UNIT-1

Introduction: - Types of Computer Networks, Reference Models - ISO-OSI Reference Model, TCP/IP Reference Model – Comparison of OSI and TCP/IP reference models.

COMPUTER NETWORK

A **computer network** is a set of computers connected together for the purpose of sharing resources. Computer Network is combination of hardware, software, and cabling, which together allow multiple computing devices to communicate (movement of meaningful **information**) with each other

Advantages of Network

- **Speed**. Sharing and transferring files within Networks are very rapid. Thus saving time, while maintaining the integrity of the file.
- **Cost**. Individually licensed copies of many popular software programs can be costly. Networkable versions are available at considerable savings. Shared programs, on a network allows for easier upgrading of the program on one single file server, instead of upgrading individual workstations.
- **Security**. Sensitive files and programs on a network are passwords protected or designated as "copy inhibit," so that you do not have to worry about illegal copying of programs.
- **Centralized Software Management**. Software can be loaded on one computer (the file server) eliminating that need to spend time and energy installing updates and tracking files on independent computers throughout the building.
- **Resource Sharing.** Resources such as, printers, fax machines and modems can be shared.
- **Electronic Mail.** E-mail aids in personal and professional communication.
- **Flexible Access.** Access their files from computers throughout the firm.
- **Workgroup Computing**. Workgroup software (such as Microsoft BackOffice) allows many users to work on a document or project concurrently.

Categories (Types) and connections of networks

TYPES OF NETWORK

Networks are discussed in terms of their (i) transmission technology/type of connection and their (ii) scale / size

Types of Network Based on the *Transmission Technology* / Type of Connection

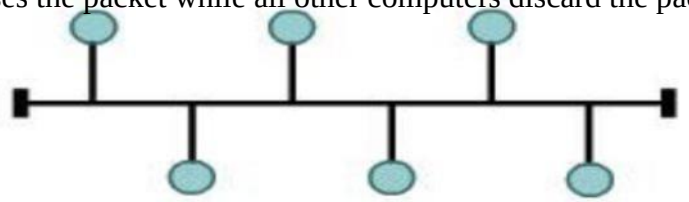
1 .Broadcast Networks (Multi point)

2 .Point-to-point Networks

1 .Broadcast Networks (Multi point)

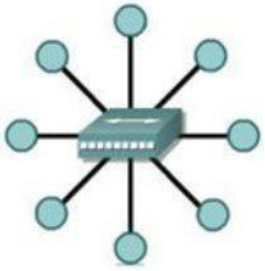
Single communication channel shared by all computers

- Packets (short messages) sent by a computer contains an address specifying the destination computer
- All computers connected to the network receive the packet. The destination computer processes the packet while all other computers discard the packet



- In multipoint (multi drop) connection the capacity of channel is shared either temporary or spatially

2. Point-to-Point Networks



- Consists of many connections between individual pairs of computers
- Sending a packet between two computer may involve the packet being forwarded by intermediate computers
- Multiple routes of different lengths possible, therefore routing algorithms are important

Types of Network Based on the SCALE / SIZE

- PAN (Personal Area Network)
- LAN (Local Area Network),
- MAN (Metropolitan area network)
- WAN (Wide Area Network)

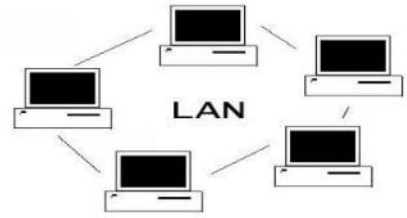
PAN (Personal Area Network)

A personal area network (PAN) is a computer network used for data transmission amongst devices such as computers, telephones, tablets and personal digital assistants.

A wireless personal area network (WPAN) is a low-powered PAN carried over a short-distance wireless network technology such as Bluetooth, ZigBee etc.

LAN (Local area networks)

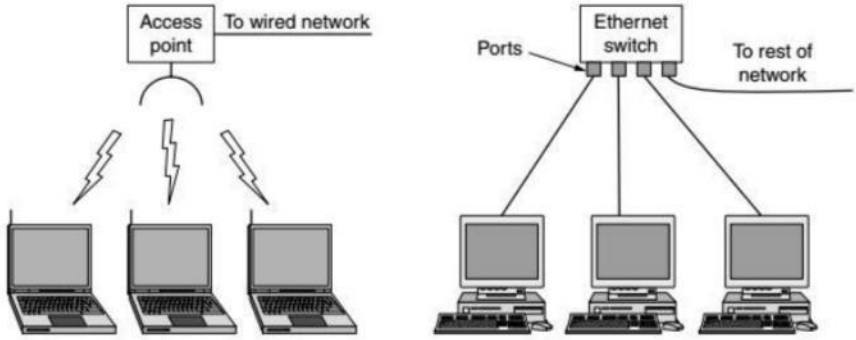
Generally called **LANs**, are privately-owned **networks within a single building or campus of up to a few kilometers in size.** They are widely used to connect personal computers and workstations in company offices and factories to share resources (e.g., printers) and exchange information.



LAN configuration consists of:

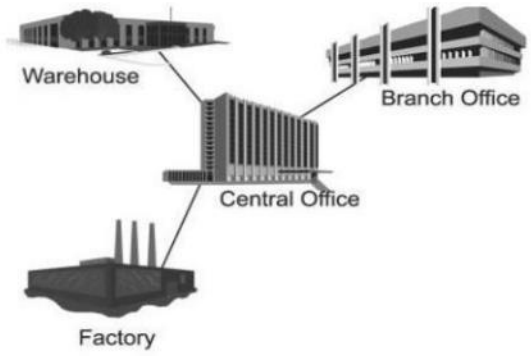
- A file server
- A workstation
- Cables

The topology of many wired LANs is built from point-to-point links. IEEE 802.3, popularly called **Ethernet**



Wireless and wired LANs. (a) 802.11. (b) Switched Ethernet.

MAN (Metropolitan area network)

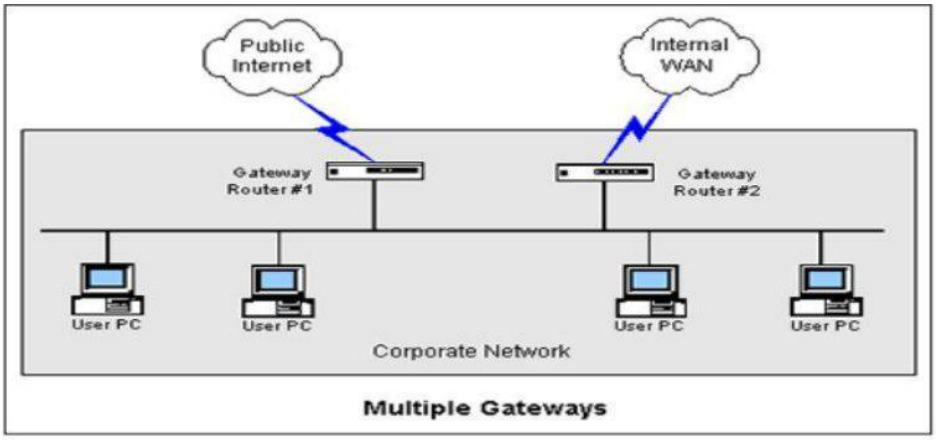


A metropolitan area network (MAN) is a large **computer network that usually spans a city or a large campus.** A MAN usually interconnects a number of local area networks (LANs) using a high-capacity backbone technology, such as fiber-optical links, and provides uplink services to wide area networks and the Internet

Eg. A metropolitan area network based on cable TV.

WAN (Wide Area Network)

A **WAN** spans a **large geographic area, such as a state, province or country.** WANs often connect multiple smaller networks, such as local area networks (LANs) or metro area networks (MANs). Eg. ISP network.



Internetwork or internet

A collection of interconnected networks is called an internetwork or internet. These terms will be used in a generic sense, in contrast to the worldwide Internet (which is one specific internet, which we will always capitalize. The Internet uses ISP networks to connect enterprise networks, home networks, and many other networks.

Connecting a LAN and a WAN or connecting two LANs is the usual way to form an internetwork, A gateway is used to performs necessary translation, both in terms of hardware and software to establish a connection between two or more networks.

Types of Network Based on the SCALE / SIZE (Summary)

	Name of Network	Inter-processor distance	Processor Located in the same
1	PAN	1m	Square meter
2	LAN	10m	Room
		100m	Building
		1Km	Campus
3	MAN	10 Km	City
4	WAN	100 Km	Country
		1000 Km	Continent
5	Internet	10,000 km	Planet

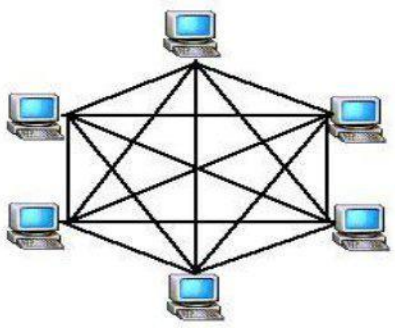
NETWORK TOPOLOGIES

Topology refers to the way a network is laid out either physically or logically. Two or more devices connect to a link; two or more links form a topology. It is the geographical representation of the relationship of all the links and linking devices to each other.

1. Mesh Topology
2. Star Topology
3. Tree (Extended Star) Topology
4. Bus Topology
5. Ring Topology
6. Hybrid Topology

1. Mesh Topology:

Here every device has a dedicated point to point link to every other device. A fully connected mesh can have n(n-1)/2 physical channels to link n devices. It must have n-1 IO ports.



Advantages:

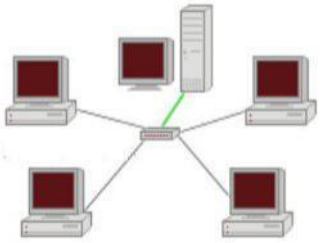
1. They use dedicated links so each link can only carry its own data load. So traffic problem can be avoided.
2. It is robust. If any one link get damaged it cannot affect others
3. It gives privacy and security
4. Fault identification and fault isolation are easy.

Disadvantages:

1. The amount of cabling and the number IO ports required are very large. Since every device is connected to each other devices through dedicated links.
2. The sheer bulk of wiring is larger then the available space
3. Hardware required to connect each device is highly expensive.

2. STAR TOPOLOGY:

Here each device has a dedicated link to the central ‘hub’. There is no direct traffic between devices. The transmission are occurred only through the central controller namely hub.



Advantages:

1. Less expensive then mesh since each device is connected only to the hub.
2. Installation and configuration are easy.
3. Less cabling is need then mesh.
4. Robustness.
5. Easy to fault identification & isolation.

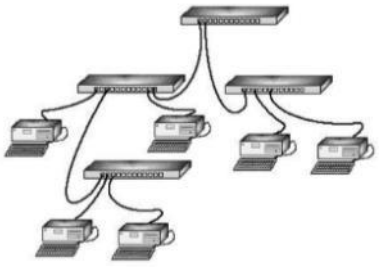
Disadvantages:

1. Even it requires less cabling then mesh when compared with other topologies it still large.

3. TREE TOPOLOGY (Extended Star Topology)

It is a variation of star. Instead of all devices connected to a central hub here most of the devices are connected to a secondary hub that in turn connected with central hub. The central hub is an active hub. An active hub contains a repeater, which regenerate the received bit pattern before sending.

The secondary hub may be active or passive. A passive hub means it just precedes a physical connection only

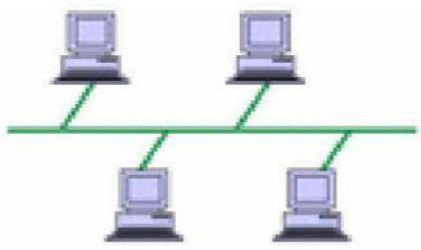


Advantages:

1. Can connect more than star.
2. The distance can be increased.
3. Can isolate and prioritize communication between different computers.

4. BUS TOPOLOGY:

A bus topology is multipoint. Here one long cable is act as a backbone to link all the devices are connected to the backbone by drop lines and taps. A drop line is the connection between the devices and the cable. A tap is the splice into the main cable or puncture the cover.



Advantages:

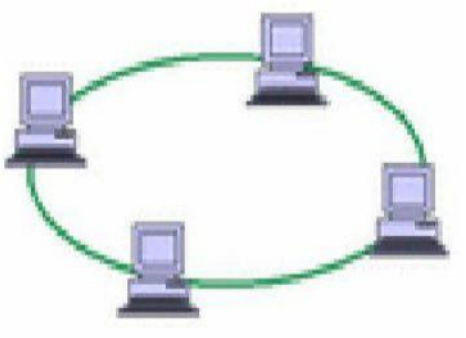
1. Ease of installation.
2. Less cabling

Disadvantages:

1. Difficult reconfiguration and fault isolation.
2. Difficult to add new devices.
3. Signal reflection at top can degradation in quality
4. If any fault in backbone can stops all transmission

5. RING TOPOLOGY

Each node is connected to exactly two other nodes, forming a ring. Can be visualized as a circular configuration. Requires at least three nodes



Advantages:

1. Easy to install.
2. Easy to reconfigure.
3. Fault identification is easy.

Disadvantages:

1. Unidirectional traffic.
2. Break in a single ring can break entire network

6. HYBRID TOPOLOGY

A combination of any two or more network topologies.

NETWORK ARCHITECTURE

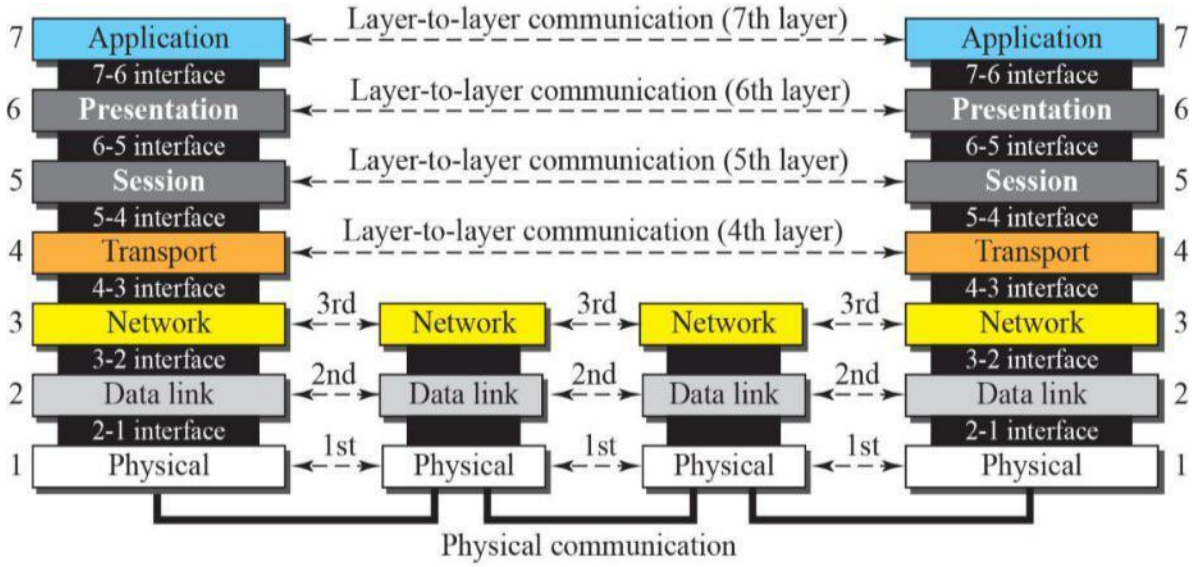
Networks do not remain fixed at single point in time, but it must evolve to accommodate changes based on the technologies on which they are based and demands made by application programmer. Network architecture guides the design and implementation of network. Two commonly used architecture are

- **OSI Architecture**
- **Internet or TCP/IP architecture**

OSI ARCHITECTURE

ISO defines a common way to connect computer by the architecture called Open System Interconnection (OSI) architecture. Network functionality is divided into seven layers. The principles that were applied to arrive at the seven layers can be briefly summarized as follows:

1. A layer should be created where a different abstraction is needed.
2. Each layer should perform a well-defined function.
3. The function of each layer should be chosen with an eye toward defining internationally standardized protocols.
4. The layer boundaries should be chosen to minimize the information flow across the interfaces.
5. The number of layers should be large enough that distinct functions need not be thrown together in the same layer out of necessity and small enough that the architecture does not become unwieldy.



Organization of the layers

The 7 layers can be grouped into 3 subgroups

1. **Network Support Layers**

   Layers 1,2,3 - Physical, Data link and Network are the network support layers. They deal with the physical aspects of moving data from one device to another such as electrical specifications, physical addressing, transport timing and reliability.
2. **Transport Layer**

   Layer4, transport layer, ensures end-to-end reliable data transmission on a single link.
3. **User Support Layers**

   Layers 5,6,7 – Session, presentation and application are the user support layers. They allow interoperability among unrelated software systems

Functions of the Layers

1. PHYSICAL LAYER

The physical layer coordinates the functions required to transmit a bit stream over a physical medium.

The physical layer is concerned with the following:

- **Physical characteristics of interfaces and media -** The physical layer defines the characteristics of the interface between the devices and the transmission medium.
- **Representation of bits -** To transmit the stream of bits, it must be encoded to signals. The physical layer defines the type of encoding.
- **Data Rate or Transmission rate -** The number of bits sent each second – is also defined by the physical layer.
- **Synchronization of bits -** The sender and receiver must be synchronized at the bit level. Their clocks must be synchronized.
- **Line Configuration -** In a point-to-point configuration, two devices are connected together through a dedicated link. In a multipoint configuration, a link is shared between several devices.
- **Physical Topology -** The physical topology defines how devices are connected to make a network. Devices can be connected using a mesh, bus, star or ring topology.
- **Transmission Mode -** The physical layer also defines the direction of transmission Between two devices: simplex, half-duplex or full-duplex.

2. DATA LINK LAYER

It is responsible for transmitting frames from one node to next node. The other responsibilities of this layer are

- **Framing -** Divides the stream of bits received into data units called frames.
- **Physical addressing** – If frames are to be distributed to different systems on the n/w , data link layer adds a header to the frame to define the sender and receiver.
- **Flow control**- If the rate at which the data are absorbed by the receiver is less than the rate produced in the sender ,the Data link layer imposes a flow ctrl mechanism.
- **Error control**- Used for detecting and retransmitting damaged or lost frames and to prevent duplication of frames. This is achieved through a trailer added at the end of the frame.
- **Access control** -Used to determine which device has control over the link at any given time.
- It is responsible for **Hop to Hop** delivery.

3. NETWORK LAYER

This layer is responsible for the delivery of packets from source to destination. It is mainly required, when it is necessary to send information from one network to another.

The other responsibilities of this layer are

- **Logical addressing -** If a packet passes the n/w boundary, we need another addressing system for source and destination called logical address.

- **Routing** – The devices which connects various networks called routers are responsible for delivering packets to final destination.
- It is responsible for **Host** to **Host** delivery.

4. TRANSPORT LAYER

- It is responsible for **Process to Process** delivery.
- It also ensures whether the message arrives in order or not. The other responsibilities of this layer are

- **Port addressing -** The header in this must therefore include a address called port address. This layer gets the entire message to the correct process on that computer.

- **Segmentation and reassembly -** The message is divided into segments and each segment is assigned a sequence number. These numbers are arranged correctly on the arrival side by this layer.

- **Connection control -** This can either be **connectionless or connection-oriented.** The connectionless treats each segment as a individual packet and delivers to the destination. The connection-oriented makes connection on the destination side before the delivery. After the delivery the termination will be terminated.
- **Flow and error control -** Similar to data link layer, but process to process take place.

5. SESSION LAYER

This layer establishes, manages and terminates connections between applications. The other responsibilities of this layer are

- **Dialog control -** This session allows two systems to enter into a dialog either in half duplex or full duplex.
- **Synchronization**-This allows to add checkpoints into a stream of data.

6. PRESENTATION LAYER

It is concerned with the syntax and semantics of information exchanged between two systems.

The other responsibilities of this layer are

- **Translation** – Different computers use different encoding system, this layer is responsible for interoperability between these different encoding methods. It will change the message into some common format.

- **Encryption and decryption**-It means that sender transforms the original information to another form and sends the resulting message over the n/w. and vice versa.

- **Compression and expansion**-Compression reduces the number of bits contained in the information particularly in text, audio and video.

7. APPLICATION LAYER

This layer enables the user to access the n/w. This allows the user to log on to remote user. The other responsibilities of this layer are

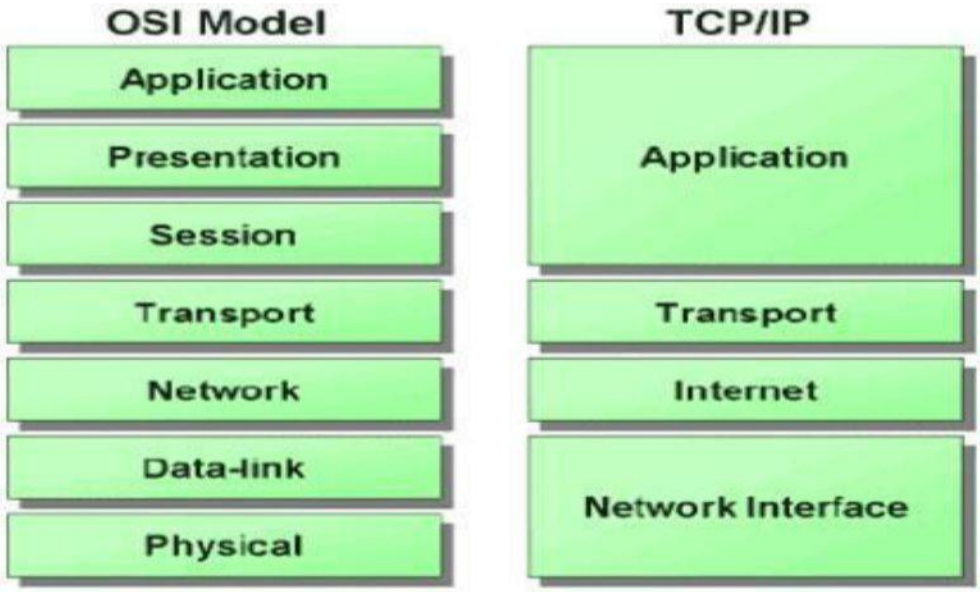
- **FTAM (file transfer, access, mgmt) -** Allows user to access files in a remote host.
- **Mail services -** Provides email forwarding and storage.

- **Directory services -** Provides database sources to access information about various sources and objects.

- **Network virtual terminal (Remote log-in)**
- **Accessing the World Wide Web**

INTERNET ARCHITECTURE (TCP/IP ARCHITECTURE)

TCP/IP protocols map to a four-layer conceptual model known as the *DARPA model* , named after the U.S. government agency that initially developed TCP/IP. The four layers of the DARPA model are: Application, Transport, Internet, and Network Interface. Each layer in the DARPA model corresponds to one or more layers of the seven-layer Open Systems Interconnection (OSI) model.



1.The Host to Network Layer:

Below the internet layer is great void. The TCP/IP reference model does not really say such about what happen here, except to point out that the host has connect to the network using some protocol so it can transmit IP packets over it. This protocol is not specified and varies from host to host and network to network.

2. Internet layer:

It is a connectionless internetwork layer forming a base for a packet-switching network. Its job is to allow hosts to insert packets into any network and have them to deliver independently to the destination. They may appear in a different order than they were sent in each case it is job of higher layers to rearrange them in order to deliver them to proper destination. TCP/IP internet layer is very similar in functionality to the OSI network layer.

Packet routing is very essential task in order to avoid congestion. For these reason it is say that TCP/IP internet layer perform same function as that of OSI network layer.

The internet layer defines an official packet format and protocol called IP (Internet Protocol) and its provides

- Best-effort delivery
  - No error checking
  - No tracking
- IP is a host-to-host protoco**l.**

3. Transport layer:

In the TCP/IP model, the layer above the internet layer is known as transport layer. It is developed to permit entities on the source and destination hosts to carry on a conversation. It specifies 2 end-to-end protocols

i) TCP (Transmission Control Protocol)

ii) UDP (User Datagram Protocol)

TCP

It is a reliable connection-oriented protocol that permits a byte stream originating on one machine to be transported without error on any machine in the internet. It divides the incoming byte stream into discrete message and passes each one onto the internet layer. At the destination, the receiving TCP process collects the received message into the output stream. TCP deals with flow control to make sure a fast sender cannot swamp a slow receiver with more message than it can handle.

UDP

It is an unreliable, connectionless protocol for applications that do not want TCP's sequencing on flow control and wish to offer their own. It is also used for client-server type request-reply queries and applications in which prompt delivery is more important than accurate delivery such as transmitting speech or video.

4. Application Layer:

In TCP/IP model, session or presentation layer are not present. Application layer is present on the top of the Transport layer. It includes all the higher-level protocols which are virtual terminal (TELNET), file transfer (FTP) and electronic mail (SMTP). The virtual terminal protocol permits a user on one machine to log into a distant machine and work there. The file transfer protocol offers a way to move data efficiently from one machine to another. Electronic mail was used for file transfer purpose but later a specialized protocol was developed for it.

The Application Layer defines following protocols

i) File Transfer Protocol (FTP)

It was designed to permit reliable transfer of files over different platforms. At the transport layer to ensure reliability, FTP uses TCP. FTP offers simple commands and makes the differences in storage methods across networks transparent to the user. The FTP client is able to interact with any FTP server; therefore the FTP server must also be able to interact with any FTP client. FTP does not offer a user interface, but it does offer an application program interface for file transfer. The client part of the protocol is called as FTP and the server part of the protocol is known as FTPd. The suffix "d" means Daemon this is a legacy from Unix computing where a daemon is a piece of software running on a server that offers a service.

ii) Hyper Text Transfer Protocol

HTTP permits applications such as browsers to upload and download web pages. It makes use of TCP at the transport layer again to check reliability. HTTP is a connectionless protocol that sends a request, receives a response and then disconnects the connection. HTTP delivers HTML

documents plus all of the other components supported within HTML such as JavaScript, Visual script and applets.

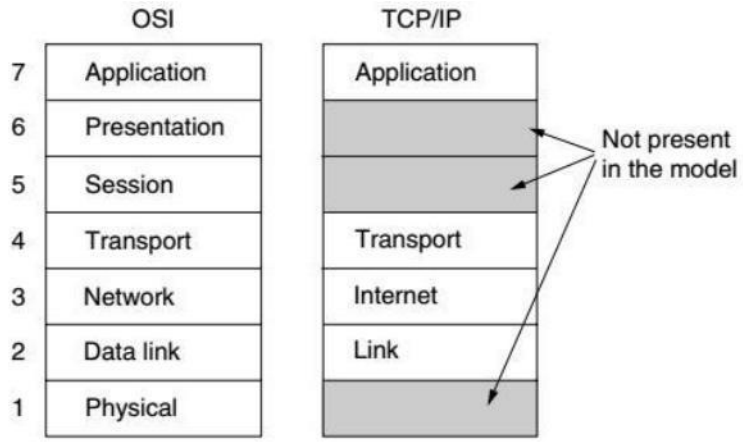
iii) Simple Mail Transfer Protocol

By using TCP, SMTP sends email to other computers that support the TCP/IP protocol suite. SMTP provides extension to the local mail services that existed in the early years of LANs. It supervises the email sending from the local mail host to a remote mail host. It is not reliable for accepting mail from local users or distributing received mail to recipients this is the responsibility of the local mail system. SMTP makes use of TCP to establish a connection to the remote mail host, the mail is sent, any waiting mail is requested and then the connection is disconnected. It can also return a forwarding address if the intended recipient no longer receives email at that destination. To enable mail to be delivered across differing systems, a mail gateway is used.

iv) Simple Network Management Protocol

For the transport of network management information, SNMP is used as standardized protocol. Managed network devices can be cross examined by a computer running to return details about their status and level of activity. Observing software can also trigger alarms if certain performance criteria drop below acceptable restrictions. At the transport layer SNMP protocol uses UDP. The use of UDP results in decreasing network traffic overheads.

Comparison of OSI and TCP/IP reference models



An obvious difference between the two models is the number of layers: the OSI model has seven layers and the TCP/IP has four layers. Both have (inter)network, transport, and application layers, but the other layers are different.

Another difference is in the area of connectionless versus connection-oriented communication. The OSI model supports both connectionless and connection-oriented

communication in the network layer, but only connection-oriented communication in the transport layer, where it counts (because the transport service is visible to the users).

The TCP/IP model has only one mode in the network layer (connectionless) but supports both modes in the transport layer, giving the users a choice. This choice is especially important for simple request-response protocols.

The OSI and TCP/IP reference models have much in common. Both are based on the concept of a stack of independent protocols. Also, the functionality of the layers is roughly similar. For example, in both models the layers up through and including the transport layer are there to provide an end-to-end, network-independent transport service to processes wishing to communicate. These layers form the transport provider. Again in both models, the layers above transport are application-oriented users of the transport service.

Three concepts are central to the OSI model:

- ∑ Services.
- ∑ Interfaces.
- ∑ Protocols.

Probably the biggest contribution of the OSI model is to make the distinction between these three concepts explicit. Each layer performs some services for the layer above it. A layer's interface tells the processes above it how to access it. It specifies what the parameters are and what results to expect. It, too, says nothing about how the layer works inside. Finally, the peer protocols used in a layer are the layer's own business. It can use any protocols it wants to, as long as it gets the job done (i.e., provides the offered services). It can also change them at will without affecting software in higher layers.

The TCP/IP model did not originally clearly distinguish between service, interface, and protocol. For example, the only real services offered by the internet layer are SEND IP PACKET and RECEIVE IP PACKET. As a consequence, the protocols in the OSI model are better hidden than in the TCP/IP model and can be replaced relatively easily as the technology changes.

The OSI reference model was devised before the corresponding protocols were invented. This ordering means that the model was not biased toward one particular set of protocols, a fact that made it quite general.

With TCP/IP the reverse was true: the protocols came first, and the model was really just a description of the existing protocols. There was no problem with the protocols fitting the model. They fit perfectly. The only trouble was that the model did not fit any other protocol stacks.

UNIT-2

The theoretical basis for data communication, Fourier analysis, Bandwidth limited signals, The maximum data rate of a channel, Guided transmission media, Digital modulation and multiplexing, Frequency division multiplexing, Time division multiplexing, code division multiplexing

Theoretical basis for data communication

Information can be transmitted on wires by varying some physical property such as current or voltage. By representing the value of this voltage or current as a single valued function of time f(t), we can model the behavior of our signal and analyze it mathematically.

This can be done by the following 3 ways.

1) Fourier analysis 2) Bandwidth-limited signals 3) Maximum data rate of a channel

1) Fourier analysis:

We model the behavior of variation of voltage or current with mathematical functions

- Fourier series is used

$g(t)=\frac{1}{2}C+\sum_{N=1}^{\infty} a_n \sin(2\pi nft)+\sum_{N=1}^{\infty} b_n \cos(2\pi nft)$

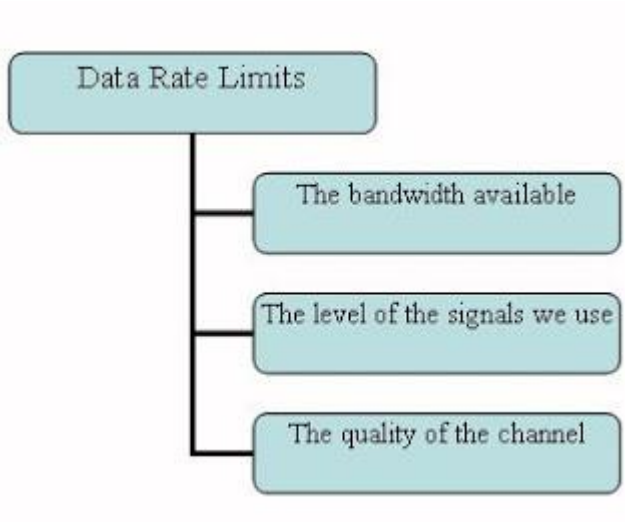
where f=1/T is the fundamental frequencies , and are cosines and sines amplitude of the nth harmonics. Such a decomposition is called fourier series.

