: AAL-SAPs
- *Connection scenario*
- *Getting TSAP addresses?*
- *From TSAP address to NSAP address?*

etp: Addressing

- Connection scenario



etp: Addressing

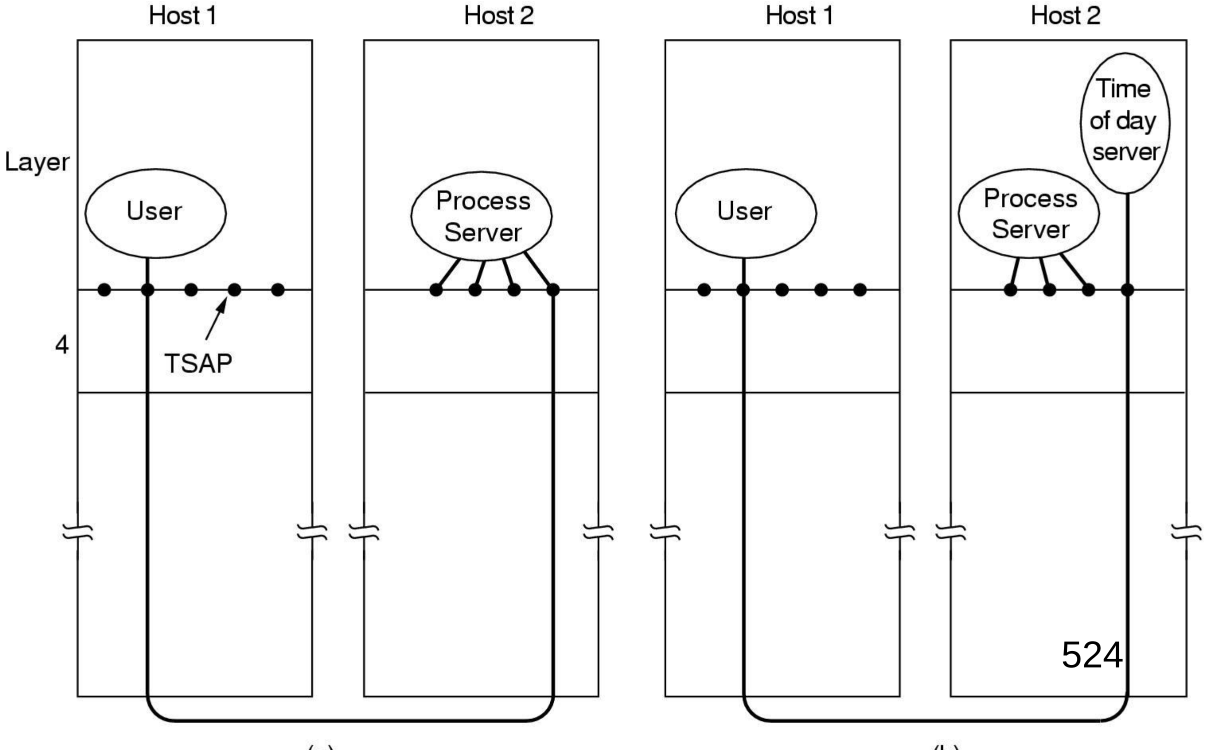
- Connection scenario
  - Host 2 (server)
    - Time-of-day server attaches itself to TSAP 1522
  - Host 1 (client)
    - Connect from TSAP 1208 to TSAP 1522
    - Setup network connection to host 2
    - Send transport connection request
  - Host 2
    - Accept connection request

etp: Addressing

- Getting TSAP addresses?
  - Stable TSAP addresses
    - For key services
    - Not for user processes
      - active for a short time
      - number of addresses limited
  - Name servers
    - to find existing servers
    - map service name into TSAP address
  - *Initial connection protocol*

etp: Addressing

- Getting TSAP addresses?
  - Initial connection protocol
    - to avoid many waiting servers ➔ one process server
      - waits on
      - creates r



etp: Addressing

- From TSAP address to NSAP address?
  - hierarchical addresses
    - address = <country> <network> <host> <port>
      - Examples: IP address + port
        Telephone numbers (<> number portability?)
    - Disadvantages:
      - TSAP bound to host!
  - flat address space
    - Advantages:
      - Independent of underlying network addresses
      - TSAP address not bound to host
    - Mapping to network addresses:
      - Name server
      - broadcast

etp: Establishing a connection

- Problem: delayed duplicates!
- Scenario:
  - – Correct bank transaction
    - connect
    - data transfer
    - disconnect
  - – Problem: same packets are received in same order a second

Recognized?

etp: Establishing a connection

- Unsatisfactory solutions:
  - throwaway TSAP addresses
    - need unlimited number of addresses?
    - process server solution impossible
  - connection identifier
    - Never reused!
      - ➔ Maintain state in hosts

- Satisfactory solutions

etp: Establishing a connection

- Satisfactory solutions
  - Ensure limited packet lifetime (incl. Acks)
  - Mechanisms
    - prevent packets from looping + bound congestion delay
    - hopcounter in each packet
    - timestamp in each packet
  - Basic assumption

Maximum packet lifetime T

If we wait a time T after sending a packet all traces of it (including Acks) are gone

etp: Establishing a connection

- Tomlinson’s method
  - requires: clock in each host
    - Number of bits > number of bits in sequence number
    - Clock keeps running, even when a hosts crashes
  - Basic idea:

2 identically numbered TPDUs are never outstanding at the same time!

etp: Establishing a connection

- Tomlin

Never reuse a sequence number x within the lifetime T for the packet with x

– Problems to solve

- Selection of the initial sequence number for a new connection
- Wrap around of sequence numbers for an active connection
- Handle host crashes

➔ Forbidden region

etp: Establishing a connection

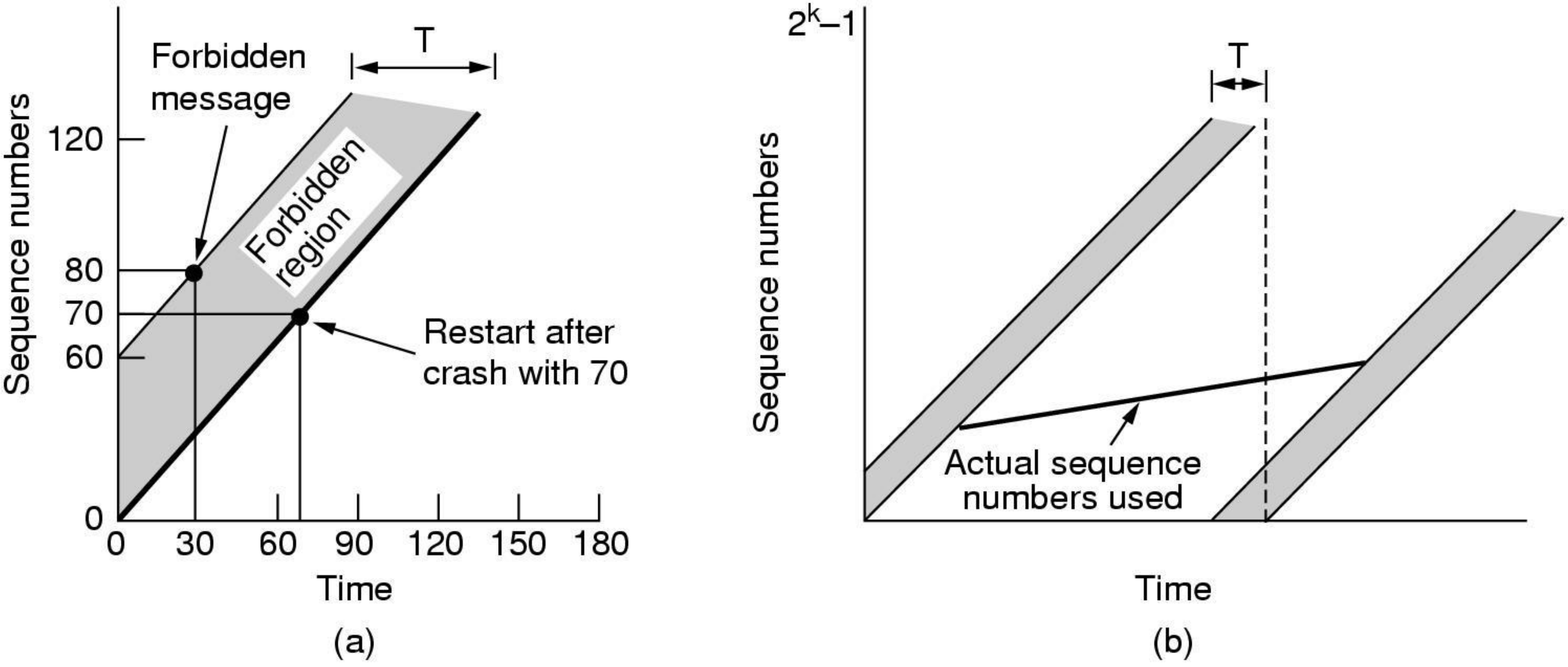
- Tomlinson’s method
  - Initial sequence number

    = lower order bits of clock
  - Ensure initial sequence numbers are always OK

    ➔ forbidden region
  - Wrap around
    - Idle
    - Resynchronize sequence numbers

etp: Establishing a connection

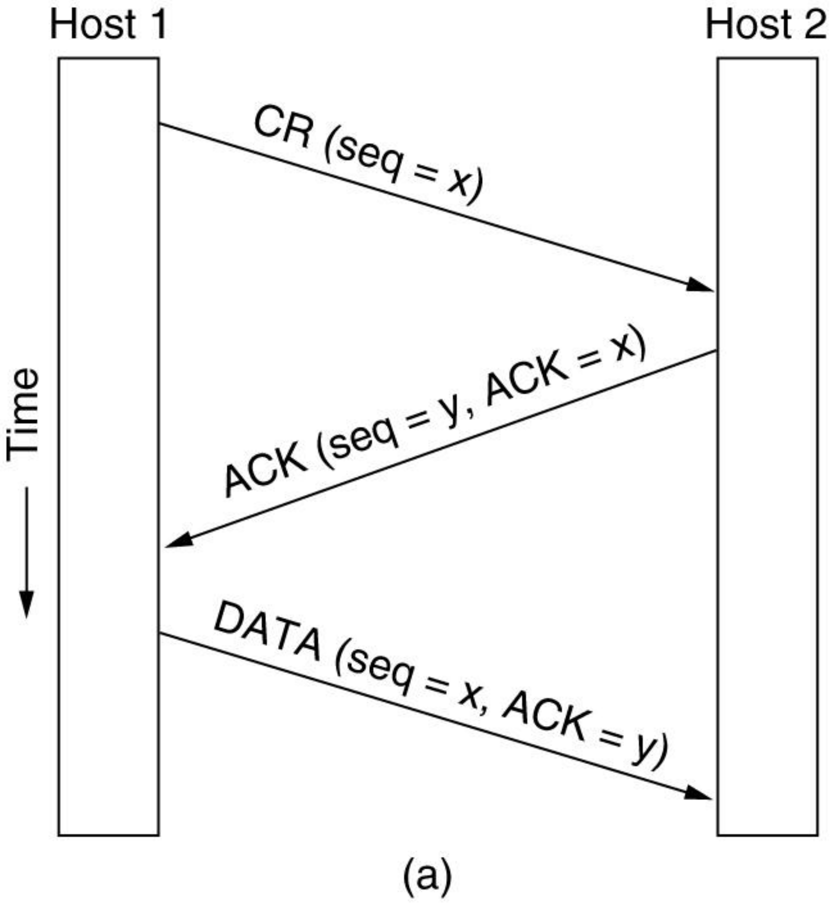
- Tomlinson - forbidden region



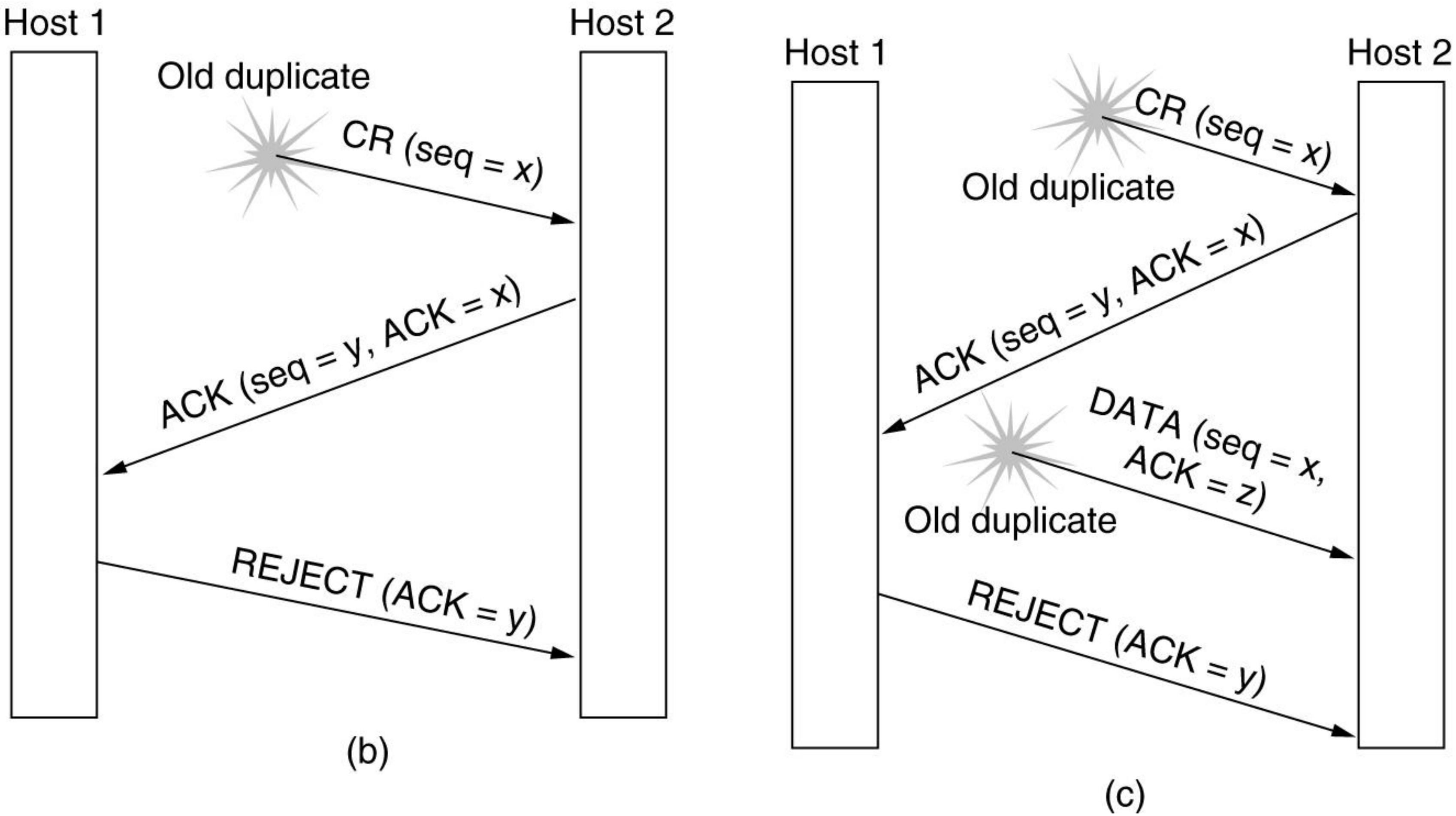
etp: Establishing a connection

- Tomlinson – thre

No combination of delayed packets can cause the protocol to fail



etp: Establishing a connection



(b)

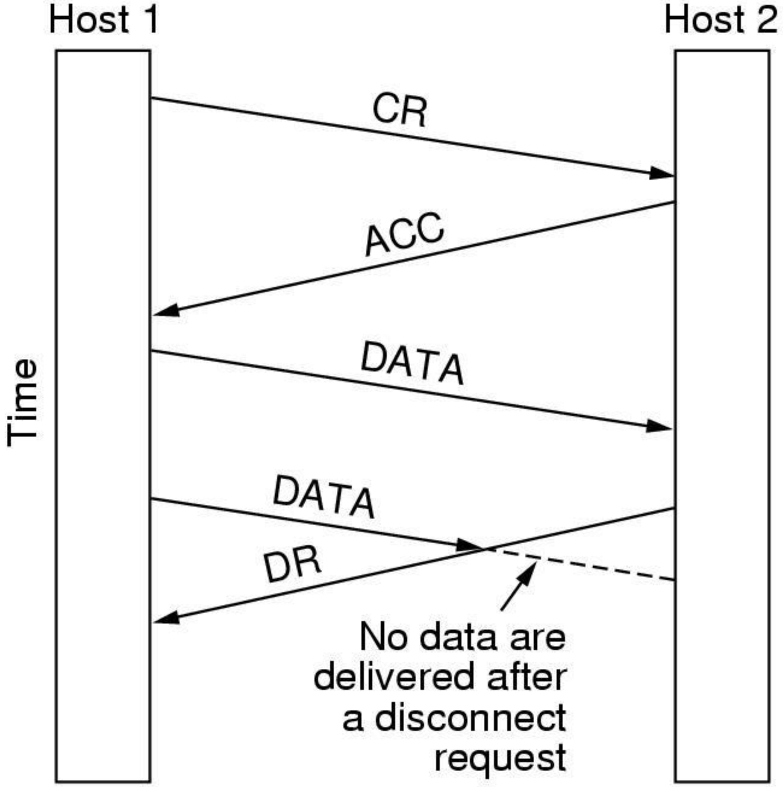
(c)

etp: Releasing a connection

- 2 styles:
  - Asymmetric
    - Connection broken when one party hangs up
    - Abrupt! ➔ may result in data loss
  - Symmetric
    - Both parties should agree to release connection
    - How to reach agreement? Two-army problem
    - Solution: three-way-handshake
  - Pragmatic approach
    - Connection = 2 unidirectional connections
    - Sender can close unidirectional connection

etp: Releasing a connection

- Asymmetric: data loss

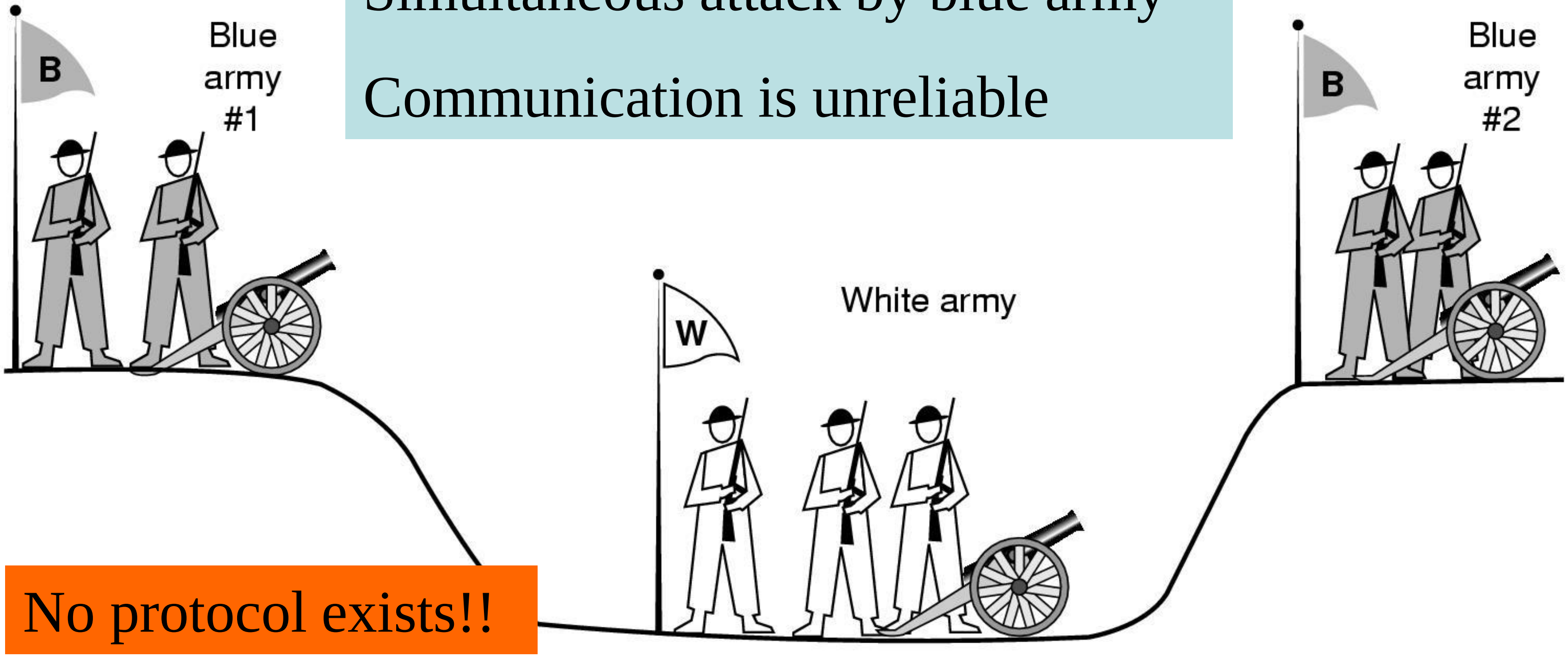


etp: Releasing a connection

- Symmetric: two-army-problem

Simultaneous attack by blue army

Communication is unreliable

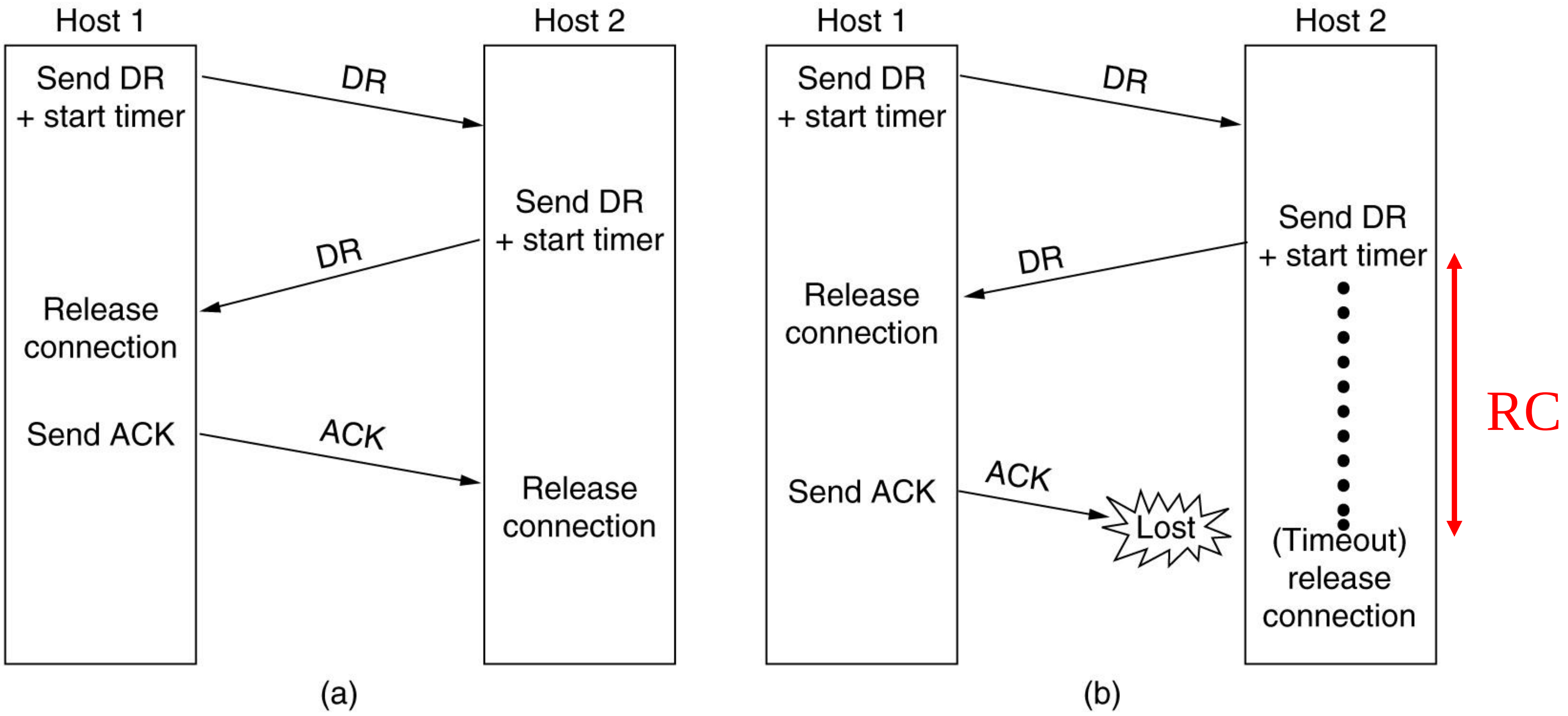


No protocol exists!!