2) **Bandwidth-limited signals**:

No transmission facility can transmit signal with losing some power in process. If all fouries components were equally diminish, the resulting signal would be reduced in amplitude, but not distorted. Unfortunately all transmission facilities diminish different Fourier components by different amount, thus introducing distortion. The range of frequencies transmitted without being strongly attenuated is called the ***bandwidth***. In practice the cut-off is not really sharp, so often the quoted bandwidth is from 0 to frequency at which half the power gets through.
The bandwidth is a physical property of the medium and usually depends on its construction, thickness and length of the medium. In some cases a filter is introduced into the circuit to limit the bandwidth available to each customer.

BPS	T(msec)	First harmonics	*Harmonics sent
300	26.67	37.5	80
600	13.33	75	40
1200	6.67	150	20
2400	3.33	300	10
4800	1.67	600	5
9600	0.83	1200	2
19200	0.42	2400	1
38400	0.21	4800	0

A data signal that has a finite duration can be handled by just imagining that it repeats the entire pattern over and over forever.



Maximum data rate of a channel

Henry nyquist derived an equation expressing the maximum data rate for a finite bandwidth noiseless channel.

Nyquist proved that if an arbitrary signal has been run through a low pass filter of bandwidth H, the filtered signal can be completely re-constructed by making only 2H samples per seconds.

"A continuous time signal may be completely represented in its samples ans recovered back if the sampling frequency is greater then or equal to twice the maximum frequency"

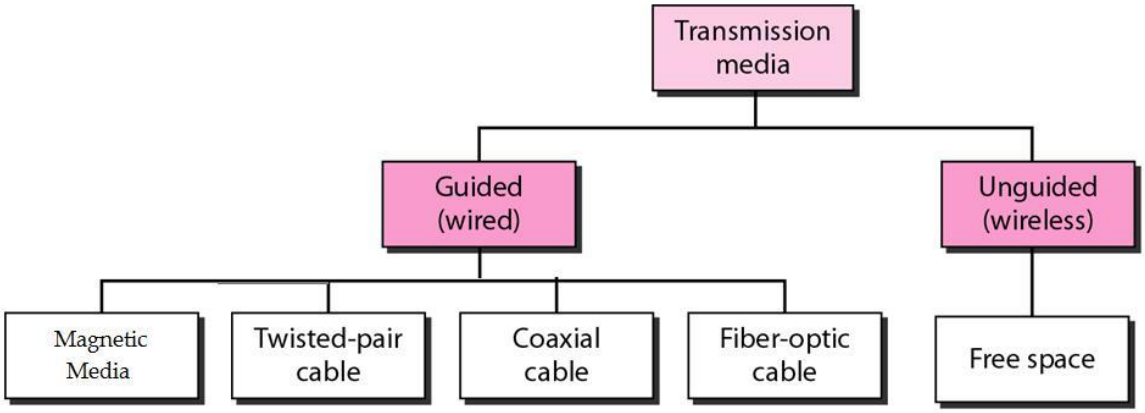
The amount of thermal noise present is measured by the ratio of the signal power (in db) to the noise power called the signal to noise ratio.

Shannon's major result is that the maximum data rate of a noisy channel whose bandwidth is H Hz is given by:-

$$C = W \log_2(1 + SNR)$$

THE PHYSICAL LAYER

Classes of transmission media



GUIDED TRANSMISSION MEDIA

1. Magnetic media and disks

One of the most common ways to transport data from one computer to another is to write them onto magnetic tape or removable media (e.g., recordable DVDs), physically transport the tape or disks to the destination machine, and read them back in again.

Although this method is not as sophisticated as using a geosynchronous communication satellite, it is often more cost effective, especially for applications in which high bandwidth or cost per bit transported is the key factor.

2. Twisted Pairs

Twisted pair is the oldest and still most common transmission medium. It consists of two insulated copper wires, typically about 1 mm thick. The wires are twisted together to reduce electrical interference from similar pairs close by (two parallel wires constitute a simple antenna, a twisted pair does not).

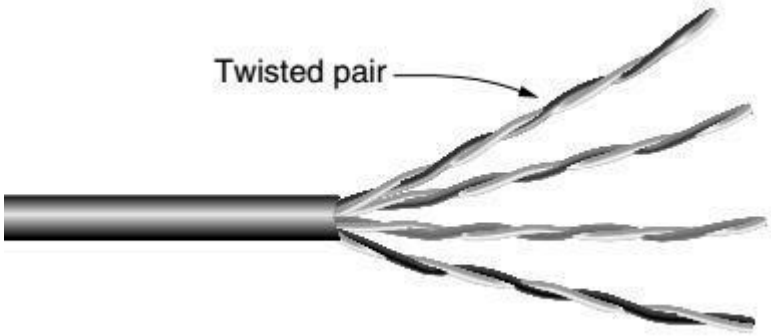
The most common application of the twisted pair is the telephone system. Twisted pairs can run several km without amplification, but for longer distances repeaters are needed.

Twisted pairs can be used for either analog or digital transmission. The bandwidth depends on the thickness of the wire and the distance traveled (several mbps for a few km can be achieved).

Due to their adequate performance and low cost, twisted pairs are widely used.

Twisted pair cabling comes in several varieties, two of which are important for computer networks:

- **Category 3** twisted pairs - gently twisted, 4 pairs typically grouped together in a plastic sheath.
- **Category 5** twisted pairs - More twists per cm than category 3 and Teflon insulation, which results in less crosstalk and better quality signal over longer distances

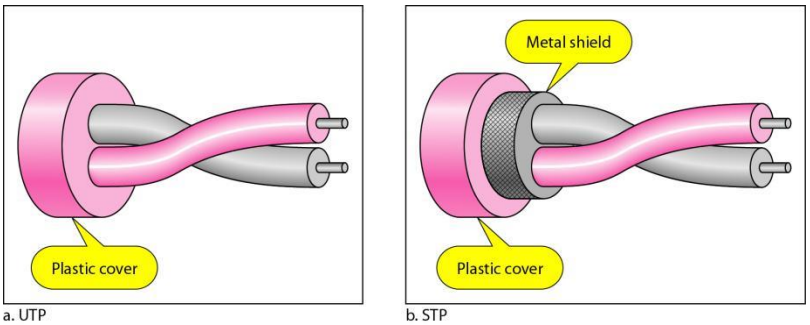


Category 5 UTP cable with four twisted pairs

Unshielded Versus Shielded Twisted-Pair Cable

The most common twisted-pair cable used in communications is referred to as unshielded twisted-pair (UTP or Category 6) as they consist simply of wires and insulators.

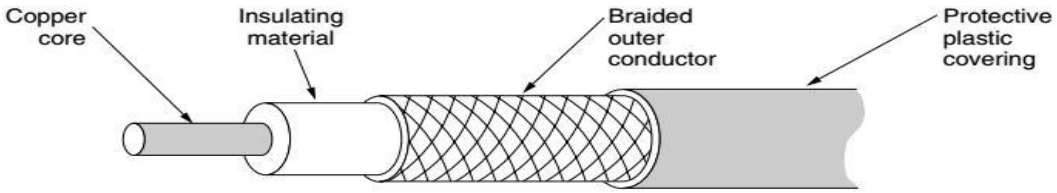
STP (shielded twisted-pair cable) produced by IBM has a metal foil or braided mesh covering that encases each pair of insulated conductors. Although metal casing improves the quality of cable by preventing the penetration of noise or crosstalk, it is bulkier and more expensive.



a. UTP b. STP

3. Coaxial Cable

Coaxial cable (or coax) carries signals of higher frequency ranges than those in twisted pair cable, in part because the two media are constructed quite differently. Instead of having two wires, coax has a central core conductor of solid or stranded wire (usually copper) enclosed in an insulating sheath, which is, in turn, encased in an outer conductor of metal foil, braid, or a combination of the two. The outer metallic wrapping serves both as a shield against noise and as the second conductor, which completes the circuit. This outer conductor is also enclosed in an insulating sheath, and the whole cable is protected by a plastic cover.



Two kinds of coaxial cable are widely used. One kind, 50-ohm cable, is commonly used when it is intended for digital transmission from the start. The original cabling standard for Ethernet that uses 50-ohm coaxial cables, called baseband coax. The other kind, 75-ohm cable, is commonly used for analog transmission and cable television. It is called broadband coax.

Categories of coaxial cables

Category	*Impedance*	*Use*
RG-59	75 Ω	Cable TV
RG-58	50 Ω	Thin Ethernet
RG-11	50 Ω	Thick Ethernet

10BASE2 (also known as cheapernet, Thin Ethernet, thinnet, and thinwire) is a variant of Ethernet that uses thin coaxial cable, terminated with BNC connectors.

10BASE5 is also called thick Ethernet, ThickWire, and ThickNet. The name derives from the fact that the maximum data transfer speed is 10 Mbps, it uses baseband transmission, and the maximum length of cables is 500 meters.

Fiber Optics

Fiber optics are used for long-haul transmission in network backbones, highspeed LANs (although so far, copper has always managed catch up eventually), and high-speed Internet access such as FttH (Fiber to the Home).

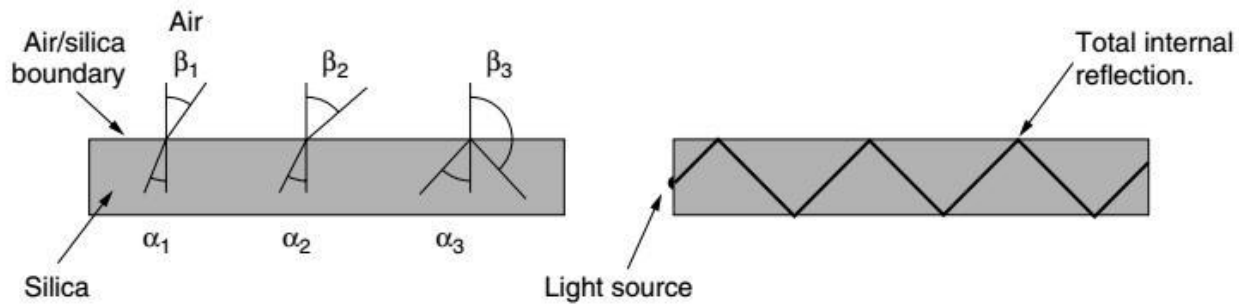
An optical transmission system has three components:

- The light source - a pulse of light indicates a 1 bit and the absence of light indicates a 0 bit,
- The transmission medium - ultra-thin fiber of glass,
- The detector - generates an electrical pulse when light falls on it.

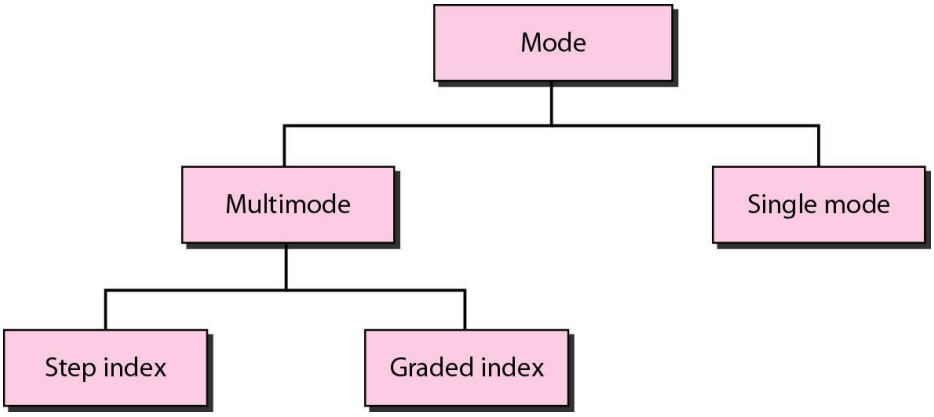
By attaching a light source to one end of an optical fiber and a detector to the other, we get a unidirectional data transmission system.

The work of this transmission system is based on the refraction of the light ray at the silica/air boundary

When a light ray passes from one medium to another—for example, from fused silica to air—the ray is refracted (bent) at the silica/air boundary, as shown in Fig. (a). Here we see a light ray incident on the boundary at an angle α1 emerging at an angle β1. The amount of refraction depends on the properties of the two media (in particular, their indices of refraction). For angles of incidence above a certain critical value, the light is refracted back into the silica; none of it escapes into the air. Thus, a light ray incident at or above the critical angle is trapped inside the fiber, as shown in Fig. (b), and can propagate for many kilometers with virtually no loss.



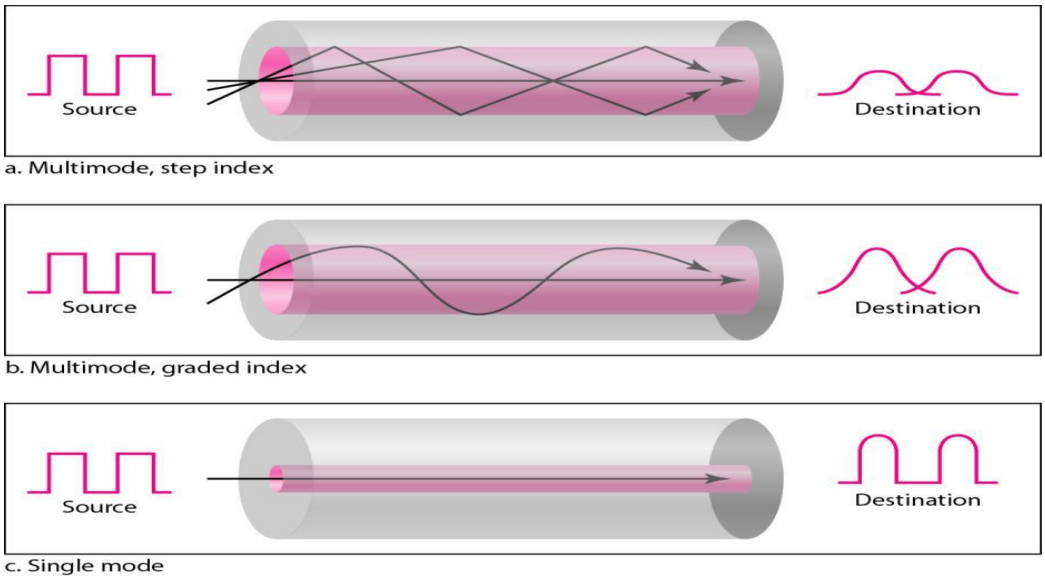
Propagation modes



In Fiber Optics any light ray incident on the boundary above the critical angle will be reflected internally, many different rays will be bouncing around at different angles. Each ray is said to have a different mode, so a fiber having this property is called **Multimode.**

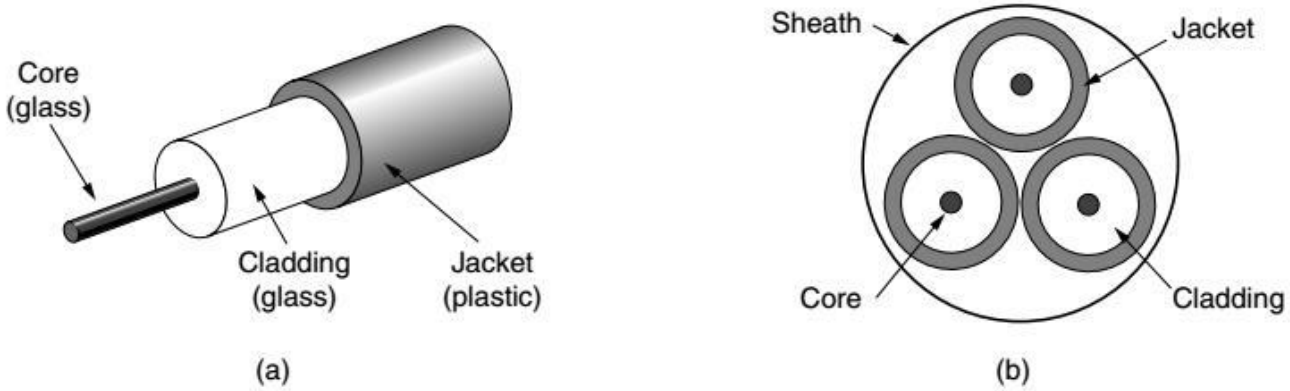
If the fiber's diameter is reduced to a few wavelengths of light the fiber acts like a wave guide and the light can propagate only in a straight line, without bouncing, yielding a **single-mode fiber**. Single-mode fibers are more ex pensive but are widely used for longer distances. Currently available single-mode fibers can transmit data at 100 Gbps for 100 km without amplification.

Modes



Fiber Cables

Fiber optics cables are similar to coax, except without the braid. In multimode fibers, the core is typically 50 microns in diameter, in single mode fibers the core is 8 - 10 microns. The cladding has a lower index of refraction than the core to keep all the light in the core.



(a) Side view of a single fiber. (b) End view of a sheath with three fibers.

Fibers can be connected in three different ways:

- Terminating in connectors and plugged into fiber sockets,
- Spliced mechanically by a clamp,
- Fu**se**d to form a solid connection.

Two kinds of light sources can be used to do the signaling:

- LEDs,
- Semiconductor lasers.

The properties of the both sources are shown below

Item	LED	Semiconductor laser
Data rate	Low	High
Fiber type	Multi-mode	Multi-mode or single-mode
Distance	Short	Long
Lifetime	Long life	Short life
Temperature sensitivity	Minor	Substantial
Cost	Low cost	Expensive

A comparison of semiconductor diodes and LEDs as light sources.

The receiving end of an optical fiber consists of a photo diode. The typical response time of a photodiode is 1 nsec which limits data rates to about 1 Gbps.

Comparison of Fiber Optics and Copper Wire

Advantages of fibers:

- Much higher bandwidth,
- Low attenuation (30 km distance of repeaters vs. 5 km for copper),
- Noise-resistance
- Not affected by corrosive chemicals,
- Much lighter than copper - easier installation and maintenance,
- Difficult to tap - higher security.

Disadvantages of fiber:

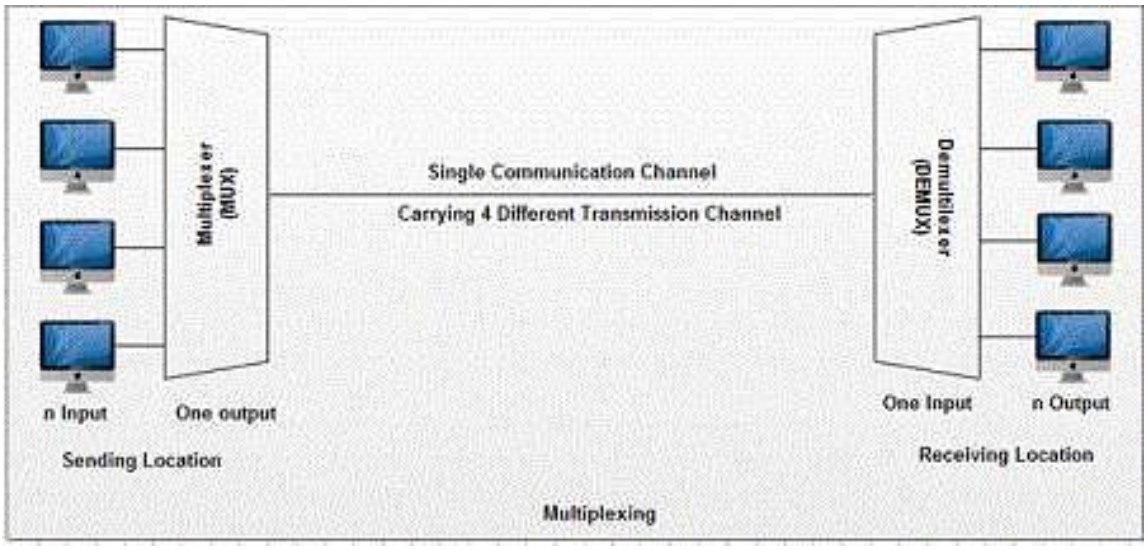
- Unfamiliar technology so far,
- Unidirectional communication,
- More expensive interfaces than electrical ones

Multiplexing:

Multiplexing is a technique by which different analog and digital streams of transmission can be simultaneously processed over a shared link. Multiplexing divides the high capacity medium into low capacity logical medium which is then shared by different streams.

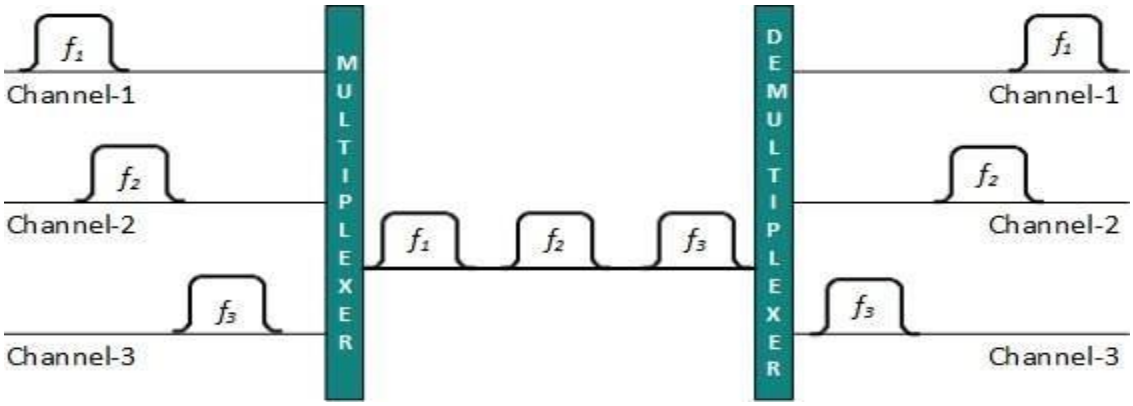
Communication is possible over the air (radio frequency), using a physical media (cable), and light (optical fiber). All mediums are capable of multiplexing.

When multiple senders try to send over a single medium, a device called Multiplexer divides the physical channel and allocates one to each. On the other end of communication, a De-multiplexer receives data from a single medium, identifies each, and sends to different receivers.



Frequency Division Multiplexing:

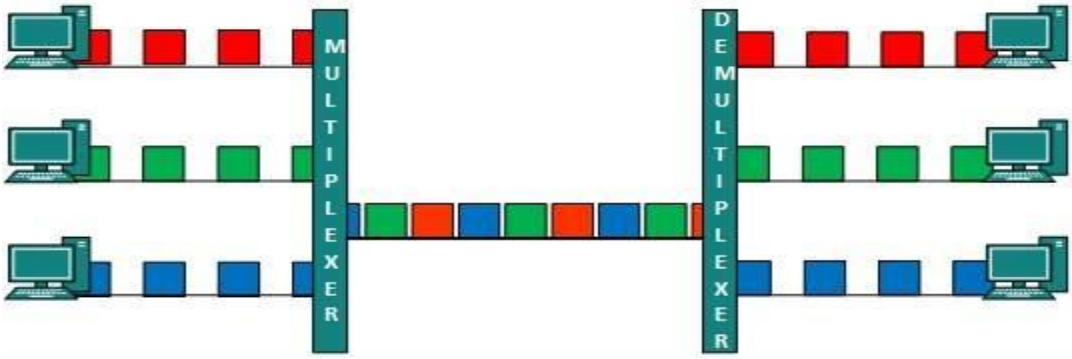
When the carrier is frequency, FDM is used. FDM is an analog technology. FDM divides the spectrum or carrier bandwidth in logical channels and allocates one user to each channel. Each user can use the channel frequency independently and has exclusive access of it. All channels are divided in such a way that they do not overlap with each other. Channels are separated by guard bands. Guard band is a frequency which is not used by either channel.



Time Division Multiplexing

TDM is applied primarily on digital signals but can be applied on analog signals as well. In TDM the shared channel is divided among its user by means of time slot. Each user can transmit data within the provided time slot only. Digital signals are divided in frames, equivalent to time slot i.e. frame of an optimal size which can be transmitted in given time slot.

TDM works in synchronized mode. Both ends, i.e. Multiplexer and De-multiplexer are timely synchronized and both switch to next channel simultaneously.



When channel A transmits its frame at one end,the De-multiplexer provides media to channel A on the other end.As soon as the channel A's time slot expires, this side switches to channel B. On the other end, the De-multiplexer works in a synchronized manner and provides media to channel B. Signals from different channels travel the path in interleaved manner.

Code Division Multiplexing:

Multiple data signals can be transmitted over a single frequency by using Code Division Multiplexing. FDM divides the frequency in smaller channels but CDM allows its users to full bandwidth and transmit signals all the time using a unique code. CDM uses orthogonal codes to spread signals. Each station is assigned with a unique code, called chip. Signals travel with these codes independently, inside the whole bandwidth. The receiver knows in advance the chip code signal it has to receive.

UNIT-3
Data Link Layer: - Data link Layer design issues - Error Detection and correction – Elementary Data link protocols, Sliding window protocols- Basic Concept, One Bit Sliding window protocol, Concept of Go Back n and Selective repeat.

DATA LINK LAYER DESIGN ISSUES

DATA LINK LAYER

The data link layer has a number of specific functions it can carry out. These functions include

1. Providing a well-defined service interface to the network layer.
2. Provides services for the reliable interchange of data across a data link established by the physical layer.
3. Link layer protocols manage the establishment, maintenance, and release of data-link connections.
4. Data-link protocols control the flow of data, dealing with transmission errors, and supervise data error recovery.
5. Regulating the flow of data so that slow receivers are not swamped by fast senders.
6. Recovery from abnormal conditions.

To accomplish these goals, the data link layer takes the packets it gets from the network layer and encapsulates them into frames for transmission. Each frame contains a frame header, a payload field for holding the packet, and a frame trailer, as illustrated in Fig. 2-1. Frame management forms the heart of what the data link layer does.

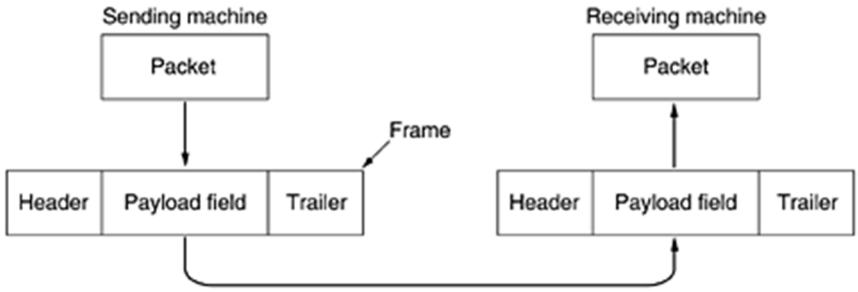


Figure 2-1. Relationship between packets and frames.

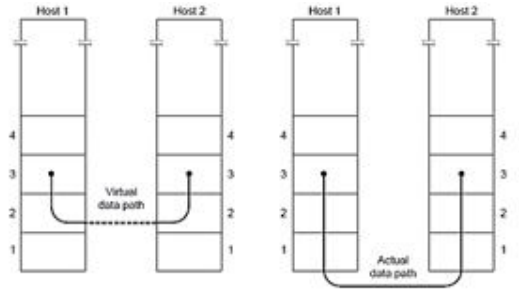
Data Link Layer Design Issues

1. Providing a well-defined service interface to the network layer
2. Determining how the bits of the physical layer are grouped into frames
3. Dealing with transmission errors
4. Regulating the flow of frames so that slow receivers are not swamped by fast senders.

1. Services Provided to the Network Layer

- The function of the data link layer is to provide service to the network layer.
- The principal service is transferring data from the network layer on the source machine to the network layer on the destination machine.
- The network layer hands some bits to the data link layer for transmission to the destination, the job of the data link layer is to transmit the bits to the destination machine, so they can be handed over to the network layer on the destination machine.

The job of the data link layer is to transmit the bits to the destination machine so they can be handed over to the network layer there, as shown in Fig.(a). The actual transmission follows the path of Fig.(b), but it is easier to think in terms of two data link layer processes communicating using a data link protocol.



(a) Virtual communication. (b) Actual communication.

- The data link layer can be designed to offer various services, Three possibilities that are commonly provided are:

1. Unacknowledged connectionless service.

2. Acknowledged connectionless service.

3. Acknowledged connection-oriented service.

Unacknowledged connectionless service

Unacknowledged connectionless service consists of having the source machine send independent frames to the destination machine without having the destination machine acknowledge them. No connection is established beforehand or released afterward. Good channels with low eror rates, for real-time traffic, such as speech.

Acknowledged connectionless service

When this service is offered, there are still no connections used, but each frame sent is individually acknowledged. This way, the sender knows whether or not a frame has arrived safely. Good for unreliable channels, such as wireless.

Acknowledged connection-oriented service

With this service, the source and destination machines establish a connection before any data are transferred. Each frame sent over the connection is numbered, and the data link layer guarantees that each frame sent is received. Furthermore, it guarantees that each frame is received exactly once and that all frames are received in the right order.

When connection-oriented service is used, transfers have three distinct phases.

1. In the first phase the connection is established by having both sides initialize variable and counter need to keep track of which frames have been received and which ones have not.
2. In the second phase, one or more frames are actually transmitted.
3. In the third phase, the connection is released, freeing up the variables, buffers, and other resources used to maintain the connection.

2. Framing

- In order to provide service to the network layer, the data link layer must use the service provided to it by the physical layer.
- What the physical layer does is accept raw bit stream and attempt to deliver it to the destination. This bit stream is not guaranteed to be error free.
- It is up to the data link layer to detect, and if necessary, correct errors.
- The usual approach is for the data link layer to break the bit stream up into discrete frames and compute the checksum for each frame. When the frames arrive at the destination , the checksum is re-computed.

There are four methods of breaking up the bit stream

1. Character count.
2. Starting and ending character stuffing.
3. Starting and ending flags, with bit stuffing.
4. Physical layer coding violations.

Character count, uses a field in the header to specify the number of characters in the frame. When the data link layer at the destination sees the character count, it knows how many characters follow. Problem: count can possible be misrepresented by a transmission error. This method is rarely used anymore.

Starting and ending character stuffing, gets around the problem of resynchronization after an error by having each frame start with the ASCII character sequence DLE STX and end with the sequence DLE ETX. (DLE is Data Link Escape, STX is Start of Text, and ETX is End of Text). **Problem:** a serious problem occurs with this method when binary data, such as object programs or floating-point numbers, are being transmitted it is possible that the DLE, STX, and ETX characters can occur, which will interfere with the framing. One way to solve this problem is to have the sender's data link layer insert and DLE character just before each "accidental" DLE and the data link layer on the other machine removes them before it gives the data to the network layer, this is called Character stuffing.

Starting and ending flags with bit stuffing, allows data frames to contain and arbitrary number of bits and allows character codes with an arbitrary number of bits per character. Each frame begins and ends with a special bit pattern, 01111110, called a flag byte. Whenever the sender's data link layer encounters five

consecutive ones in the data, it automatically stuffs a 0 bit into the outgoing bit stream, which is called bit stuffing. The receiving machine destuffs the 0 bit. • The fourth method, Physical coding violations, is only applicable to networks in which the encoding on the physical medium contains some redundancy. For example, some LANs encode 1 bit of data by using 2 physical bits.

3. **Error Control**
   - The next problem to deal with is to make sure all frames are eventually delivered to the network layer at the destination, and in proper order.
   - The usual way to ensure reliable delivery is to provide the sender with some feedback about what is happening at the other end of the line.
   - One complication with this is that the frame may vanish completely, in which case, the receiver will not react at all, since it has no reason to react.
   - This possibility is dealt with by introducing timers into the data link layer. When the sender transmits a frame, it generally also starts a timer. The timer is set to go off after an interval long enough for the frame to reach the destination machine. If the frame or acknowledgment is lost the timer will go off. The obvious solution is to transmit the frame again. This creates the problem of possible sending frames multiple times. To prevent this from happening, it is generally necessary to assign sequence numbers to outgoing frames, so that the receiver can distinguish retransmission from originals.
   - The whole issue of managing the timers and sequence numbers so as to ensure that each frame is ultimately passed to the network layer at the destination exactly one, no more no less, is an important part of the data link layer's duties.

4. **Flow Control**
   - Another important design issue that occurs in the data link layer (and higher layers as well) is what to do with a sender that systematically wants to transmit frames faster than a receiver can accept them.
   - This situation can easily occur when the sender is running on a fast computer and the receiver is running on a slow machine.
   - The usual solution is to introduce flow control to throttle the sender into sending no faster than the receiver can handle the traffic.
   - Various flow control schemes are known, but most of them use the same basic principle, eg HDLC. The protocol contains well-defined rules about when a sender may transmit the next frame.

ERROR DETECTION AND CORRECTION

Error sources are present when data is transmitted over a medium. Even if all possible error-reducing measures are used during the transmission, an error invariably creeps in and begins to disrupt data transmission. Any computer or communication network must deliver accurate messages.

Error detection is applied mostly in the data-link layer but is also performed in other layers. In some cases, the transport layer includes some sort of error-detection scheme. When a packet arrives at the destination, the destination may extract an error-checking code from the transport header and perform error detection. Sometimes, network-layer protocols apply an error-detection code in the network-layer header. In this case, the error detection is performed only on the IP header, not on the data field. At the application layer, some

type of error check, such as detecting lost packets, may also be possible. But the most common place to have errors is still the data-link layer. Possible and common forms of errors at this level are described here and are shown in Figure 2.1

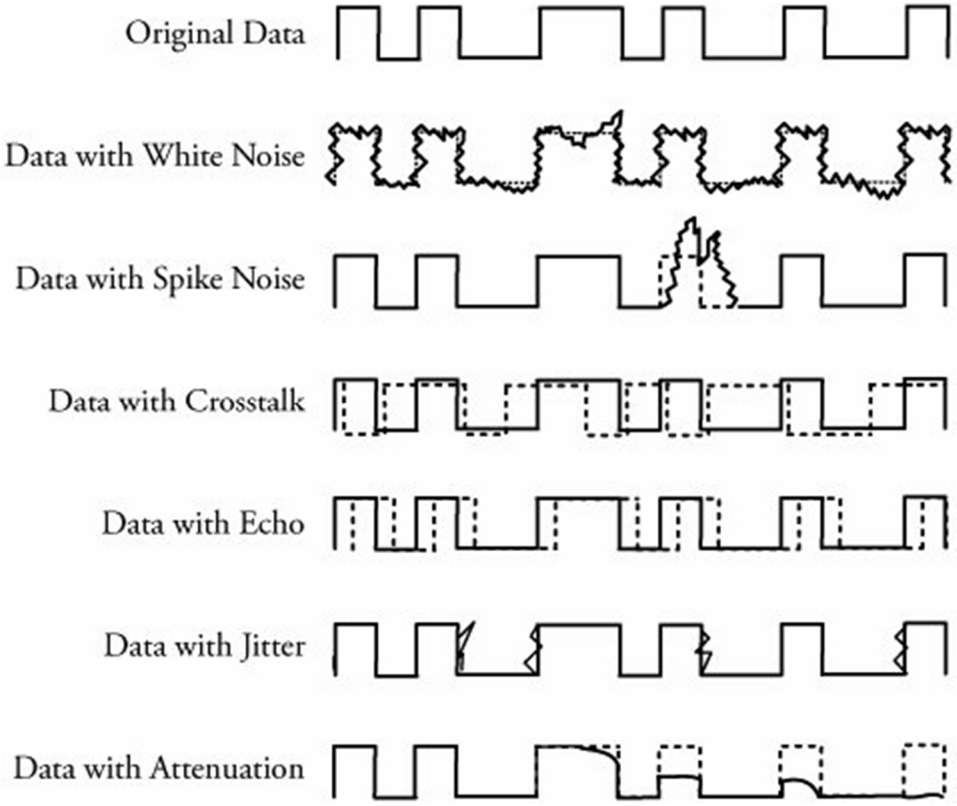


Figure 2.1. Common forms of data errors at the data-link level

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.

TYPES OF ERRORS

Whenever bits flow from one point to another, they are subject to unpredictable changes because of interference. This interference can change the shape of the signal; leads to an error in data transmission. Basic types of errors are

i. Single-Bit Error
ii. Burst Error

i. Single-Bit Error

The term single-bit error means that only one bit of a given data unit (such as a byte, character, data unit, or packet) is changed from 1 to 0 or from 0 to I. In a single-bit error, only one bit in the data unit has changed.

Figure 2.1 shows the effect of a single-bit error on a data unit. To understand the impact of the change, imagine that each group of 8 bits is an ASCII character with a 0 bit added to the left. In the figure, 00000010 (ASCII STX) was sent, meaning start of text, but 00001010 (ASCII LF) was received, meaning line feed.

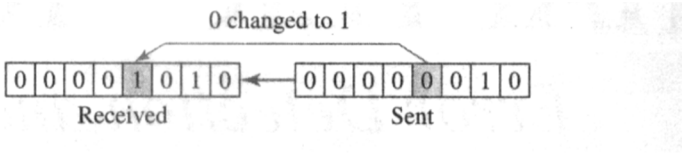


Fig 2.1 Single-bit error

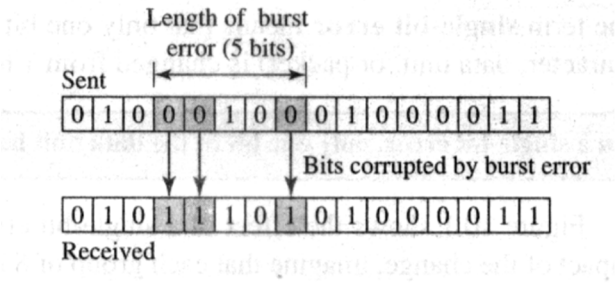
Single-bit errors are the least likely type of error in serial data transmission. To understand Why, imagine a sender sends data at 1 Mbps means that each bit lasts only 1/1,000,000 s, or 1 µs. For a single-bit error to occur, the noise must have a duration of on1y 1 µs, which is very rare; noise normally lasts much longer than this.

However, a single-bit error can happen if we are sending data using parallel transmission. For example, if eight wires are used to send all 8 bits of 1 byte at the same time and one of the wires is noisy, one bit can be corrupted in each byte. Think of parallel transmission inside a computer, between CPU and memory, for example.

ii. Burst Error

The term burst error means that 2 or more bits in the data unit have changed from 1 to 0 or from 0 to 1.

Figure below shows the effect of a burst error on a data unit. In this case, 0100010001000011 was sent, but 0101110101000011 was received. Note that a burst error 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 been corrupted.



Burst error is most likely to occur in a serial transmission. The duration of noise is normally longer than the duration of one bit, which means that when noise affects data, it affects a set of bits. The number of bits affected depends on the data rate and duration of noise. For example, if we are sending data at 1 Kbps, a noise of 1/100 s can affect 10 bits; if we are sending data at 1 Mbps, the same noise can affect 10,000 bits.

ERROR DETECTION

Most networking equipment at the data-link layer inserts some type of error-detection code. When a frame arrives at the next hop in the transmission sequence, the receiving hop extracts the error-detection code and

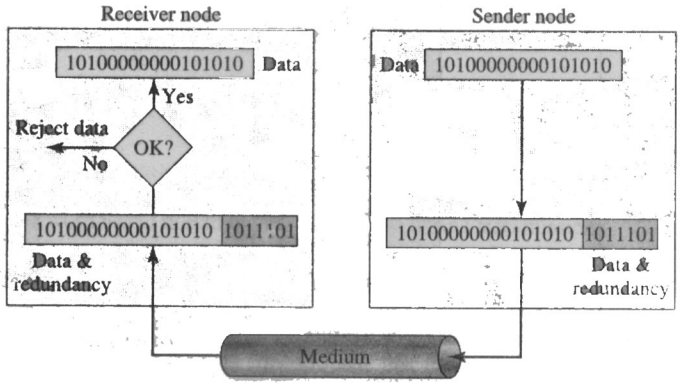
applies it to the frame. When an error is detected, the message is normally discarded. In this case, the sender of the erroneous message is notified, and the message is sent again. However, in real-time applications, it is not possible to resend messages. The most common approaches to error detection are

Redundancy

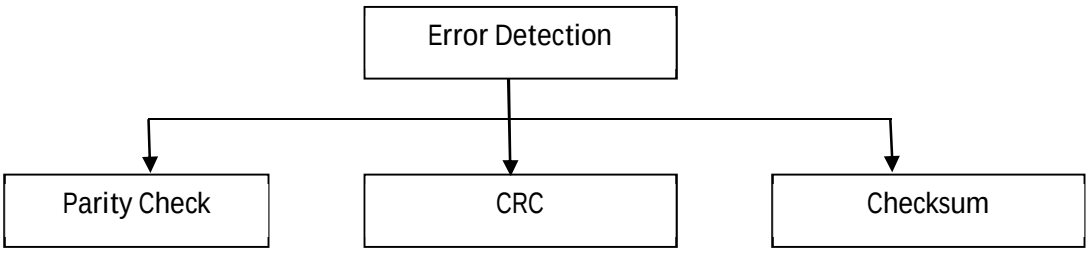
One error detection mechanism would be to send every data unit twice. The receiving device would then able to do a bit for bit comparison between the two versions of the data. Any discrepancy would indicate an error, and an appropriate correction mechanism could be set in place The system would be completely accurate (the odds of errors being introduced on to exactly the same bits in both sets of data are infinitesimally small), but it would also Not only would the transmission time double, but also the time it takes to compare every unit bit by bit must be added.

Error detection uses the concept of redundancy, which means adding extra bits for detecting errors at the destination.. This technique is called redundancy because the extra bits are redundant to the information; they are discarded as soon as the accuracy of the transmission has been determined.

Figure 1.3 shows the process of using redundant bits to check the accuracy of a data unit. Once the data stream has been generated, it passes through a device that analyzes it and adds on an appropriately coded data unit, now enlarged by several bits, travels over the link to the receiver. The receiver puts the entire stream through a checking function, if the received bit stream passes the checking criteria the data portion of the data unit is accepted and the redundant bits are discarded.



Three types of redundancy checks are common in data communications: Parity check, Cyclic Redundancy Check (CRC), and Checksum.

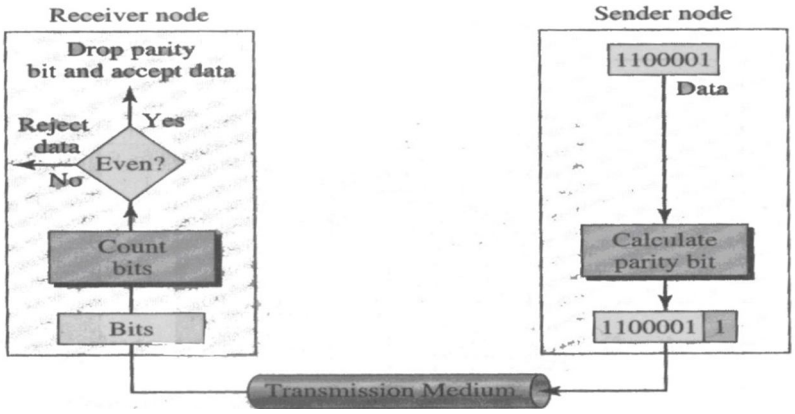


1. PARITY CHECK

The most common and least expensive mechanism for error detection is the parity check. Parity checking can be simple or two dimensional.

Simple Parity Check

In this technique, a redundant bit, called a parity bit, unit so that the total number of Is in the unit (including the parity bit) becomes even (or odd). Suppose we want to transmit the binary data unit 1100001 [ASII a (97)];



Adding the number of 1s gives us 3 an odd number. Before transmitting we pass the data unit through a parity generator. The parity generator counts the 1's and appends the parity bit (a 1 in this case end. The total number of is now 4 an even number. The system now transmits the entire expanded unit across the network link. When it reaches its destination, the receiver puts all 8 bits through an even-parity checking function. If the receiver sees 11000011, it counts four is, an even number, and the data unit passes. But if the data unit has been damaged in transit, means instead of 11000011, the receiver sees 11001011. Then when the parity checker counts the 1s, it gets 5, an odd number. The receiver knows that an error has been introduced into the data somewhere and therefore rejects the whole unit.

Some systems may use odd-parity checking, where the number of Is should be odd. The principle is the same.

Example: Suppose the sender wants to send the word world. In ASCII, the five characters are coded as

1110111 1101111 1110010 1101100 1100100

w o r 1 d

Each of the first four characters has an even number of is, so the parity bit is a 0. The last character (d), however, has three 1s (an odd number), so the parity bit is a 1 to make the total number of 1s even. The following shows the actual bits sent (the parity bits are underlined).

11101110 11011110 11100100 11011000 11001001

Now suppose the word *world* is received by the receiver without being corrupted in transmission as 1110111<u>0</u> 1101111<u>0</u> 1110010<u>0</u> 1101100<u>0</u> 1100100<u>1</u>, The receiver counts the is in each character and comes up with even numbers (6, 6, 4, 4, 4). The data are accepted.

If the word *world* in is corrupted during transmission and receiver get the data stream as,

111**1**111<u>0</u> 1101111<u>0</u> 1110**1**10<u>0</u> 1101100<u>0</u> 1100100<u>1</u>

The receiver counts the Is in each character and comes up with even and odd numbers (7, 6, 5, 4, 4). The receiver knows that the data are corrupted, discards them, and asks for retransmission.

Performance

Simple parity check can detect all single bit error It can also detect burst errors as long as the total number of bits change to odd (1, 3, 5, etc.). Let's say we have an even-parity data unit where the total number of 1s, including the parity bit, is 6: 1000111011. If any 3 bits change value, the resulting parity will be odd and the error will be detected: 1**1**1**1**111011:9, 0110111011:7, 1**l**00**0**1**0**011:5 a11 odd. The checker would return a result of 1, and the data unit would be rejected. The same holds true for any odd number of errors.

Suppose, however, that 2 bits of the data unit are changed: 1**11**0111011:8, 1**1**00**0**11011:6, 1000**0**1101**0**:4. In each case the number of is in the data unit is still even. The parity checker will add them and return an even number although the data unit contains two errors. This method cannot detect errors where the total number of bits changed is even. If any two bits change in transmission, the changes cancel each other and the data unit will pass a parity check even though the data unit is damaged. The same holds true for any even number of errors.

Two Dimensional Parity Check

A better approach is the two dimensional parity check. In this method, block of bits is organized as a table (rows and columns). First we calculate the parity bit for each data unit. Then we organize them into a table format.

For example, shown in Figure 1.6, we have four data units shown in four rows and eight columns. We then calculate parity bit for each column and create a new row of 8 bit; they are the parity bits for the whole block. Note that the first parity bit in the fifth row is calculated based on all first bits; the second parity bit is calculated based on all second bits; and so on. We then attach the parity bits to the original data and send them to the receiver.

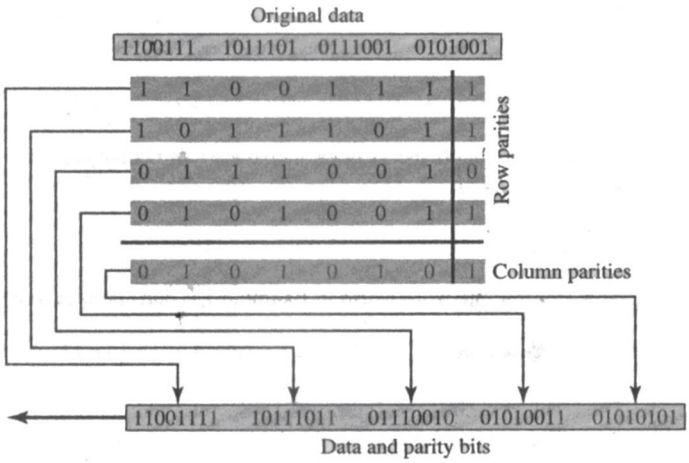


Figure 1.6 Two Dimensional Parity check

Performance

Two-dimensional parity check increases the likelihood of detecting burst errors. A redundancy of *n* bits can easily detect a burst error of n bits. A burst error of more than n bits is also detected by this method with a very high probability. There is, however, one pattern of errors that remains elusive. If 2 bits in one data unit are damaged and two bits in exactly the same positions in another data unit are also damaged, the checker will not detect an error.

2. CYCLIC REDUNDANCY CHECK (CRC)

The most powerful of the redundancy checking techniques is the cyclic redundancy check (CRC). Unlike the parity check which is based on addition, CRC is based on binary division. In CRC, instead of adding bits to achieve a desired parity, a sequence of redundant bits, called the CRC or the CRC remainder, is appended to the end of a data unit so that the resulting data unit becomes exactly divisible by a second, predetermined binary number. At its destination, the incoming data unit is divided by the same number. If at this step there is no remainder the data unit is assumed to be intact and is therefore accepted. A remainder indicates that the data unit has been damaged in transmit therefore must be rejected.

The redundancy bits used by CRC are derived by dividing the data unit by a pre determined divisor; the remainder is the CRC.

To be valid, a CRC must have two qualities; it must have exactly one less bit than the divisor, and appending it to the end of the data string must make the resulting bit sequence exactly divisible by the divisor.

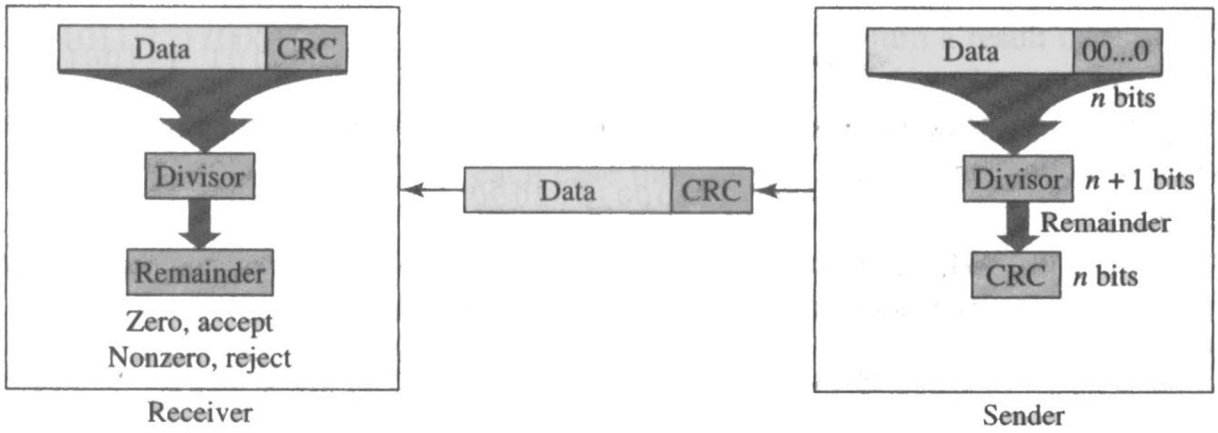


Figure 2.7 CRC generator and checker

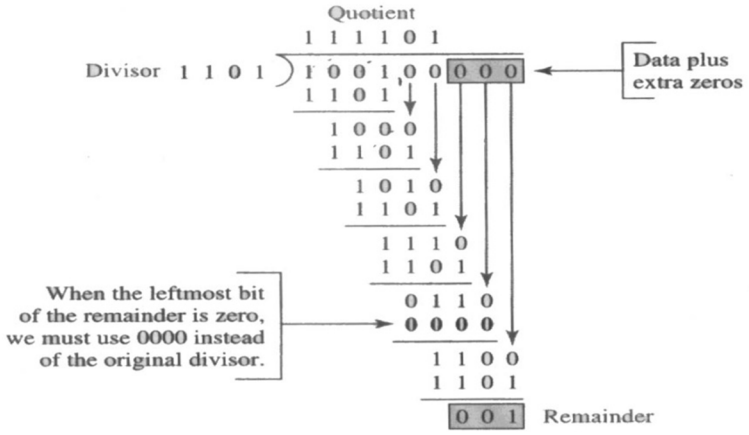
Figure 2.7 provides an outline of the three basic steps in CRC.

1. A string of *n* 0s is appended to the data unit. The number *n* is 1 less than the number of bits in the predetermined divisor, which is *n + 1* bit.
2. The newly elongated data unit is divided by the divisor, using a process called binary division. The remainder resulting from this division is the CRC.
3. The CRC of n bits derived in step 2 replaces the appended 0s at the end of the data unit. Note that the CRC may consist of all 0s.

The data unit arrives at the receiver data first, followed by the CRC. The receiver treats the whole string as a unit and divides it by the same divisor that was used to find the CRC remainder.

If the string arrives without error, the CRC checker yields a remainder of zero and the data unit passes. If the string has been changed in transit, the division yields a nonzero remainder and the data unit does not pass.

The CRC Generator



A CRC generator uses modulo-2 division. Above figure shows this process. In the first step, the 4-bit divisor is subtracted from the first 4 bits of the dividend. Each bit of the divisor is subtracted from the corresponding bit of the dividend without disturbing the next-higher bit. In our example, the divisor, 1101, is subtracted from the first 4 bits of the dividend, 1001, yielding 100 (the leading 0 of the remainder is dropped). The next unused bit from the dividend is then pulled down to make the number of bits in the remainder equal to the number of bits in the divisor. The next step, therefore, is 1000 — 1101, which yields 101, and so on.

In this process, the divisor always begins with a 1; the divisor is subtracted from a portion of the previous that is equal to it in length; the divisor can only be subtracted from a dividend/remainder whose leftmost bit is 1. Anytime the leftmost bit of the dividend/remainder is 0, a string of 0s, of the same length as the divisor, replaces the divisor in that step of the process. For example, if the divisor is 4 bits long, it is replaced by four 0s. (Remember, we are dealing with bit patterns not with quantitative values; 0000 is not the same as 0.) This restriction means that, at any step, the leftmost subtraction will be either 0 - 0 or 1 - 1, both of which equal 0. So, after subtraction, the leftmost bit of the remainder will always be a leading zero, which is dropped, and the next unused bit of the dividend is pulled down to fill out the remainder. Note that only the first bit of the remainder is dropped—if the second bit is also 0, it is retained, and the dividend/remainder for the next step will begin with 0. This process repeats until the entire dividend has been used.

The CRC Checker

A CRC checker functions exactly as the generator does. After receiving the data appended with the CRC, it does the same modulo-2 division. If the remainder is all 0s, the CRC is dropped and the data are accepted; otherwise, the received stream of bits is discarded and data are resent.

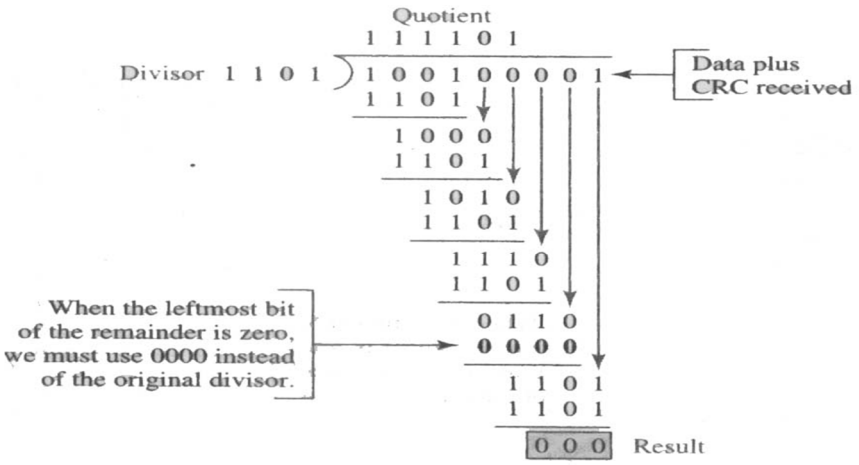


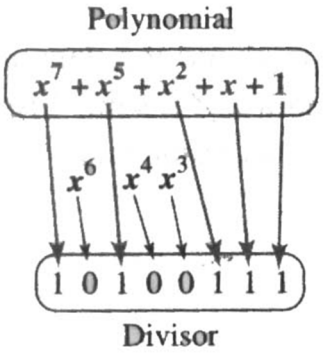
Figure above shows the same process of division in the receiver. We assume that there is no error. The remainder is therefore all 0s, and the data are accepted.

Polynomials

The divisor in the CRC generator is most often represented *not as a string* of 1s and 0s, but as an algebraic polynomial, like $\mathbf{x^7+x^5+x^2+x+1}$

The polynomial format is useful for two reasons: It is short, and it can be used to prove the concept mathematically.

The relationship of a polynomial to its corresponding binary representation is shown below.



A polynomial should be selected to have at least the following properties:

1. It should not be divisible by x.
2. It should be divisible by x + 1.

The first condition guarantees that all burst errors of a length equal to the degree of the polynomial are detected. The second condition guarantees that all burst errors affecting an odd number of bits are detected.

Example: It is obvious that we cannot choose x (binary 10) or x2 + x (binary 110) as the polynomial because both are divisible by x. However, we can choose x + 1 (binary 11) because it is not divisible by x, but is divisible by x + 1. We can also choose x + 1 (binary 101) because it is divisible by x + I (binary division).

Standard Polynomials

Some standard polynomials used by popular protocols for CRC generation are given below.

CRC-8 for ATM: $x^8 + x^2 + x + 1$

CRC-12: $x^{12} + x^{11} + x^3 + x + 1$

CRC-16: $x^{16} + x^{15} + x^2 + 1$

CRC-CCITT: $x^{16} + x^{15} + x^5 + 1$

CRC-32 used in IEEE 802:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

Performance

CRC is a very effective error detection method. If the divisor is chosen according to the previously mentioned rules,

1. CRC can detect all burst errors that affect an odd number of bits.
2. CRC can detect all burst errors of length less than or equal to the degree of the polynomial.
3. CRC can detect, with a very high probability, burst errors of length greater than the degree of the polynomial.

Example The CRC- 12 ($x^{12} + x^{11} + x^3 + x + 1$), which has a degree of 12, will detect all burst errors affecting an odd number of bits, will detect all burst errors with a length less than or equal to 12, and will detect, 99.97 percent of the time, burst errors with a length of 12 or more.

3. CHECKSUM

The third error detection method we discuss here is called the checksum. Like the parity checks and CRC, the checksum is based on the concept of redundancy.

Checksum Generator

In the sender, the checksum generator did the following actions.

1. The unit is subdivided into *k* sections, each of *n* bits.
2. All these *k* sections are added using ones complement arithmetic in such a way that the total is also *n* bits long.
3. The sum is complemented and appended to the end of the original data unit as redundancy bits, called the checksum field
4. The extended data unit is transmitted across the network. So, if the sum of the data segment is T, the checksum will be -T.

Checksum Checker

The receiver follows these steps

1. The unit is divided into k sections, each of n bits as the checksum generation.
2. All sections are added using ones complement to get the sum.
3. The sum is complemented.
4. If the result is zero, the data are accepted: otherwise, they are rejected.

Example: Suppose the following block of 16 bits is to be sent using a checksum of 8 bits.

10101001 00111001

The numbers are added using ones complement arithmetic

	10101001
	00111001
Sum	11100010
Checksum	00011101

The pattern sent is: 10101001 00111001 00011101
Data Checksum

Performance

The checksum detects all errors involving an odd number of bits as well as most errors involving an even number of bits. However, if one or more bits of a segment are damaged and the corresponding bit or 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. If the last digit of one segment is a 0 and it gets changed to a 1 in transit, then the last 1 in another segment must be changed to a 0 if the error is to go undetected.

In two-dimensional parity check, two 0s could both change to is without altering the parity because carries were discarded. Checksum retains all carries; so although two Os becoming is would not alter the value of their own column, it would change the value of the next-higher column. But anytime a bit inversion is balanced by an opposite bit inversion in the corresponding digit of another data segment, the error is invisible.

FLOW AND ERROR CONTROL

Flow and error control are the main functions of the data link layer. Let us informally define each.

Flow Control

Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment. Flow control coordinates the amount of data that can be sent before receiving acknowledgment and is one of the most important duties of the data link layer. The flow of data must not be allowed to overwhelm the receiver.

Any receiving device has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data. The receiving device must be able to inform the sending device before those limits are reached and to request that the transmitting device send fewer frames or stop temporarily. Incoming data must be checked and processed before they can be used. The rate of such processing is often slower than the rate of transmission. For this reason, each receiving device has a block of memory, called a buffer, reserved for storing incoming data until they are processed. If the buffer begins to fill up, the receiver must be able to tell the sender to halt transmission until it is once again able to receive.

Error Control

Error control is both error detection and error correction. It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. In the data link layer, the term error control refers primarily to methods of error detection and retransmission. Error control in the data link layer is often implemented simply: Anytime an error is detected in an exchange, specified frames are retransmitted. This process is called ***automatic repeat request*** (**ARQ**).

FLOW AND ERROR CONTROL MECHANISMS

Three common flow and error control mechanisms are,

1. Stop- and-Wait ARQ,
2. Go-Back-N ARQ
3. Selective-Repeat ARQ.

These are sometimes referred to as ***sliding window protocols***.

1. STOP-AND-WAIT ARQ

Stop-and-Wait ARQ is the simplest flow and error control mechanism. It has the following features:

- The sending device keeps a copy of the last frame transmitted until it receives an acknowledgment for that frame. Keeping a copy allows the sender to retransmit lost or damaged frames until they are received correctly.
- For identification purposes, both data frames and acknowledgment (ACK) frames are numbered alternately 0 and 1. A data 0 frame is acknowledged by an ACK 1 frame, indicating that the receiver has received data frame 0 and is now expecting data frame 1. This numbering allows for identification of data frames in case of duplicate transmission (important in the case of lost acknowledgment or delayed acknowledgment).
- A damaged or lost frame is treated in the same manner by the receiver. If the receiver detects an error in the received frame, it simply discards the frame and sends no acknowledgment. If the receiver receives a frame that is out of order (0 instead of 1 or 1 instead of 0), it knows that a frame is lost. It discards the out-of-order received frame.
- The sender has a control variable, which we call S, that holds the number of the recently sent frame (0 or 1). The receiver has a control variable, which we call R, that holds the number of the next frame expected (0 or 1).
- The sender starts a timer when it sends a frame. If an acknowledgment is not received within an allotted time period, the sender assumes that the frame was lost or damaged and resends it.
- The receiver sends only positive acknowledgment for frames received safe and Sound; it is silent about the frames damaged or lost. The acknowledgment number always defines the number of the next expected frame. If frame 0 is received, ACK I is sent; if frame 1 is received, ACK 0 is sent.

Operation

In the transmission of a frame, we can have four situations: *normal operation*, the *frame is lost*, *the acknowledgement is lost*, or *the acknowledgment is delayed*.

Normal Operation

In a normal transmission, the sender sends frame 0 and waits to receive ACK 1. When ACK 1 is received, it sends frame 1 and then waits to receive ACK 0, and so on. The ACK must be received before the timer set for each frame expires. Figure below shows successful frame transmissions

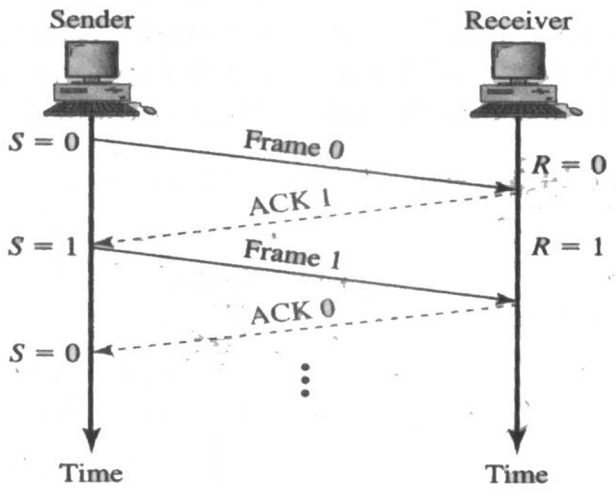


Figure: Stop and Wait ARQ, Normal operation

Lost or Damaged Frame

A lost or damaged frame is handled in the same way by the receiver; when the receiver receives a damaged frame, it discards it, which essentially means the frame is lost. The receiver remains silent about a lost frame and keeps its value of R. For example, in Figure below, the sender transmits frame 1, but it is lost. The receiver does nothing, retaining the value of R (1). After the timer at the sender site expires, another copy of frame 1 is sent.