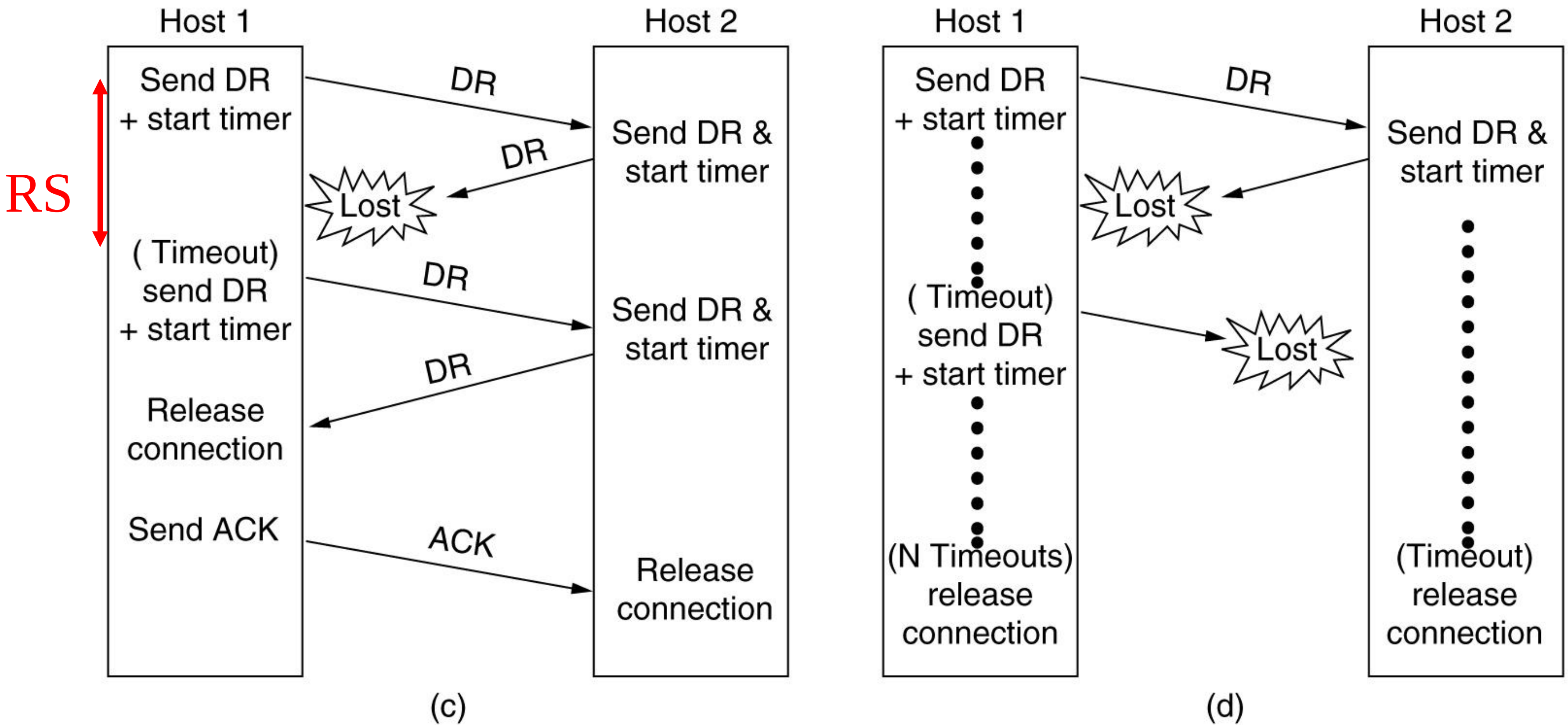
etp: Releasing a connection

- Three-way-handshake + timers
  - Send disconnection request
    + start timer RS to resend (at most N times)
    the disconnection request
  - Ack disconnection request
    + start timer RC to release connection

etp: Releasing a connection



etp: Releasing a connection

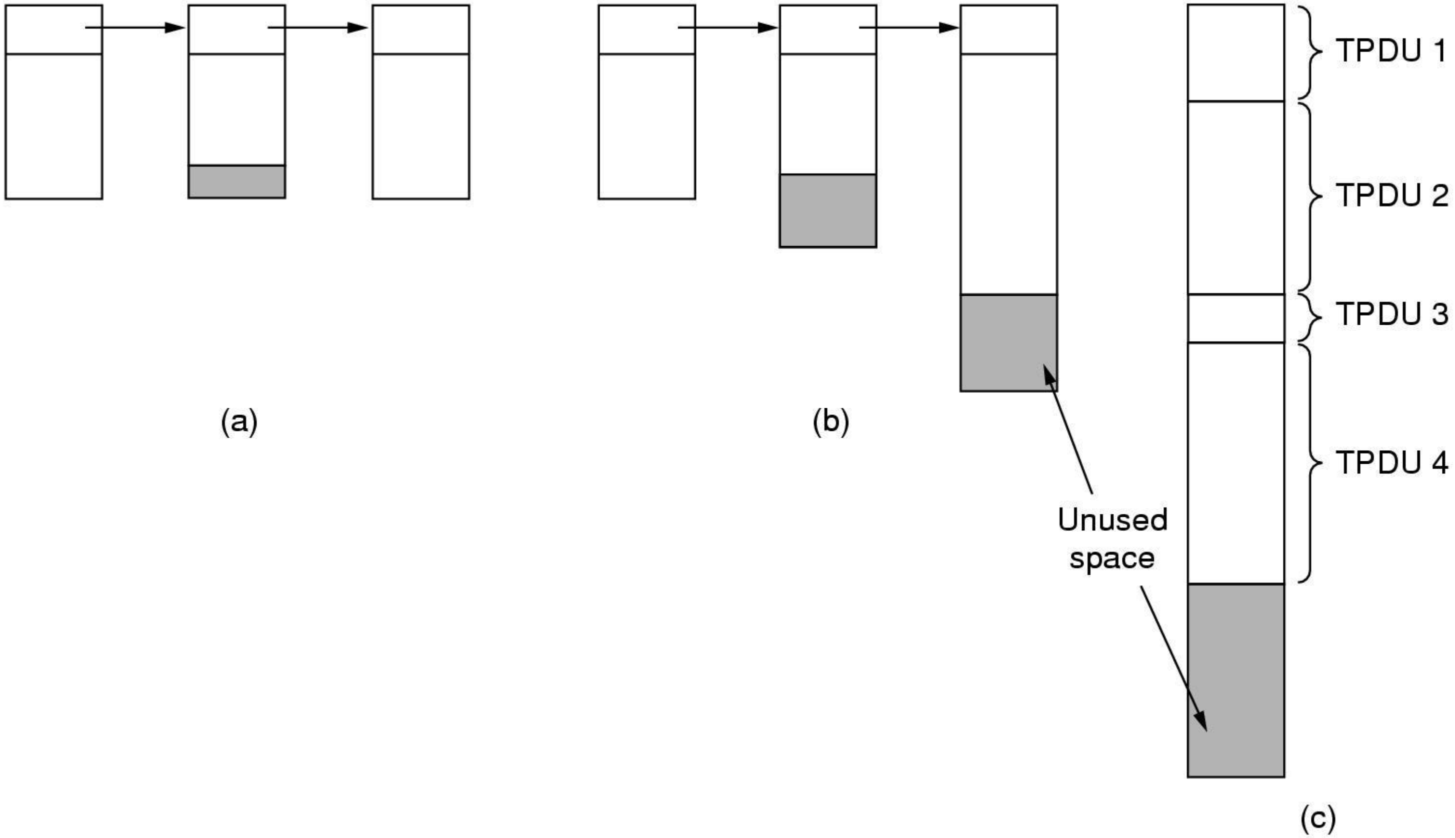


etp: Flow control and buffering

	Transport	**Data link**
connections, lines	many varying	few fixed
(sliding) window size	varying	fixed
buffer management	different sizes?	fixed size

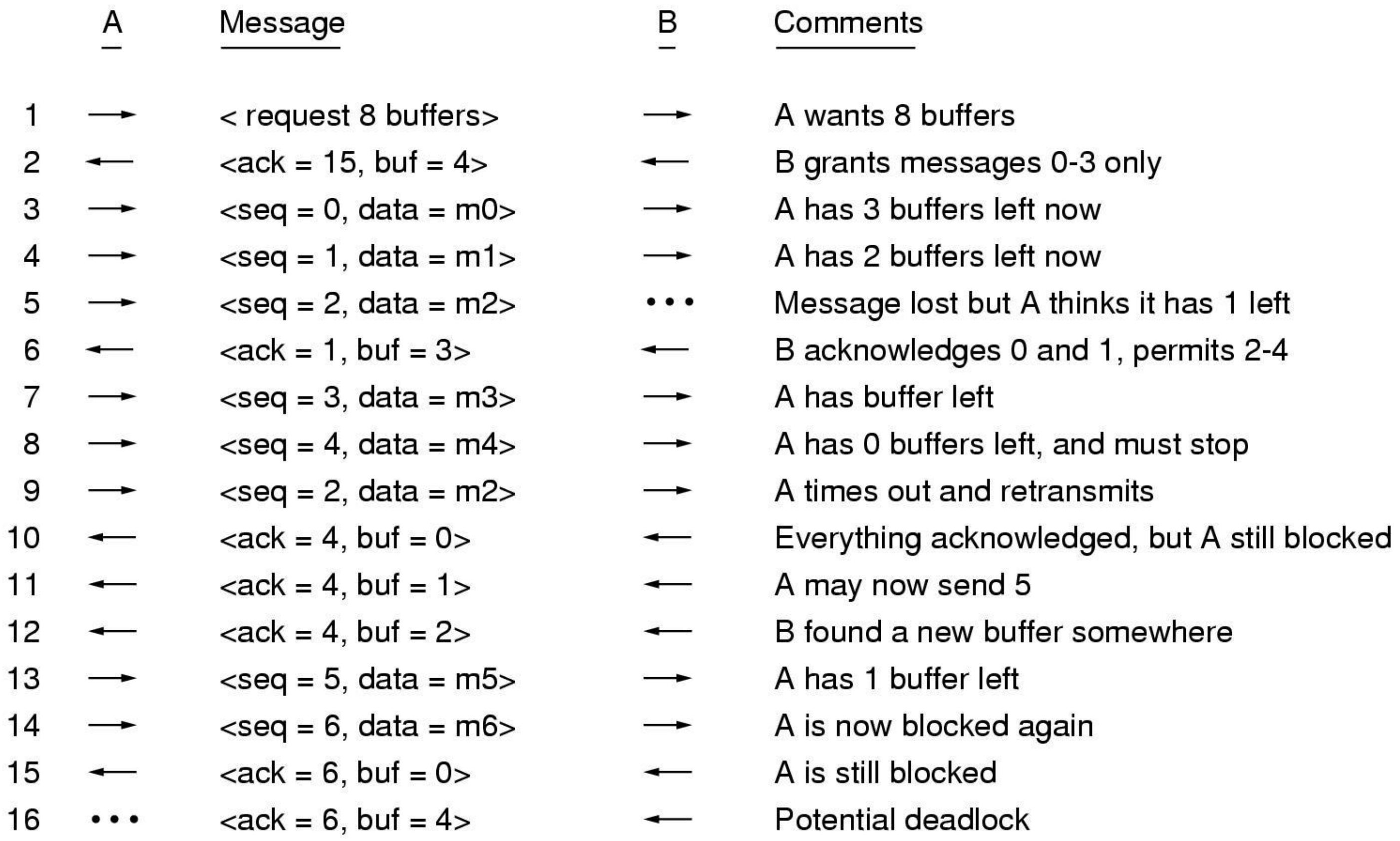
etp: Flow control and buffering

- Buffer organization



etp: Flow control and buffering

- Buffer management: decouple buffering from Acks



	A	Message	B	Comments
1	→	< request 8 buffers>	→	A wants 8 buffers
2	←	<ack = 15, buf = 4>	←	B grants messages 0-3 only
3	→	<seq = 0, data = m0>	→	A has 3 buffers left now
4	→	<seq = 1, data = m1>	→	A has 2 buffers left now
5	→	<seq = 2, data = m2>	• • •	Message lost but A thinks it has 1 left
6	←	<ack = 1, buf = 3>	←	B acknowledges 0 and 1, permits 2-4
7	→	<seq = 3, data = m3>	→	A has buffer left
8	→	<seq = 4, data = m4>	→	A has 0 buffers left, and must stop
9	→	<seq = 2, data = m2>	→	A times out and retransmits
10	←	<ack = 4, buf = 0>	←	Everything acknowledged, but A still blocked
11	←	<ack = 4, buf = 1>	←	A may now send 5
12	←	<ack = 4, buf = 2>	←	B found a new buffer somewhere
13	→	<seq = 5, data = m5>	→	A has 1 buffer left
14	→	<seq = 6, data = m6>	→	A is now blocked again
15	←	<ack = 6, buf = 0>	←	A is still blocked
16	• • •	<ack = 6, buf = 4>	←	Potential deadlock

etp: Flow control and buffering

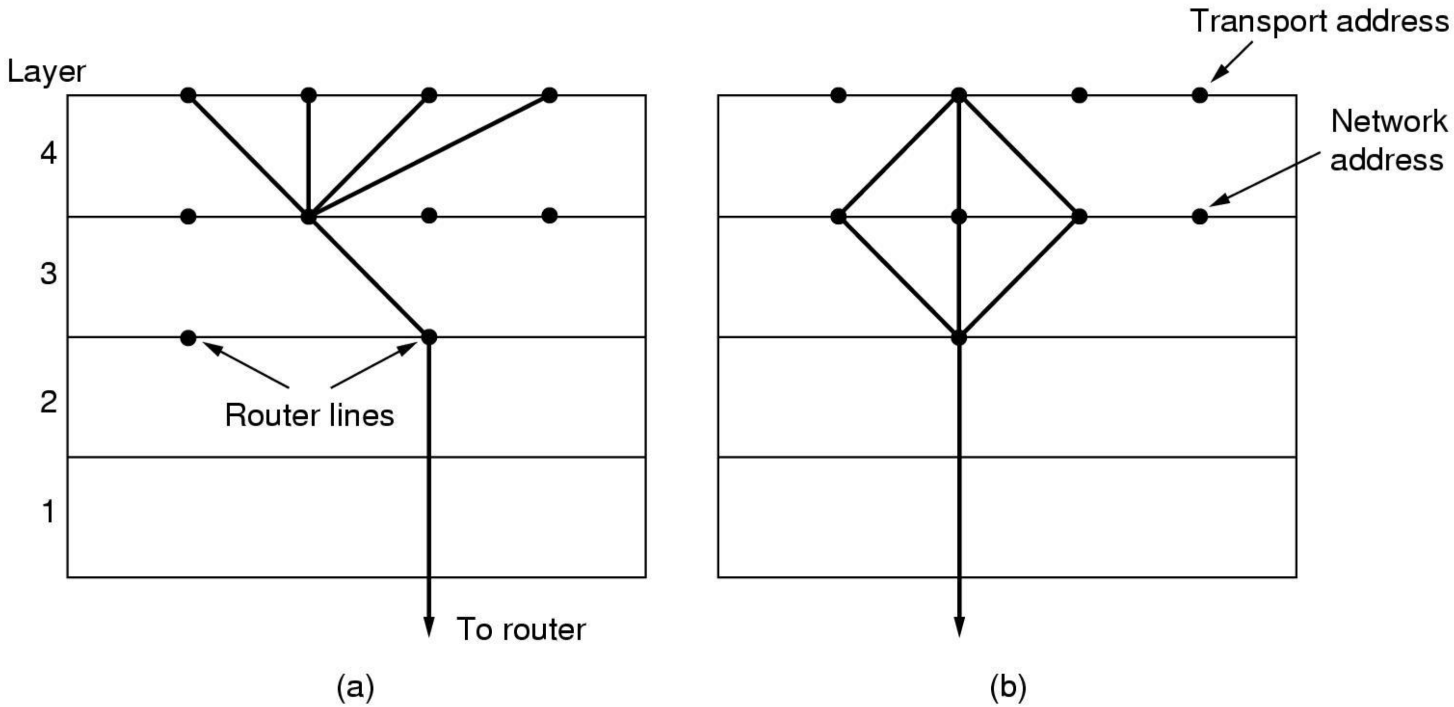
- Where to buffer?
  - – datagram network ➔ @ sender
  - – reliable network
  - + Receiver process guarantees free buffers?
    - No: for low-bandwidth bursty traffic
➔ @ sender
    - Yes: for high-bandwidth smooth traffic
➔ @ receiver

etp: Flow control and buffering

- Window size?
  - Goal:
    - Allow sender to continuously send packets
    - Avoid network congestion
  - Approach:
    - maximum window size = c * r
      - network can handle c TPDUs/sec
      - r = cycle time of a packet
    - measure c & r and adapt window size

etp: Multiplexing

- Upward: reduce number of network connections to reduce cost
- Downward: increase bandwidth to avoid per connection limits



etp: Crash recovery

- recovery from network, router crashes?
  - No problem
    - Datagram network: loss of packet is always handled
    - Connection-oriented network: establish new connection + use state to continue service
- recovery from host crash?
  - server crashes, restarts: implications for client?
  - assumptions:

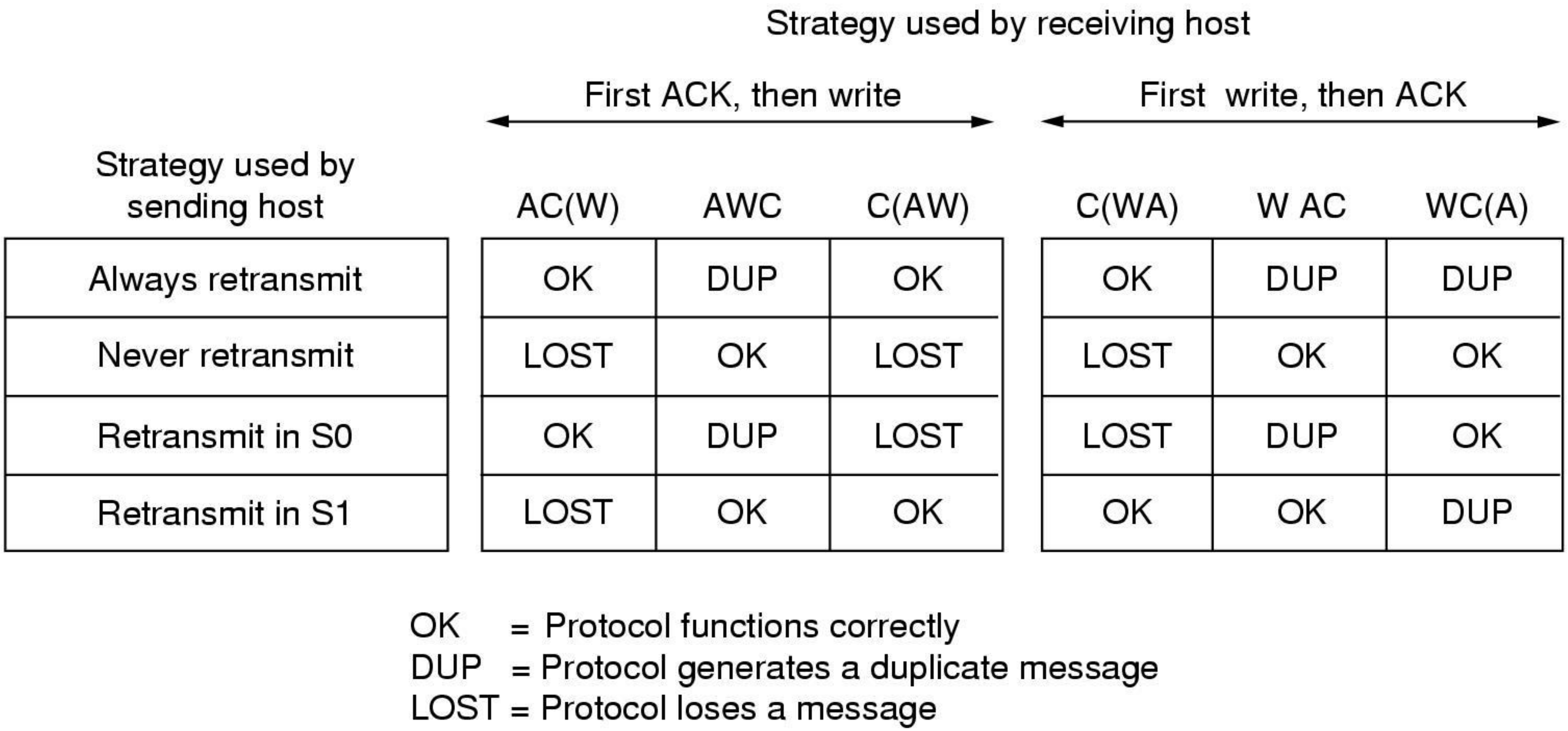
Recovery from a layer N crash can only be done by layer N+1 and only if the higher layer retains enough status information.

etp: Crash recovery

- Illustration of problem: File transfer:
  - Sender: 1 bit window protocol: states S0, S1
    - packet with seq number 0 transmitted; wait for ack
  - Receiver: actions
    - Ack packet
    - Write data to disk
    - Order?

etp: Crash recovery

Illustration of problem: File transfer



Strategy used by sending host	Strategy used by receiving host					
	First ACK, then write			First write, then ACK		
	AC(W)	AWC	C(AW)	C(WA)	W AC	WC(A)
Always retransmit	OK	DUP	OK	OK	DUP	DUP
Never retransmit	LOST	OK	LOST	LOST	OK	OK
Retransmit in S0	OK	DUP	LOST	LOST	DUP	OK
Retransmit in S1	LOST	OK	OK	OK	OK	DUP

OK = Protocol functions correctly
DUP = Protocol generates a duplicate message
LOST = Protocol loses a message

Transport Layer

- Services
- Elements of transport protocol
- Simple transport protocol
- UDP
- Remote Procedure Call (see Distributed Systems)
- TCP

Simple transport protocol

- Service primitives:
  - connum = LISTEN (local)
    - Caller is willing to accept connection
    - Blocked till request received
  - connum = CONNECT (local, remote)
    - Tries to establish connection
    - Returns identifier (nonnegative number)
  - status = SEND (connum, buffer, bytes)
    - Transmits a buffer
    - Errors returned in status
  - status = RECEIVE (connum, buffer, bytes)
    - Indicates caller's desire to get data
  - status = DISCONNECT (connum)
    - Terminates connection

Simple transport protocol

- Transport entity
  - Uses a connection-oriented reliable network
  - Programmed as a library package
  - Network interface
    - ToNet(…)
    - FromNet(…)
    - Parameters:
      - Connection identifier (connum = VC)
      - Q bit: 1 = control packet
      - M bit: 1 = more data packets to come
      - Packet type
      - Pointer to data
      - Number of bytes of data

Simple transport protocol

- Transport entity: packet types

Network packet	**Meaning**
Call request	Sent to establish a connection
Call accepted	Response to Call Request
Clear Request	Sent to release connection
Clear confirmation	Response to Clear request
Data	Used to transport data
Credit	Control packet to manage window

Simple transport protocol

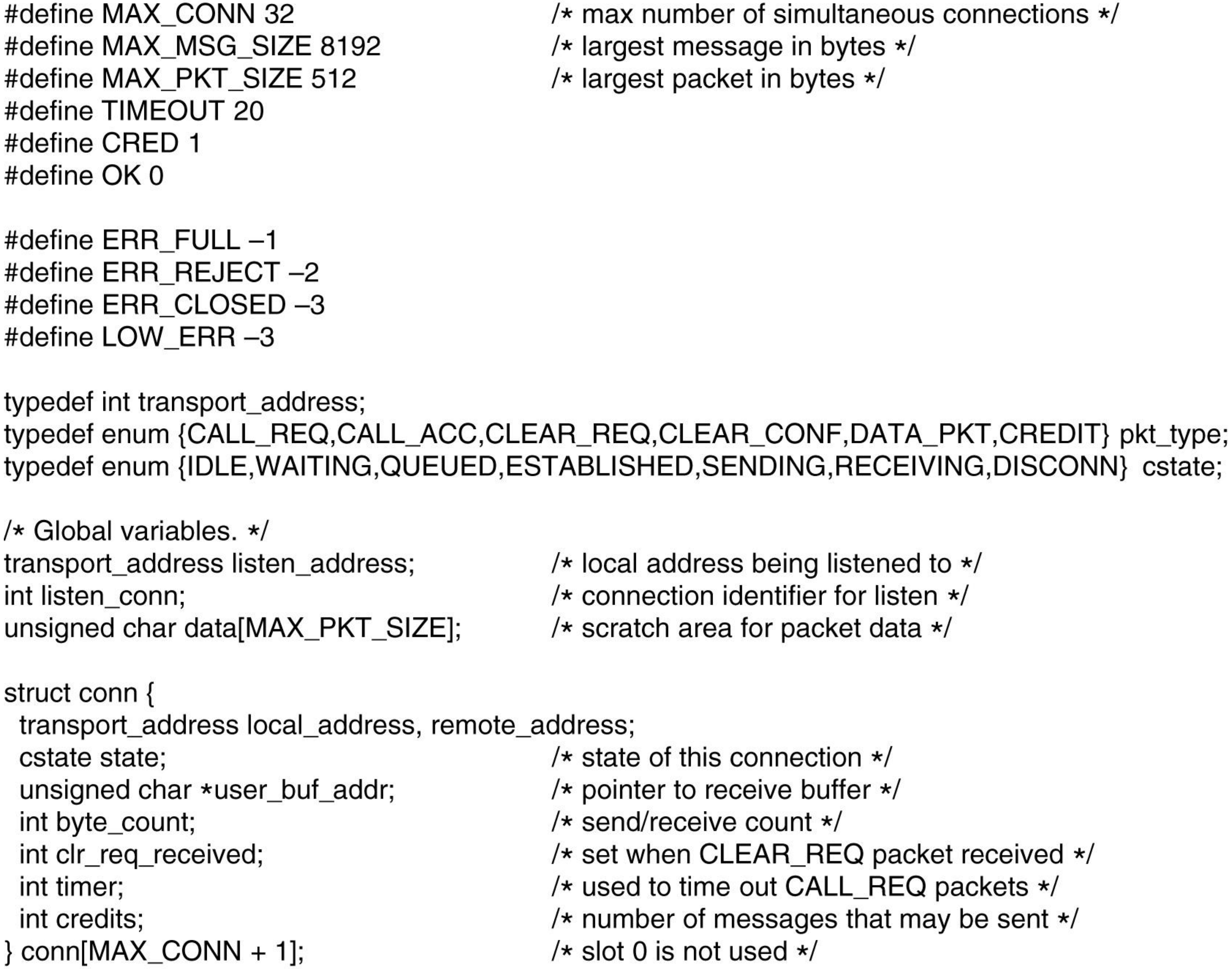
- Transport entity: state of a connection:

State	**Meaning**
Idle	Connection not established
Waiting	CONNECT done; Call Request sent
Queued	Call Request arrived; no LISTEN yet
Established	
Sending	Waiting for permission to send a packet
Receiving	RECEIVE has been done
Disconnecting	DISCONNECT done locally

Simple transport protocol

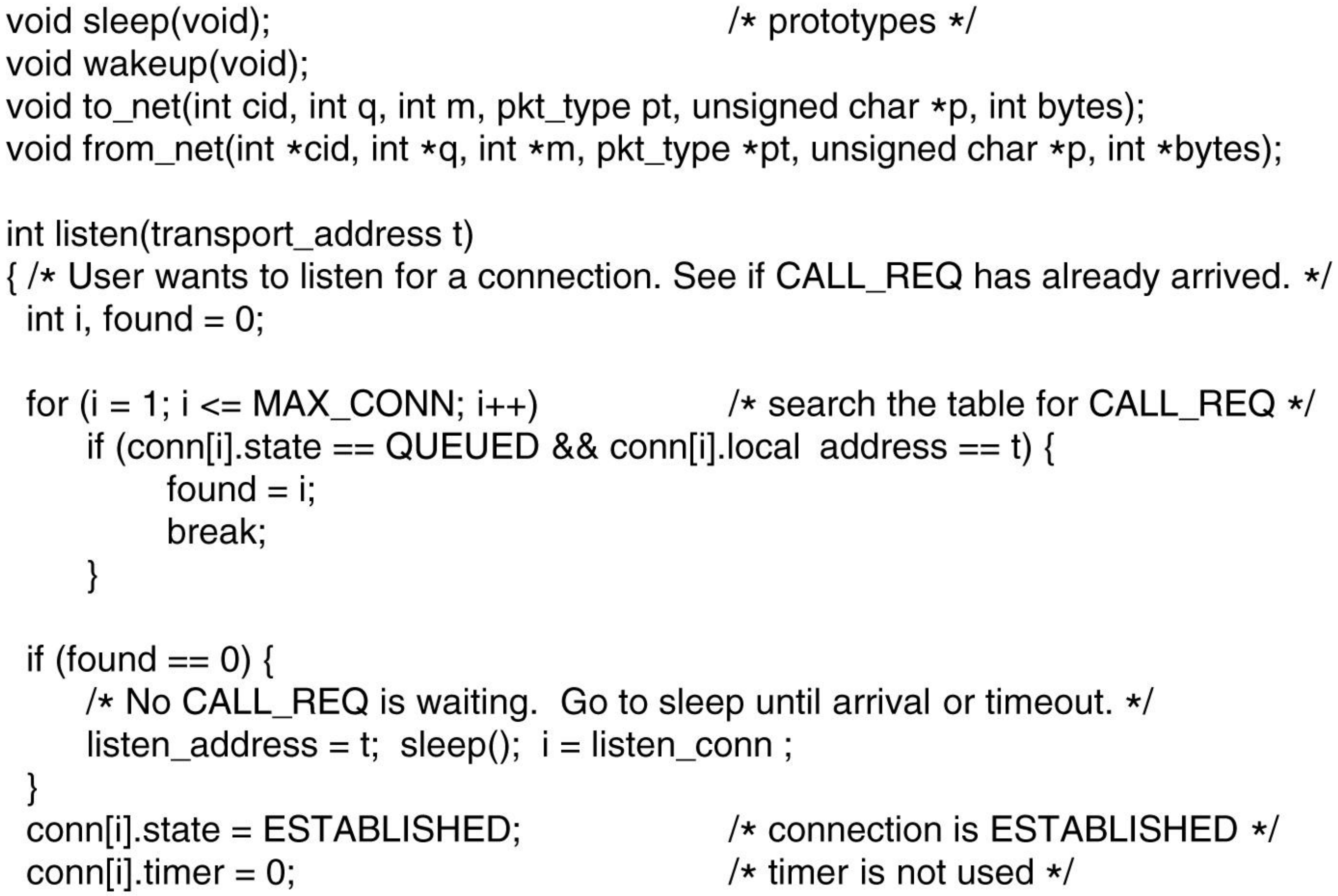
- Transport entity: code
  - See fig 6-20, p. 514 – 517
  - To read and study at home!
  - Questions?
    - Is it acceptable not to use a transport header?
    - How easy would it be to use another network protocol?

Example Transport Entity (1)



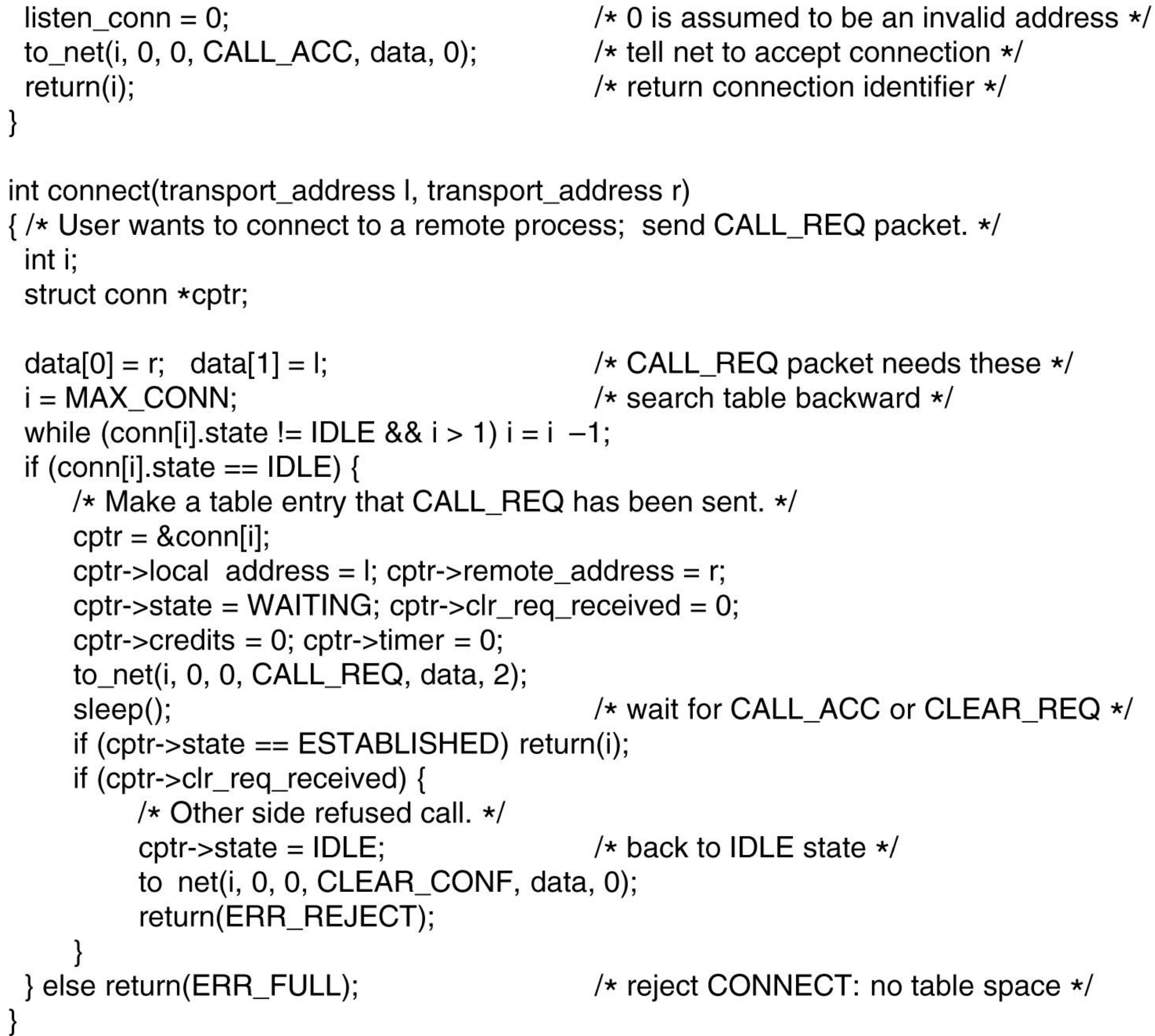
```
#define MAX_CONN 32                          /* max number of simultaneous connections */
#define MAX_MSG_SIZE 8192                    /* largest message in bytes */
#define MAX_PKT_SIZE 512                     /* largest packet in bytes */
#define TIMEOUT 20
#define CRED 1
#define OK 0

#define ERR_FULL –1
#define ERR_REJECT –2
#define ERR_CLOSED –3
#define LOW_ERR –3

typedef int transport_address;
typedef enum {CALL_REQ,CALL_ACC,CLEAR_REQ,CLEAR_CONF,DATA_PKT,CREDIT} pkt_type;
typedef enum {IDLE,WAITING,QUEUED,ESTABLISHED,SENDING,RECEIVING,DISCONN}  cstate;

/* Global variables. */
transport_address listen_address;            /* local address being listened to */
int listen_conn;                             /* connection identifier for listen */
unsigned char data[MAX_PKT_SIZE];            /* scratch area for packet data */

struct conn {
  transport_address local_address, remote_address;
  cstate state;                              /* state of this connection */
  unsigned char *user_buf_addr;              /* pointer to receive buffer */
  int byte_count;                            /* send/receive count */
  int clr_req_received;                      /* set when CLEAR_REQ packet received */
  int timer;                                 /* used to time out CALL_REQ packets */
  int credits;                               /* number of messages that may be sent */
} conn[MAX_CONN + 1];                        /* slot 0 is not used */
```

Example Transport Entity (2)



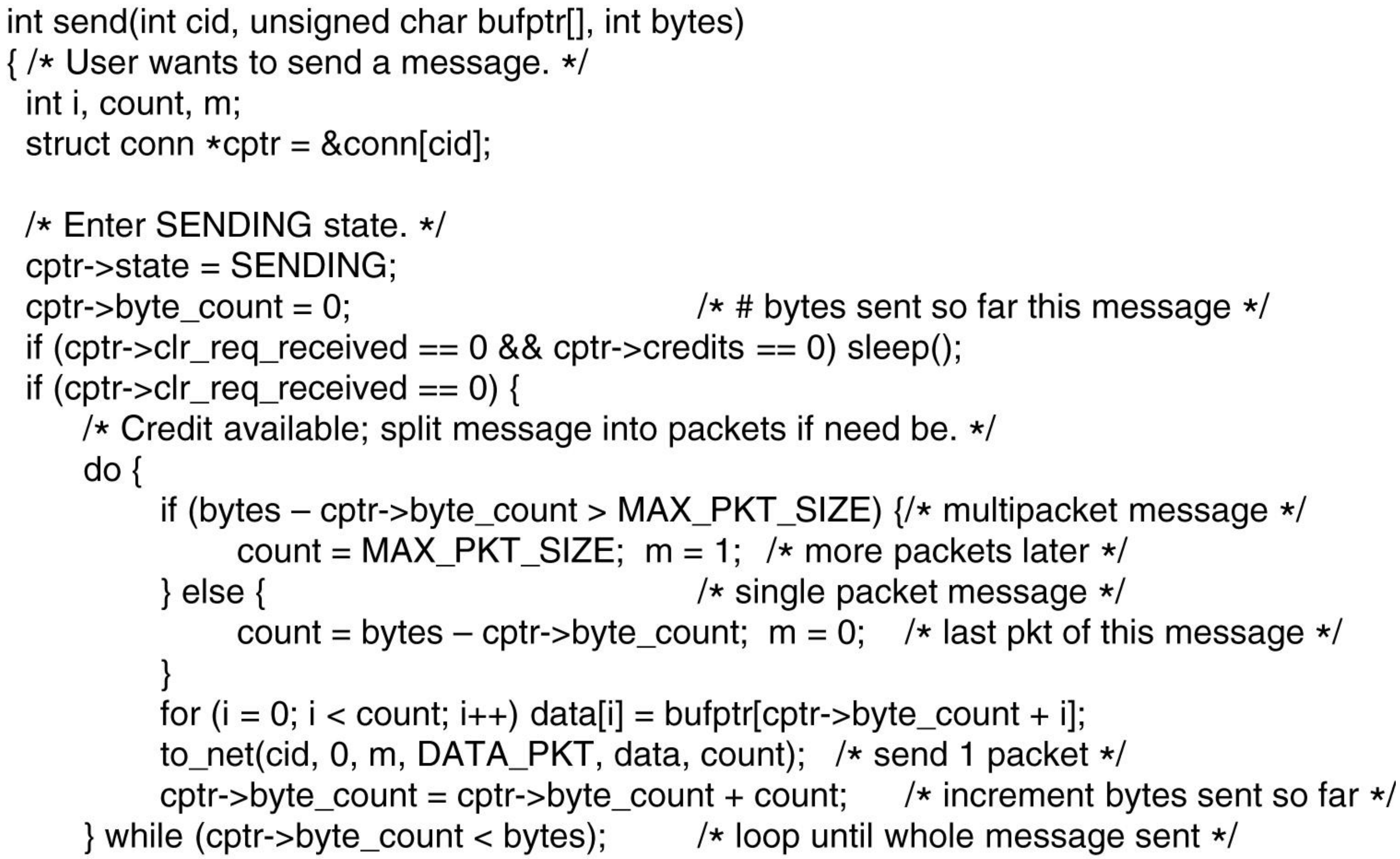
```
void sleep(void);                                    /* prototypes */
void wakeup(void);
void to_net(int cid, int q, int m, pkt_type pt, unsigned char *p, int bytes);
void from_net(int *cid, int *q, int *m, pkt_type *pt, unsigned char *p, int *bytes);

int listen(transport_address t)
{ /* User wants to listen for a connection. See if CALL_REQ has already arrived. */
  int i, found = 0;

  for (i = 1; i <= MAX_CONN; i++)                    /* search the table for CALL_REQ */
       if (conn[i].state == QUEUED && conn[i].local  address == t) {
             found = i;
             break;
       }

  if (found == 0) {
       /* No CALL_REQ is waiting.  Go to sleep until arrival or timeout. */
       listen_address = t;  sleep();  i = listen_conn ;
  }
  conn[i].state = ESTABLISHED;                       /* connection is ESTABLISHED */
  conn[i].timer = 0;                                 /* timer is not used */
```

Example Transport Entity (3)



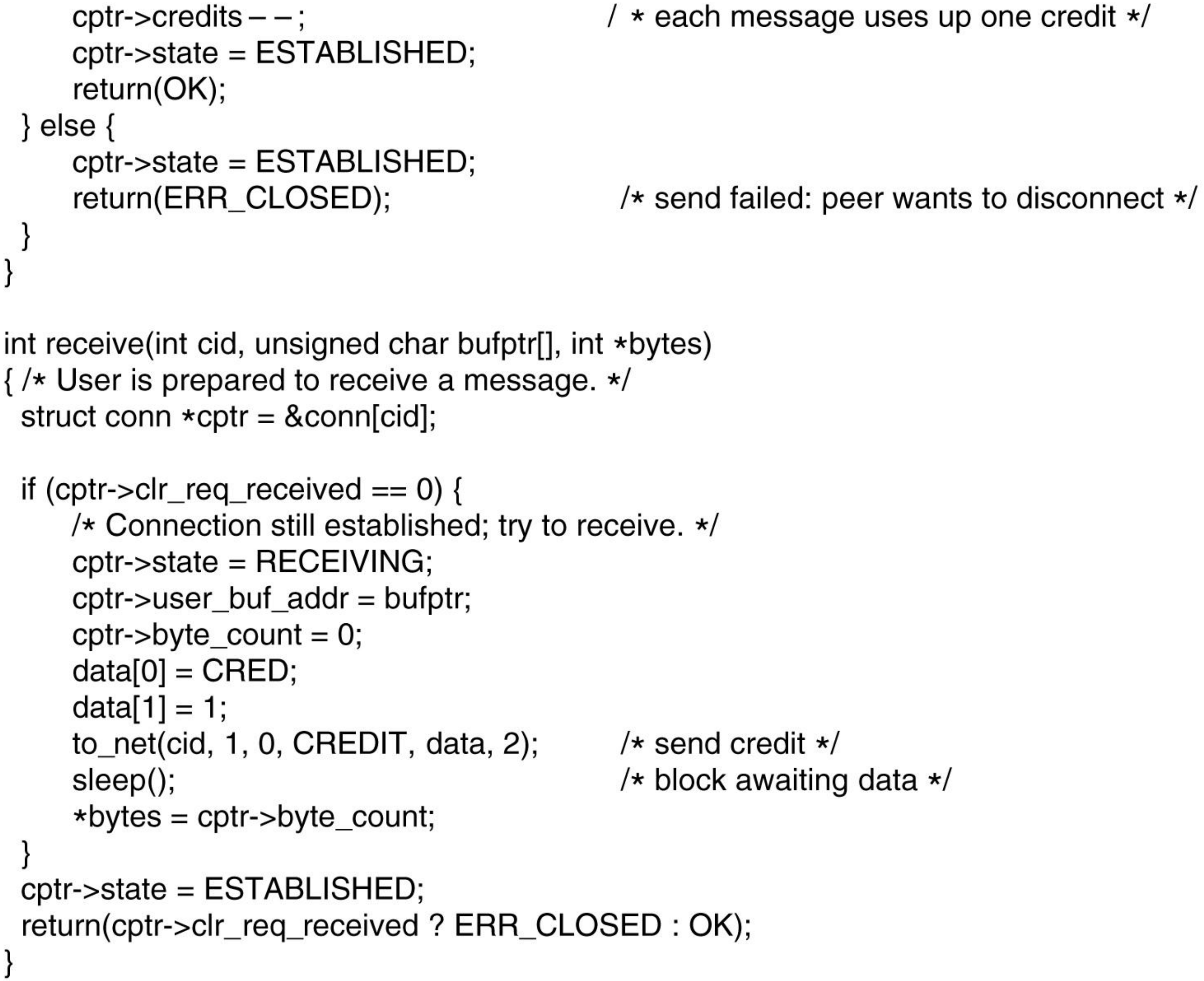
```
  listen_conn = 0;                          /* 0 is assumed to be an invalid address */
  to_net(i, 0, 0, CALL_ACC, data, 0);       /* tell net to accept connection */
  return(i);                                /* return connection identifier */
}

int connect(transport_address l, transport_address r)
{ /* User wants to connect to a remote process;  send CALL_REQ packet. */
  int i;
  struct conn *cptr;

  data[0] = r;   data[1] = l;               /* CALL_REQ packet needs these */
  i = MAX_CONN;                             /* search table backward */
  while (conn[i].state != IDLE && i > 1) i = i  –1;
  if (conn[i].state == IDLE) {
      /* Make a table entry that CALL_REQ has been sent. */
      cptr = &conn[i];
      cptr->local  address = l; cptr->remote_address = r;
      cptr->state = WAITING; cptr->clr_req_received = 0;
      cptr->credits = 0; cptr->timer = 0;
      to_net(i, 0, 0, CALL_REQ, data, 2);
      sleep();                              /* wait for CALL_ACC or CLEAR_REQ */
      if (cptr->state == ESTABLISHED) return(i);
      if (cptr->clr_req_received) {
          /* Other side refused call. */
          cptr->state = IDLE;               /* back to IDLE state */
          to  net(i, 0, 0, CLEAR_CONF, data, 0);
          return(ERR_REJECT);
      }
  } else return(ERR_FULL);                  /* reject CONNECT: no table space */
}
```

Example Transport Entity (4)



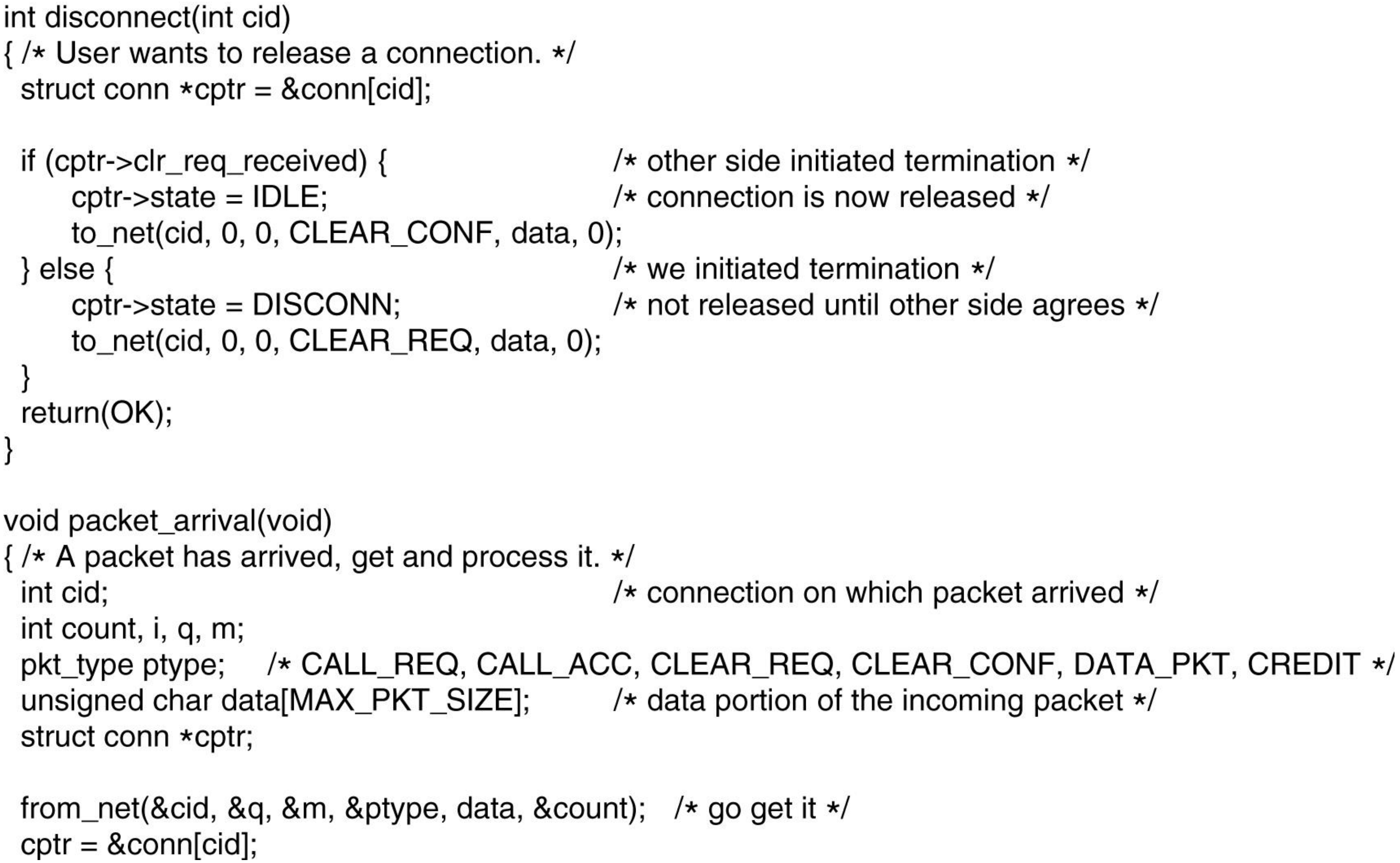
```
int send(int cid, unsigned char bufptr[], int bytes)
{ /* User wants to send a message. */
  int i, count, m;
  struct conn *cptr = &conn[cid];

  /* Enter SENDING state. */
  cptr->state = SENDING;
  cptr->byte_count = 0;                          /* # bytes sent so far this message */
  if (cptr->clr_req_received == 0 && cptr->credits == 0) sleep();
  if (cptr->clr_req_received == 0) {
       /* Credit available; split message into packets if need be. */
       do {
            if (bytes – cptr->byte_count > MAX_PKT_SIZE) {/* multipacket message */
                 count = MAX_PKT_SIZE;  m = 1;   /* more packets later */
            } else {                              /* single packet message */
                 count = bytes – cptr->byte_count;  m = 0;    /* last pkt of this message */
            }
            for (i = 0; i < count; i++) data[i] = bufptr[cptr->byte_count + i];
            to_net(cid, 0, m, DATA_PKT, data, count);   /* send 1 packet */
            cptr->byte_count = cptr->byte_count + count;      /* increment bytes sent so far */
       } while (cptr->byte_count < bytes);        /* loop until whole message sent */
```