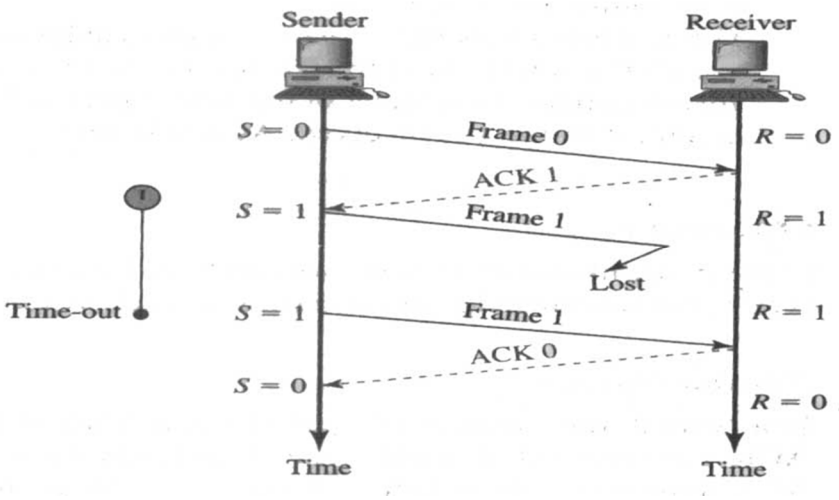


Figure: Stop and Wait ARQ, Lost or Damaged Frame

A lost or damaged acknowledgment

A lost or damaged acknowledgment is handled in the same way by the sender; if the sender receives a damaged acknowledgment, it discards it. Figure below shows a lost ACK 0. The waiting sender does not know if frame I has been received. When the timer for frame 1 expires, the sender retransmits frame 1. Note that the receiver has already received frame 1 and is expecting to receive frame 0 (R = 0). Therefore, it silently discards the second copy of frame 1.

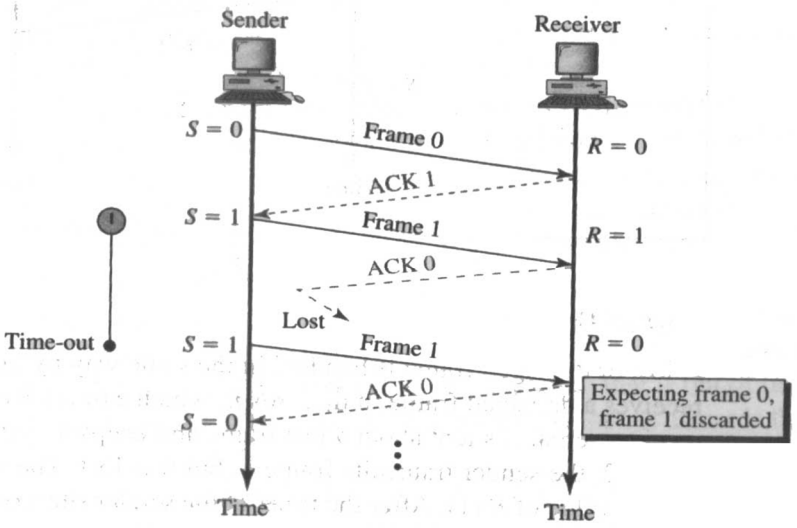


Figure: Stop and Wa1t ARQ, lost ACK frame

Delayed acknowledgment

Another problem that may occur is delayed acknowledgment. An acknowledgment can be delayed at the receiver or by some problem with the link. Figure 4 shows the delay of ACK 1; it is received after the timer for frame 0 has already expired. The sender has already retransmitted a copy of frame 0. However, the value of R at the receiver site is still 1, which means that the receiver expects to see frame 1. The receiver, therefore, discards the duplicate frame 0.

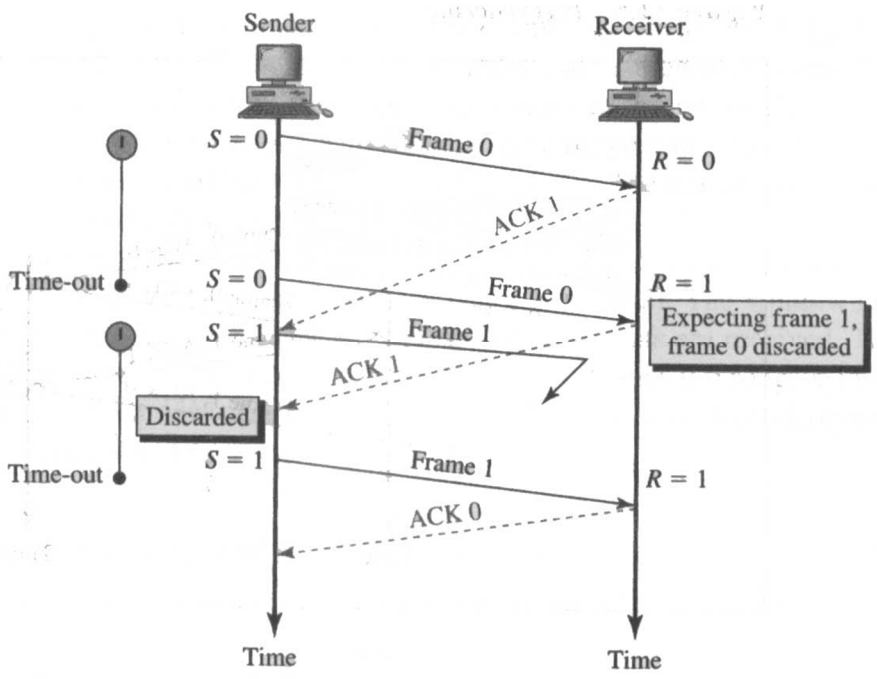


Figure 4: Stop and Wa1t ARQ, delayed ACK frame

The sender has now received two ACKs, one that was delayed and one that was sent after the duplicate frame 0 arrived. The second ACK 1 is discarded.

After the delayed ACK 1 reaches the sender, frame I is sent. However, frame 1 is lost and never reaches the receiver. The sender then receives an ACK 1 for the duplicate frame sent. If the ACKs were not numbered, the sender would interpret the second ACK as the acknowledgment for frame 1. Numbering the ACKs provides a method to keep track of the received data frames.

Bidirectional Transmission

If the two parties have two separate channels for full- duplex transmission or share the same channel for half-duplex transmission. In this case, each party needs both S and R variables to track frames sent and expected.

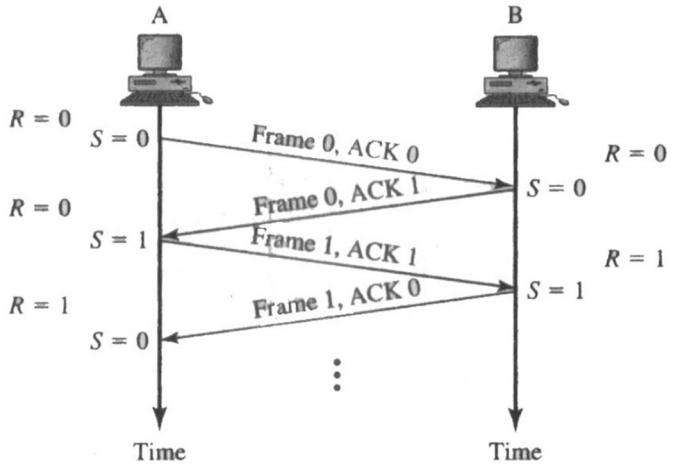


Figure 5: Bidirectional Transmission, Piggybacking

Piggybacking

Piggybacking is a method to combine a data frame with an acknowledgment. For example, in Figure 5, Stations A and B both have data to send. Instead of sending separate data and ACK frames, station A sends a data frame that includes an ACK. Station B behaves in a similar manner.

Piggybacking can save bandwidth because the overhead from a data frame and an ACK frame (addressees, CRC, etc.,) can be combined into just one frame.

2. GO-BACK-N ARQ

In Stop-and-Wait ARQ, at any point in time for a sender, there is only one frame, the outstanding frame, that is sent and waiting to be acknowledged. This is not a good use of the transmission medium. To improve the efficiency, multiple frames should be in transition while waiting for acknowledgment. In other words, we need to let more than one frame be outstanding. Two protocols use this concept: Go-Back-N ARQ and Selective Repeat ARQ.

In Go-Back-N ARQ, we can send up to W frames before worrying about acknowledgments; we keep a copy of these frames until the acknowledgments arrive. This procedure requires additional features to be added to Stop-and-Wait ARQ.

Sequence Numbers

Frames from a sending station are numbered sequentially. However, because we need to include the sequence number of each frame in the header, we need to set a limit. If the header of the frame allows m bits for the sequence number, the sequence numbers range from 0 to 2^m-1. For example, if m is 3, the only sequence numbers are 0 through 7 inclusive. However, we can repeat the sequence. So the sequence numbers are

0,l,2,3,4,5,6,7, 0,l,2,3,4,5,6,7….1

Sender Sliding Window

At the sender site, to hold the outstanding frames until they are acknowledged, we use the concept of a window. We imagine that all frames are stored in a buffer. The outstanding frames are enclosed in a window. The frames to the left of the window are those that have already been acknowledged and can be purged; those to the right of the Window cannot be sent until the window slides over them. The size of the window is at most 2^m — 1.

The size of the window in this protocol is fixed, although we can have a variable- size window in other protocols such as TCP. The window slides to include new unsent frames when the correct acknowledgments are received. The Window is a sliding window. For example, in Figure 6a, frames 0 through 6 have been sent. In part b, the window slides two frames over because an acknowledgment was received for frames 0 and 1.

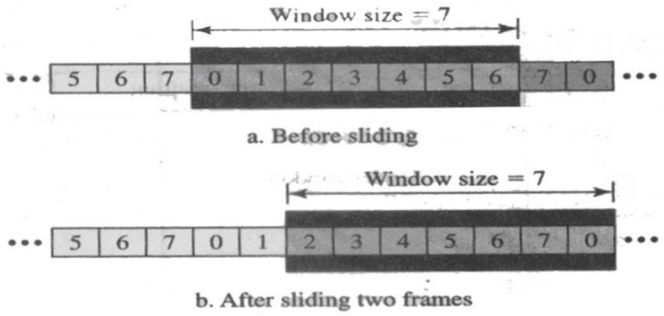


Figure 6: Sender Sliding Window

Receiver Sliding Window

The size of the window at the receiver side in this protocol is always 1. The receiver is always looking for a specific frame to arrive in a specific order. Any frame arriving out of order is discarded and needs to be resent. The receiver window also slides as shown in Figure 7. In part *a* the receiver is waiting for frame 0. When that arrives, the window slides over.

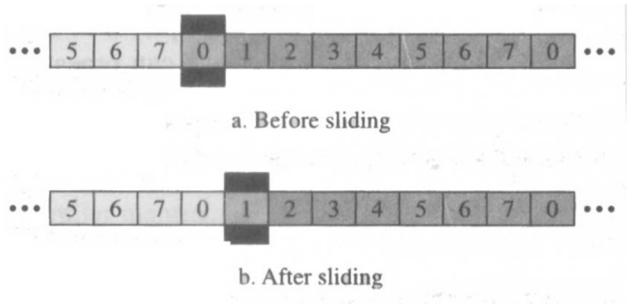


Figure 6: Sender Sliding Window

Control variables

The sender has three variables, S, S_F, and S_L. The S variable holds the sequence number of the recently sent frame; S_F holds the sequence number of the first frame in the window; and S_L holds the sequence number of the last frame in the window. The size of the window is W, where W= S_L — S_F+ 1.

The receiver only has one variable, R that holds the sequence number of the frame it expects to receive. If the sequence number of the received frame is the same as the value of R, the frame is accepted; if not, it is rejected. Figure 8 shows the sender and receiver window with their control variables.

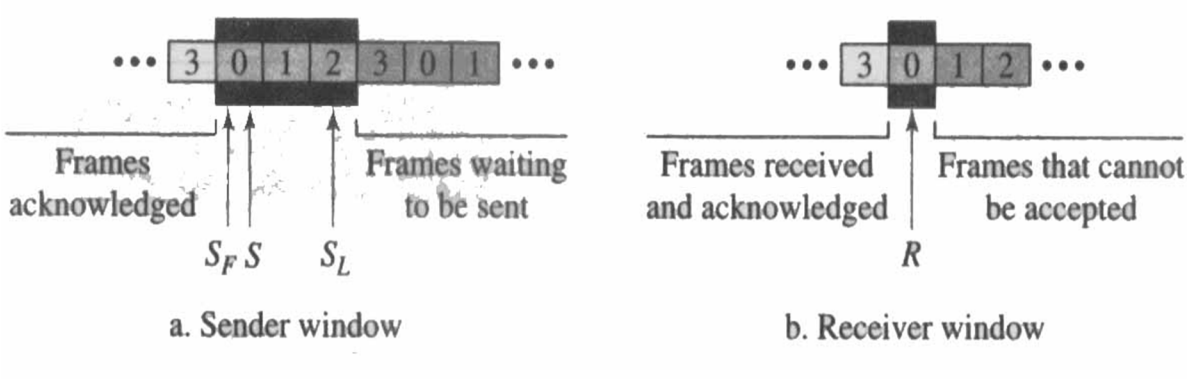


Figure 8, Control variables

Timers

The sender sets a timer for each frame sent. The receiver has no timers.

Acknowledgment

The receiver sends positive acknowledgments if a frame has arrived safe and sound and in order. If a frame is damaged or is received out of order, the receiver is silent and will discard all subsequent frames until it receives the one it is expecting. The silence of the receiver causes the timer of the unacknowledged frame to expire. This, in turn, causes the sender to go back and resend all frames, beginning with the one with the expired timer. The receiver does not have to acknowledge each frame received. It can send one cumulative acknowledgment for several frames.

Resending Frame

When a frame is damaged, the sender goes back and sends a set of frames starting from the damaged one up to the last one sent. For example, suppose the sender has already sent frame 6, but the timer for frame 3 expires. This means that frame 3 has not been acknowledged, so the sender goes back and sends frames 3, 4, 5, 6 again. That is why the protocol is called Go-Back-N ARQ.

OPERATION

Normal Operation

Figure 9 shows a normal operation of Go-Back-N ARQ. The sender keeps track of the outstanding frames and updates the variables and windows as the acknowledgments arrive.

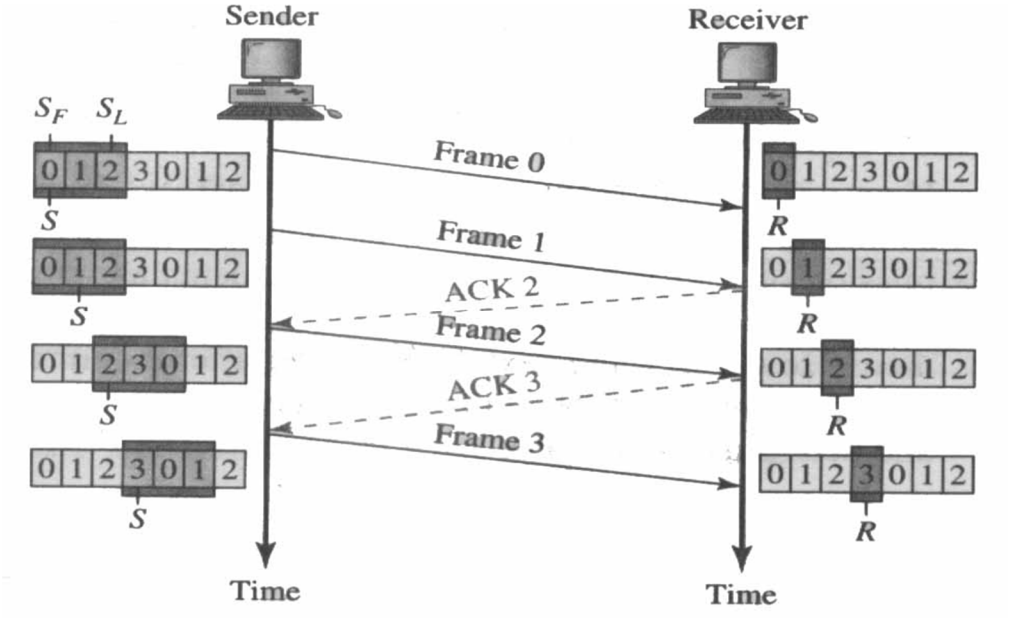


Figure 9 Go-Back-N ARQ, normal operation

Damaged or Lost Frame

Figure 10 shows, the situation when a frame is lost. It shows that frame 2 is lost. Note that when the receiver receives frame 3, it is discarded because the receiver is expecting frame 2, not frame 3 (according to its window). After the timer for frame 2 expires at the sender site, the sender sends frames 2 and 3 (it goes back to 2).

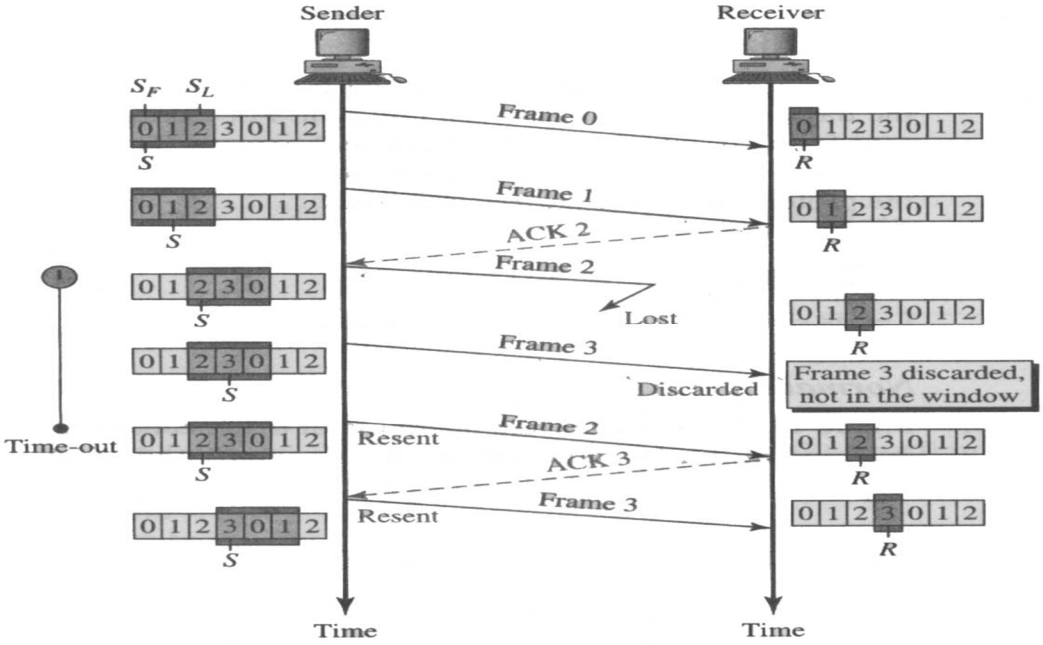


Figure 10: Damaged or Lost Frame

Damaged or lost Acknowledgment

If an acknowledgment is damaged or lost, we can have two situations. If the next acknowledgment arrives before the expiration of any timer, there is no need for retransmission of frames because acknowledgments are cumulative in this protocol. ACK 4 means ACK 1 to ACK 4. So if ACK 1, ACK 2, and ACK 3 are lost, ACK 4 covers them. However, if the next ACK arrives after the time-out, the frame and all the frames after that are resent. The receiver never resends an ACK.

Delayed Acknowledgment

A delayed acknowledgment triggers the resending of frames.

Sender Window Size

We can now show why the size of the sender window must be less than 2^m. As an example, we choose m = 2, which means the size of the window can be 2^m — 1, or 3. Figure 11 compares a window size of 3 and 4.

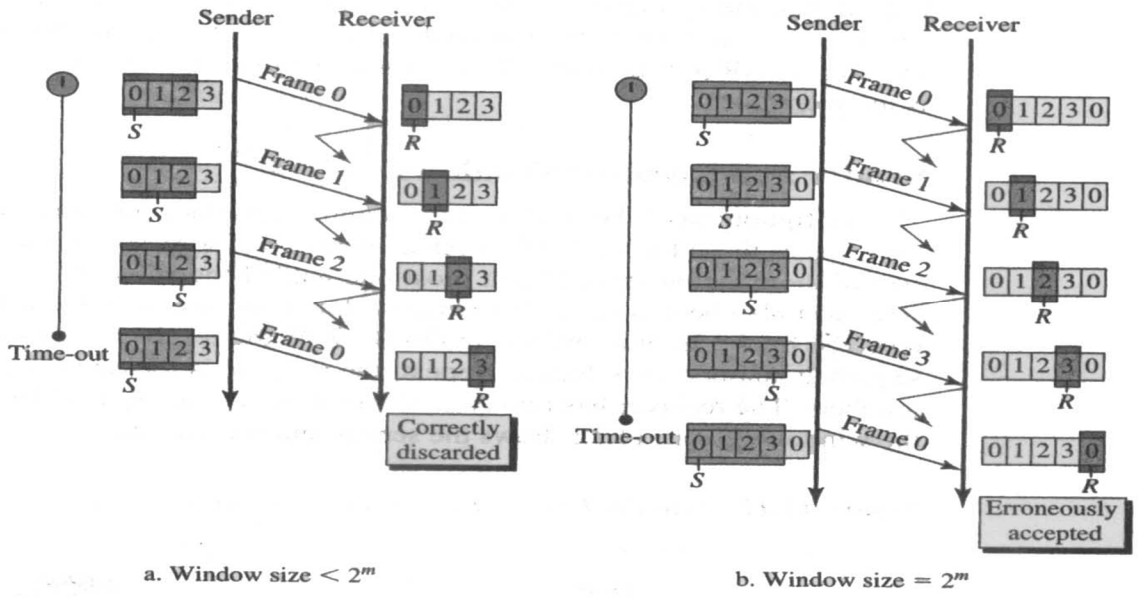


Figure 11 Go-Back-N ARQ: sender Window size

If the size of the window is 3 (less than 22) and all three acknowledgments are lost, the frame 0 timer expires and all three frames are resent. However, the window of the receiver is now expecting frame 3, not frame 0, so the duplicate frame is correctly discarded. On the other hand, if the size of the window is 4 (equal to 22) and all acknowledgments are lost, the sender will send the duplicate of frame 0. However, this time the window of the receiver expects to receive frame 0, so it accepts frame 0, not as a duplicate, but as the first frame in the next cycle. This is an error.

Bidirectional Transmission and Piggybacking

As in the case of Stop-and-Wait ARQ, Go-Back N ARQ can also be bidirectional We can also use piggybacking to improve the efficiency of the transmission However, each direction needs both a sender window and a receiver window.

3. SELECTIVE REPEAT ARQ

Go-Back-N ARQ simplifies the process at the receiver site. The receiver keeps3 track of only one variable, and there is no need to buffer out-of-order frames; they are simply discarded. However, this protocol is very inefficient for a noisy link. In a noisy link a frame has a higher probability of damage, which means the resending of multiple frames. This resending uses up the bandwidth and slows down the transmission. For noisy links, there is another mechanism that does not resend N frames when just one frame is damaged; only the damaged frame is resent. This mechanism is called Selective Repeat ARQ. It is more efficient for noisy links, but the processing at the receiver is more complex.

Sender and Receiver Windows

The configuration of the sender and its control variables for Selective Repeat ARQ is the same as those for Go-Back-N ARQ. But the size of the window should be at most one-half of the value $2^{m.}$ The receiver window size must also be this size. This window, however, specifies the range of the accepted received frame. In other words, in Go-Back-N, the receiver is looking for one specific sequence number; in Selective Repeat, the receiver is looking for a range of sequence numbers. The receiver has two control variables R_F and R_L to define the boundaries of the window. Figure 12 shows the sender and receiver windows.

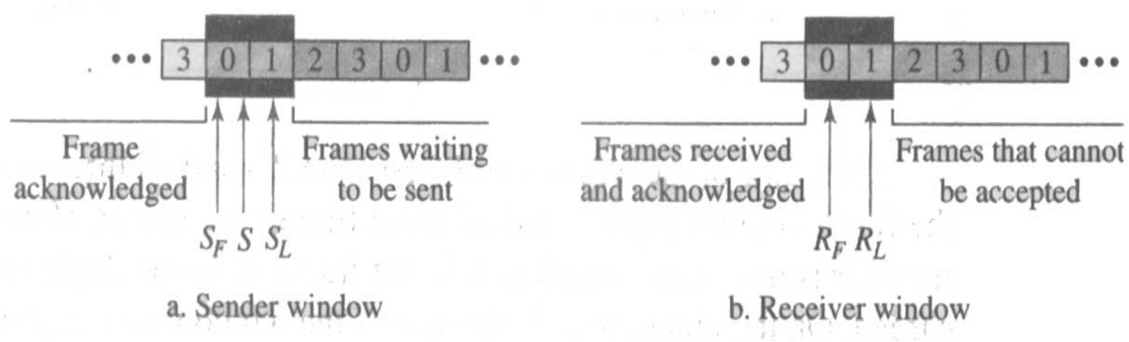


Figure 12 Selective Repeat ARQ, sender and receiver windows

Selective Repeat ARQ also defines a negative acknowledgment (NAK) that reports the sequence number of a damaged frame before the timer expires.

Operation

Figure 13 show the operation of the mechanism of Selective Repeat ARQ with an example of a lost frame.

Frames 0 and I are accepted when received because they are in the range specified by the receiver window. When frame 3 is received, it is also accepted for the same reason. However, the receiver sends a NAK 2 to show that frame 2 has not been received. When the sender receives the NAK 2, it resends only frame 2, which is then accepted because it is in the range of the window.

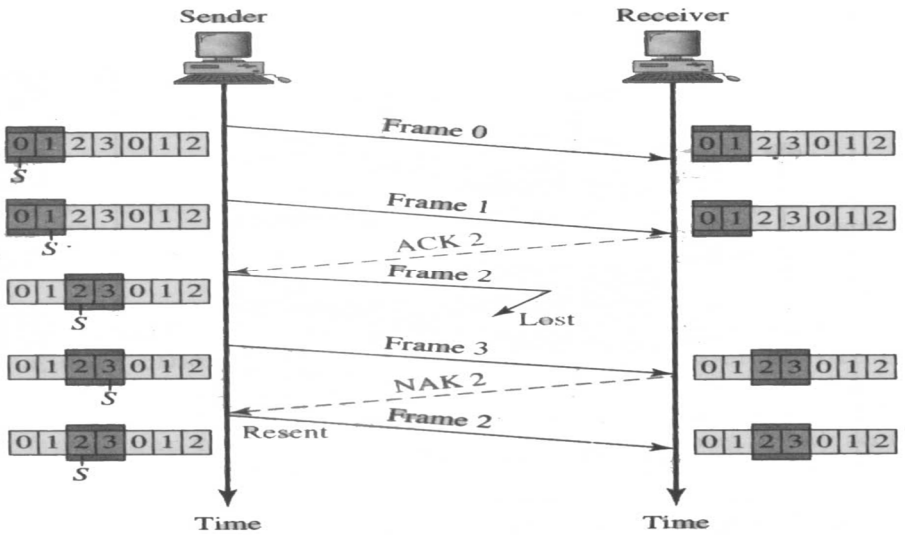


Figure 13 Selective Repeat ARQ, lost frame

Sender Window Size

We can now show why the size of the sender and receiver Windows must be at most one-half of 2^m. For an example, we choose m = 2, which means the size of the Window should be $2^m/2$ or 2. Figure 14 compares a window size of 2 with a window size of 3.

If the size of the window is 2 and all acknowledgments are lost, the timer for frame 0 expires and frame 0 is resent. However, the window of the receiver is now expecting frame 2, not frame 0, so this duplicate frame is correctly discarded. When the size of the window is 3 and all acknowledgments are lost, the sender sends a duplicate of frame 0. However, this time, the window of the receiver expects to receive frame 0 (0 is part of the Window), so it accepts frame 0, not as a duplicate, but as the first frame in the next cycle. This is clearly an error.

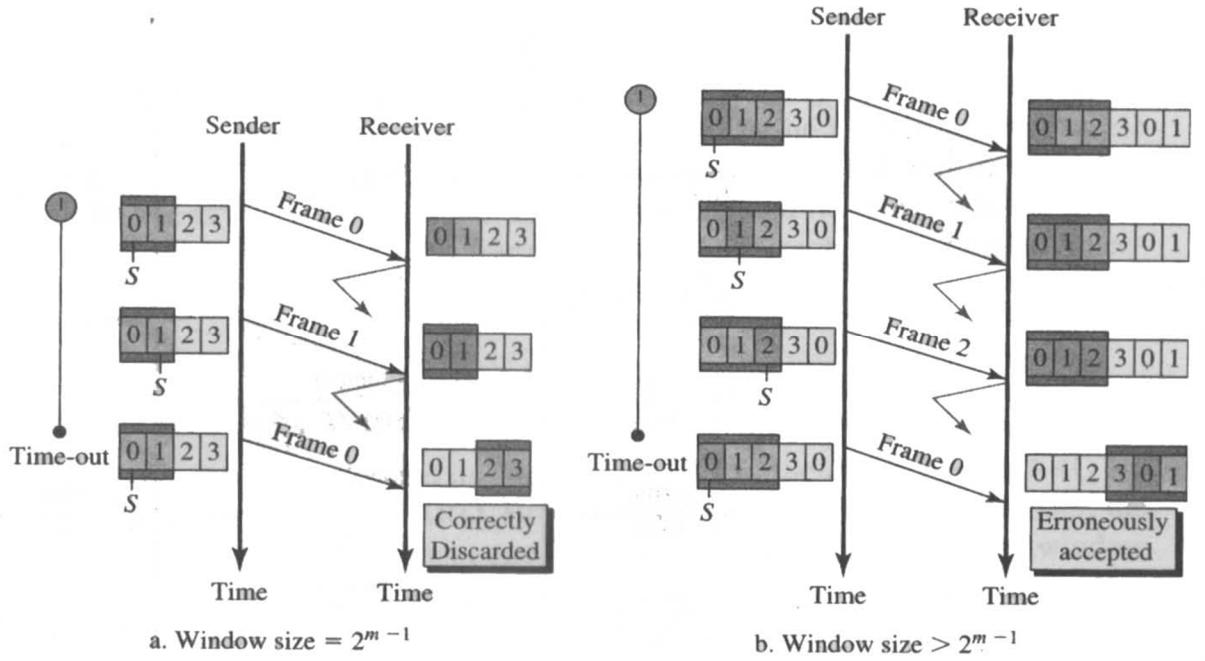


Figure 14 Selective Repeat ARQ, sender window size

Bidirectional Transmission and Piggybacking

As in the case of Stop-and-Wait ARQ and the Go-Back-N ARQ, Selective Repeat ARQ can also be bidirectional. We can use piggybacking to improve the efficiency of the transmission. However each direction needs both a sender window and a receiver window.