Example Transport Entity (5)



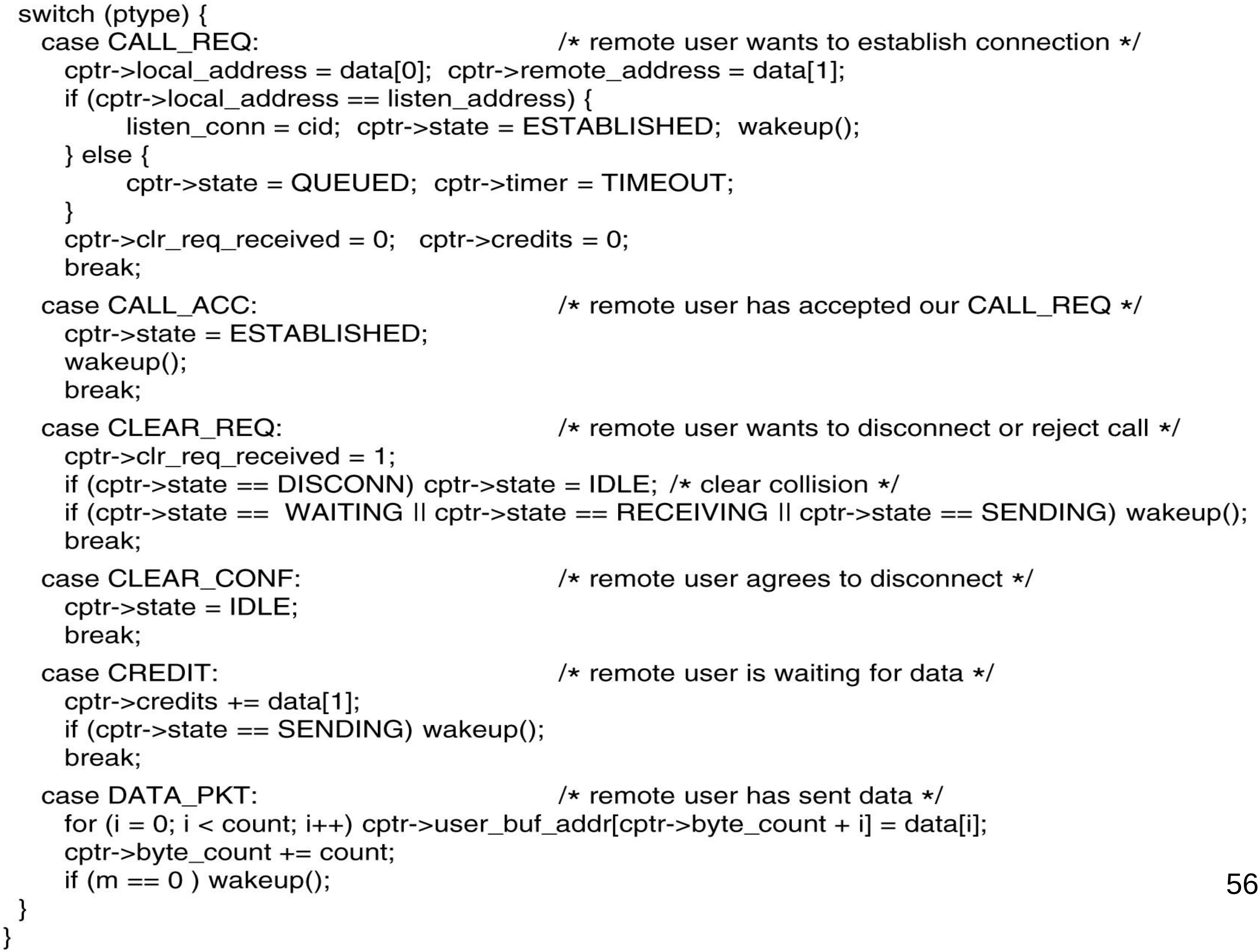
```
        cptr->credits – –;                  / * each message uses up one credit */
        cptr->state = ESTABLISHED;
        return(OK);
  } else {
        cptr->state = ESTABLISHED;
        return(ERR_CLOSED);                  /* send failed: peer wants to disconnect */
  }
}

int receive(int cid, unsigned char bufptr[], int *bytes)
{ /* User is prepared to receive a message. */
  struct conn *cptr = &conn[cid];

  if (cptr->clr_req_received == 0) {
        /* Connection still established; try to receive. */
        cptr->state = RECEIVING;
        cptr->user_buf_addr = bufptr;
        cptr->byte_count = 0;
        data[0] = CRED;
        data[1] = 1;
        to_net(cid, 1, 0, CREDIT, data, 2);     /* send credit */
        sleep();                                 /* block awaiting data */
        *bytes = cptr->byte_count;
  }
  cptr->state = ESTABLISHED;
  return(cptr->clr_req_received ? ERR_CLOSED : OK);
}
```

Example Transport Entity (6)



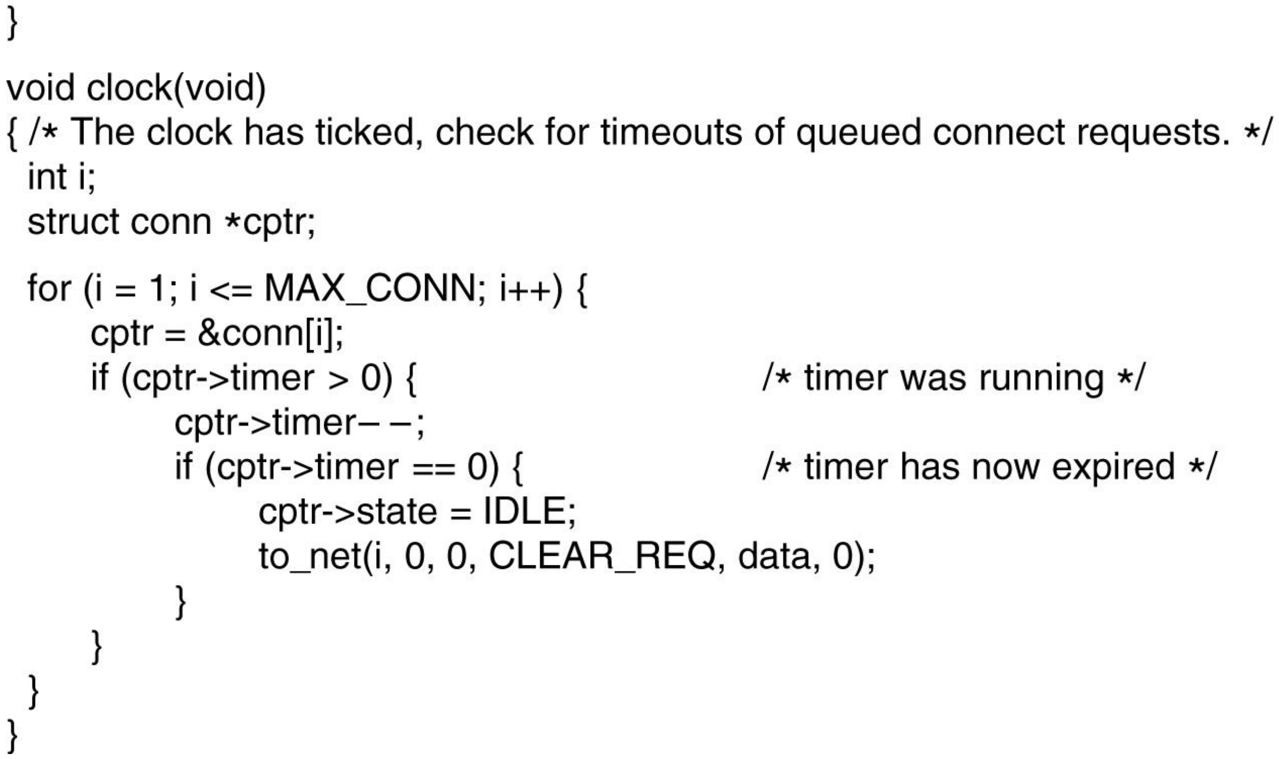
```
int disconnect(int cid)
{ /* User wants to release a connection. */
  struct conn *cptr = &conn[cid];

  if (cptr->clr_req_received) {                    /* other side initiated termination */
        cptr->state = IDLE;                        /* connection is now released */
        to_net(cid, 0, 0, CLEAR_CONF, data, 0);
  } else {                                         /* we initiated termination */
        cptr->state = DISCONN;                     /* not released until other side agrees */
        to_net(cid, 0, 0, CLEAR_REQ, data, 0);
  }
  return(OK);
}

void packet_arrival(void)
{ /* A packet has arrived, get and process it. */
  int cid;                                         /* connection on which packet arrived */
  int count, i, q, m;
  pkt_type ptype;     /* CALL_REQ, CALL_ACC, CLEAR_REQ, CLEAR_CONF, DATA_PKT, CREDIT */
  unsigned char data[MAX_PKT_SIZE];                /* data portion of the incoming packet */
  struct conn *cptr;

  from_net(&cid, &q, &m, &ptype, data, &count);    /* go get it */
  cptr = &conn[cid];
```

Example Transport Entity (7)



```
switch (ptype) {
  case CALL_REQ:                         /* remote user wants to establish connection */
    cptr->local_address = data[0];  cptr->remote_address = data[1];
    if (cptr->local_address == listen_address) {
          listen_conn = cid;  cptr->state = ESTABLISHED;  wakeup();
    } else {
          cptr->state = QUEUED;  cptr->timer = TIMEOUT;
    }
    cptr->clr_req_received = 0;   cptr->credits = 0;
    break;

  case CALL_ACC:                         /* remote user has accepted our CALL_REQ */
    cptr->state = ESTABLISHED;
    wakeup();
    break;

  case CLEAR_REQ:                        /* remote user wants to disconnect or reject call */
    cptr->clr_req_received = 1;
    if (cptr->state == DISCONN) cptr->state = IDLE; /* clear collision */
    if (cptr->state ==  WAITING || cptr->state == RECEIVING || cptr->state == SENDING) wakeup();
    break;

  case CLEAR_CONF:                       /* remote user agrees to disconnect */
    cptr->state = IDLE;
    break;

  case CREDIT:                           /* remote user is waiting for data */
    cptr->credits += data[1];
    if (cptr->state == SENDING) wakeup();
    break;

  case DATA_PKT:                         /* remote user has sent data */
    for (i = 0; i < count; i++) cptr->user_buf_addr[cptr->byte_count + i] = data[i];
    cptr->byte_count += count;
    if (m == 0 ) wakeup();
 }
}
```

Example Transport Entity (8)



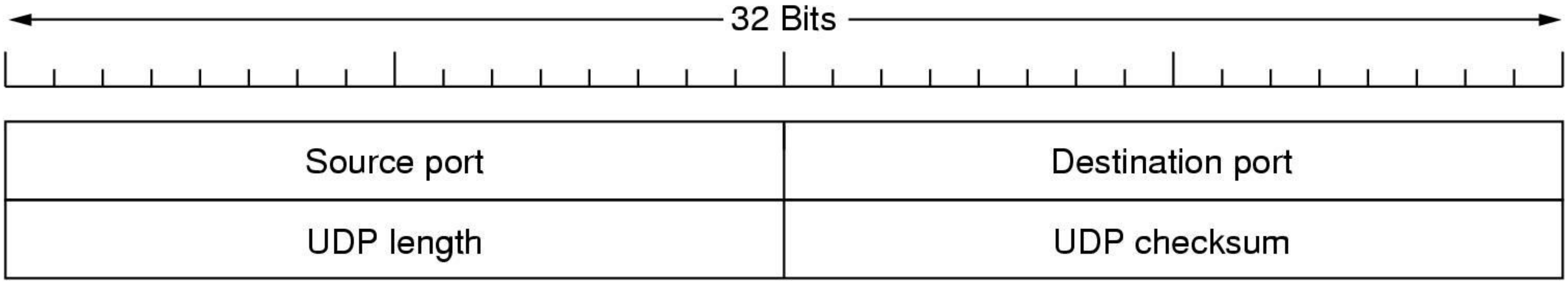
```
}

void clock(void)
{ /* The clock has ticked, check for timeouts of queued connect requests. */
  int i;
  struct conn *cptr;

  for (i = 1; i <= MAX_CONN; i++) {
        cptr = &conn[i];
        if (cptr->timer > 0) {                          /* timer was running */
                cptr->timer– –;
                if (cptr->timer == 0) {                 /* timer has now expired */
                        cptr->state = IDLE;
                        to_net(i, 0, 0, CLEAR_REQ, data, 0);
                }
        }
  }
}
```

Transport Layer

- Services
- Elements of transport protocol
- Simple transport protocol
- UDP
- Remote Procedure Call (see Distributed Systems)
- TCP

UDP

- User Data Protocol
  - Datagram service between processes
    - No connection overhead
  - UDP header:
    - Ports = identification of end points



UDP

- Some characteristics
  - Supports broadcasting, multicasting (not in TCP)
  - Packet oriented (TCP gives byte stream)
  - Simple protocol
  - Why needed above IP?

Transport Layer

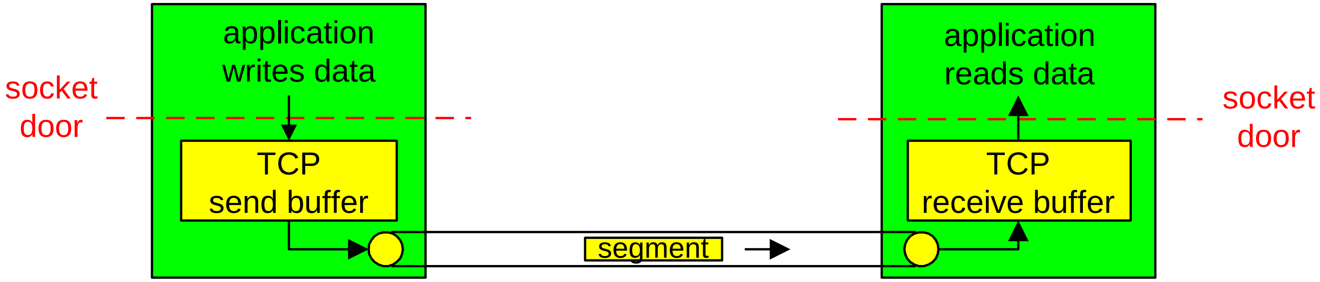
- Services
- Elements of transport protocol
- Simple transport protocol
- UDP
- Remote Procedure Call (see Distributed Systems)
- TCP

TCP service model

- point-to-point
  - one sender, one receiver
- reliable, in-order byte stream
  - no *message/packet boundaries*
- pipelined & flow controlled
  - window size set by TCP congestion and flow control algorithms
- connection-oriented
  - handshaking to get at initial state
- full duplex data
  - bi-directional data flow in same connection

TCP service model

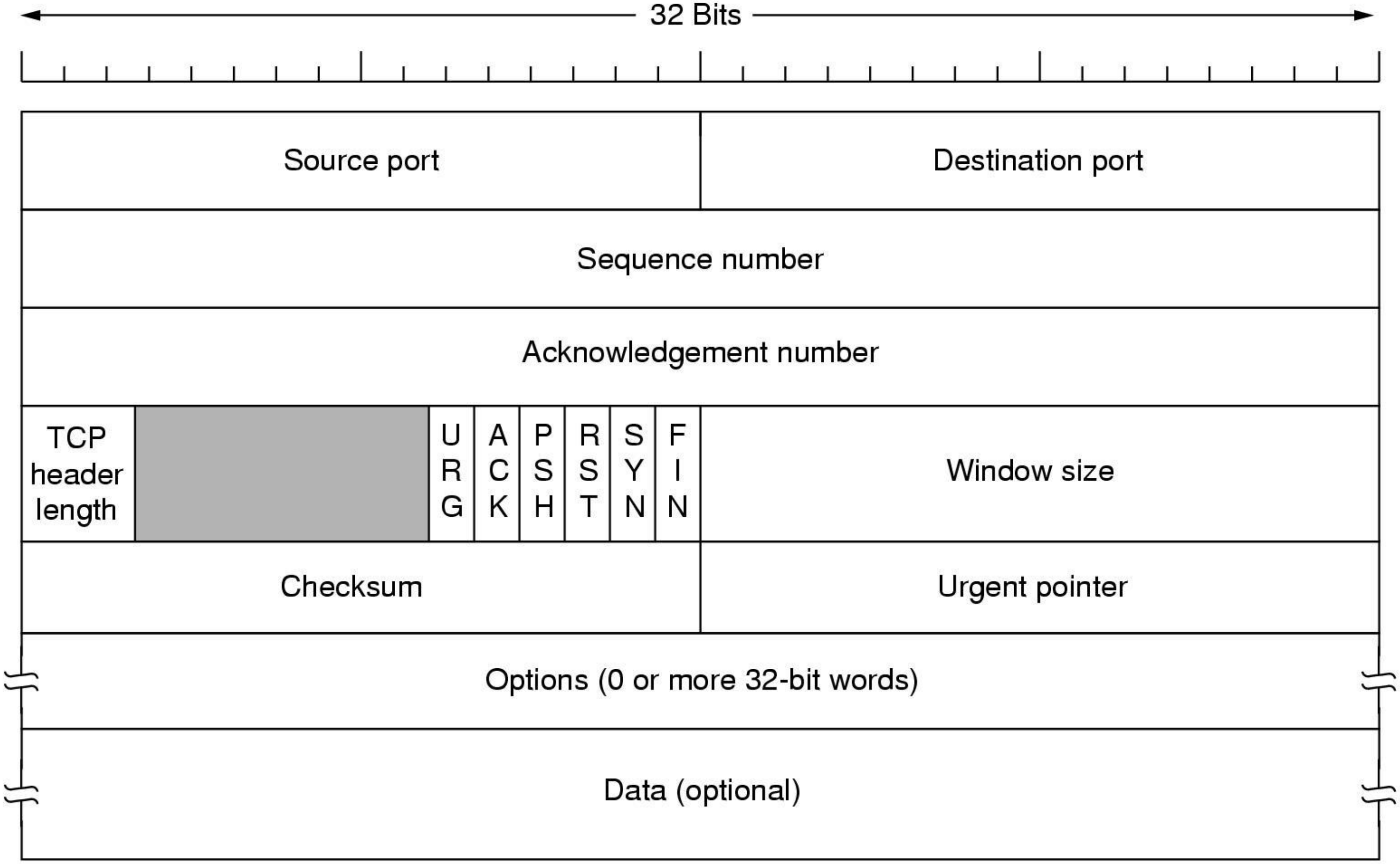
- …
- send & receive buffers



TCP protocol

- Three-way handshake to set up connections
- Every byte has its own 32-bit sequence number
  - Wrap around
  - 32-bit Acks; window size in bytes
- Segment = unit of data exchange
  - 20-byte header + options + data
  - Limits for size
    - 64Kbyte
    - MTU, agreed upon for each direction
  - Data from consecutive writes may be accumulated in a single segment
  - Fragmentation possible
- Sliding window protocol

TCP header



32 Bits

Source port

Destination port

Sequence number

Acknowledgement number

TCP header length

URG

ACK

PSH

RST

SYN

FIN

Window size

Checksum

Urgent pointer

Options (0 or more 32-bit words)

Data (optional)

TCP header

- source & destination ports (16 bit)
- sequence number (32 bit)
- Acknowledgement number (32 bit)
- Header length (4 bits) in 32-bit words
- 6 flags (1 bit)
- window size (16 bit): number of bytes the sender is allowed to send starting at byte acknowledged
- checksum (16 bit)
- urgent pointer (16 bit) : byte position of urgent data

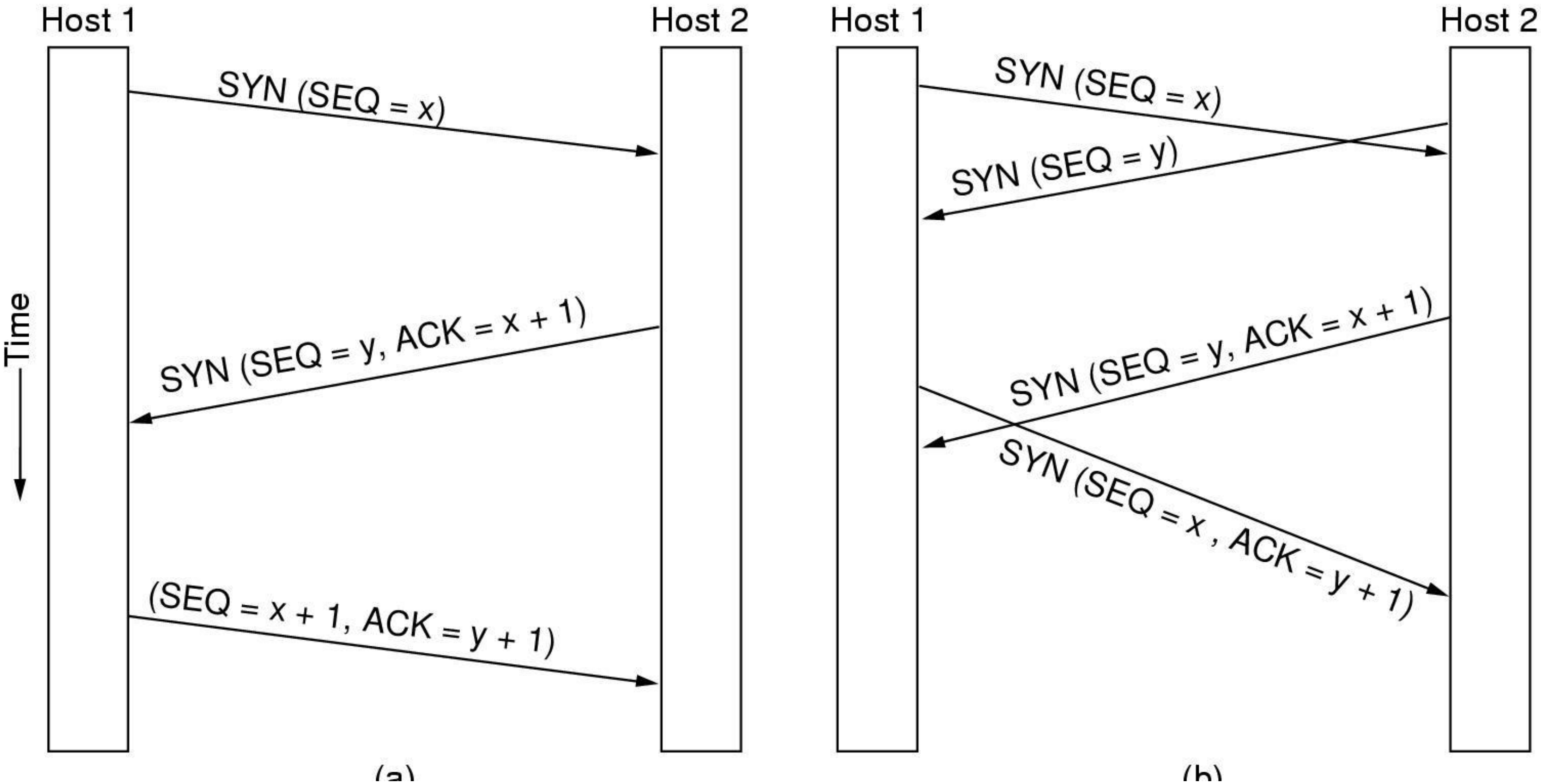
TCP header

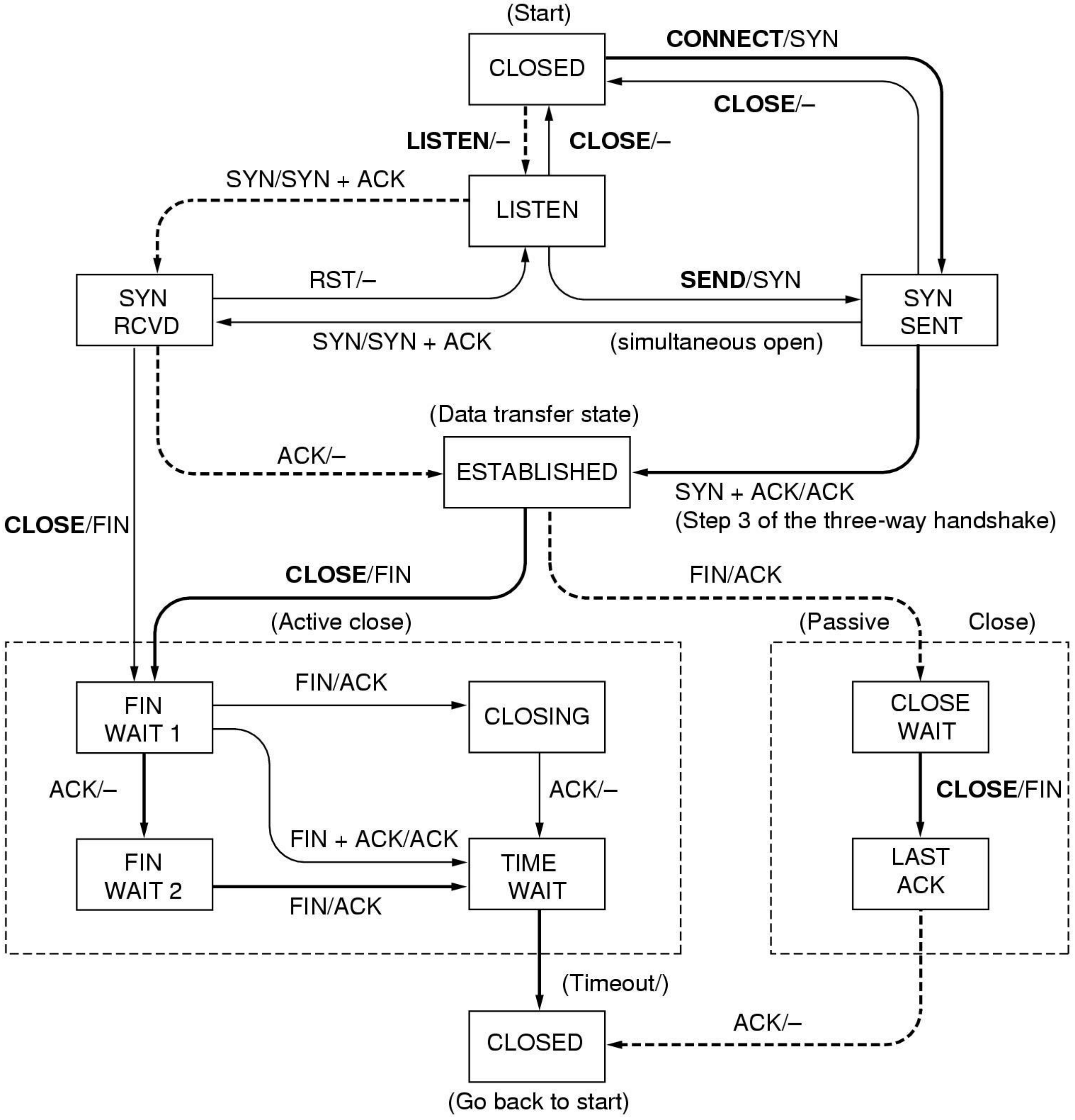
- Flags:
  - URG: urgent pointer in use
  - ACK: valid Acknowledgement number
  - PSH: receiver should deliver data without delay to user
  - RST: reset connection
  - SYN: used when establishing connections
  - FIN: used to release connection
- Options:
  - Maximum payload a host is willing to receive
  - Scale factor window size
  - Use selective repeat instead of go back n

TCP connection management

- Three-way handshake
  - Initial sequence number: clock based
  - No reboot after crash for T (maximum packet lifetime=120 sec)
  - Wrap around?
- Connection identification
  - Pair of ports of end points
- Connection release
  - Both sides are closed separately
  - No response to FIN: release after 2*T
  - Both sides closed: wait for time 2 * T

TCP connection management


Host 1
Host 2
SYN (SEQ = x)
SYN (SEQ = y, ACK = x + 1)
(SEQ = x + 1, ACK = y + 1)
Time
(a)
Host 1
Host 2
SYN (SEQ = x)
SYN (SEQ = y)
SYN (SEQ = y, ACK = x + 1)
SYN (SEQ = x , ACK = y + 1)
(b)



(Start)

CLOSED

CONNECT/SYN

CLOSE/–

LISTEN/–

CLOSE/–

SYN/SYN + ACK

LISTEN

RST/–

SEND/SYN

SYN RCVD

SYN SENT

SYN/SYN + ACK

(simultaneous open)

(Data transfer state)

ACK/–

ESTABLISHED

SYN + ACK/ACK

(Step 3 of the three-way handshake)

CLOSE/FIN

CLOSE/FIN

FIN/ACK

(Active close)

(Passive Close)

FIN WAIT 1

FIN/ACK

CLOSING

CLOSE WAIT

ACK/–

ACK/–

CLOSE/FIN

FIN + ACK/ACK

FIN WAIT 2

TIME WAIT

LAST ACK

FIN/ACK

(Timeout/)

ACK/–

CLOSED

(Go back to start)

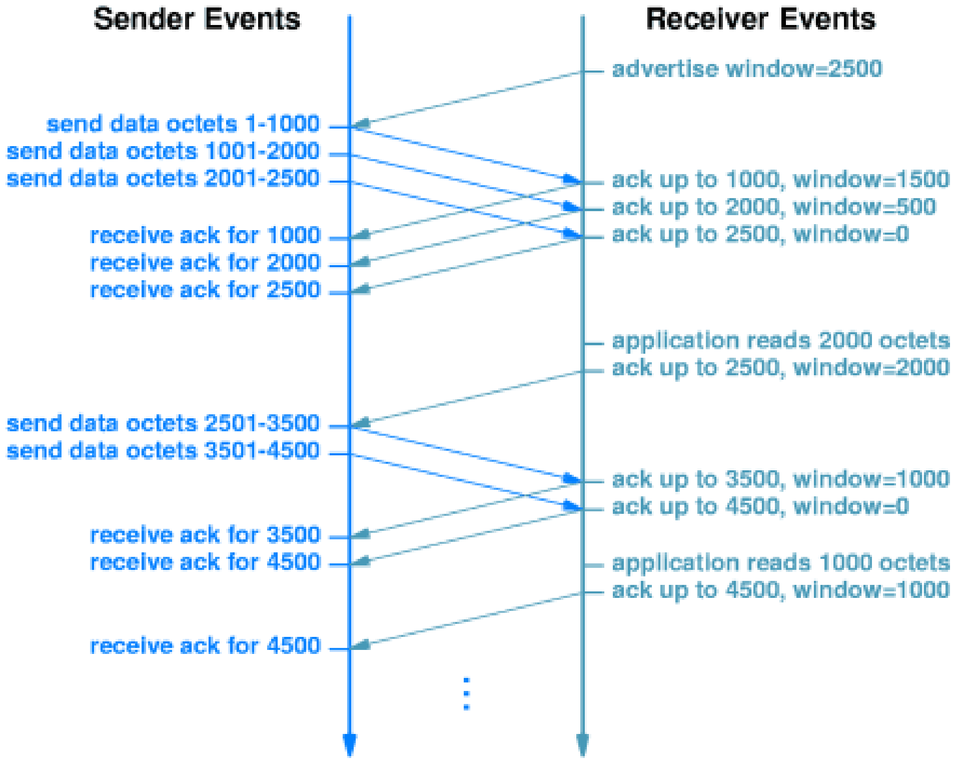
TCP connection management

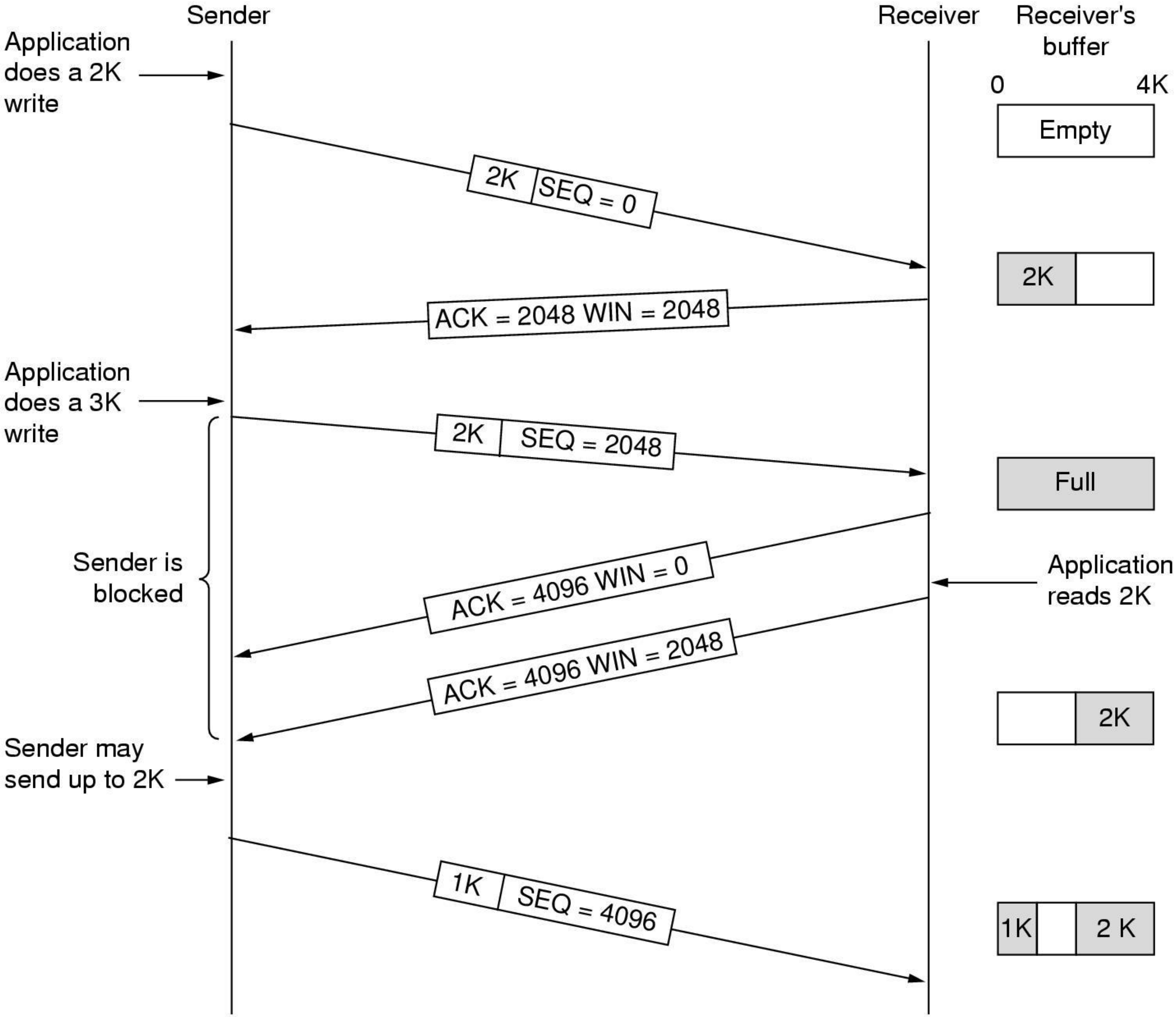
State	**Description**
Closed	No connection is active or pending
Listen	The server is waiting for an incoming call
SYN rcvd	A connection request has arrived; wait for ACK
SYN sent	The application has started to open a connection
Established	The normal data transfer state
FIN wait 1	The application has said it is finished
FIN wait 2	The other side has agreed to release
Timed wait	Wait for all packets to die off
Closing	Both sides have tried to close simultaneously
Close wait	The other side has initiated a release
Last Ack	Wait for all packets to die off

TCP transmission policy

- Window size decoupled from Acks (ex. next slides)
- Window = 0 ➔ no packets except for
  - Urgent data
  - 1 byte segment to send Ack & window size
- Incoming user data may be buffered
  - May improve performance: less segments to send
- Ways to improve performance:
  - Delay acks and window updates for 500 msec
  - Nagle’s algorithm
  - Silly window syndrome

TCP transmission policy


Sender Events
Receiver Events
advertise window=2500
send data octets 1-1000
send data octets 1001-2000
send data octets 2001-2500
ack up to 1000, window=1500
ack up to 2000, window=500
ack up to 2500, window=0
receive ack for 1000
receive ack for 2000
receive ack for 2500
application reads 2000 octets
ack up to 2500, window=2000
send data octets 2501-3500
send data octets 3501-4500
ack up to 3500, window=1000
ack up to 4500, window=0
receive ack for 3500
receive ack for 4500
application reads 1000 octets
ack up to 4500, window=1000
receive ack for 4500
⋮


Sender
Receiver
Receiver's buffer
0
4K
Application does a 2K write
Empty
2K SEQ = 0
2K
ACK = 2048 WIN = 2048
Application does a 3K write
2K SEQ = 2048
Full
Sender is blocked
ACK = 4096 WIN = 0
Application reads 2K
ACK = 4096 WIN = 2048
2K
Sender may send up to 2K
1K SEQ = 4096
1K
2 K

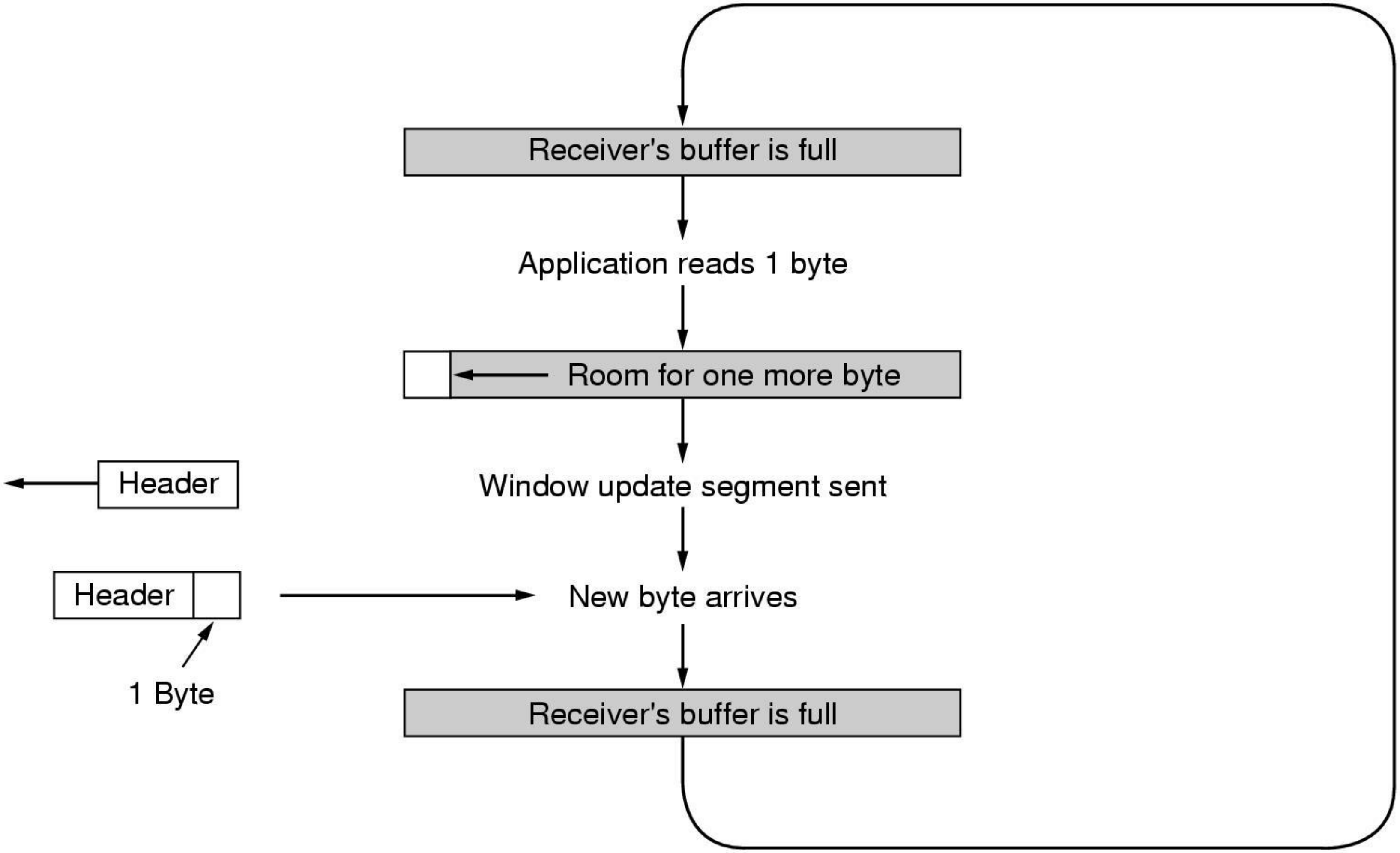
TCP transmission policy

- Telnet scenario: interactive editor reacting on each keystroke: One character typed
  - ➔ 21 byte segment or 41 byte IP packet
  - 🡐 (packet received) 20 byte segment with Ack
  - 🡐 (editor has read byte) 20 byte segment with window update
  - 🡐 (editor has processed byte; sends echo) 21 byte segment
  - ➔ (client gets echo) 20 byte segment with Ack
- Solutions:
  - – delay acks + window updates for 500 msec
  - – Nagle's algorithm:
    - Receive one byte from user; send it in segment
    - Buffer all other chars till Ack for first char arrives
    - Send other chars in a single segment
    - Disable algorithm for X-windows applications (do not delay sending of mouse movements)

TCP transmission policy

- Silly window syndrome
  - Problem:
    - Sender transmits data in large blocks
    - Receiver reads data 1 byte at a time
  - Scenario: next slide
  - Solution:
    - Do not send window update for 1 byte
    - Wait for window update till
      - Receiver can accept MTU
      - Buffer is half empty

TCP transmission policy



TCP transmission policy

- General approach:
  - Sender should not send small segments
    - Nagle: buffer data in TCP send buffer
  - Receiver should not ask for small segments
    - Silly window: do window updates in large units

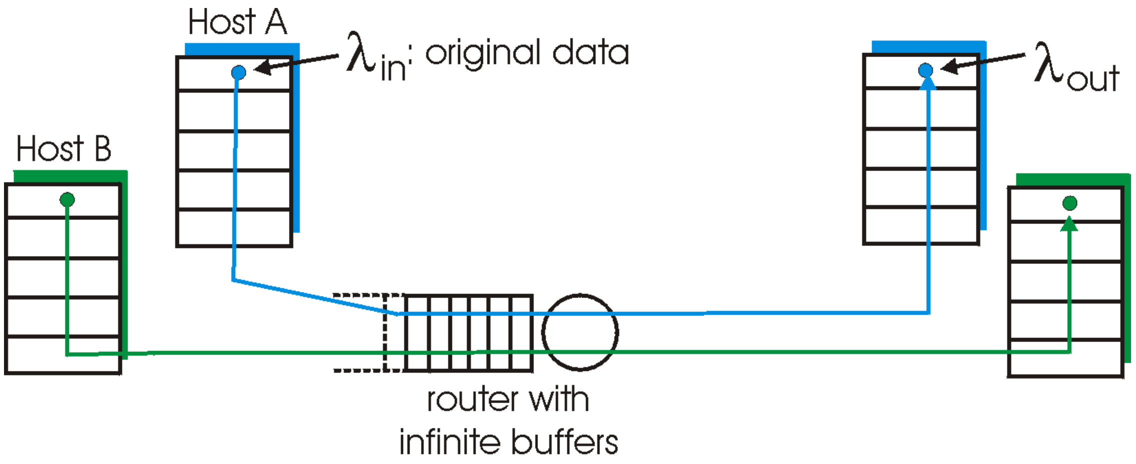
Principles of Congestion Control

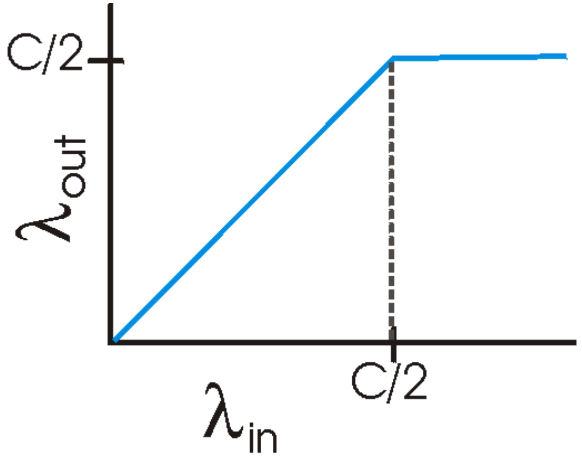
Congestion:

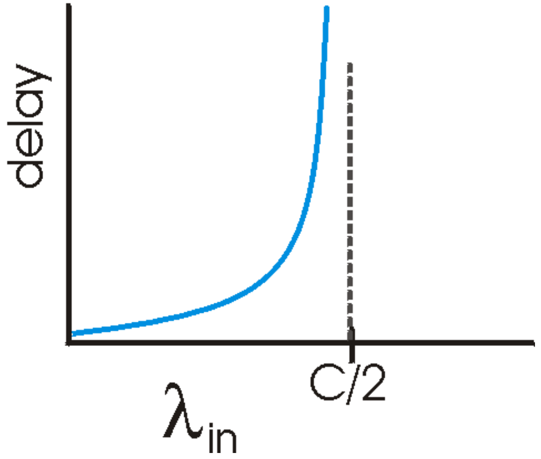
- informally: “too many sources sending too much data too fast for *network* to handle”
- different from flow control!
  - = end-to-end issue!
- manifestations:
  - – lost packets (buffer overflow at routers)
  - – long delays (queue-ing in router buffers)
- a top-10 problem!

Causes/costs of congestion: scenario

- two senders, two receivers
- one router, infinite buffers
- no retransmission







- large delays when congested
- maximum achievable throughput

Approaches towards congestion control

Two broad approaches towards congestion control:

end-to-end congestion control:

- no explicit feedback from network
- congestion inferred from end-system observed loss, delay
- approach taken by TCP

Network-assisted congestion control:

- routers provide feedback to end systems
  - single bit indicating congestion (SNA, ATM)
  - explicit rate sender should send at

TCP Congestion Control

- How to detect congestion?
- Timeout caused by packet loss: reasons

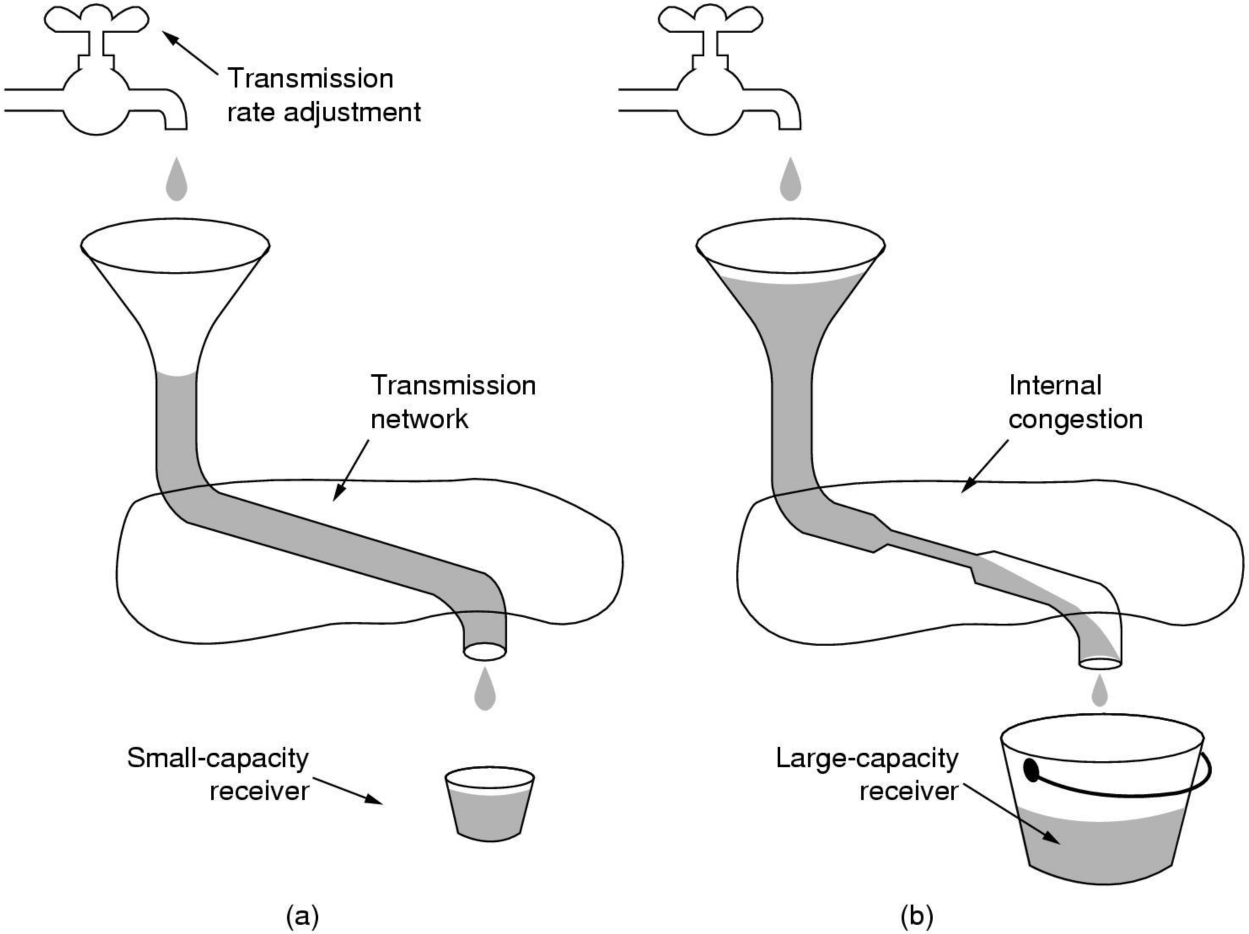
Rare for wired networks

  - Transmission errors
  - Packed discarded at congested router

Packet loss

Hydraulic example illustrating two limitations for sender!