Chapter 4

The Medium Access Control Sublayer

The Channel Allocation Problem

- Static Channel Allocation in LANs and MANs
- Dynamic Channel Allocation in LANs and MANs

Dynamic Channel Allocation in LANs and MANs

1. Station Model.

2. Single Channel Assumption.

3. Collision Assumption.

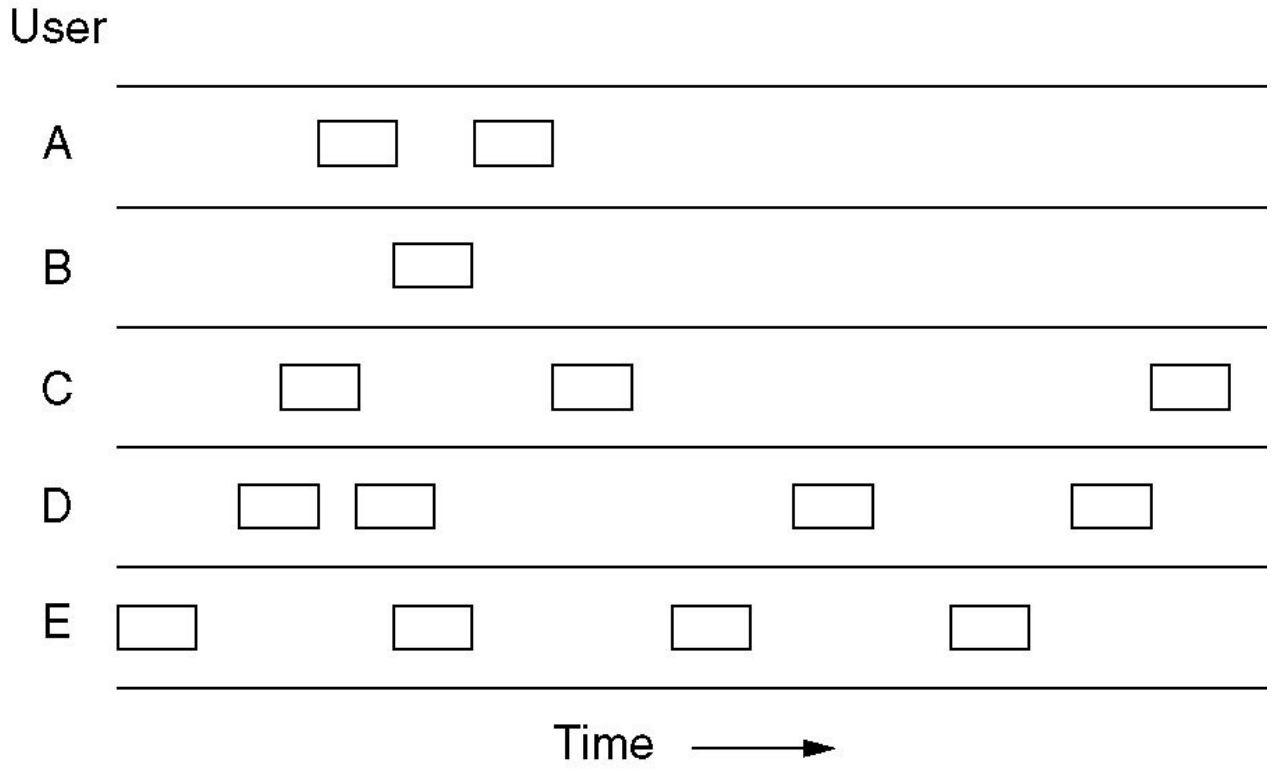
4. (a) Continuous Time.
   (b) Slotted Time.

5. (a) Carrier Sense.
   (b) No Carrier Sense.

Multiple Access Protocols

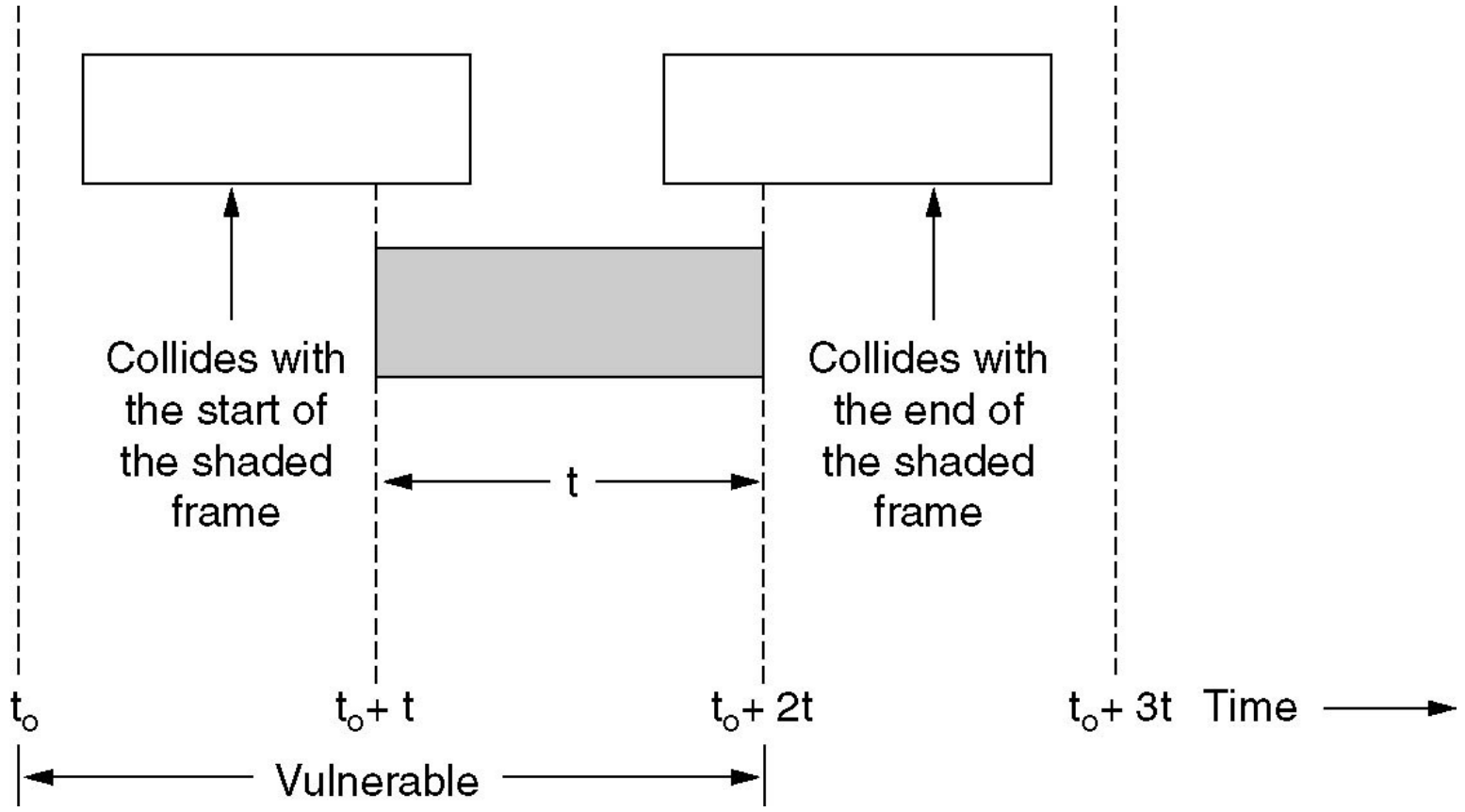
- ALOHA
- Carrier Sense Multiple Access Protocols
- Collision-Free Protocols
- Limited-Contention Protocols
- Wavelength Division Multiple Access Protocols
- Wireless LAN Protocols

Pure ALOHA



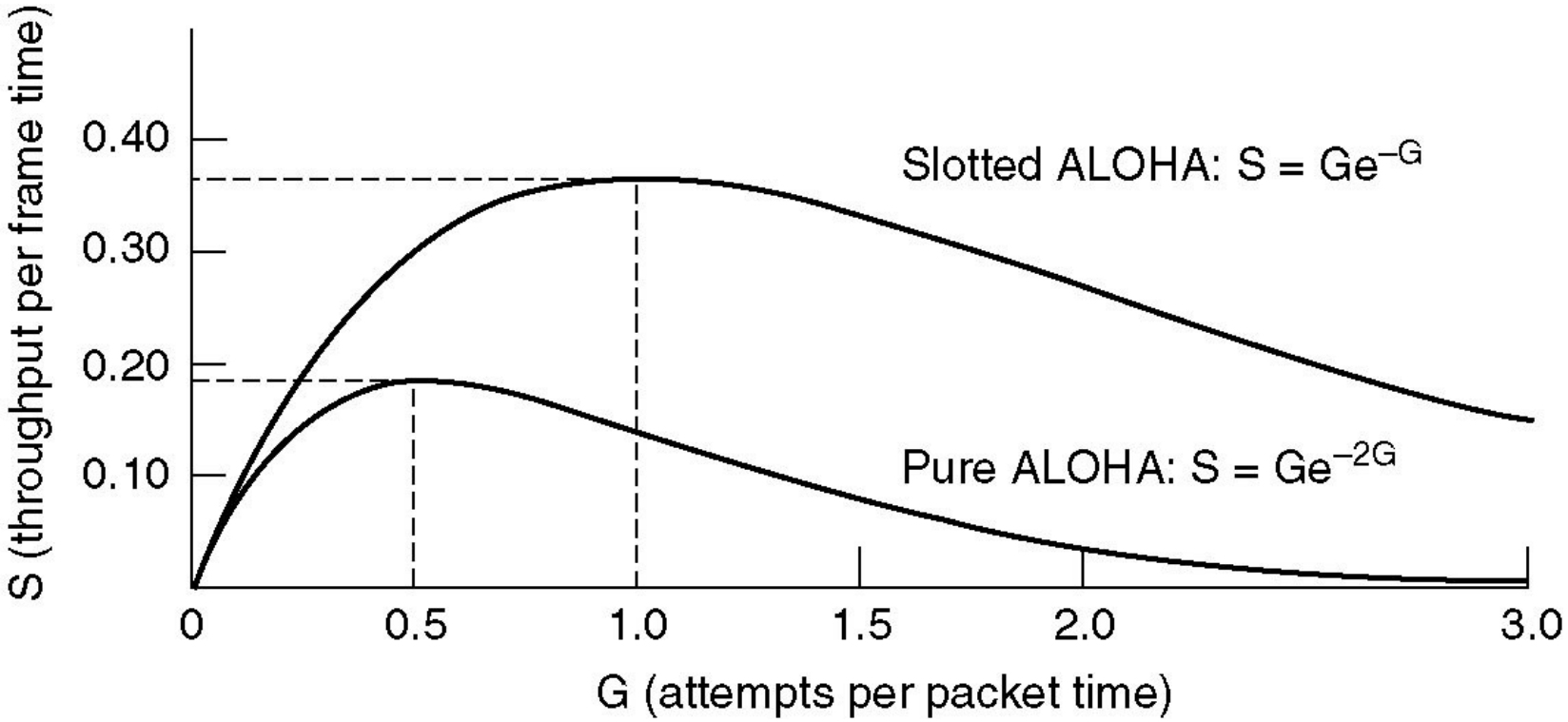
In pure ALOHA, frames are transmitted at completely arbitrary times.

Pure ALOHA (2)



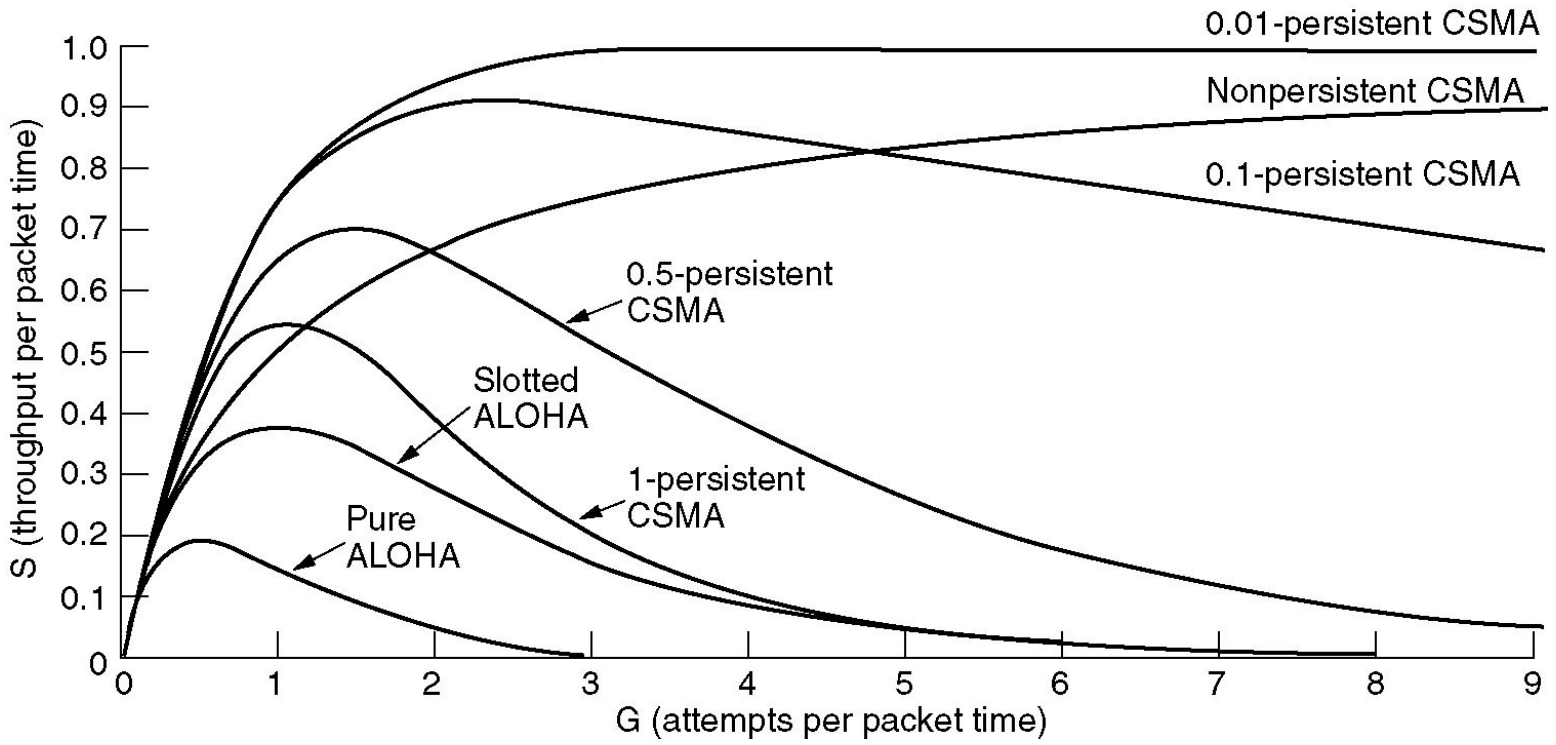
Vulnerable period for the shaded frame.

Pure ALOHA (3)



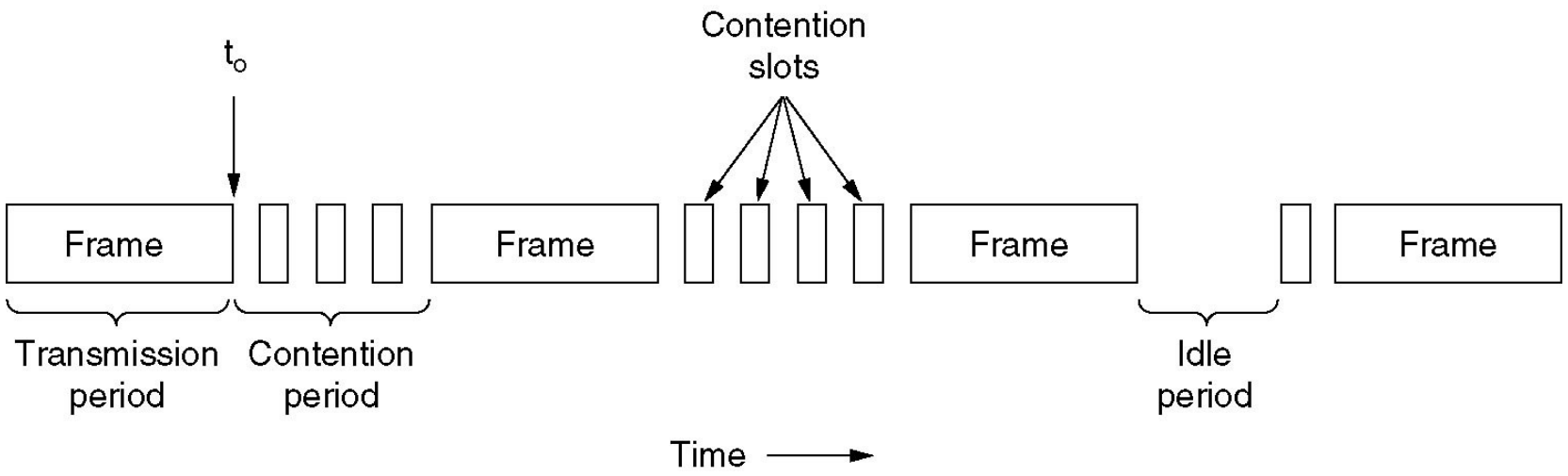
Throughput versus offered traffic for ALOHA systems.

Persistent and Nonpersistent CSMA



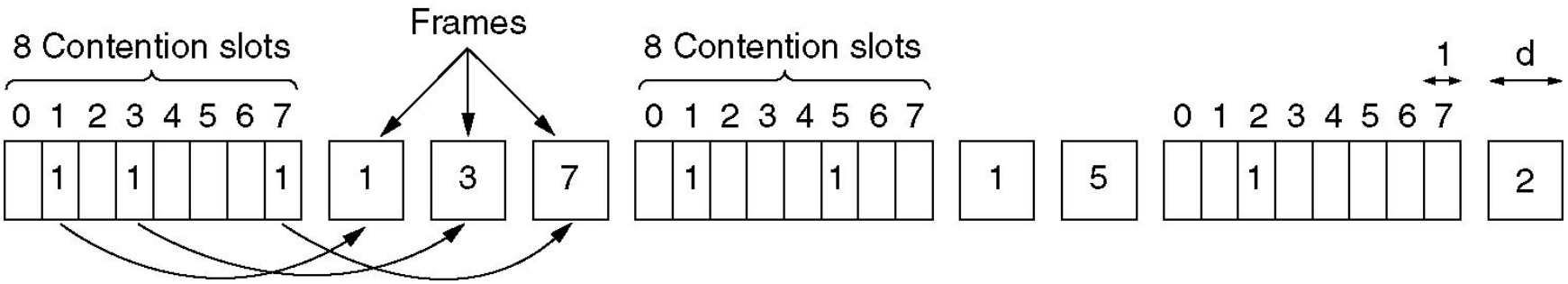
Comparison of the channel utilization versus load for various random access protocols.

CSMA with Collision Detection



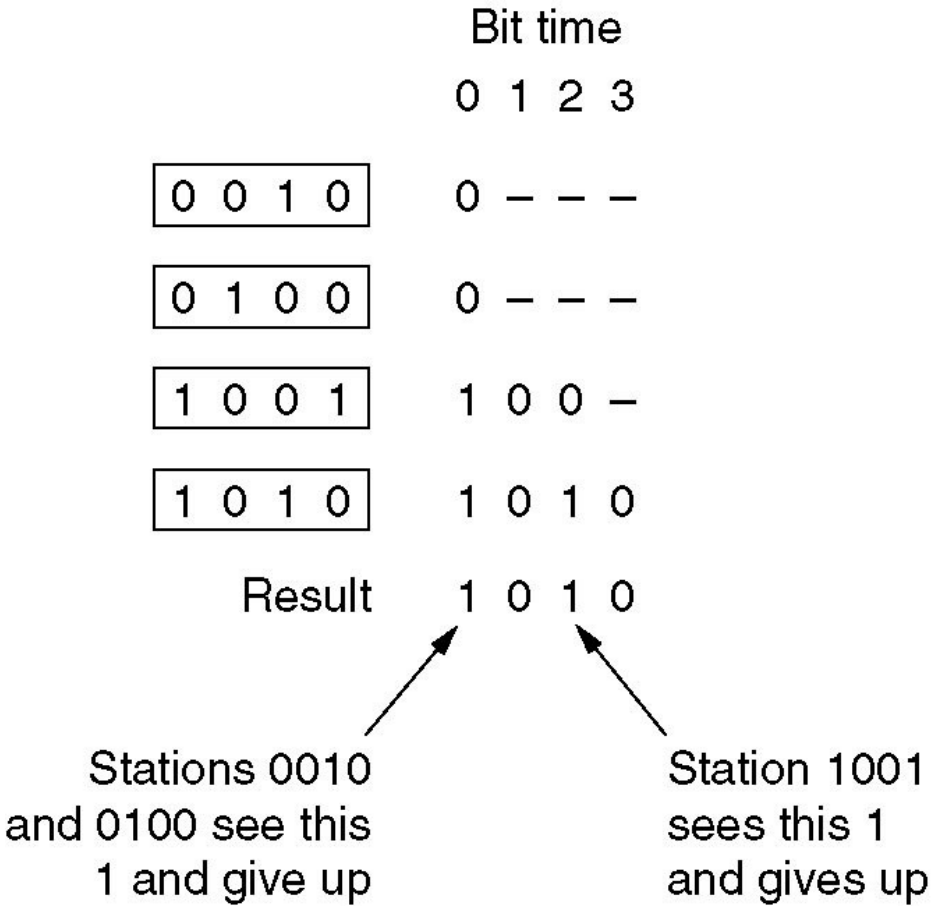
CSMA/CD can be in one of three states: contention, transmission, or idle.

Collision-Free Protocols



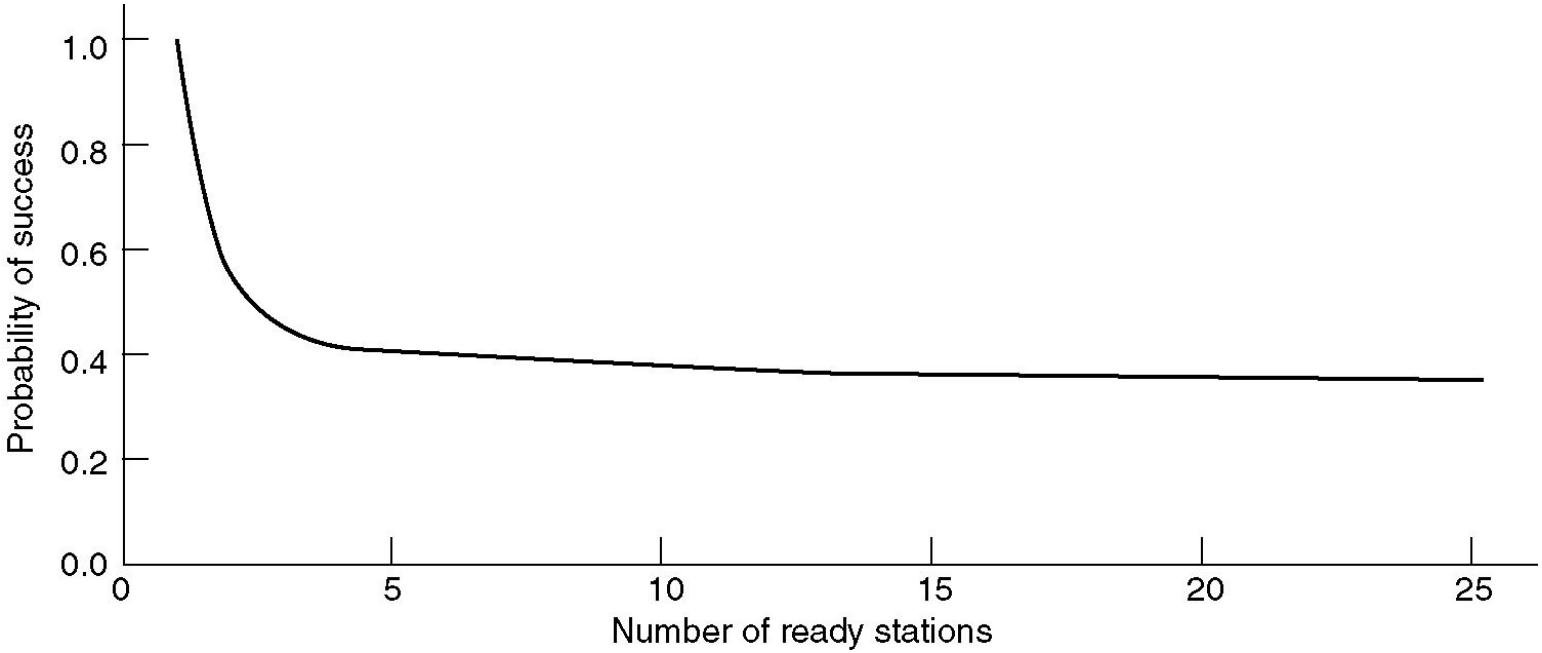
The basic bit-map protocol.

Collision-Free Protocols (2)



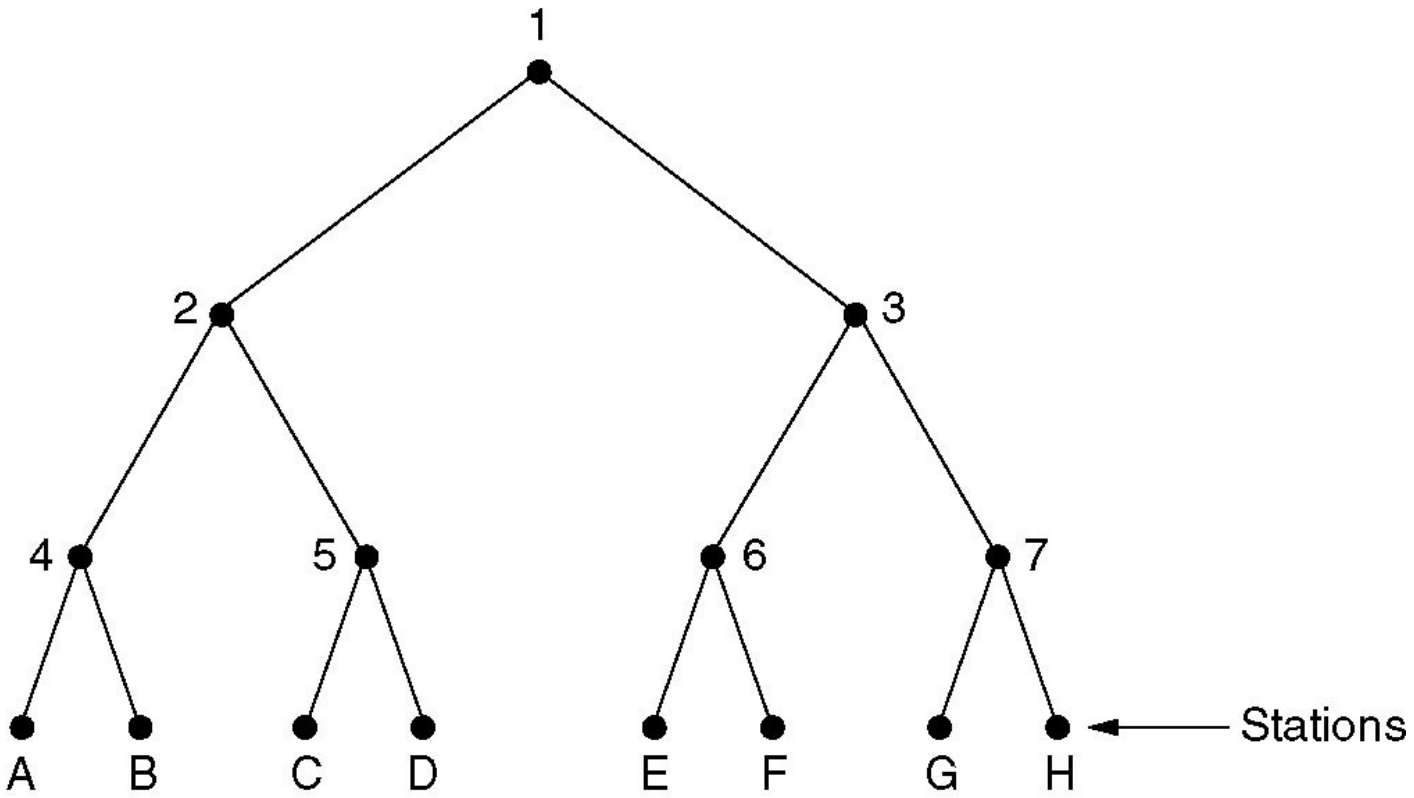
The binary countdown protocol. A dash indicates silence.

Limited-Contention Protocols



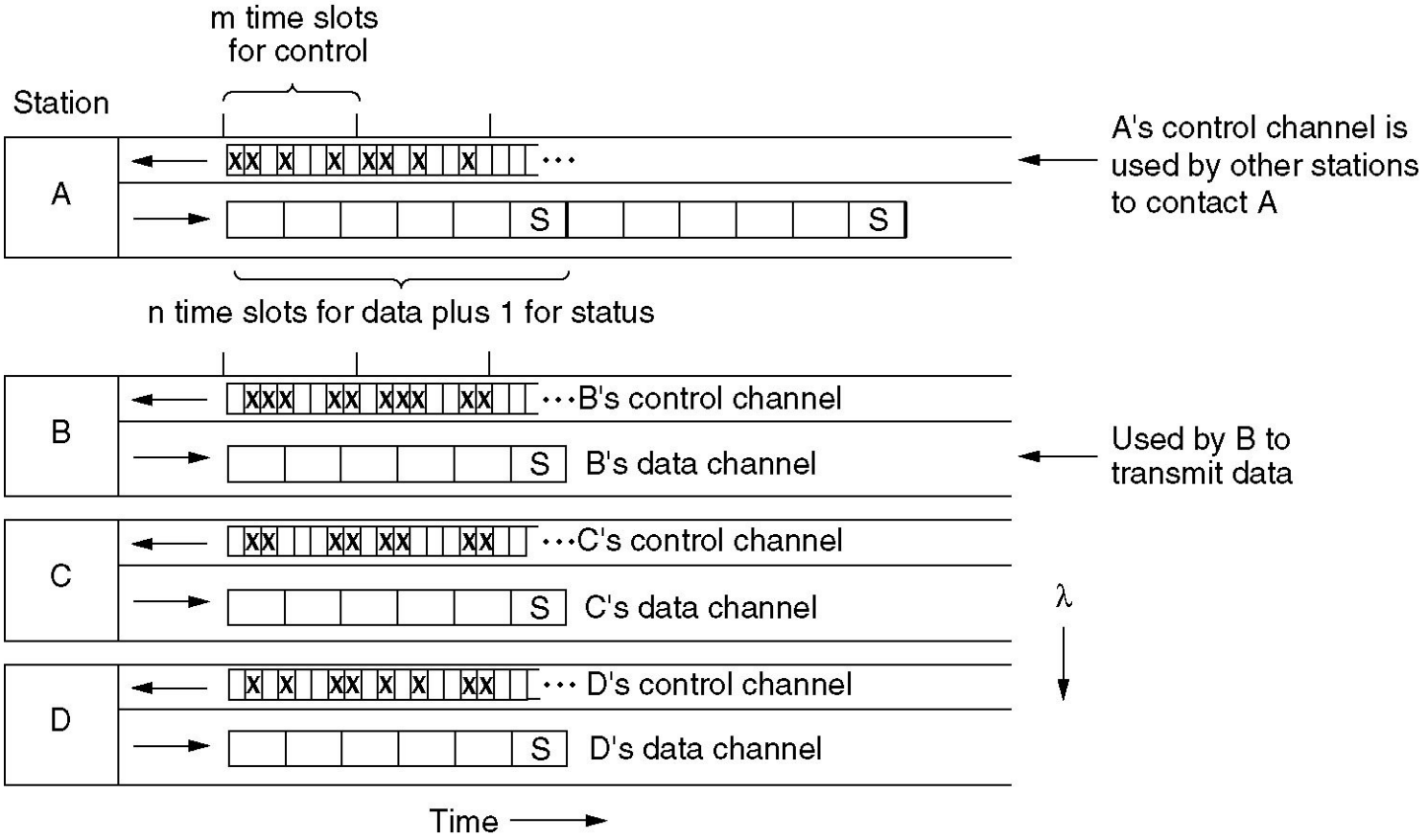
Acquisition probability for a symmetric contention channel.

Adaptive Tree Walk Protocol



The tree for eight stations.