TCP congestion control



TCP Congestion Control

- How to detect congestion?
- Timeout caused by packet loss: reasons

  Rare

  - Transmission errors
  - Packed discarded at congested router

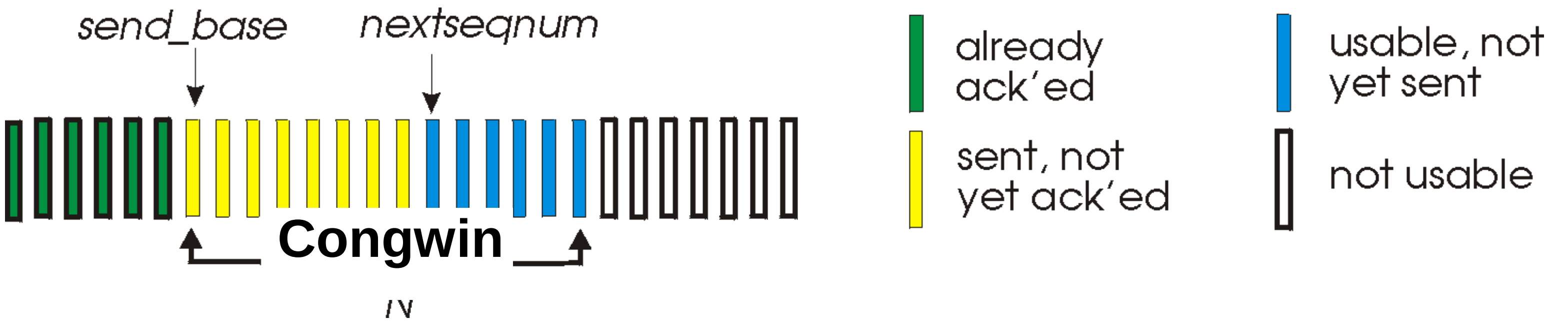
  Packet loss → congestion

Approach: 2 windows for sender

Minimum of { Receiver window / Congestion window

TCP Congestion Control

- end-end control (no network assistance)
- transmission rate limited by congestion window size, **`Congwin`**, over segments:



❑ w segments, each with MSS bytes sent in one RTT:

$$\boxed{\text{throughput} = \frac{w * MSS}{RTT} \text{ Bytes/sec}}$$

TCP Congestion Control:

- "probing" for usable bandwidth:
  - ideally: transmit as fast as possible (**`Congwin`** as large as possible) without loss
  - *increase* **`Congwin`** until loss (congestion)
  - loss: *decrease* **`Congwin`**, then begin probing (increasing) again

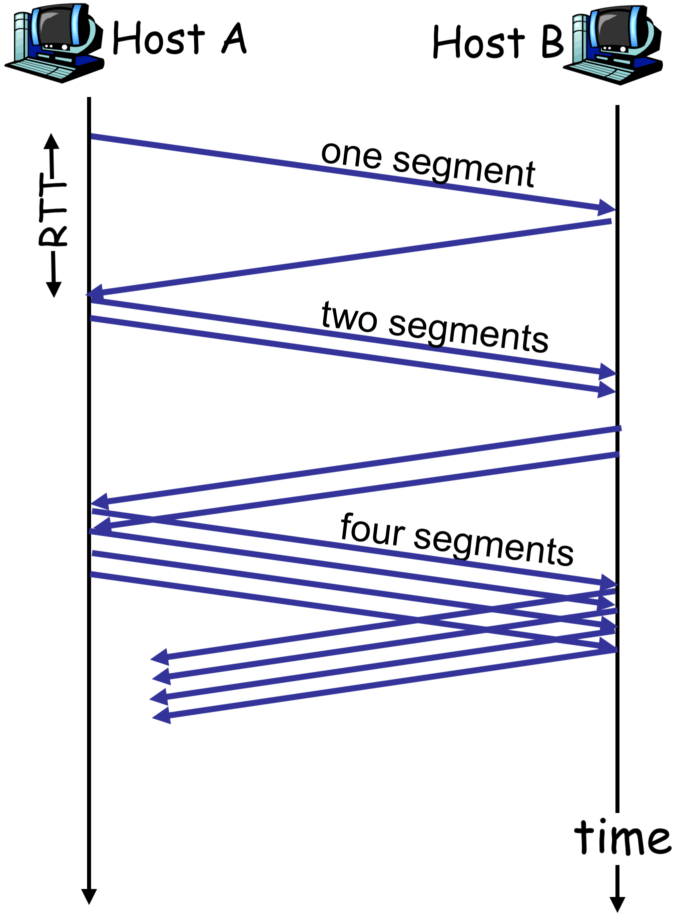
- two "phases"
  - slow start
  - congestion avoidance
- important variables:
  - **`Congwin`**
  - **`threshold:`** defines threshold between two phases:
    - slow start phase
    - congestion control phase

TCP Slow start

Slow start algorithm

```
initialize: Congwin = 1
for (each segment ACKed)
      Congwin++
until (loss event OR
        CongWin > threshold)
```

- exponential increase (per RTT) in window size (not so slow!)
- loss event: timeout (Tahoe TCP) and/or three duplicate ACKs (Reno TCP)

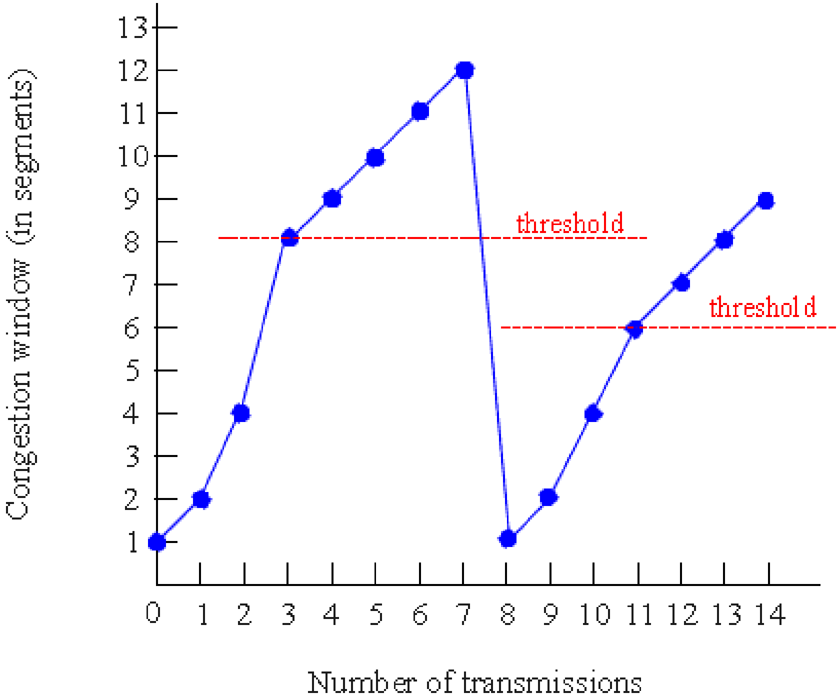


TCP Congestion Avoidance

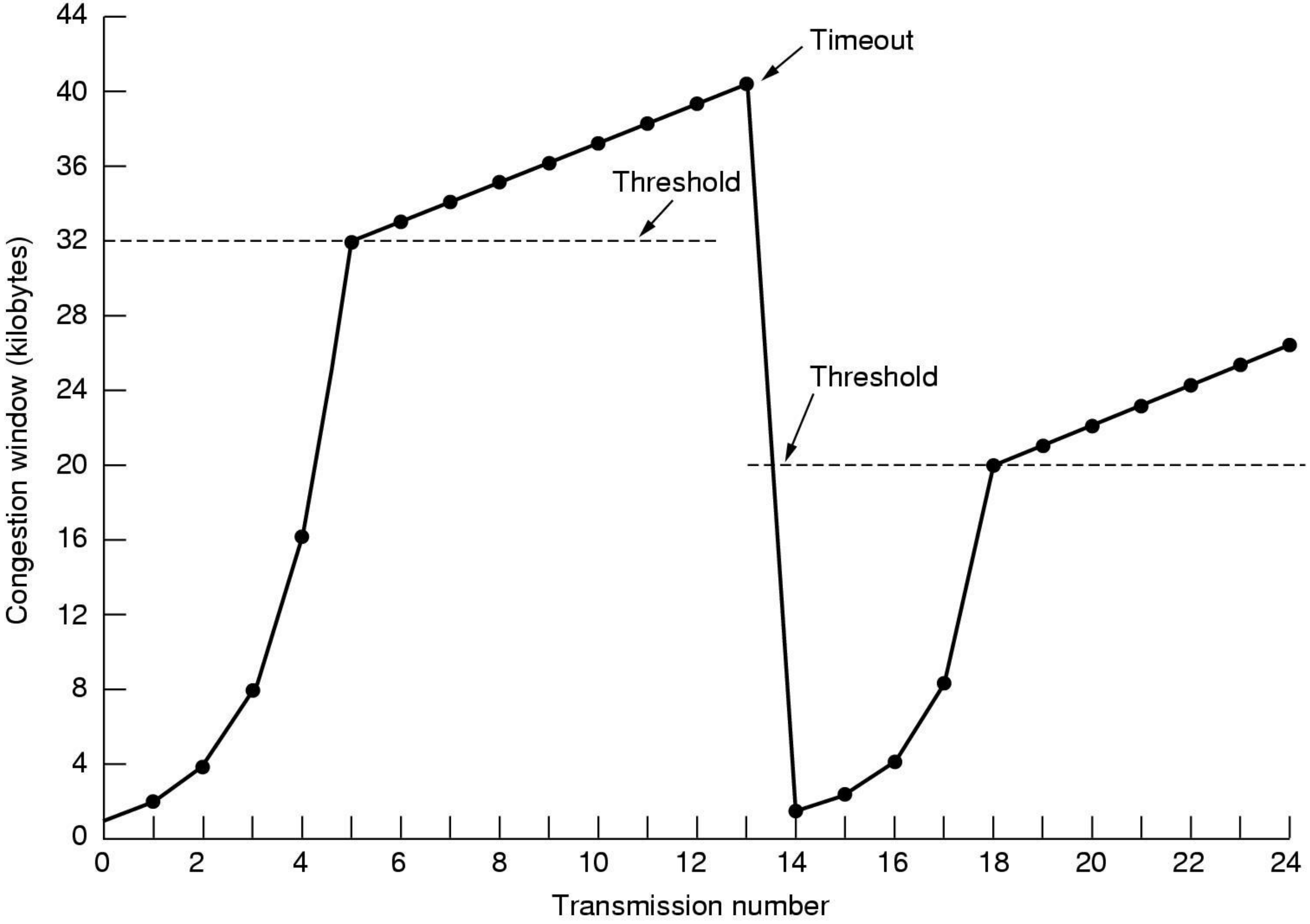
Congestion avoidance

```
/* slowstart is over       */
/* Congwin > threshold */
Until (loss event) {
  every w segments ACKed:
      Congwin++
  }
threshold = Congwin/2
Congwin = 1
perform slowstart[1]
```

1: TCP Reno skips slowstart (fast recovery) after three duplicate ACKs



TCP congestion control

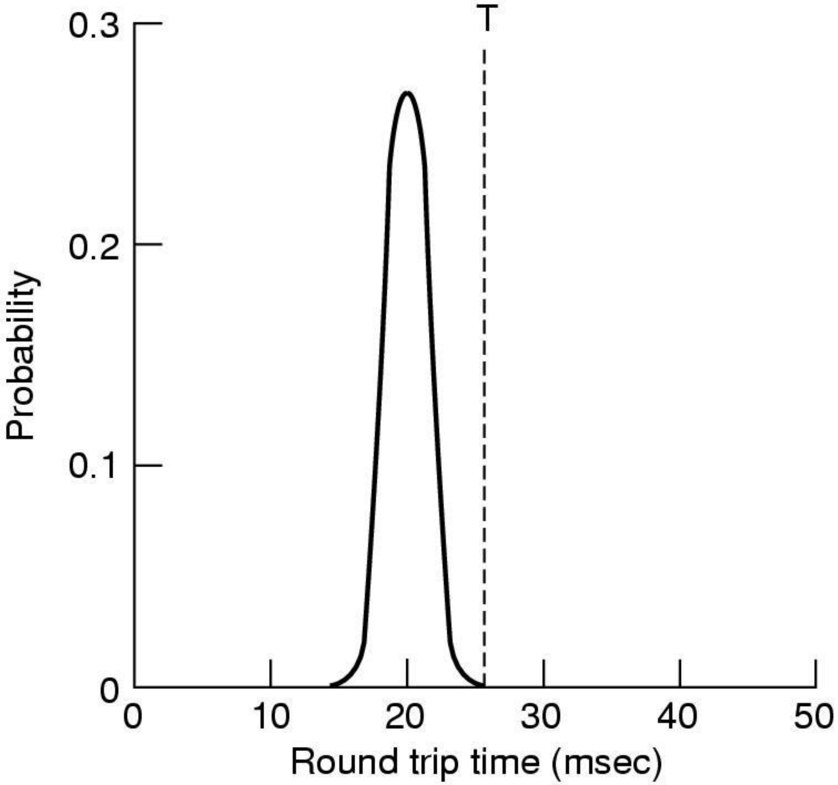


TCP timer management

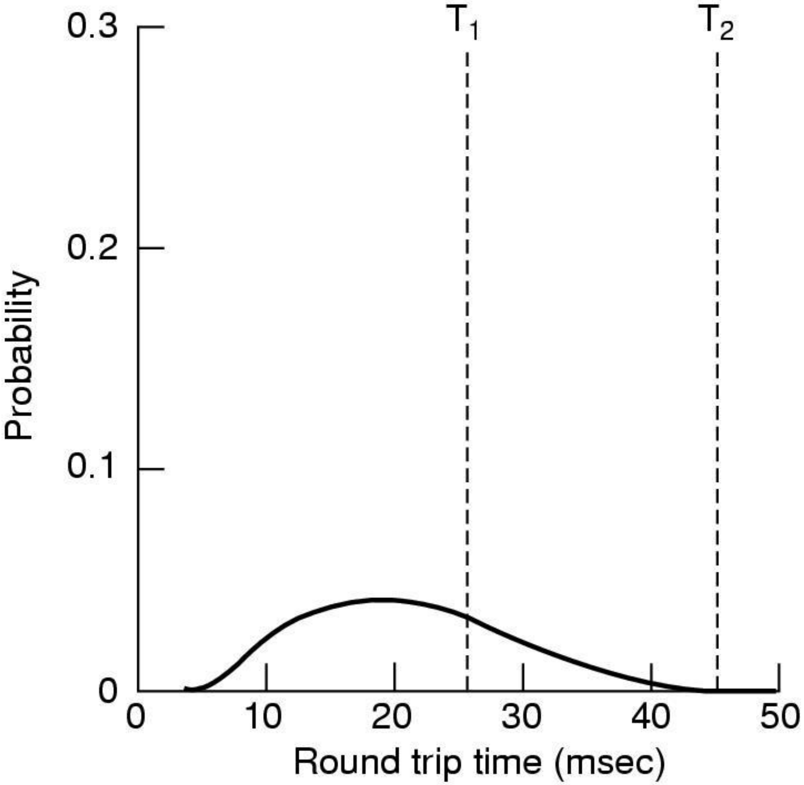
- How long should the timeout interval be?
  - Data link: expected delay predictable
  - Transport: different environment; impact of
    - Host
    - Network (routers, lines)

    unpredictable
- Consequences
  - Too small: unnecessary retransmissions
  - Too large: poor performance
- Solution: adjust timeout interval based on continuous measurements of network performance

TCP timer management



Data link layer



Transport layer

TCP timer management

- Algorithm of Jacobs

Timeout = RTT + 4 * D

  - RTT = best current estimate of the round-trip time
  - D = mean deviation (cheap estimator of the standard variance)
  - 4?
    - Less than 1% of all packets come in more than 4 standard deviations late
    - Easy to compute

TCP timer management

Timeout = RTT + 4 * D

- Algorithm of Jacobson:
  - $RTT = \alpha\ RTT + (1 - \alpha)\ M \qquad \alpha = 7/8$

    M = last measurement of round-trip time
  - $D = \alpha\ D + (1 - \alpha)\ |RTT - M|$
- Karn's algorithm: how handle retransmitted segments?
  - Do not update RTT for retransmitted segments
  - Double timeout

TCP timer management

- Other timers:
  - Persistence timer
    - Problem: lost window update packet when window is 0
    - Sender transmits probe; receivers replies with window size
  - Keep alive timer
    - Check whether other side is still alive if connection is idle for a long time
    - No response: close connection
  - Timed wait
    - Make sure all packets are died off when connection is closed
    - = 2 T

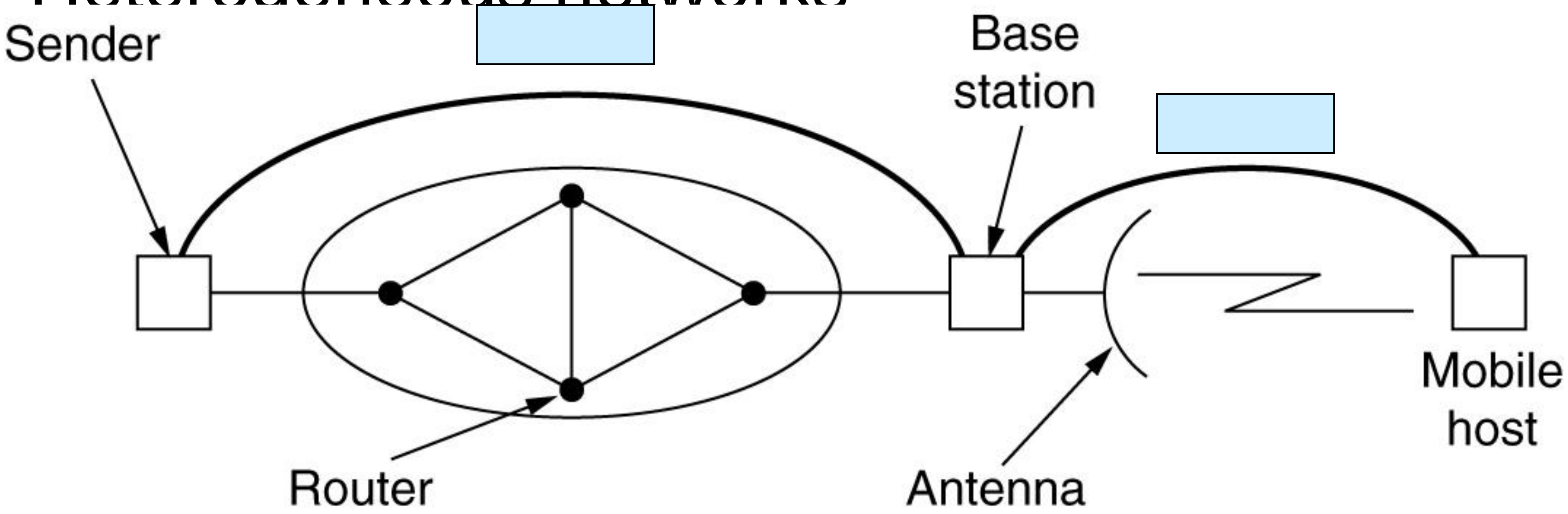
Wireless TCP & UDP

- Transport protocols
  - Independent of underlying network layer
  - BUT: carefully optimized for wired networks
  - Assumption:
    - Packet loss caused by congestion
    - Invalid for wireless networks: always loss of packets
- Congestion algorithm:
  - Timeout (= congestion) ➔ slowdown
- Soluti
  - Retransmit asap

Wireless: Lower throughput – same loss ➔ **NO** solution

Wireless TCP

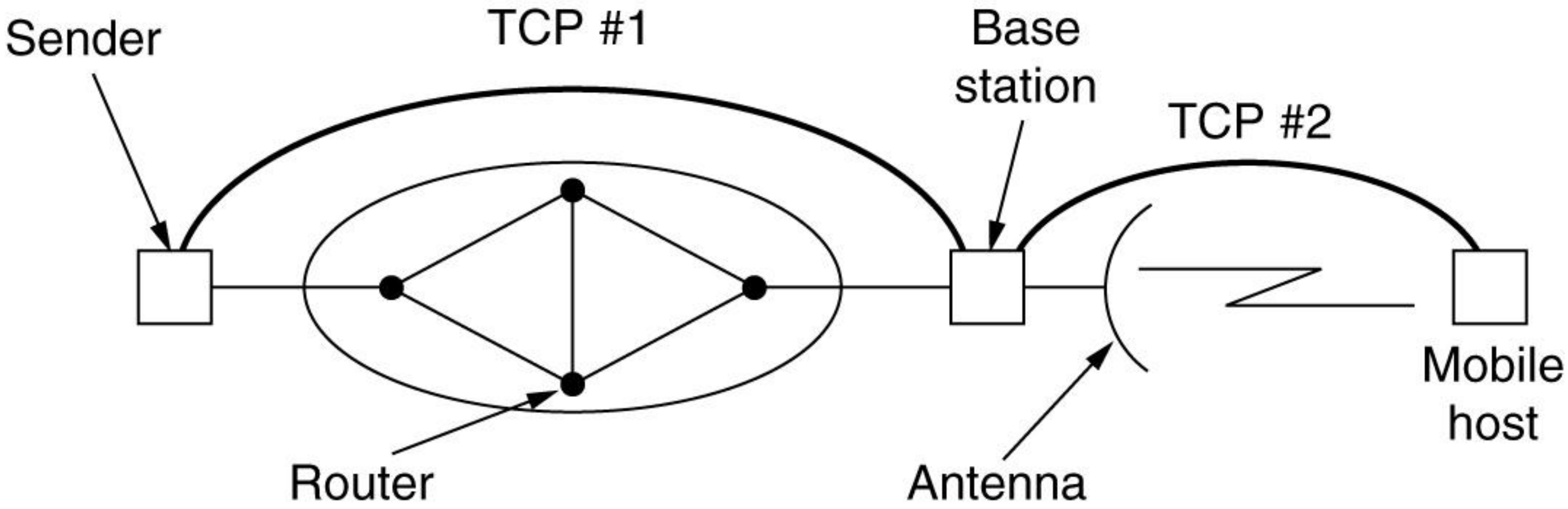
- Heterogeneous networks



- Solutions?
  - Retransmissions can cause congestion in wired network

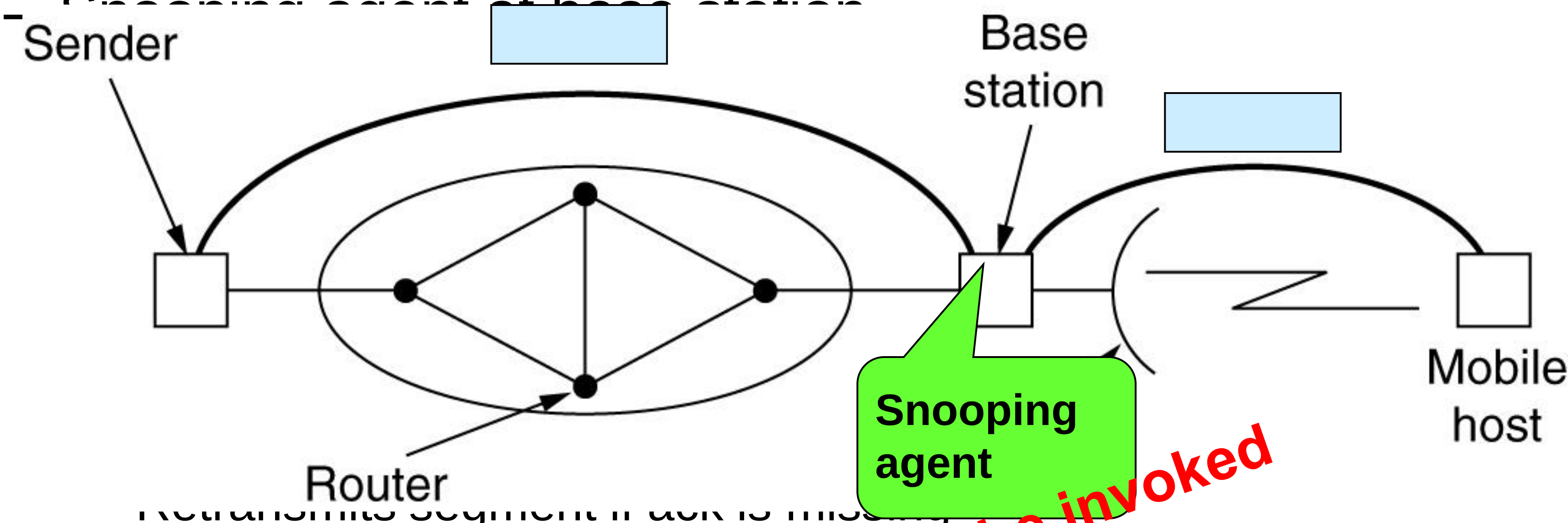
Wireless TCP

- Solutions for heterogeneous networks
  - Indirect TCP
    - + 2 homogeneous connections



Wireless TCP

- Solutions for heterogeneous networks
  - Snooping agent at base station



  - Retransmits segment if ack is missing
  - Removes duplicate acks
  - Generates selective repeat requests for segments originating at mobile host

Congestion algorithm may be invoked

Wireless UDP

- UDP = datagram service ➔ loss permitted
  no problems?

- Programs using UDP expect it to be
  highly reliable
- Wireless UDP: far from perfect!!!

➔Implications for programs recovering from lost UDP messages

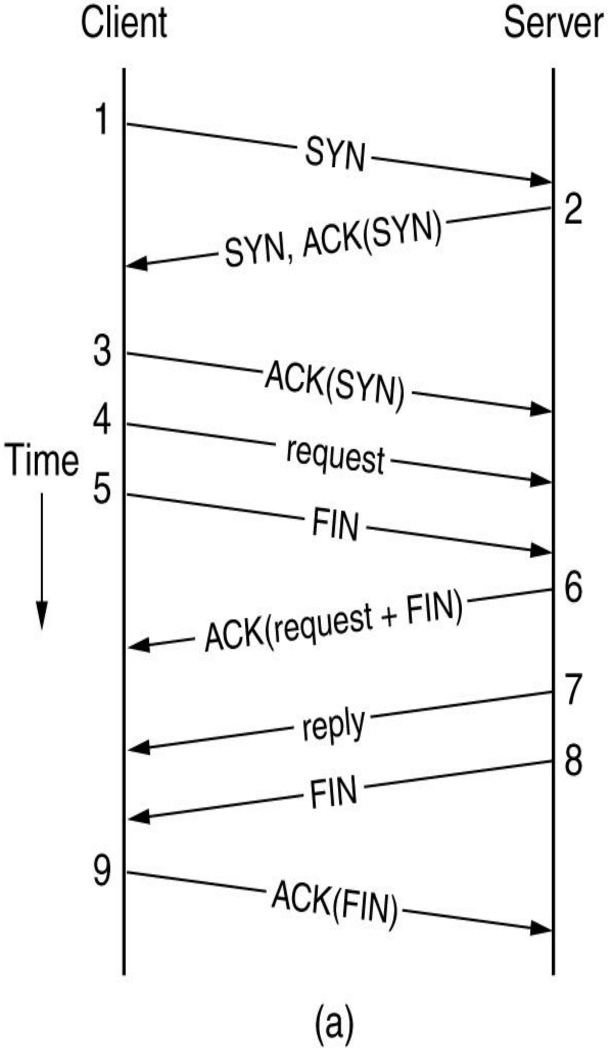
Transactional TCP

- How to implement RPC?
  - On top of UDP?
  - Yes if
    - Request and reply fit in a single packet
    - Operations are idempotent
  - Otherwise
    - Reimplementation of reliability
  - On top of TCP?

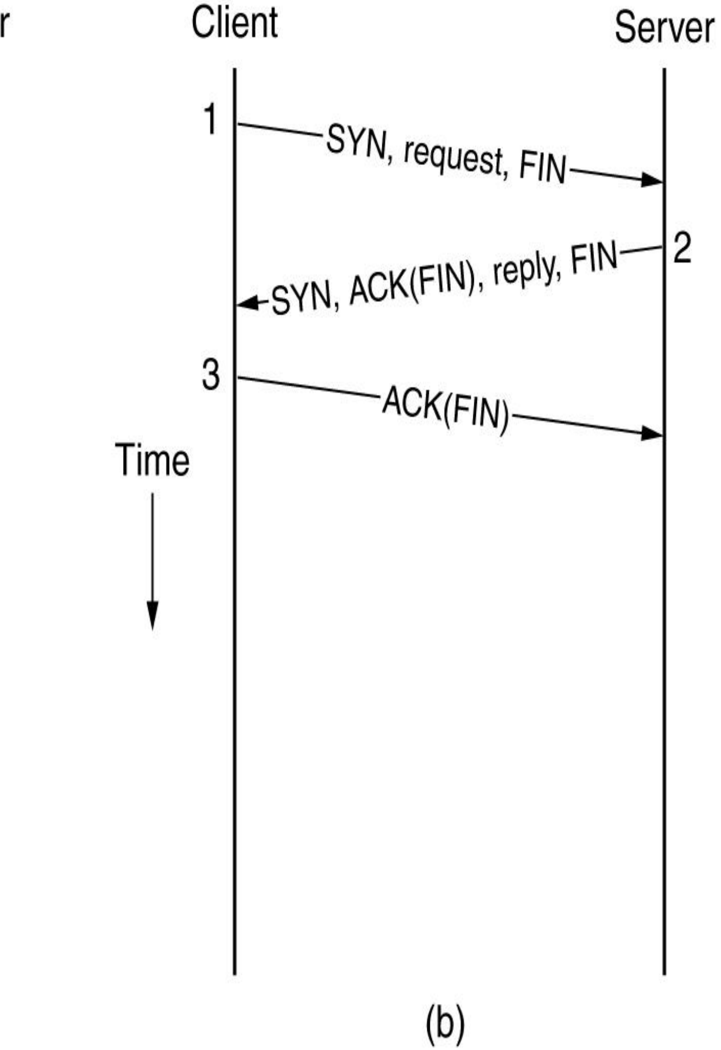
Transactional TCP

How to implement RPC?

- On top of UDP?
  - Yes if
    - Request and reply fit in a single packet
    - Operations are idempotent
  - Otherwise
    - Reimplementation of reliability
- On top of TCP?
  - Unattractive because of connection set up
- Solution: transactional TCP



Transactional TCP



How to implement RPC?

- On top of UDP?
  - Problems withreliability
- On top of TCP?
  - Overhead of connection set up
- Solution: transactional TCP
  - Allow data transfer during setup
  - Immediate close of stream

UNIT 5

INTRODUCTION TO APPLICATION LAYER

Chapter 2: Application layer

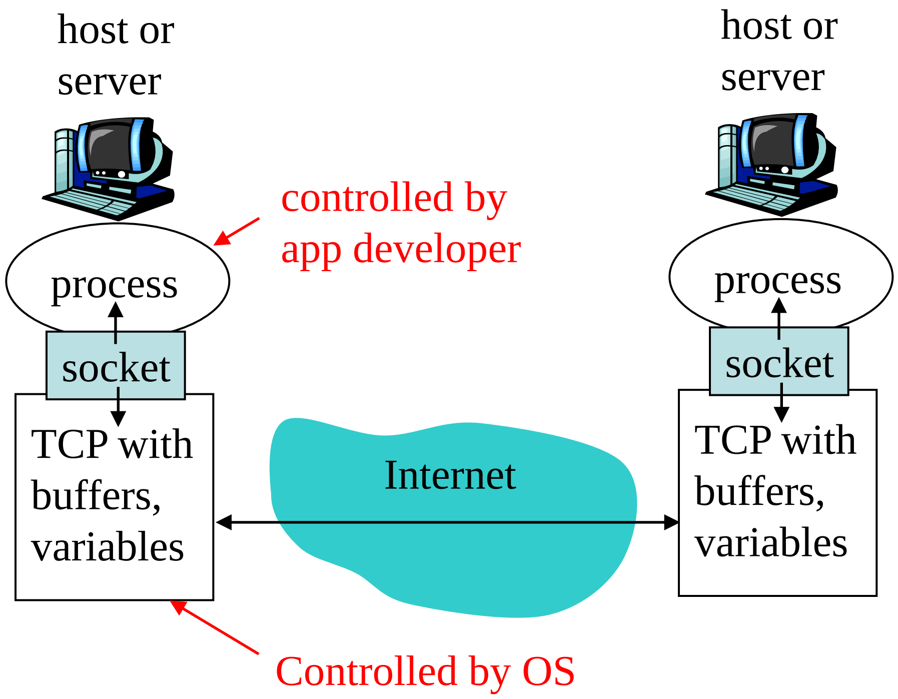
- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
    - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP

Processes communicating

Process:
: program running within a host

Client process:
: initiates communication

Server process:
: waits to be contacted


host or server
process
socket
TCP with buffers, variables
controlled by app developer
Internet
host or server
process
socket
TCP with buffers, variables
Controlled by OS

process sends/receives messages to/from its socket

identifier includes both IP address and port numbers associated with process on host.

App-layer protocol defines

- Types of messages exchanged,
  - e.g., request, response
- Message syntax:
  - what fields in messages & how fields are delineated
- Message semantics
  - meaning of information in fields
- Rules for when and how processes send & respond to messages

Public-domain protocols:

- ♦ defined in RFCs
- ♦ allows for interoperability
- ♦ e.g., HTTP, SMTP

Proprietary protocols:

- ♦ e.g., Skype

Transport service requirements of common apps

Application	Data loss	Throughput	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100's msec
instant messaging	no loss	elastic	yes and no

Internet transport protocols services

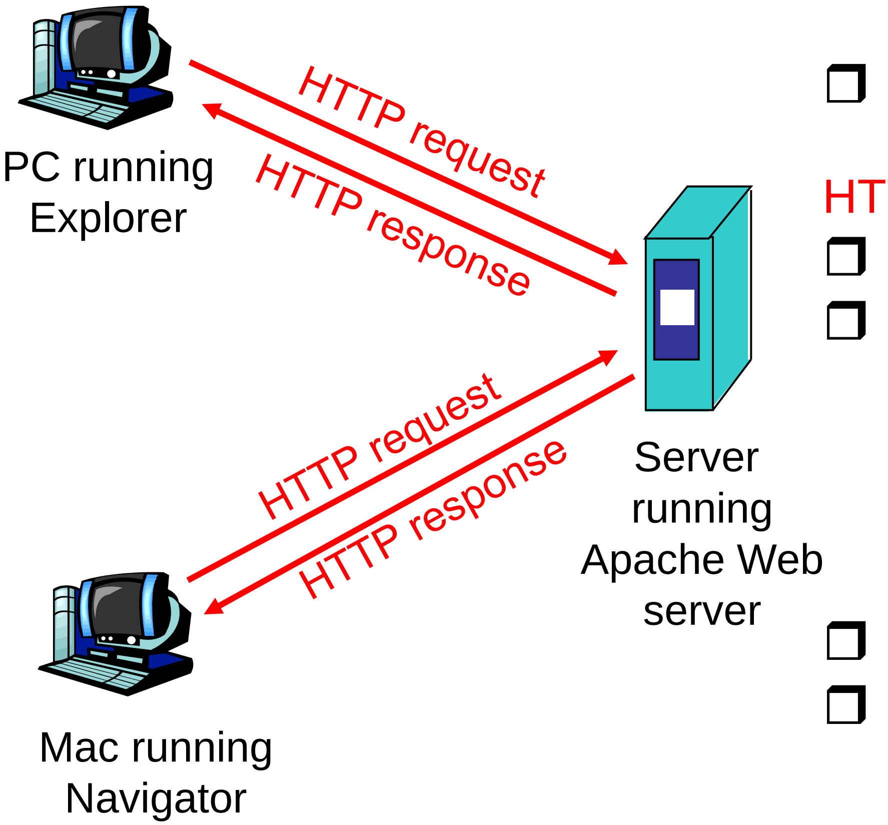
TCP service:

- *connection-oriented:* setup required between client and server processes
- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not provide:* timing, minimum throughput guarantees, security

UDP service:

- unreliable data transfer between sending and receiving process
- does not provide: connection setup, reliability, flow control, congestion control, timing, throughput guarantee, or security

HTTP overview



- ❒ Web page consists of base HTML-file which includes several referenced objects
- ❒ Each object is addressable by a URL

HTTP: hypertext transfer protocol

- ❒ Web's application layer protocol
- ❒ client/server model
  - ❖ *client:* browser that requests, receives, "displays" Web objects
  - ❖ *server:* Web server sends objects in response to requests
- ❒ uses TCP
- ❒ is "stateless"

HTTP connections

Nonpersistent HTTP

- At most one object is sent over a TCP connection.

Persistent HTTP

- Multiple objects can be sent over single TCP connection between client and server.

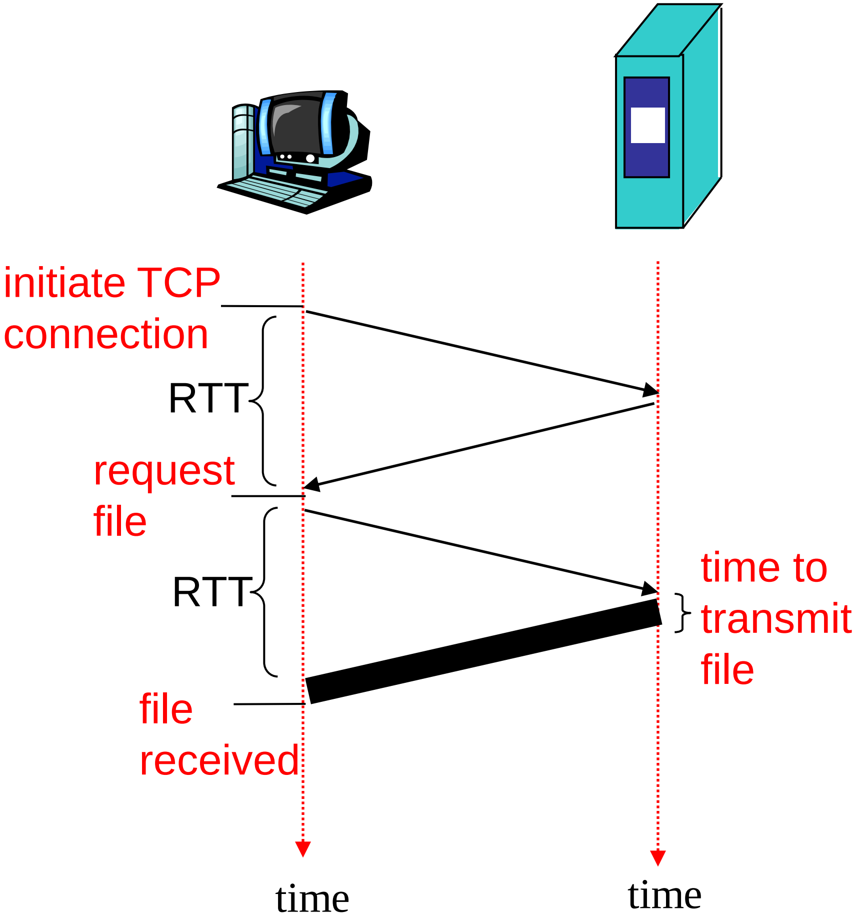
Non-Persistent HTTP: Response time

Definition of RTT: time for a small packet to travel from client to server and back.

Response time:

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- file transmission time

total = 2RTT+transmit time



Persistent HTTP

Nonpersistent HTTP issues:

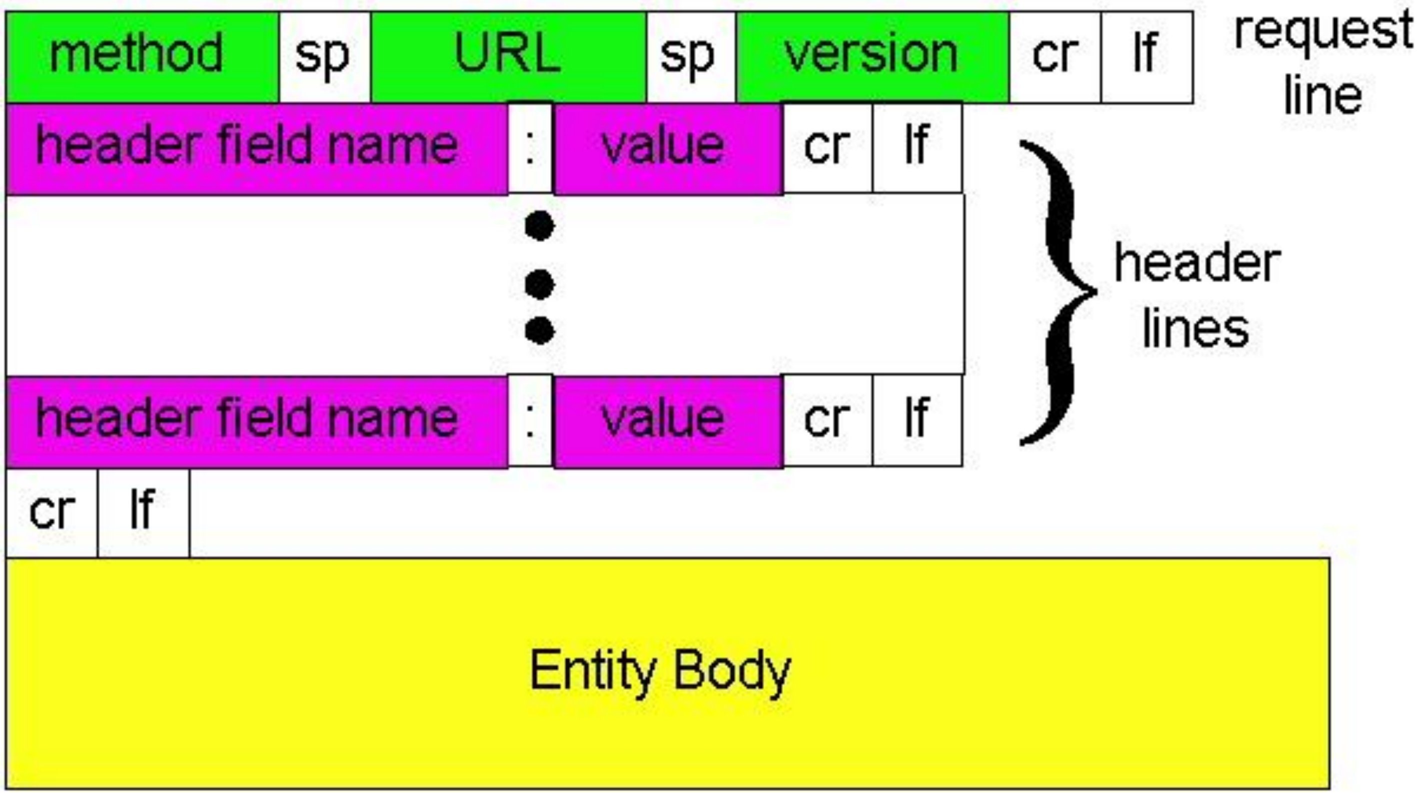
- requires 2 RTTs per object
- OS overhead for *each* TCP connection
- browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

HTTP messages

- two types of HTTP messages: *request*, *response*
- HTTP request message:
  - ASCII (human-readable format)



Method types

HTTP/1.0

- GET
  - request an object from server
- POST
  - upload information using forms
- HEAD
  - asks server to leave requested object out of response

HTTP/1.1

- GET, POST, HEAD
- PUT
  - uploads file in entity body to path specified in URL field
- DELETE
  - deletes file specified in the URL field

Cookies: Keeping state

What cookies can bring:

- authorization
- shopping carts
- recommendations
- user session state (Web e-mail)

aside

Cookies and privacy:

- cookies permit sites to learn a lot about you
- you may supply name and e-mail to sites

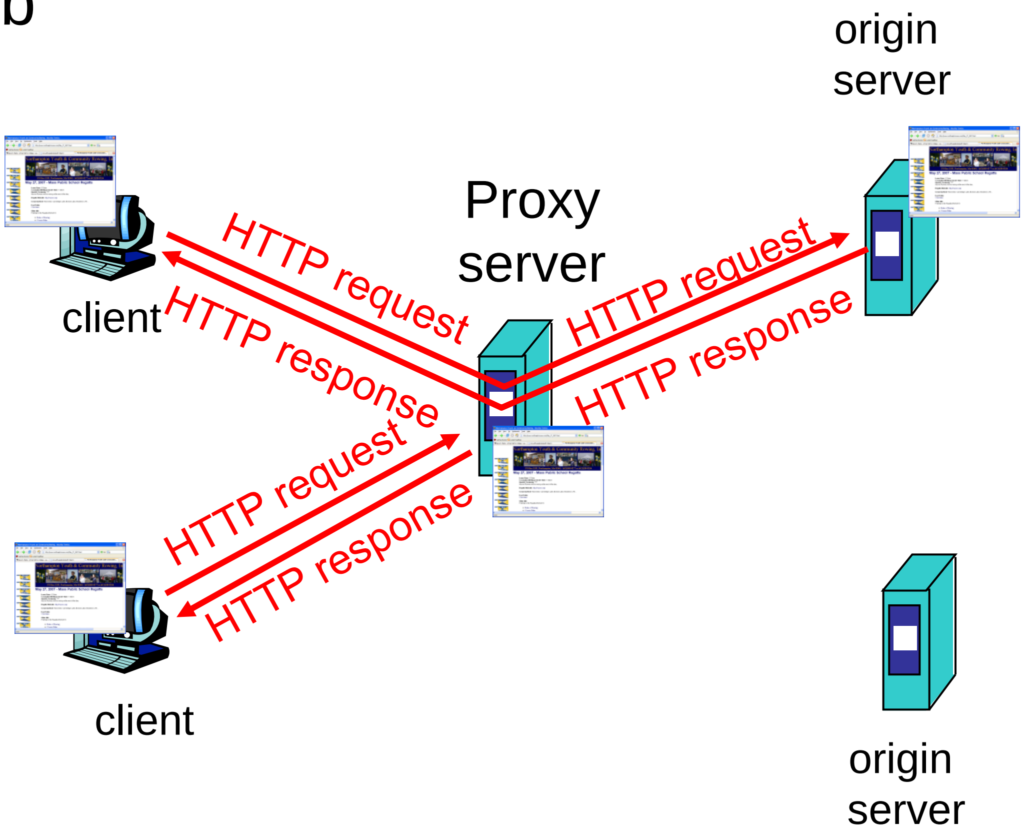
How to keep "state":

- protocol endpoints: maintain state at sender/receiver over multiple transactions
- cookies: http messages carry state

Web caches (proxy server)

Goal: satisfy client request without involving origin server

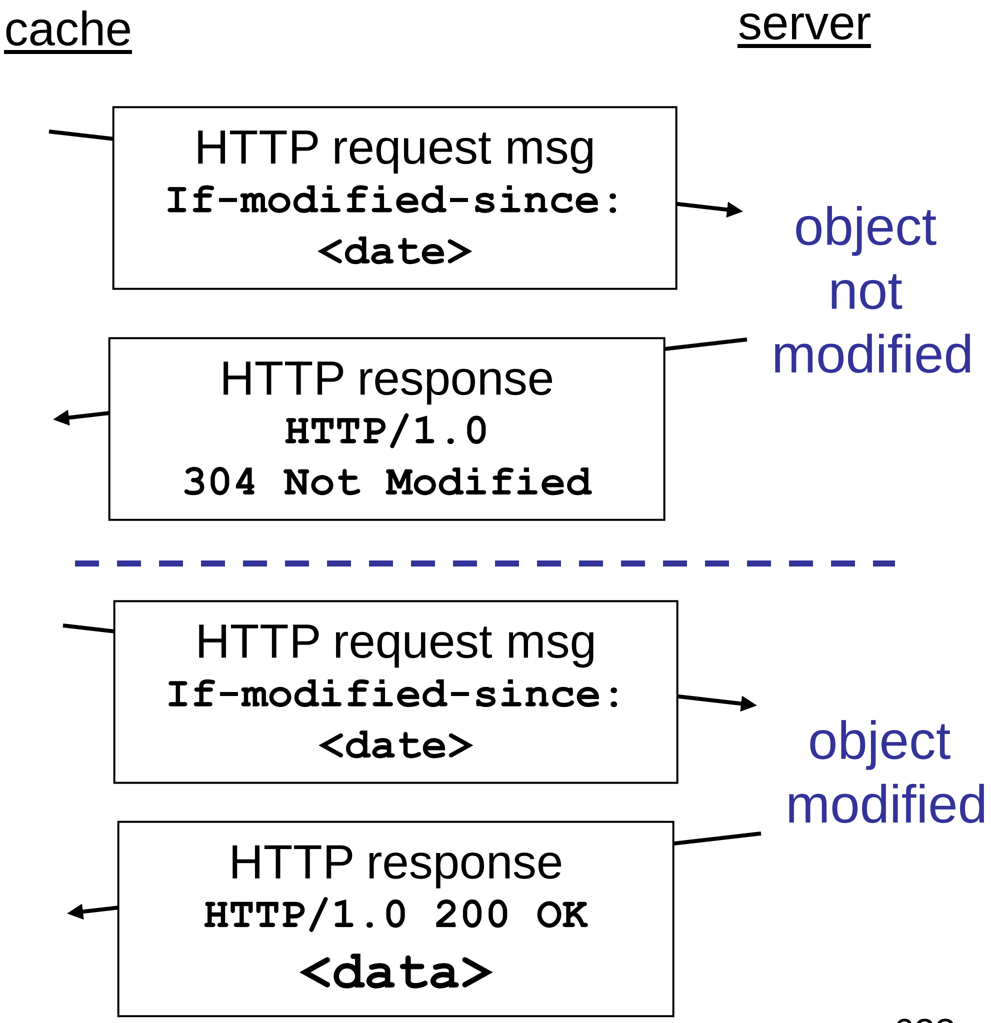
- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache



- Why Web caching?
  - reduce response time for client request
  - reduce traffic on an institution’s access link.
  - enables “poor” content providers to effectively deliver content