Wavelength Division Multiple Access Protocols

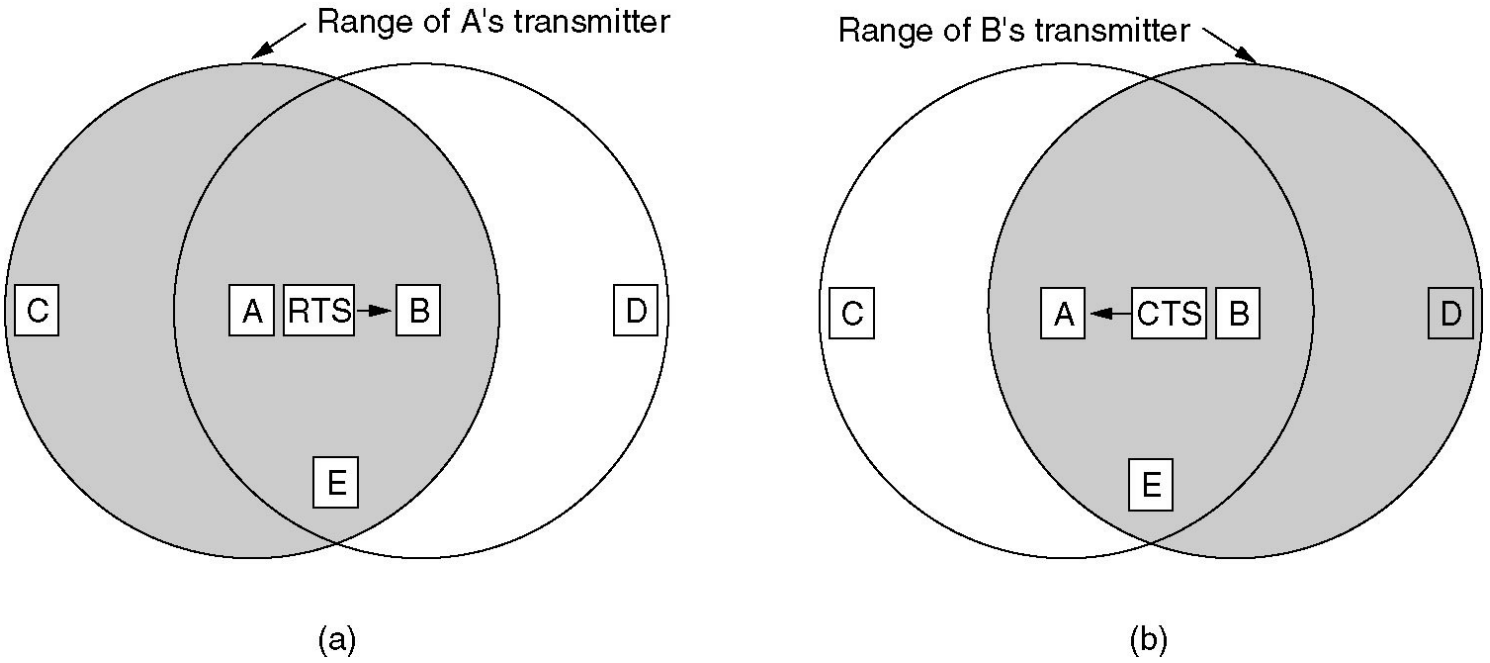


Wavelength division multiple access.

Wireless LAN Protocols

A wireless LAN. (a) A transmitting. (b) B transmitting.

Wireless LAN Protocols (2)



The MACA protocol. (a) A sending an RTS to B. (b) B responding with a CTS to A.

Ethernet

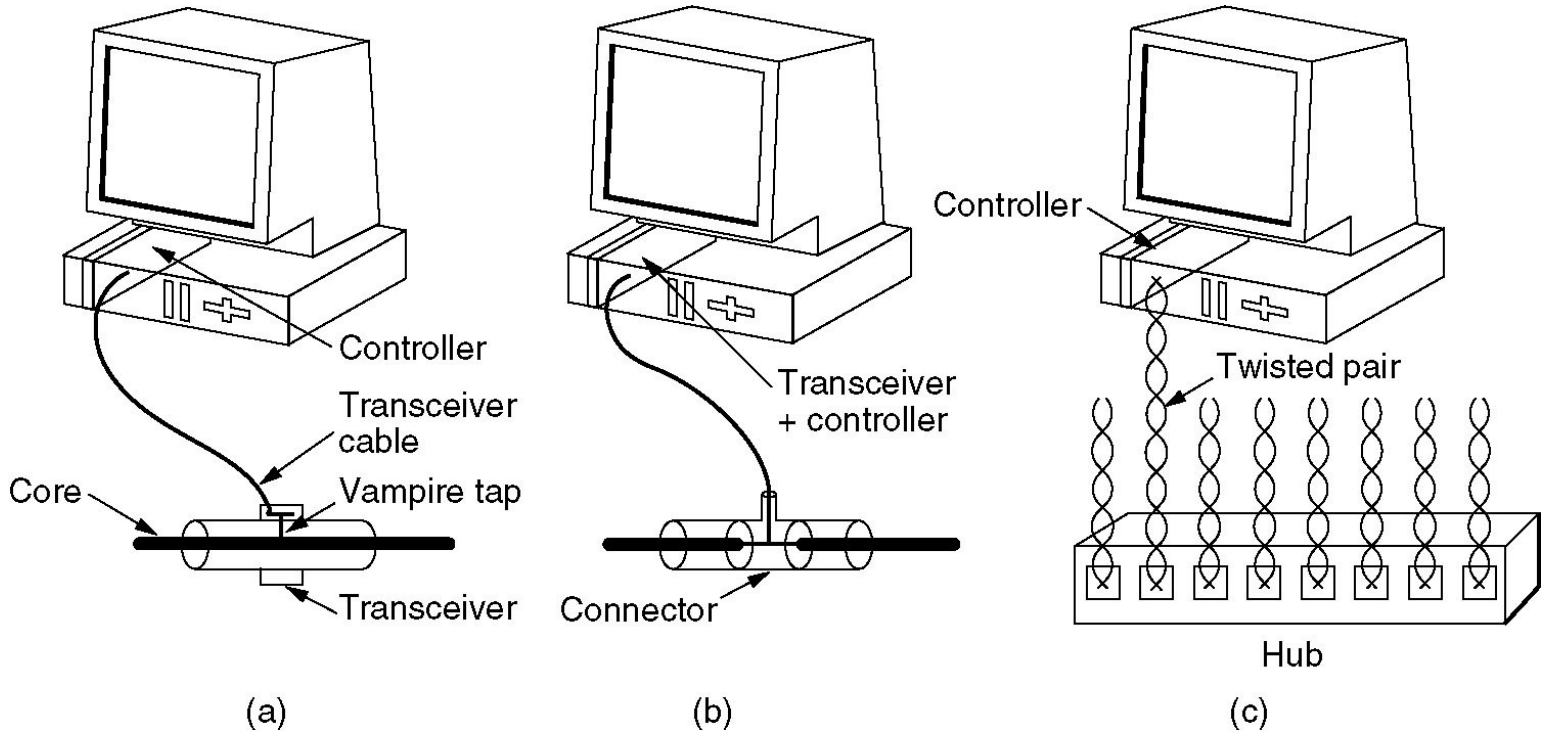
- Ethernet Cabling
- Manchester Encoding
- The Ethernet MAC Sublayer Protocol
- The Binary Exponential Backoff Algorithm
- Ethernet Performance
- Switched Ethernet
- Fast Ethernet
- Gigabit Ethernet
- IEEE 802.2: Logical Link Control
- Retrospective on Ethernet

Ethernet Cabling

Name	Cable	Max. seg.	Nodes/seg.	Advantages
10Base5	Thick coax	500 m	100	Original cable; now obsolete
10Base2	Thin coax	185 m	30	No hub needed
10Base-T	Twisted pair	100 m	1024	Cheapest system
10Base-F	Fiber optics	2000 m	1024	Best between buildings

The most common kinds of Ethernet cabling.

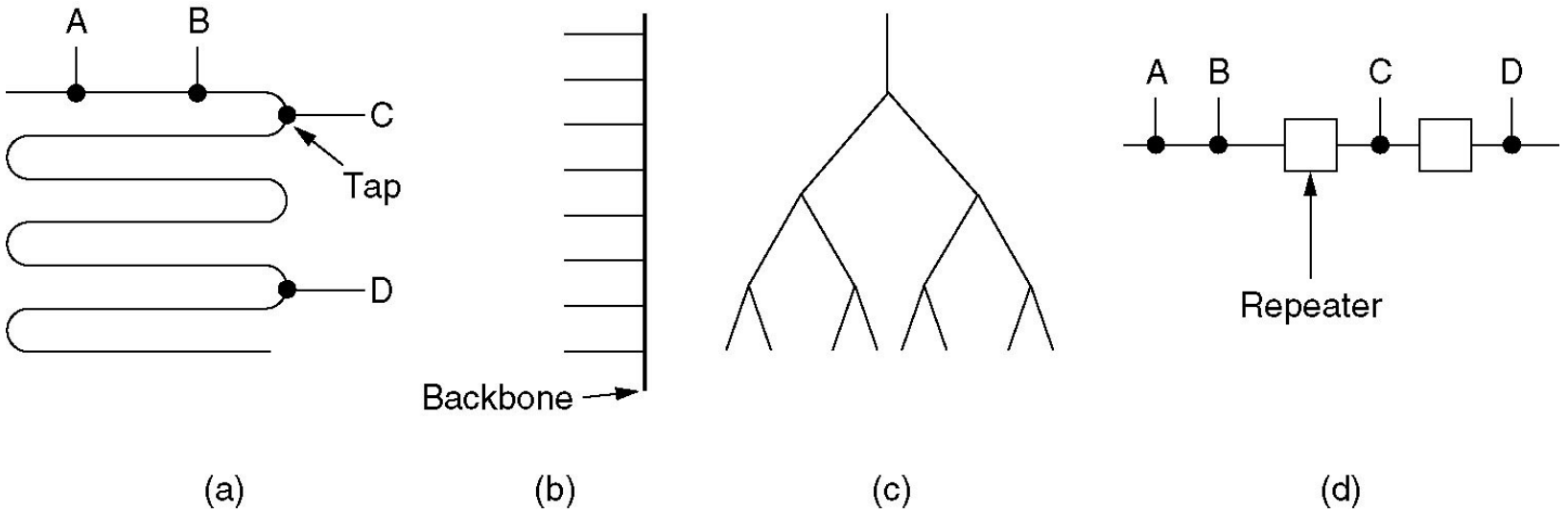
Ethernet Cabling (2)



Three kinds of Ethernet cabling.

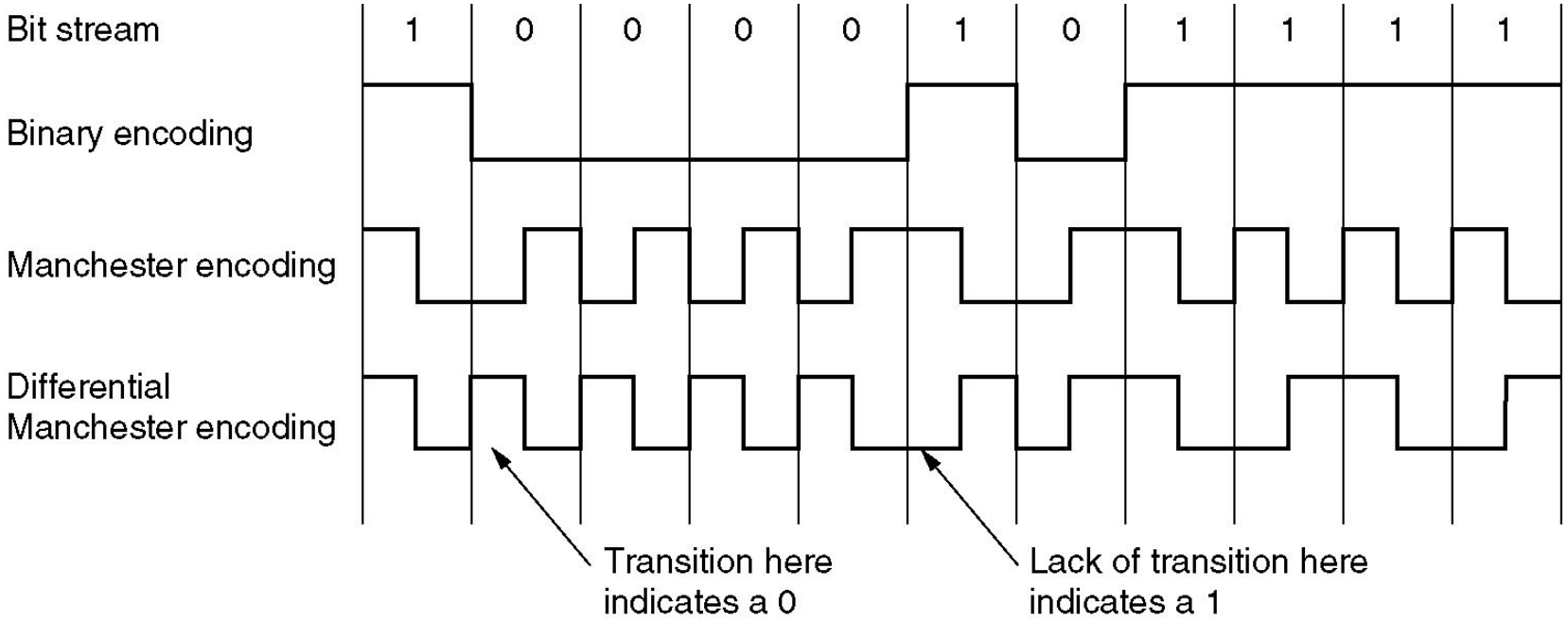
(a) 10Base5, (b) 10Base2, (c) 10Base-T.

Ethernet Cabling (3)



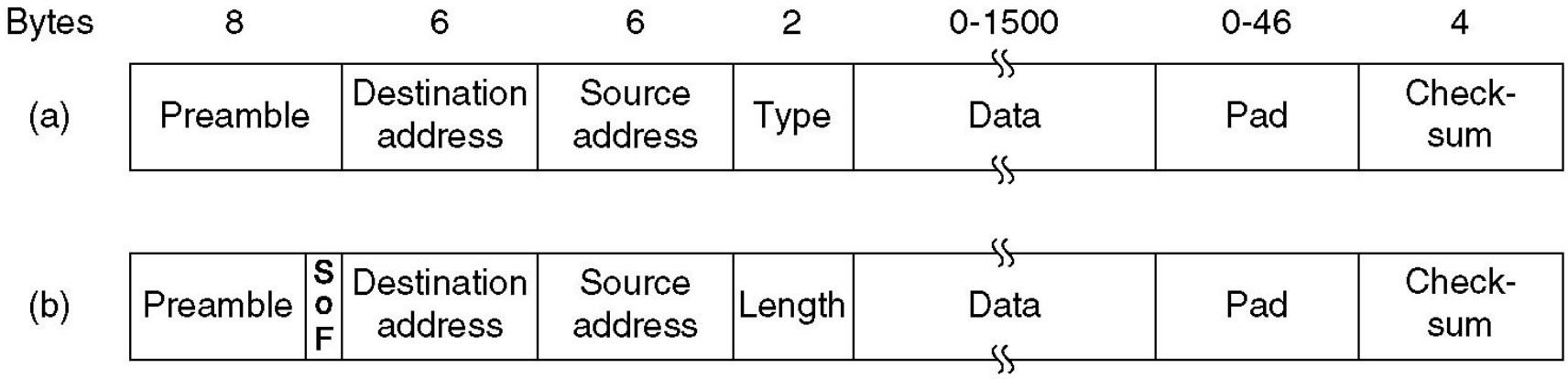
Cable topologies. (a) Linear, (b) Spine, (c) Tree, (d) Segmented.

Ethernet Cabling (4)



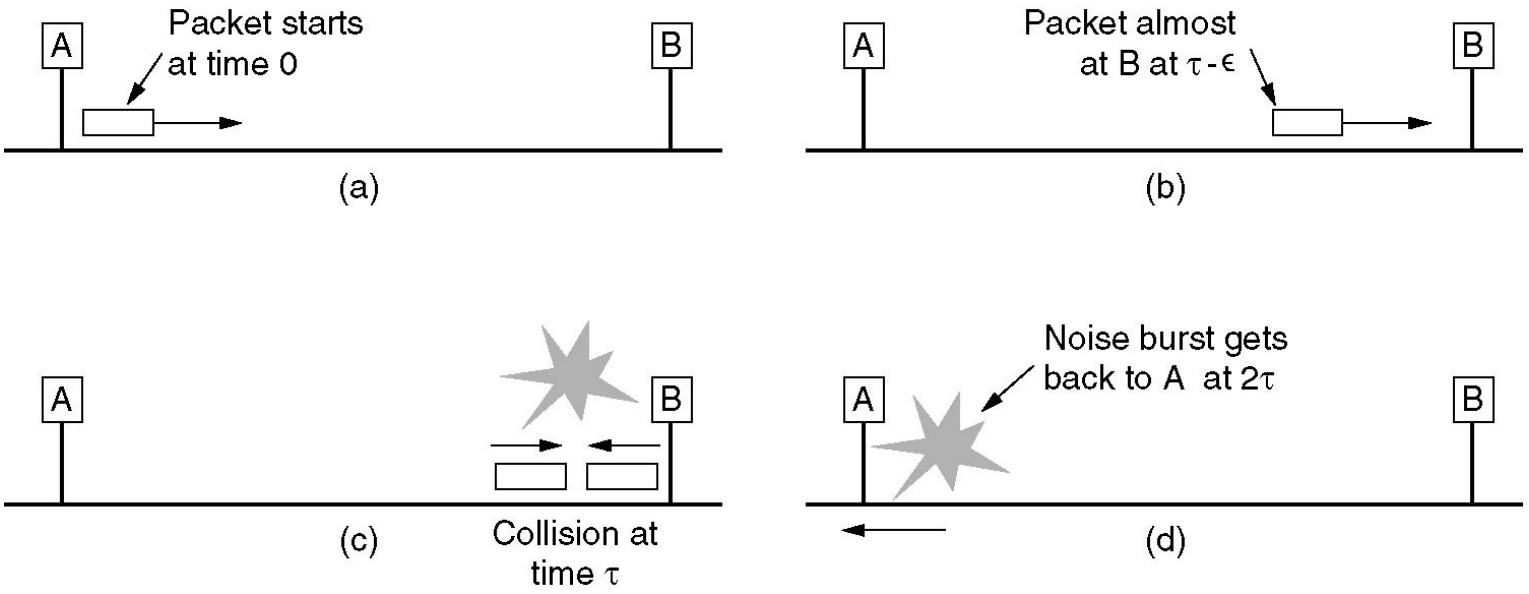
(a) Binary encoding, (b) Manchester encoding,
(c) Differential Manchester encoding.

Ethernet MAC Sublayer Protocol



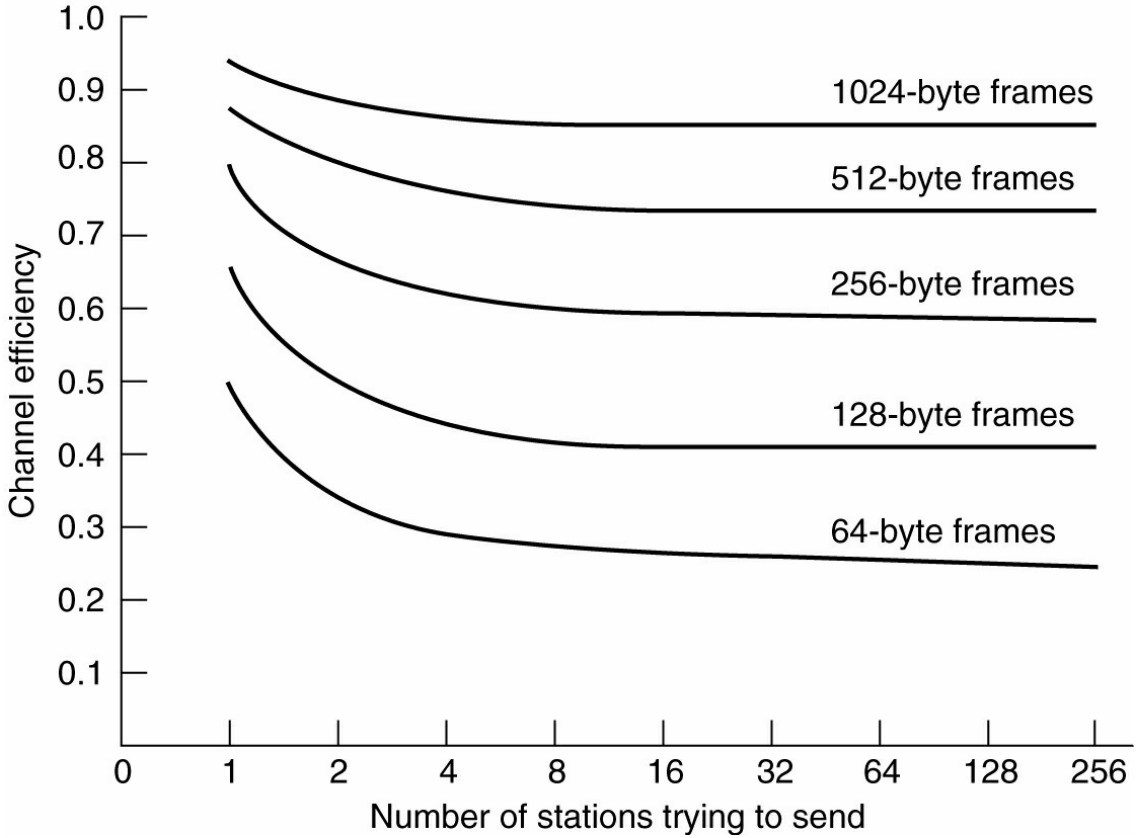
Frame formats. (a) DIX Ethernet, (b) IEEE 802.3.

Ethernet MAC Sublayer Protocol (2)



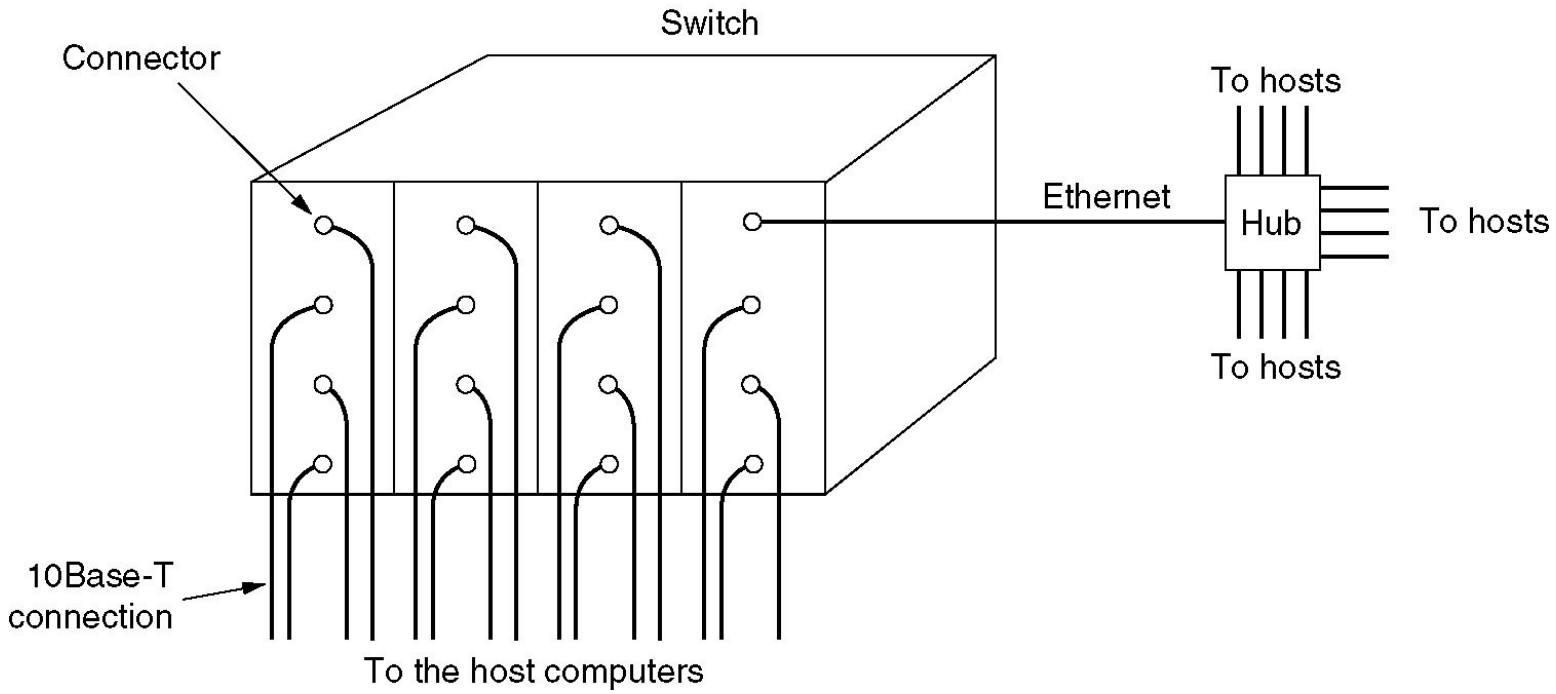
Collision detection can take as long as 2τ.

Ethernet Performance



Efficiency of Ethernet at 10 Mbps with 512-bit slot times.

Switched Ethernet



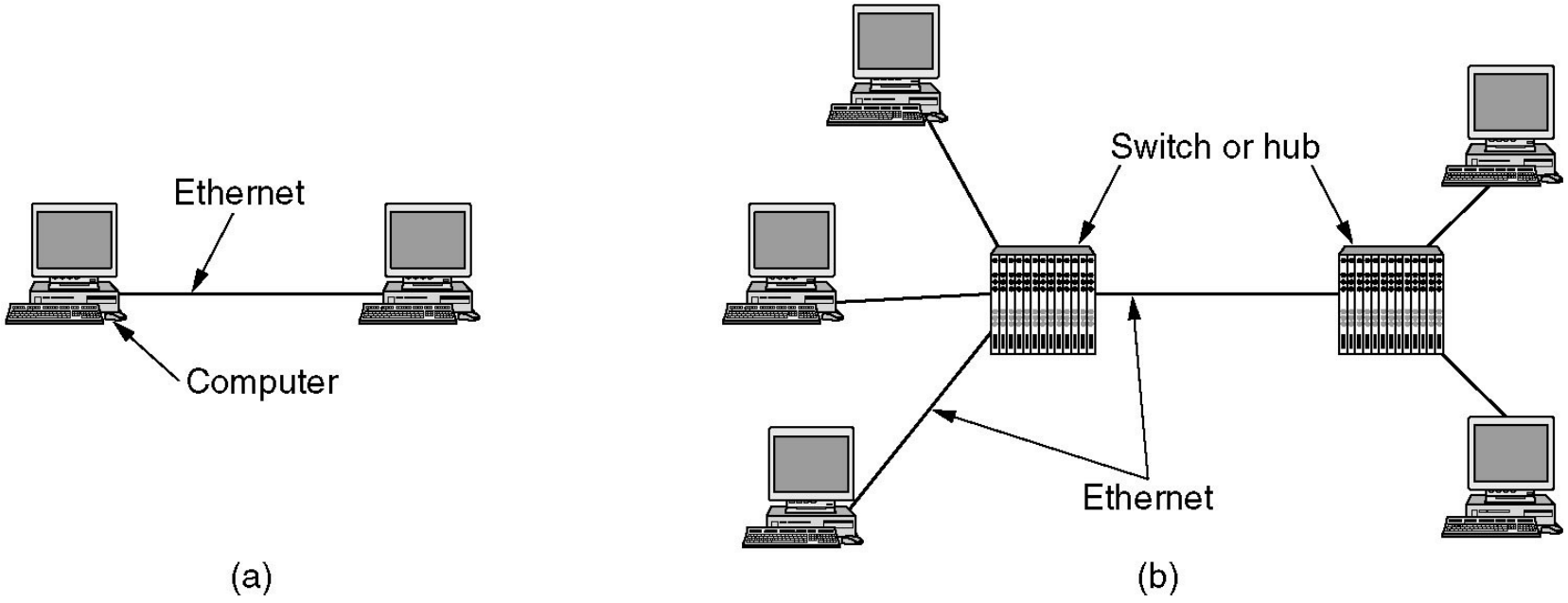
A simple example of switched Ethernet.

Fast Ethernet

Name	Cable	Max. segment	Advantages
100Base-T4	Twisted pair	100 m	Uses category 3 UTP
100Base-TX	Twisted pair	100 m	Full duplex at 100 Mbps
100Base-FX	Fiber optics	2000 m	Full duplex at 100 Mbps; long runs

The original fast Ethernet cabling.

Gigabit Ethernet



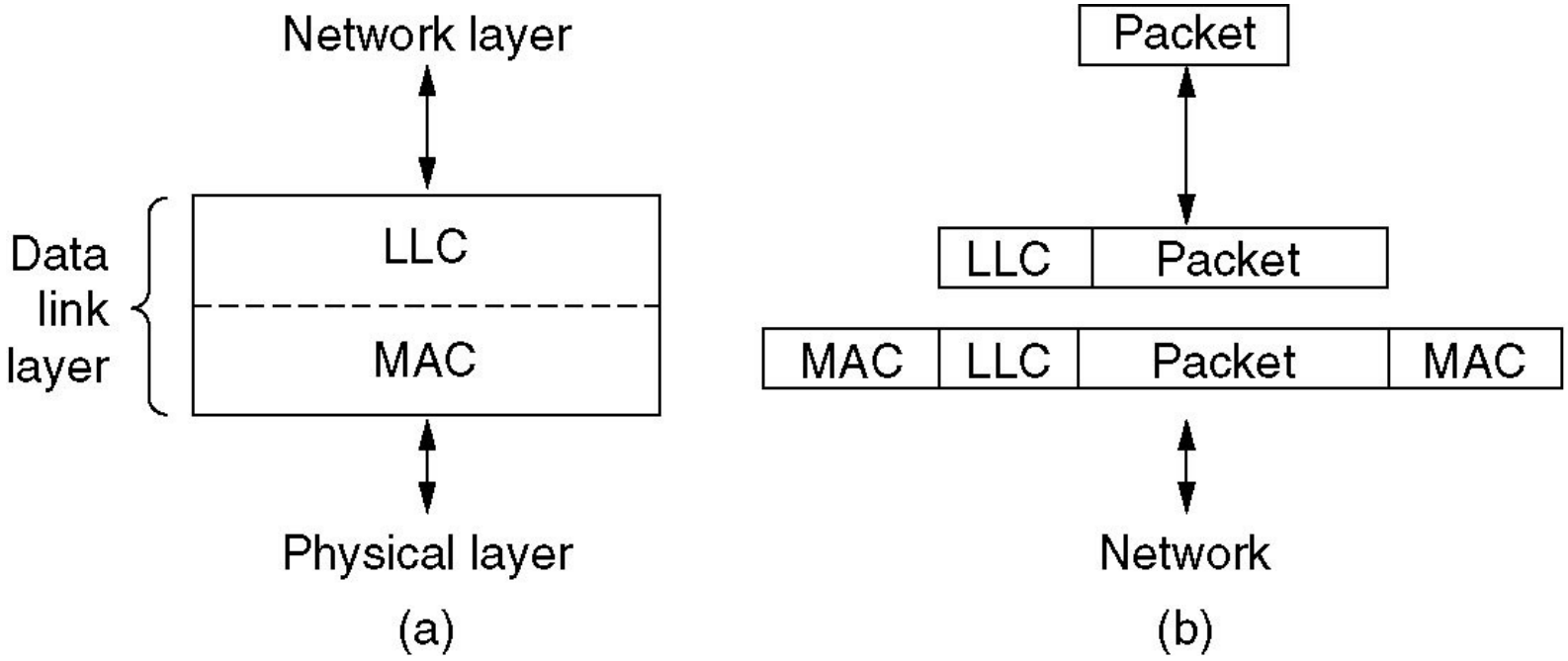
(a) A two-station Ethernet. (b) A multistation Ethernet.

Gigabit Ethernet (2)

Name	Cable	Max. segment	Advantages
1000Base-SX	Fiber optics	550 m	Multimode fiber (50, 62.5 microns)
1000Base-LX	Fiber optics	5000 m	Single (10 μ) or multimode (50, 62.5 μ)
1000Base-CX	2 Pairs of STP	25 m	Shielded twisted pair
1000Base-T	4 Pairs of UTP	100 m	Standard category 5 UTP

Gigabit Ethernet cabling.