Conditional GET

- Goal: don’t send object if cache has up-to-date cached version
- **cache:** specify date of cached copy in HTTP request
  `If-modified-since: <date>`
- **server:** response contains no object if cached copy is up-to-date:
  `HTTP/1.0 304 Not Modified`



cache

server

HTTP request msg
`If-modified-since: <date>`

object not modified

HTTP response
`HTTP/1.0`
`304 Not Modified`

HTTP request msg
`If-modified-since: <date>`

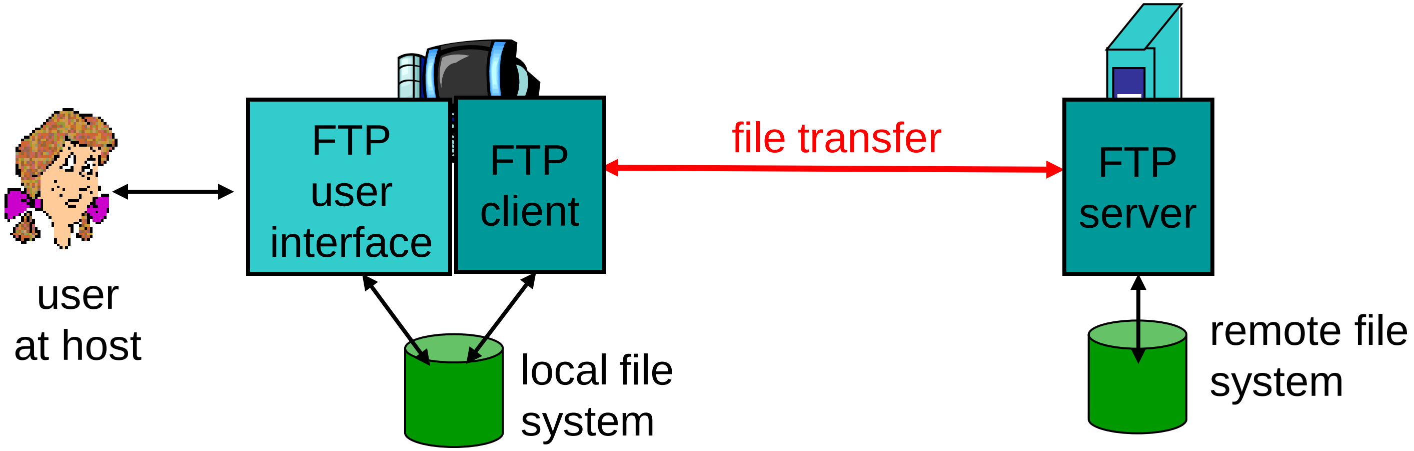
object modified

HTTP response
`HTTP/1.0 200 OK`
`<data>`

Lecture 5: Outline

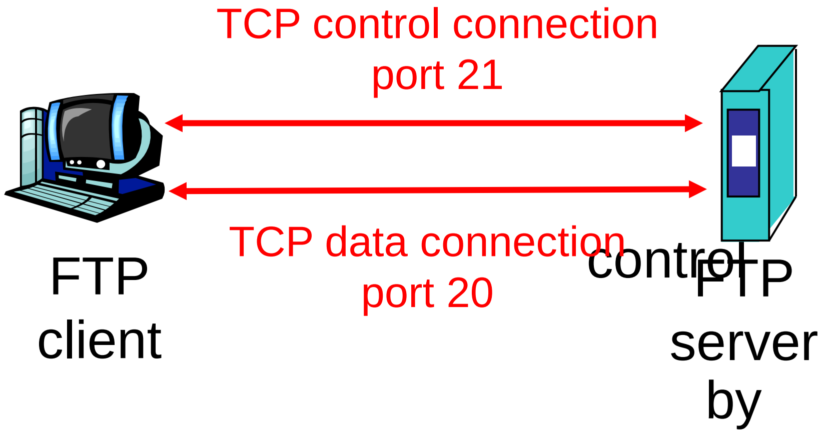
- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
  - SMTP, POP3, IMAP
- 2.5 DNS

FTP: the file transfer protocol



- transfer file to/from remote host
- client/server model
  - *client:* side that initiates transfer (either to/from remote)
  - *server:* remote host
- ftp: RFC 959
- ftp server: port 21

FTP: separate control, data connections



- FTP client contacts FTP server port 21
- client authorized over connection
- client browses remote directory sending commands over control connection.
- when server receives file transfer command, server opens *2nd* TCP connection (for file) to client
- after transferring one file, server closes data connection.
- server opens another TCP data connection to transfer another file.
- control connection: "out of band"
- FTP server maintains "state": current directory, earlier authentication

FTP commands, responses

Sample commands:

- sent as ASCII text over control channel
- `USER` ***username***
- `PASS` ***password***
- `LIST` return list of file in current directory
- `RETR filename` retrieves (gets) file
- `STOR filename` stores (puts) file onto remote host

Sample return codes

- status code and phrase (as in HTTP)
- `331 Username OK, password required`
- `125 data connection already open; transfer starting`
- `425 Can't open data connection`
- `452 Error writing file`

FTP issues

- Multiple connections are used
  - for each directory listing and file transmission
- No integrity check at receiver
- Messages are sent in clear text
  - including Passwords and file contents
  - can be sniffed by eavesdroppers
- Solution
  - Secure FTP (SSH FTP)
    - allows a range of operations on remote files
  - FTPS (FTP over Secure Sockets Layer (SSL))
  - Transport Layer Security (TLS) encryption

Lecture 5: Outline

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
  - SMTP
  - POP3
  - IMAP
- 2.5 DNS

Electronic Mail

🟩 outgoing message queue

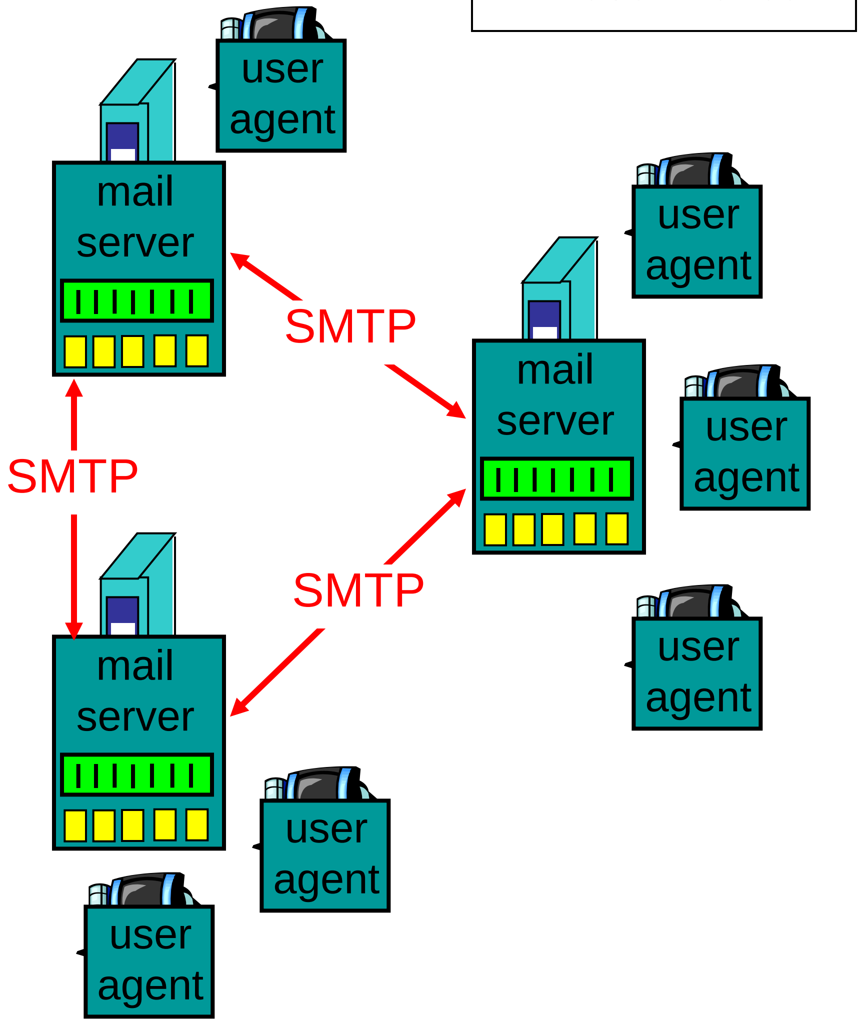
🟨 user mailbox

Three major components:

- user agents
- mail servers
- simple mail transfer protocol: SMTP

User Agent

- a.k.a. “mail reader”
- composing, editing, reading mail messages
- e.g., Eudora, Outlook, elm, Mozilla Thunderbird
- outgoing, incoming messages stored on server

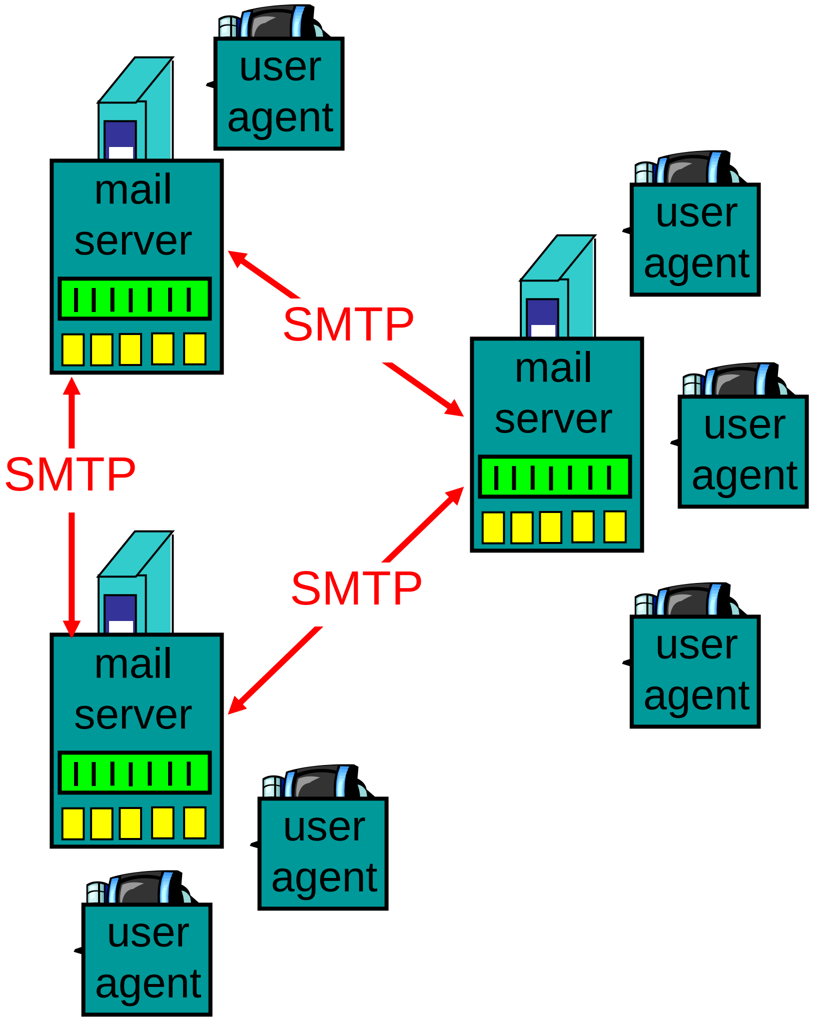


Electronic Mail: mail servers

Mail Servers

- mailbox contains incoming messages for user
- message queue of outgoing (to be sent) mail messages

- SMTP protocol between mail servers to send email messages
  - client: sending mail server
  - "server": receiving mail server



Electronic Mail: SMTP [RFC 2821]

- uses TCP to reliably transfer email message from client to server (port 25)
- direct transfer: sending server to receiving server
- three phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - closure
- command/response interaction
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCII

Scenario: Alice sends message to Bob

1) Alice uses UA to compose message and "to"
   `bob@someschool.edu`

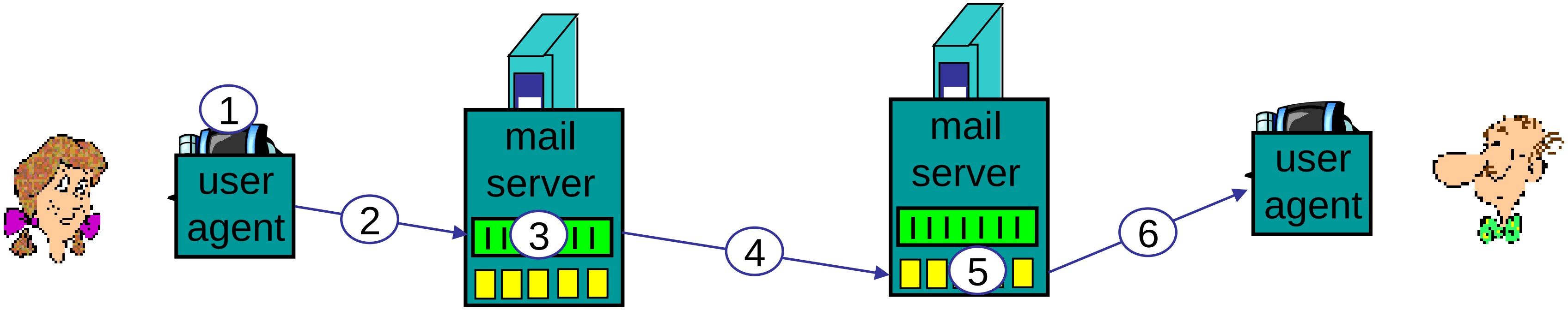
2) Alice's UA sends message to her mail server; message placed in message queue

3) Client side of SMTP opens TCP connection with Bob's mail server

4) SMTP client sends Alice's message over the TCP connection

5) Bob's mail server places the message in Bob's mailbox

6) Bob invokes his user agent to read message



Sample SMTP interaction

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

SMTP: final words

- SMTP uses persistent connections
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses `CRLF.CRLF` to determine end of message

Comparison with HTTP:

- HTTP: pull
- SMTP: push
- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response msg
- SMTP: multiple objects sent in multipart msg

Mail message format

SMTP: protocol for exchanging email msgs

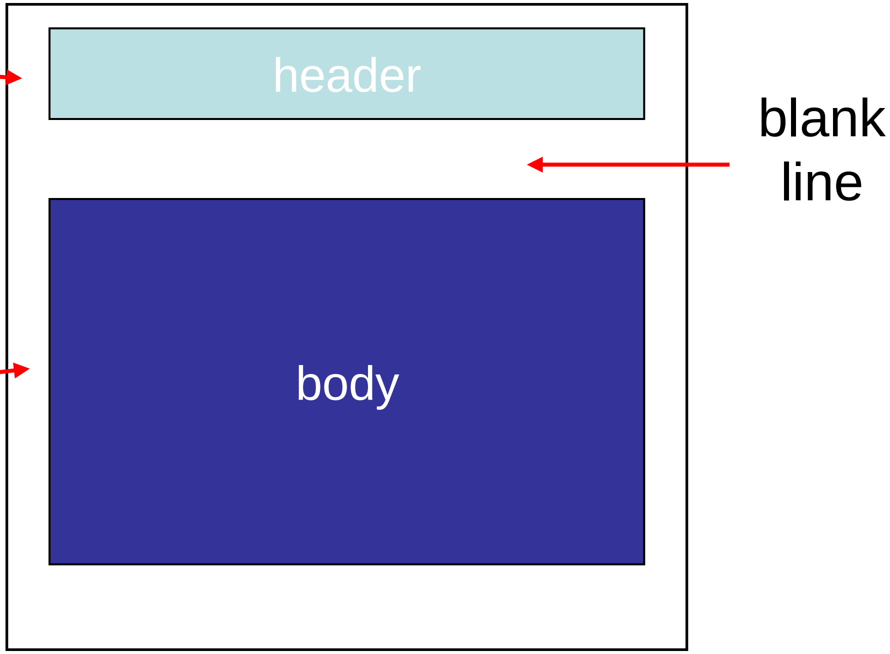
RFC 822: standard for text message format:

- header lines, e.g.,
  - To:
  - From:
  - Subject:

  different from SMTP commands!

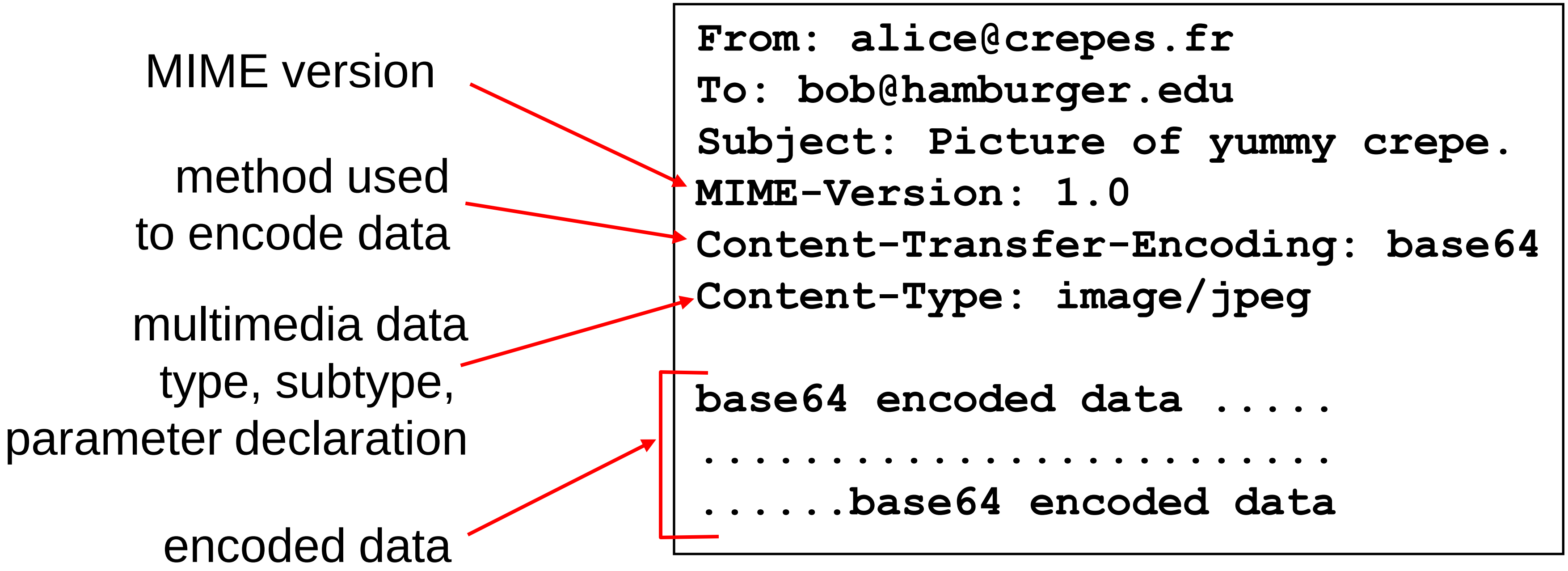
- body
  - the “message”,

  ASCII characters only



Message format: multimedia extensions

- MIME: multimedia mail extension, RFC 2045, 2056
- additional lines in msg header declare MIME content type



MIME version

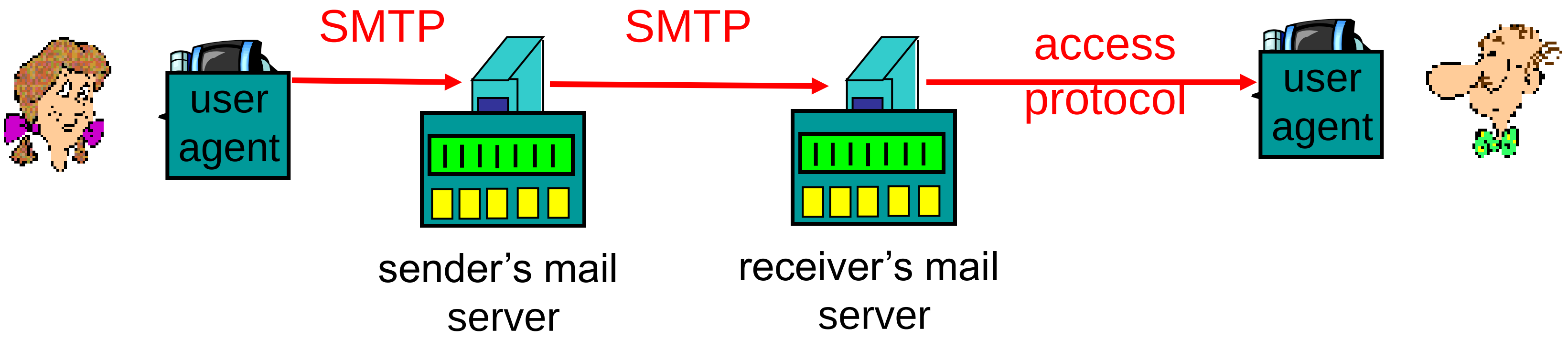
method used to encode data

multimedia data type, subtype, parameter declaration

encoded data

```
From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
```

Mail access protocols



- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]
    - authorization (agent <-->server) and download
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - more features (more complex)
    - manipulation of stored msgs on server
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.

POP3 protocol

authorization phase

- client commands:
  - **`user`**: declare username
  - **`pass`**: password
- server responses
  - **`+OK`**
  - **`-ERR`**

transaction phase, client:

- **`list`**: list message numbers
- **`retr`**: retrieve message by number
- **`dele`**: delete
- **`quit`**

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 (more) and IMAP

More about POP3

- Previous example uses “download and delete” mode.
- Bob cannot re-read e-mail if he changes client
- “Download-and-keep”: copies of messages on different clients
- POP3 is stateless across sessions

IMAP

- Keep all messages in one place: the server
- Allows user to organize messages in folders
- IMAP keeps user state across sessions:
  - names of folders and mappings between message IDs and folder name

Try SMTP interaction for yourself:

- **`telnet servername 25`**
- see 220 reply from server
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

above lets you send email without using email client (reader)

Lecture 5: Outline

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
  - SMTP
  - POP3
  - IMAP
- 2.5 DNS

DNS: Domain Name System

People: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- "name", e.g., ww.yahoo.com - used by humans

Domain Name System:

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol* host, routers, name servers to communicate to *resolve* names (address/name translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"

DNS services

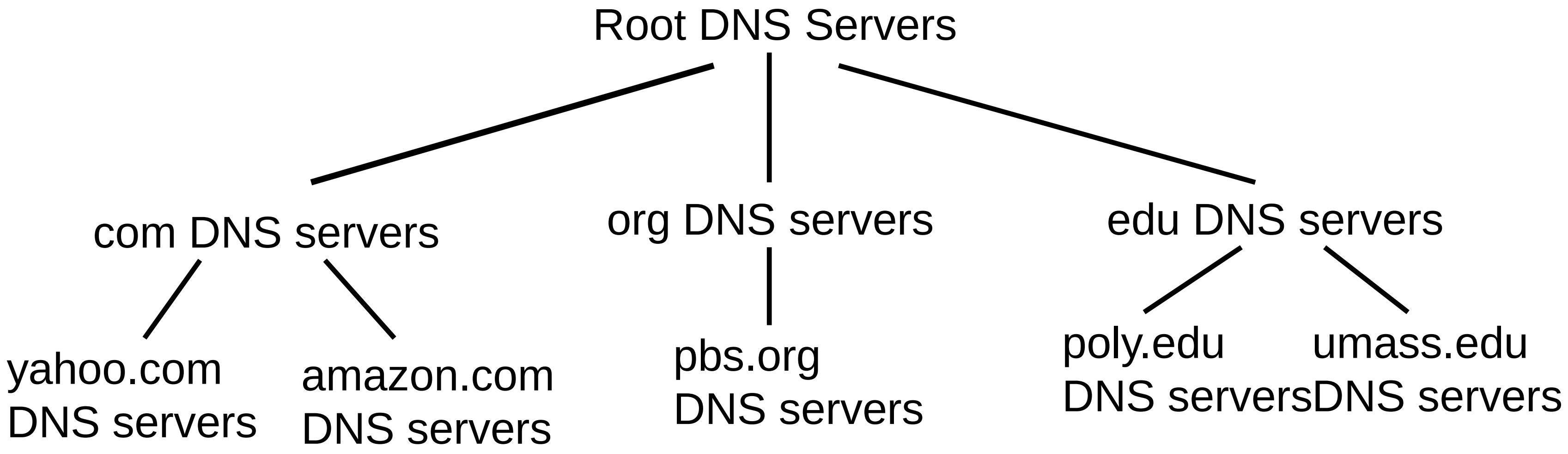
- hostname to IP address translation
- host aliasing
  - Canonical, alias names
- mail server aliasing
- load distribution
  - replicated Web servers: set of IP addresses for one canonical name

Why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

doesn’t *scale!*

Distributed, Hierarchical Database



Client wants IP for www.amazon.com; 1st approx:

- client queries a root server to find com DNS server
- client queries com DNS server to get amazon.com DNS server
- client queries amazon.com DNS server to get IP address for www.amazon.com

Lecture 5: Summary

- Application
- Web and HTTP
- File Transfer Protocol
- Electronic Mail
  - SMTP
  - POP3
  - IMAP
- Domain Name Service