IEEE 802.2: Logical Link Control

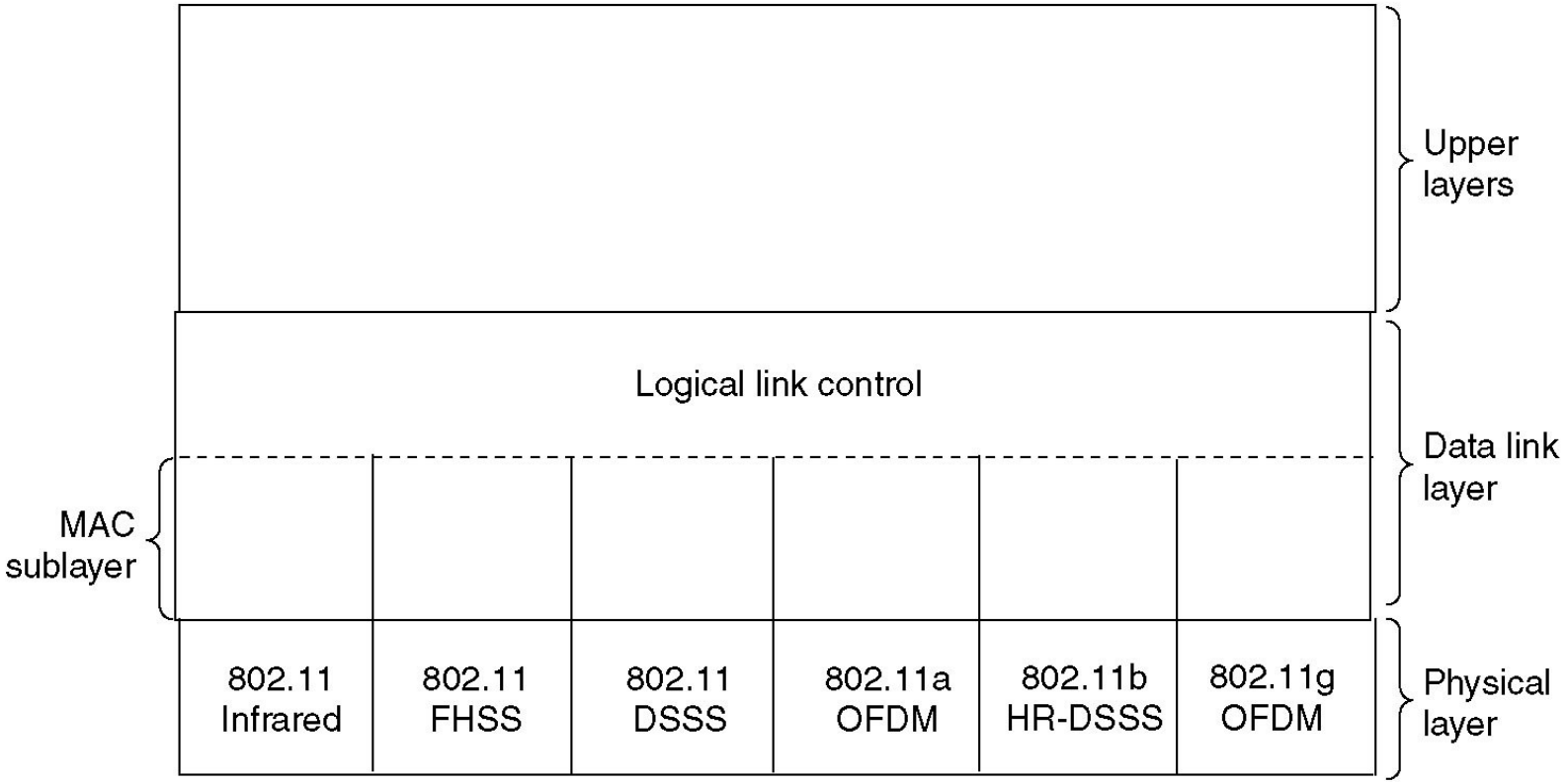


(a) Position of LLC. (b) Protocol formats.

Wireless LANs

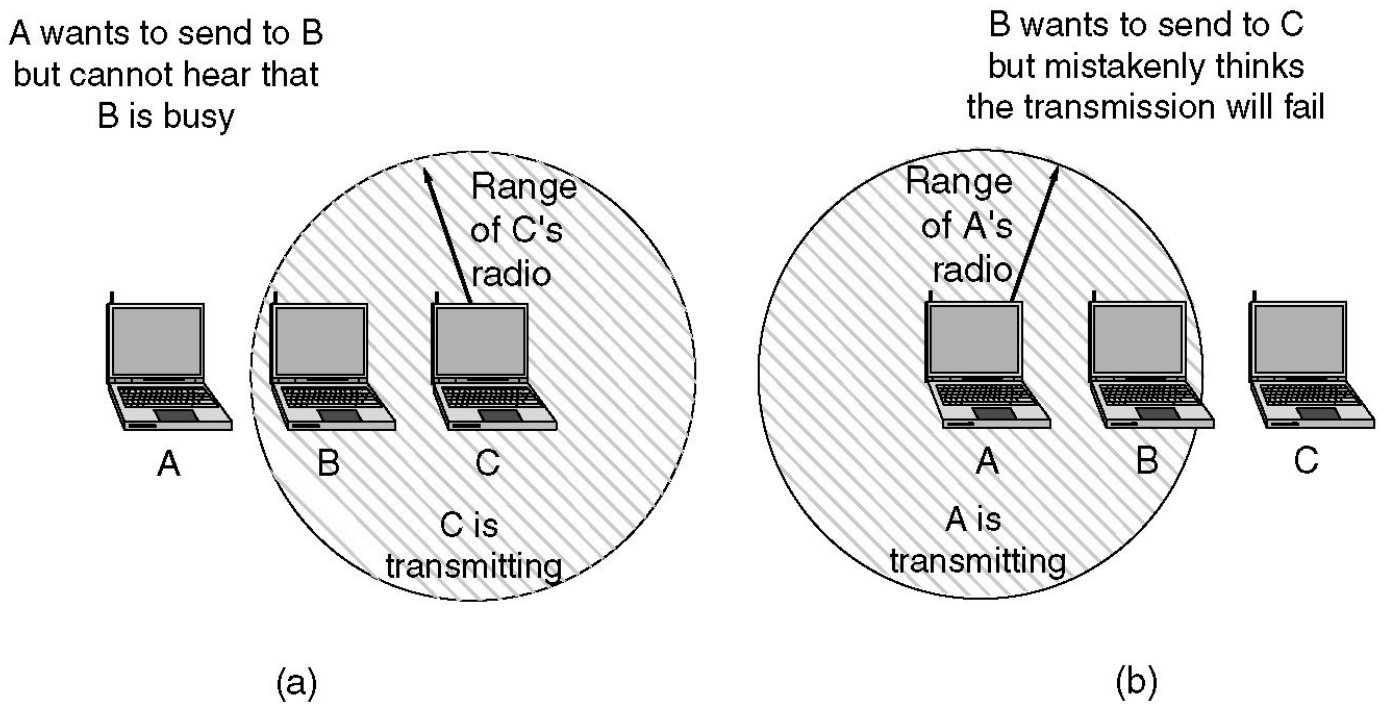
- The 802.11 Protocol Stack
- The 802.11 Physical Layer
- The 802.11 MAC Sublayer Protocol
- The 802.11 Frame Structure
- Services

The 802.11 Protocol Stack



Part of the 802.11 protocol stack.

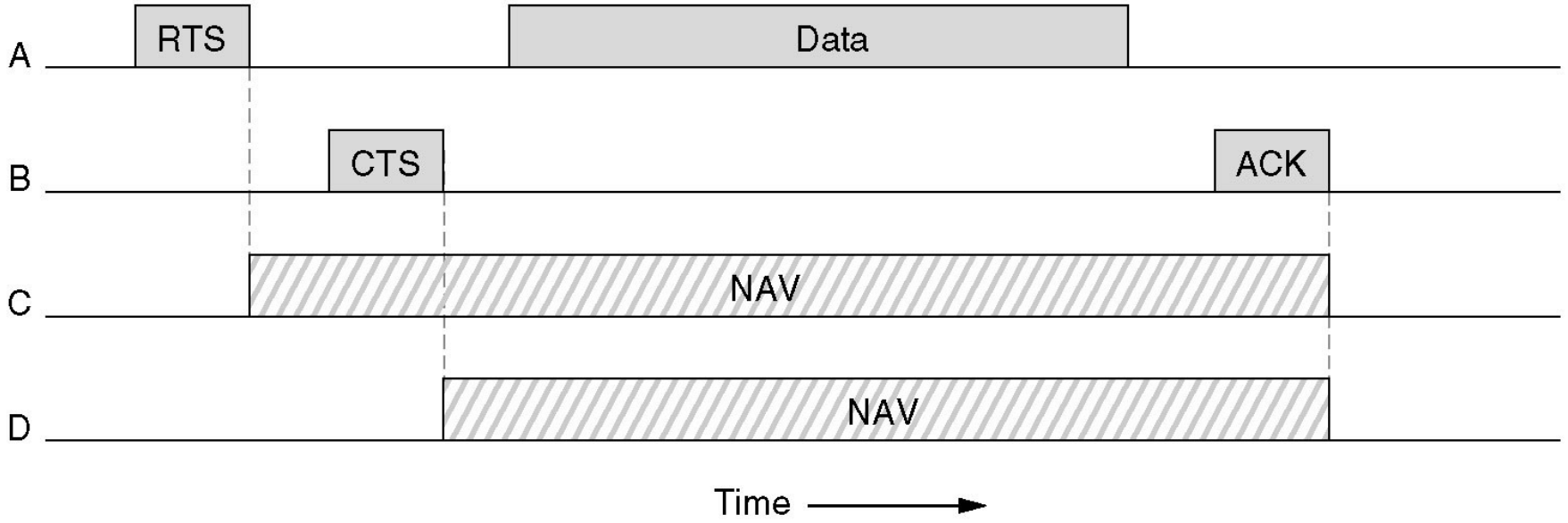
The 802.11 MAC Sublayer Protocol



(a) The hidden station problem.

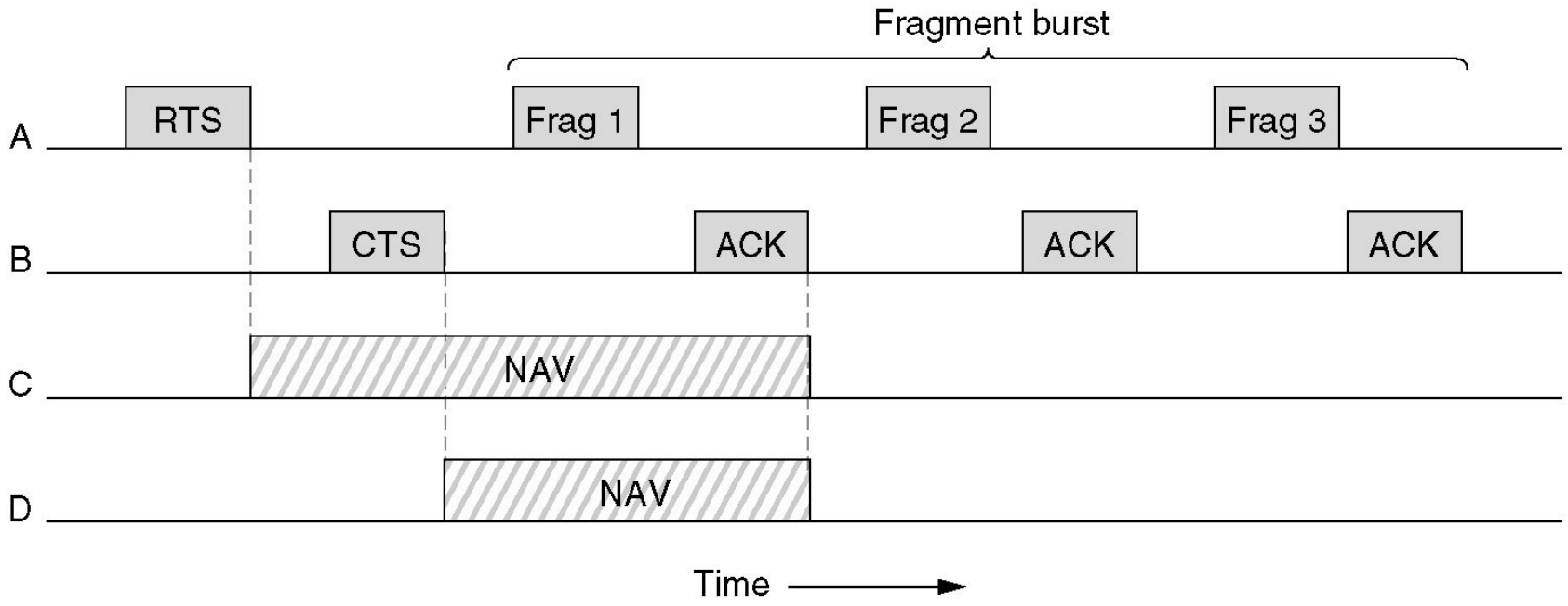
(b) The exposed station problem.

The 802.11 MAC Sublayer Protocol (2)



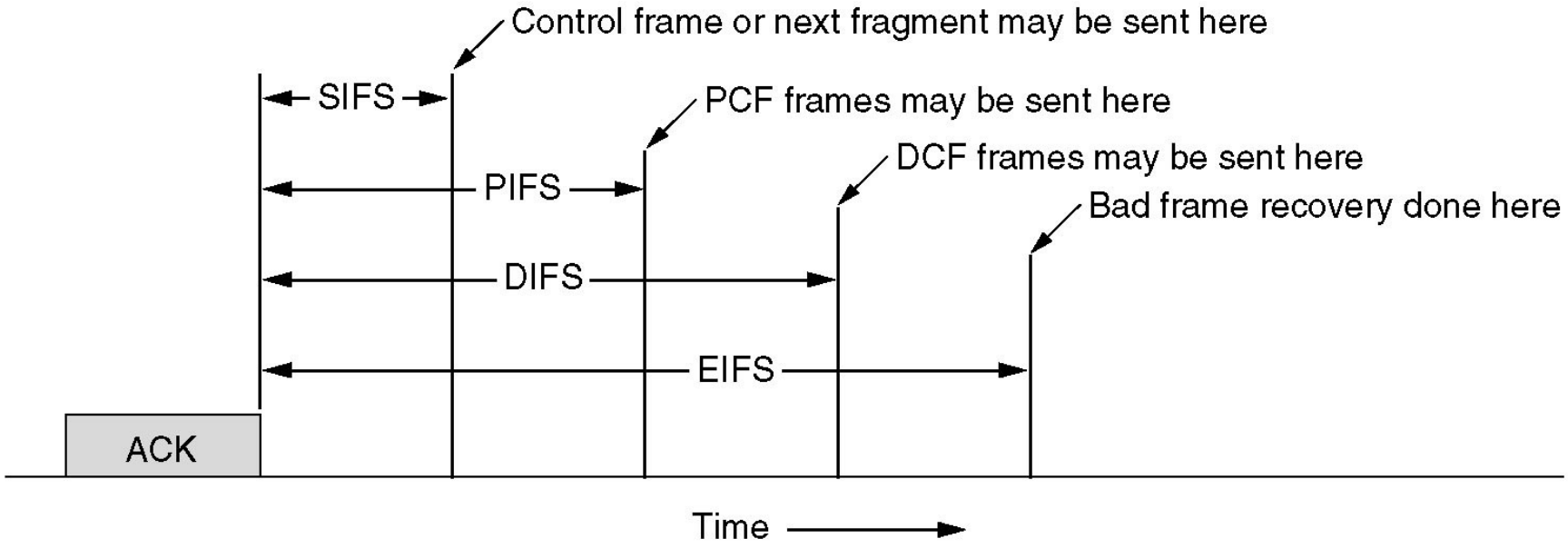
The use of virtual channel sensing using CSMA/CA.

The 802.11 MAC Sublayer Protocol (3)



A fragment burst.

The 802.11 MAC Sublayer Protocol (4)



Interframe spacing in 802.11.

The 802.11 Frame Structure

The 802.11 data frame.

802.11 Services

Distribution Services

- Association
- Disassociation
- Reassociation
- Distribution
- Integration

802.11 Services

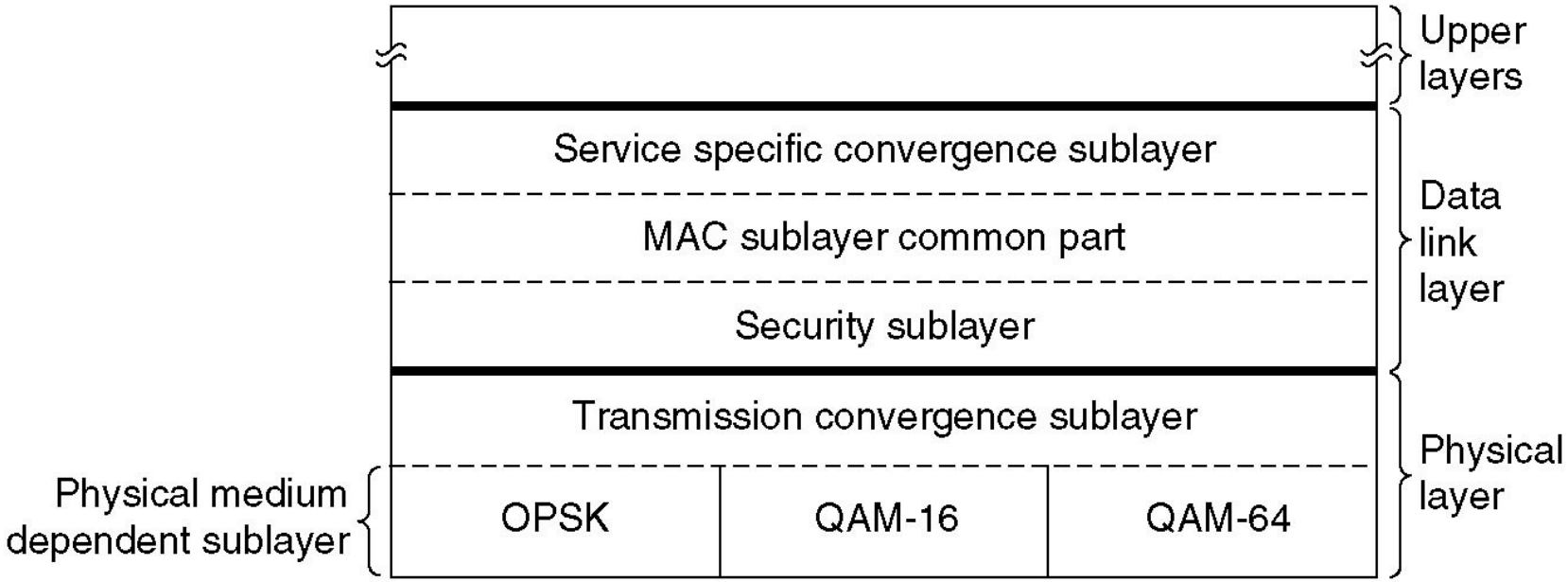
Intracell Services

- Authentication
- Deauthentication
- Privacy
- Data Delivery

Broadband Wireless

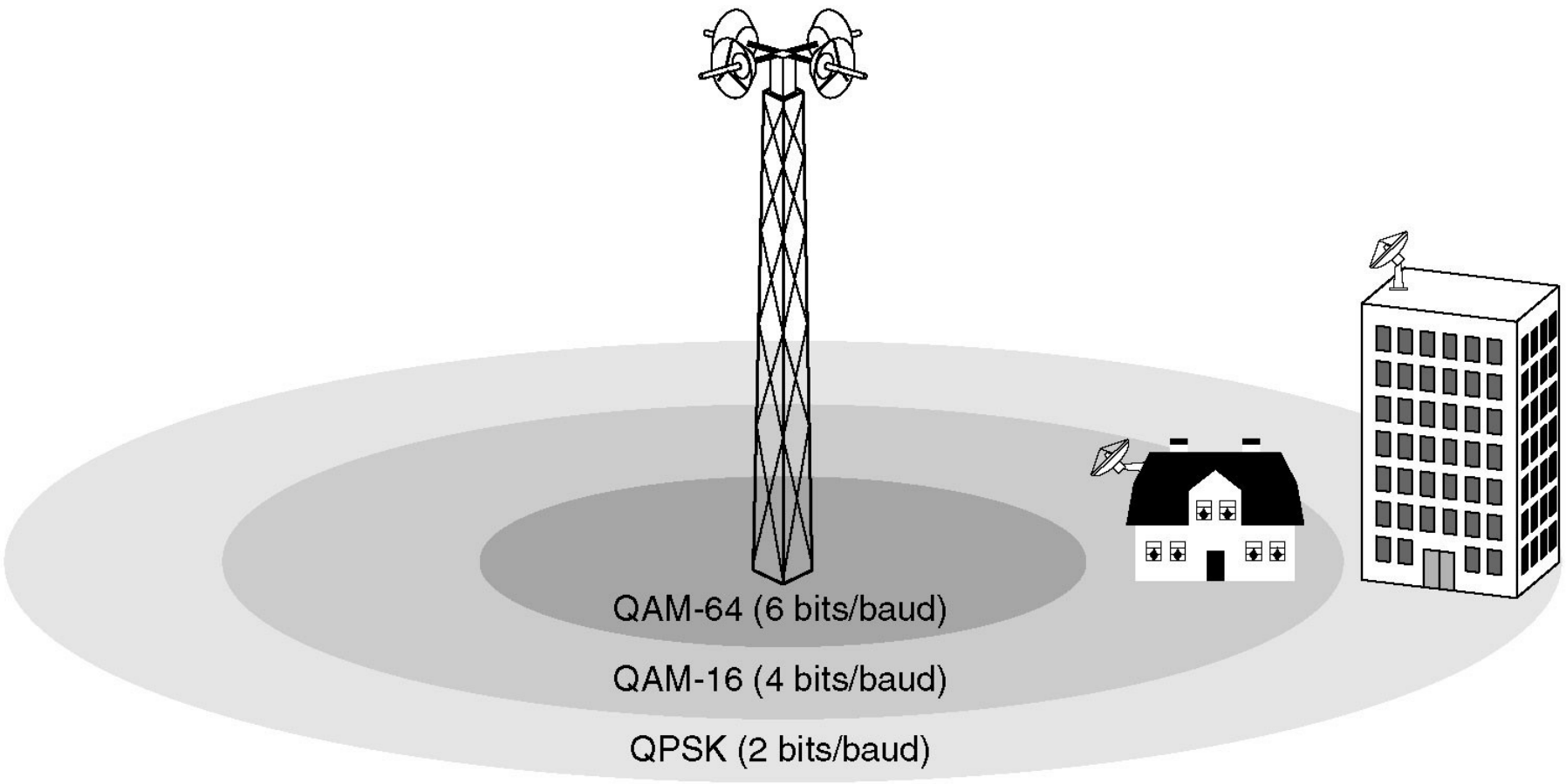
- Comparison of 802.11 and 802.16
- The 802.16 Protocol Stack
- The 802.16 Physical Layer
- The 802.16 MAC Sublayer Protocol
- The 802.16 Frame Structure

The 802.16 Protocol Stack



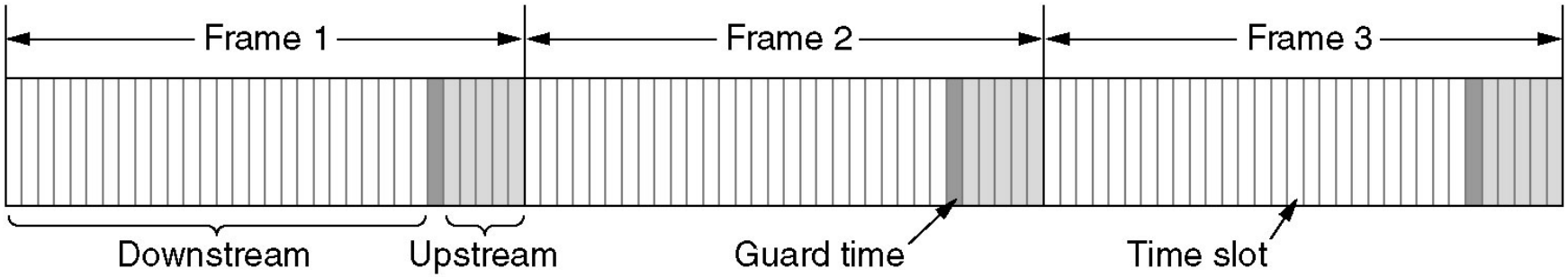
The 802.16 Protocol Stack.

The 802.16 Physical Layer



The 802.16 transmission environment.

The 802.16 Physical Layer (2)



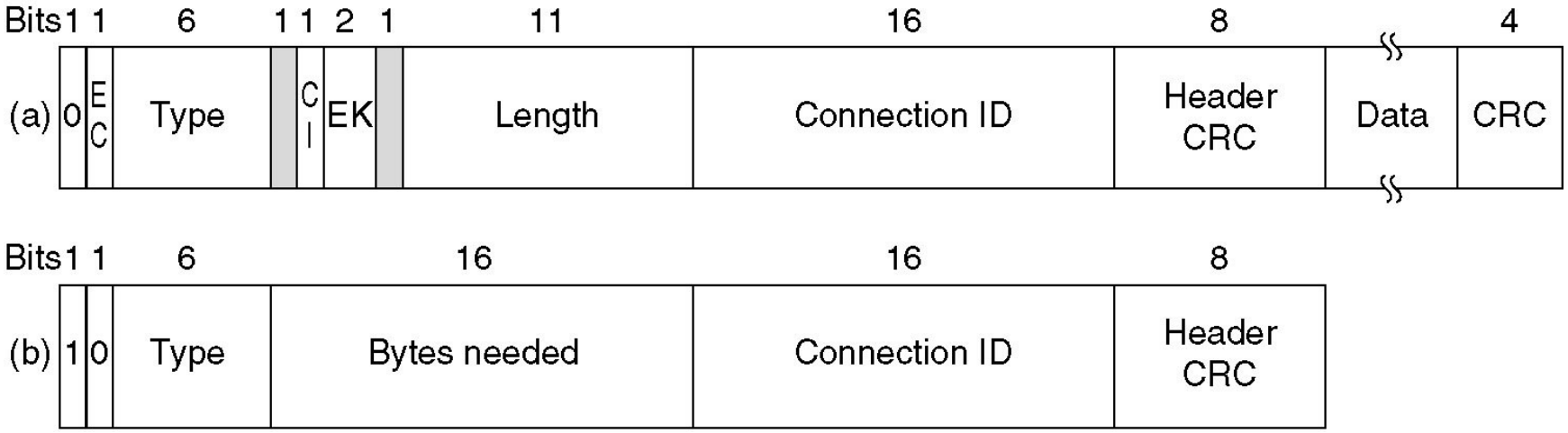
Frames and time slots for time division duplexing.

The 802.16 MAC Sublayer Protocol

Service Classes

- Constant bit rate service
- Real-time variable bit rate service
- Non-real-time variable bit rate service
- Best efforts service

The 802.16 Frame Structure

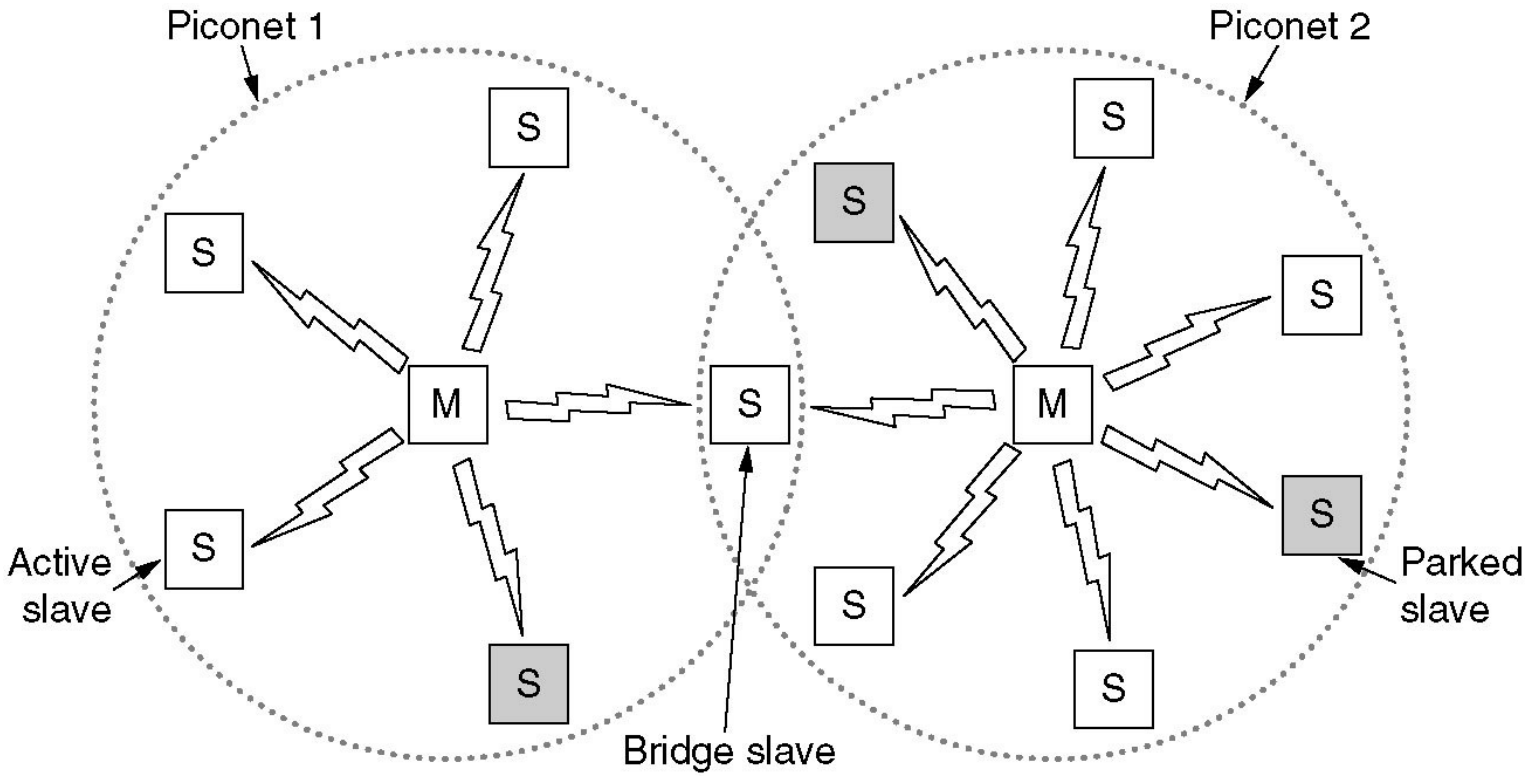


(a) A generic frame. (b) A bandwidth request frame.

Bluetooth

- Bluetooth Architecture
- Bluetooth Applications
- The Bluetooth Protocol Stack
- The Bluetooth Radio Layer
- The Bluetooth Baseband Layer
- The Bluetooth L2CAP Layer
- The Bluetooth Frame Structure

Bluetooth Architecture



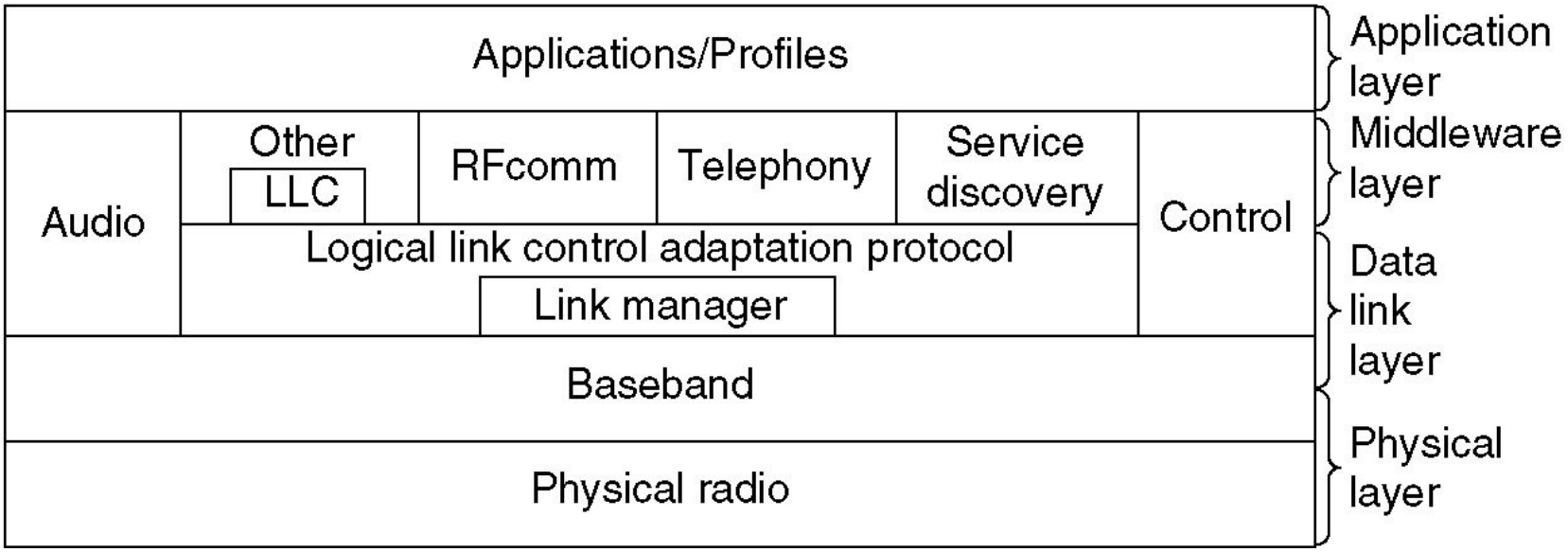
Two piconets can be connected to form a scatternet.

Bluetooth Applications

Name	Description
Generic access	Procedures for link management
Service discovery	Protocol for discovering offered services
Serial port	Replacement for a serial port cable
Generic object exchange	Defines client-server relationship for object movement
LAN access	Protocol between a mobile computer and a fixed LAN
Dial-up networking	Allows a notebook computer to call via a mobile phone
Fax	Allows a mobile fax machine to talk to a mobile phone
Cordless telephony	Connects a handset and its local base station
Intercom	Digital walkie-talkie
Headset	Intended for hands-free voice communication
Object push	Provides a way to exchange simple objects
File transfer	Provides a more general file transfer facility
Synchronization	Permits a PDA to synchronize with another computer

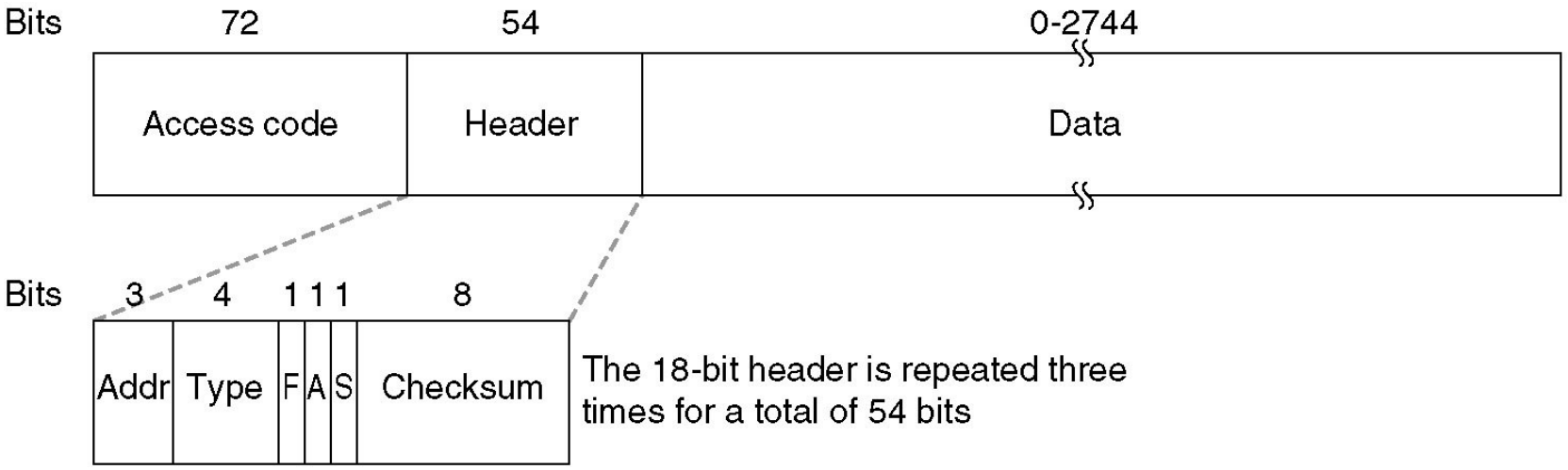
The Bluetooth profiles.

The Bluetooth Protocol Stack



The 802.15 version of the Bluetooth protocol architecture.

The Bluetooth Frame Structure

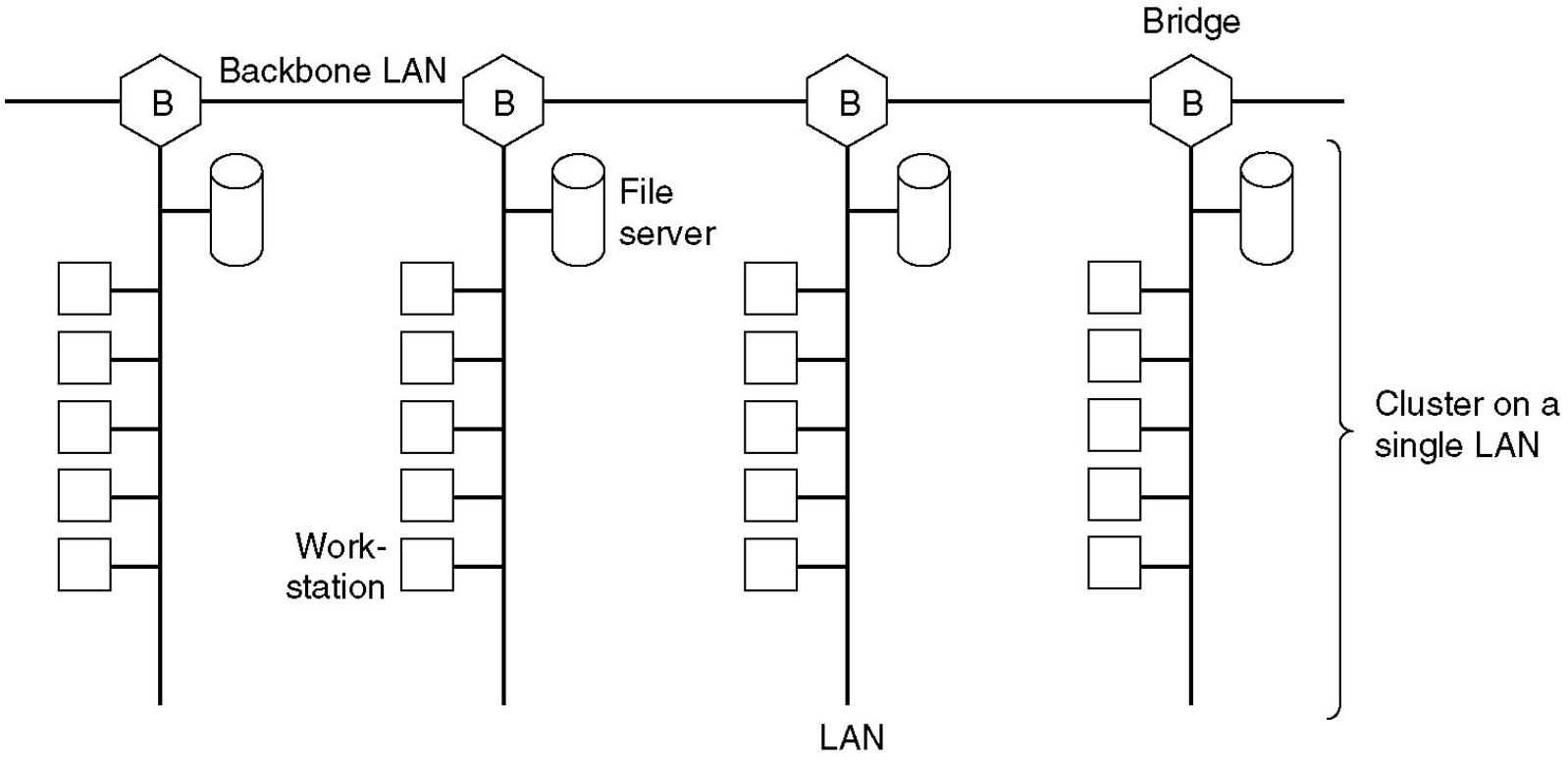


A typical Bluetooth data frame.

Data Link Layer Switching

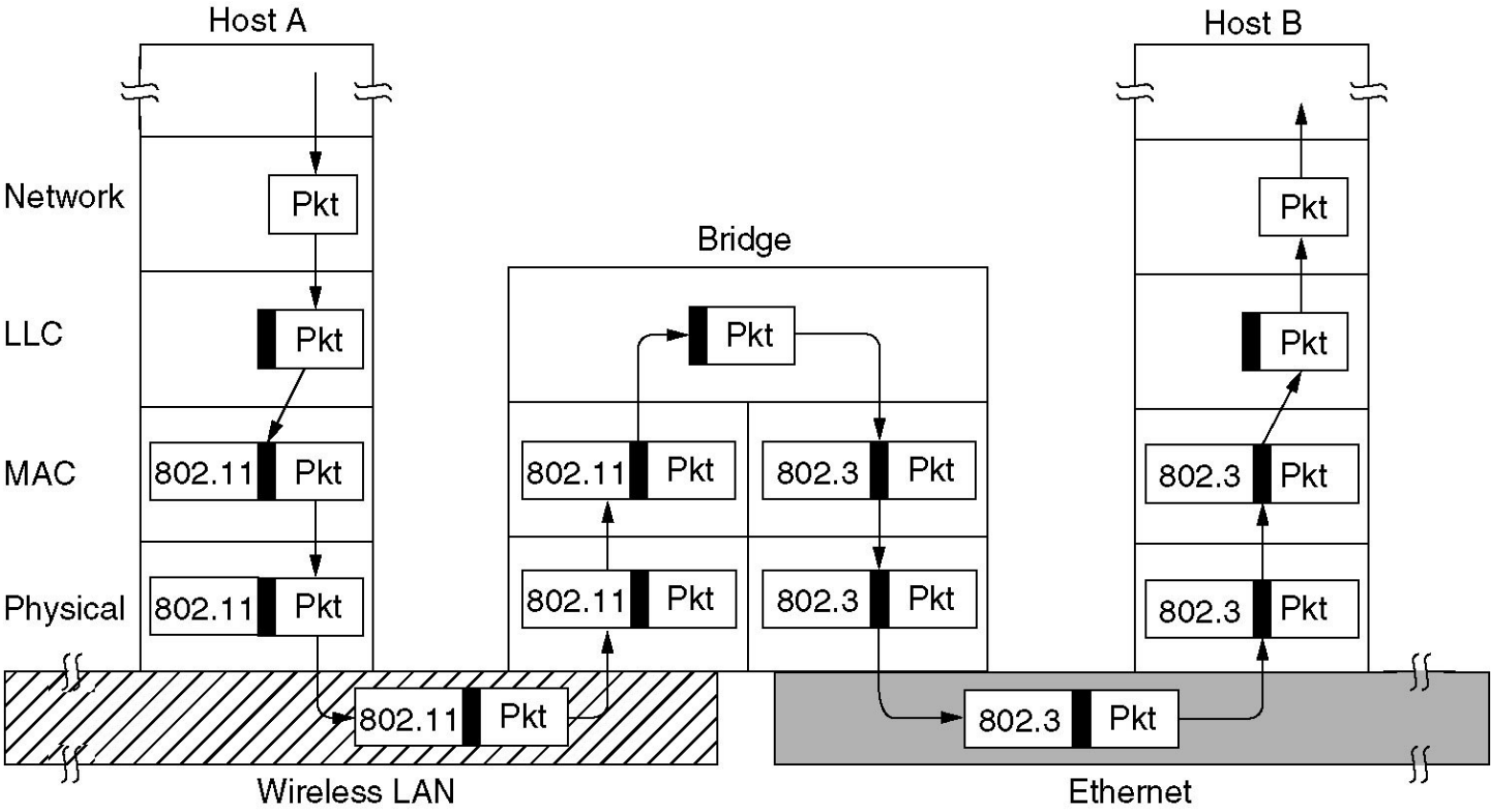
- Bridges from 802.x to 802.y
- Local Internetworking
- Spanning Tree Bridges
- Remote Bridges
- Repeaters, Hubs, Bridges, Switches, Routers, Gateways
- Virtual LANs

Data Link Layer Switching



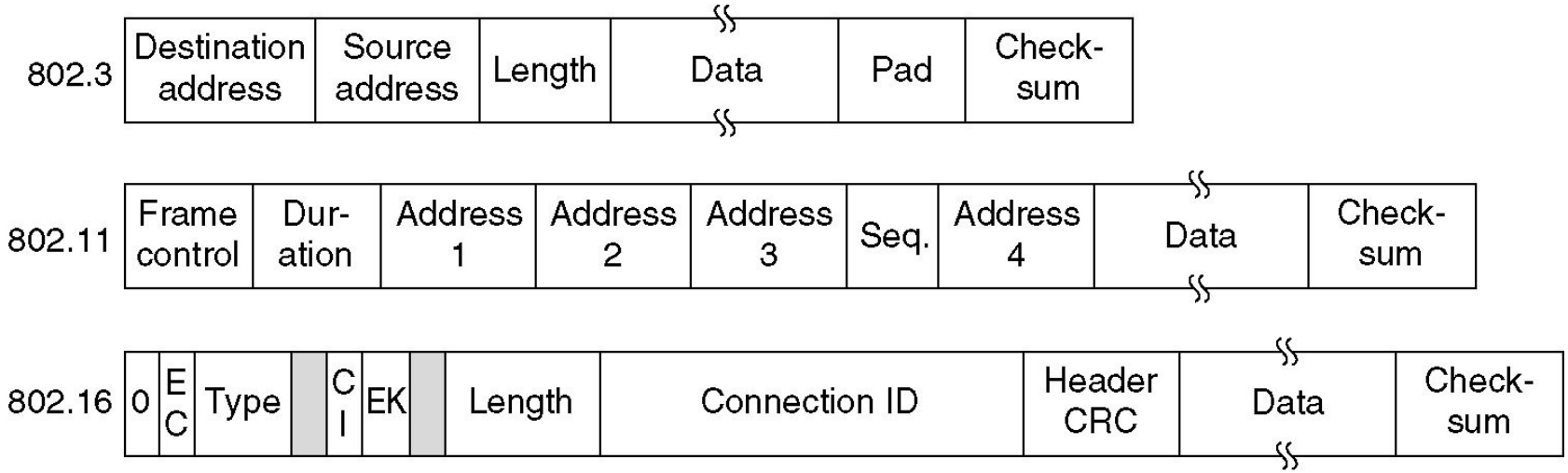
Multiple LANs connected by a backbone to handle a total load higher than the capacity of a single LAN.

Bridges from 802.x to 802.y



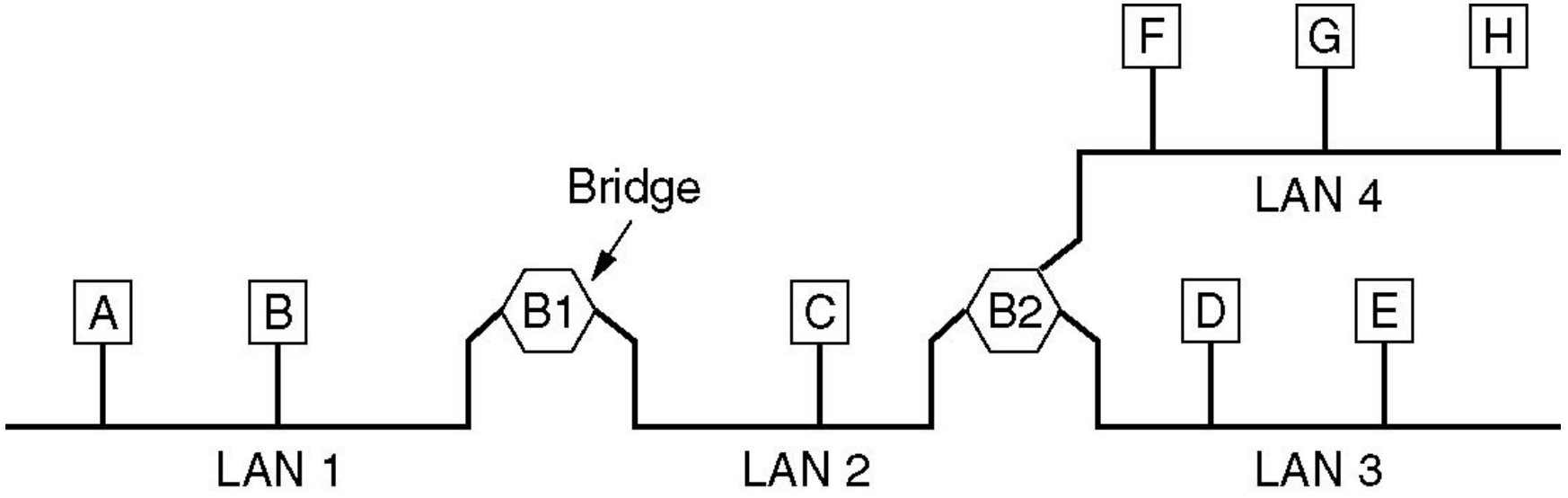
Operation of a LAN bridge from 802.11 to 802.3.

Bridges from 802.x to 802.y (2)



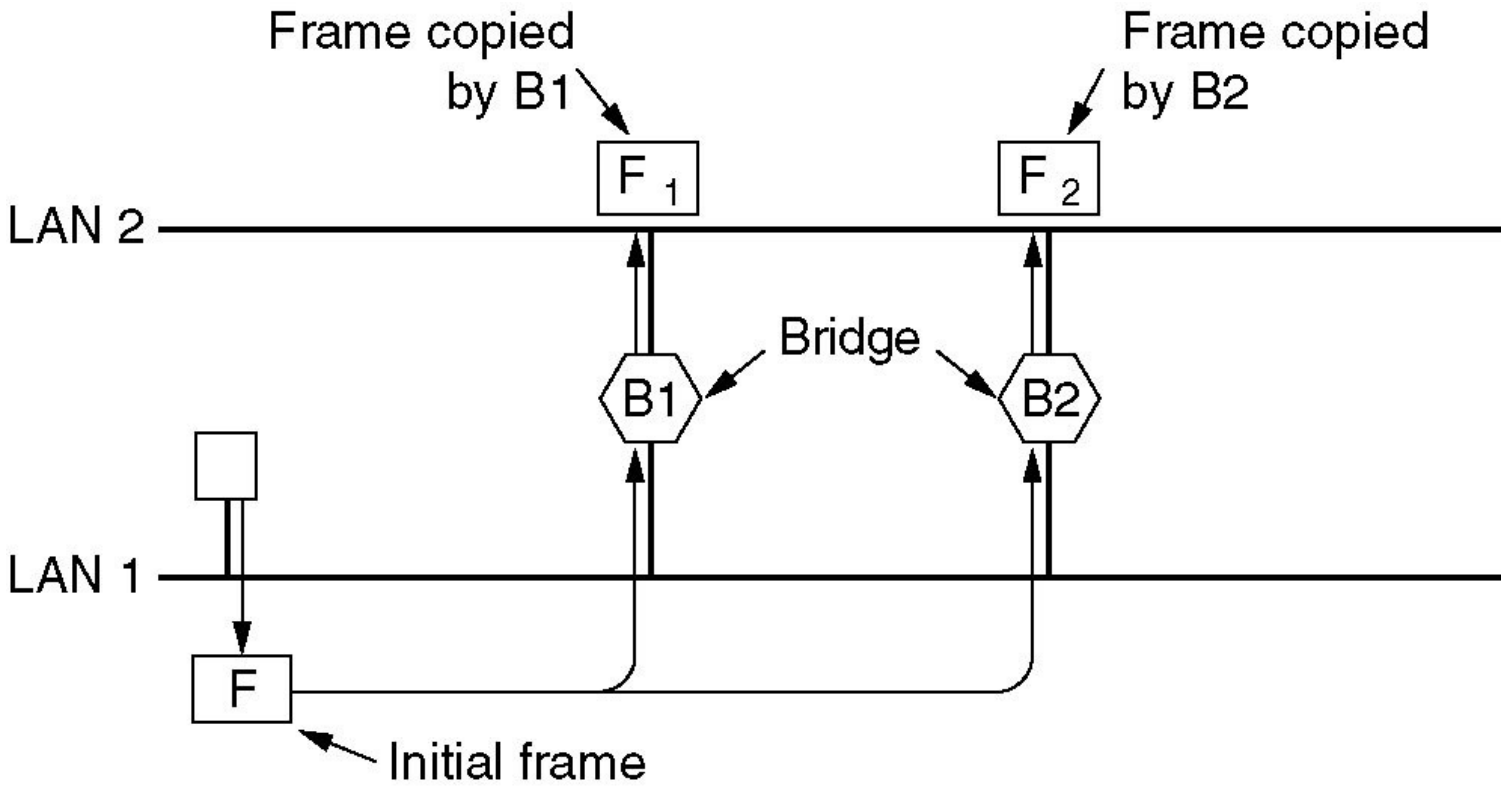
The IEEE 802 frame formats. The drawing is not to scale.

Local Internetworking



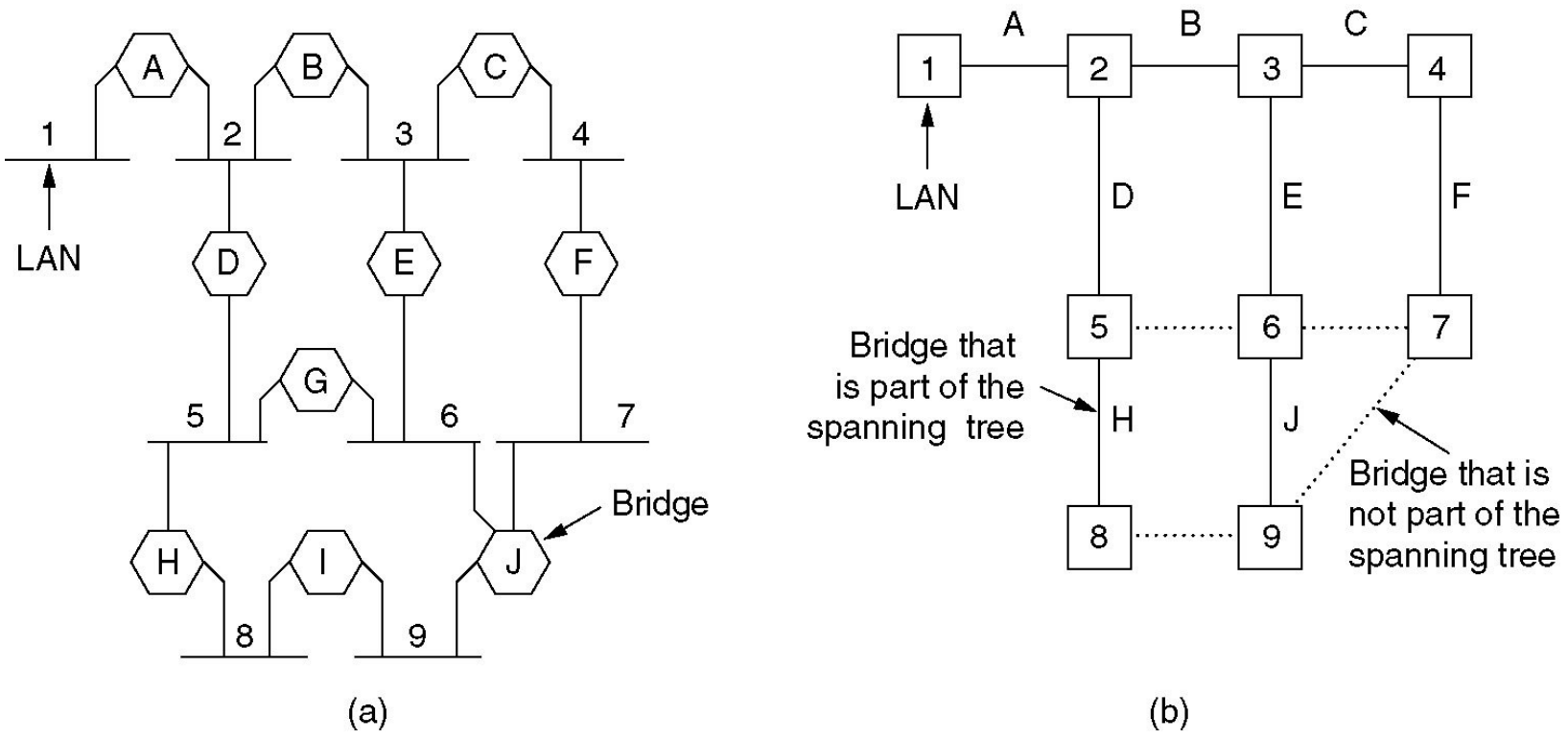
A configuration with four LANs and two bridges.

Spanning Tree Bridges



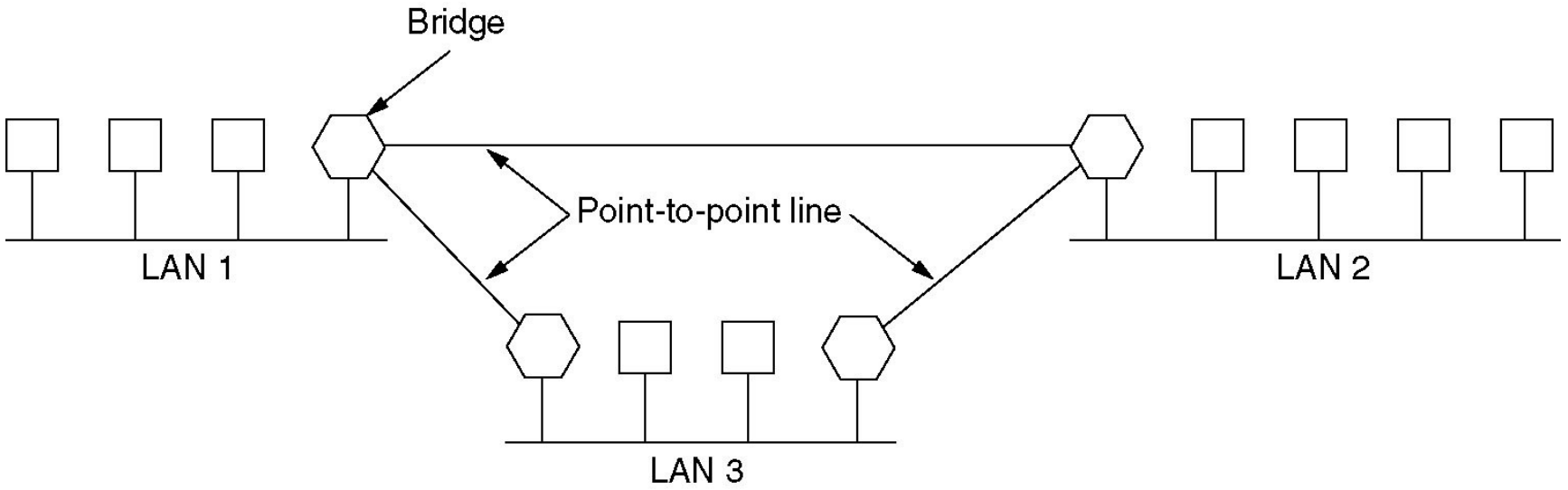
Two parallel transparent bridges.

Spanning Tree Bridges (2)



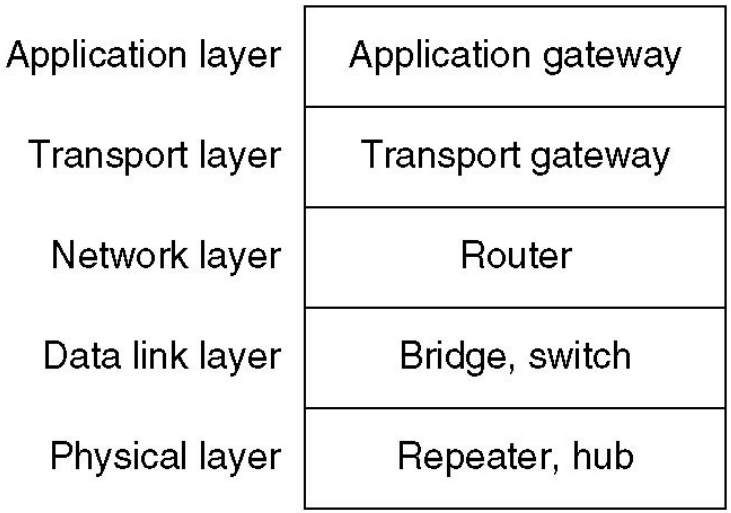
(a) Interconnected LANs. (b) A spanning tree covering the LANs. The dotted lines are not part of the spanning tree.

Remote Bridges

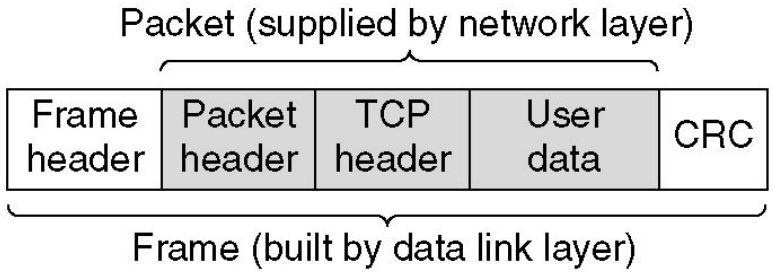


Remote bridges can be used to interconnect distant LANs.

Repeaters, Hubs, Bridges, Switches, Routers and Gateways



(a)

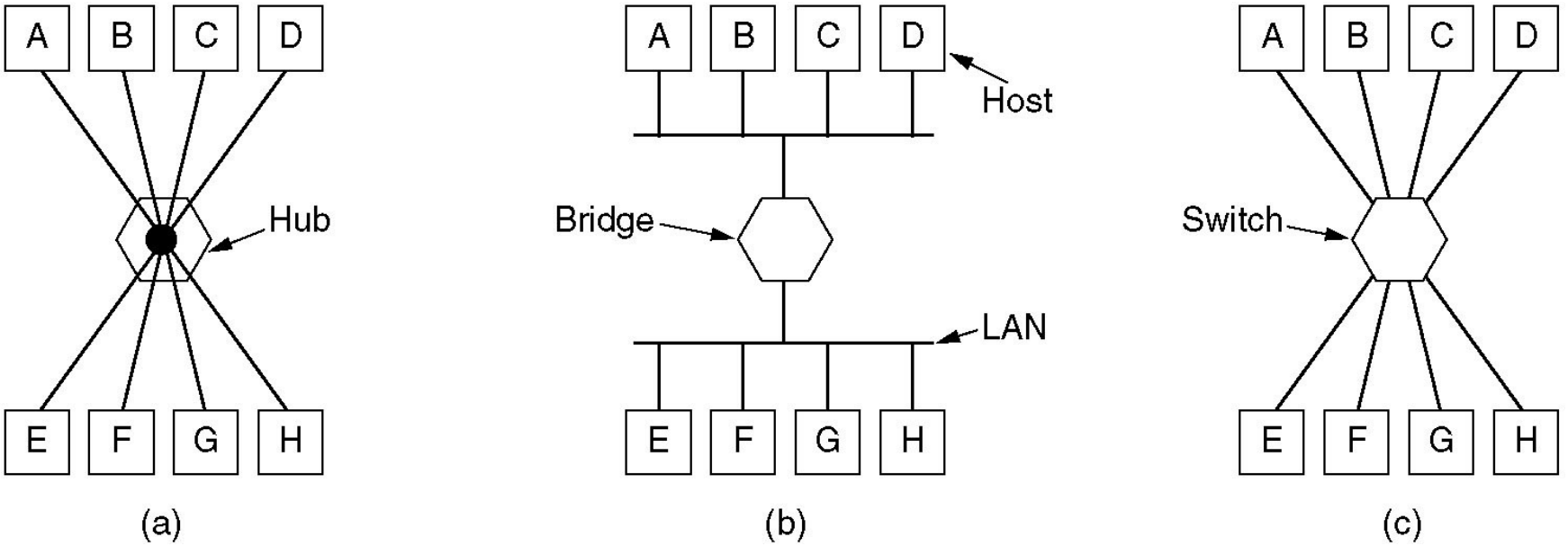


(b)

(a) Which device is in which layer.

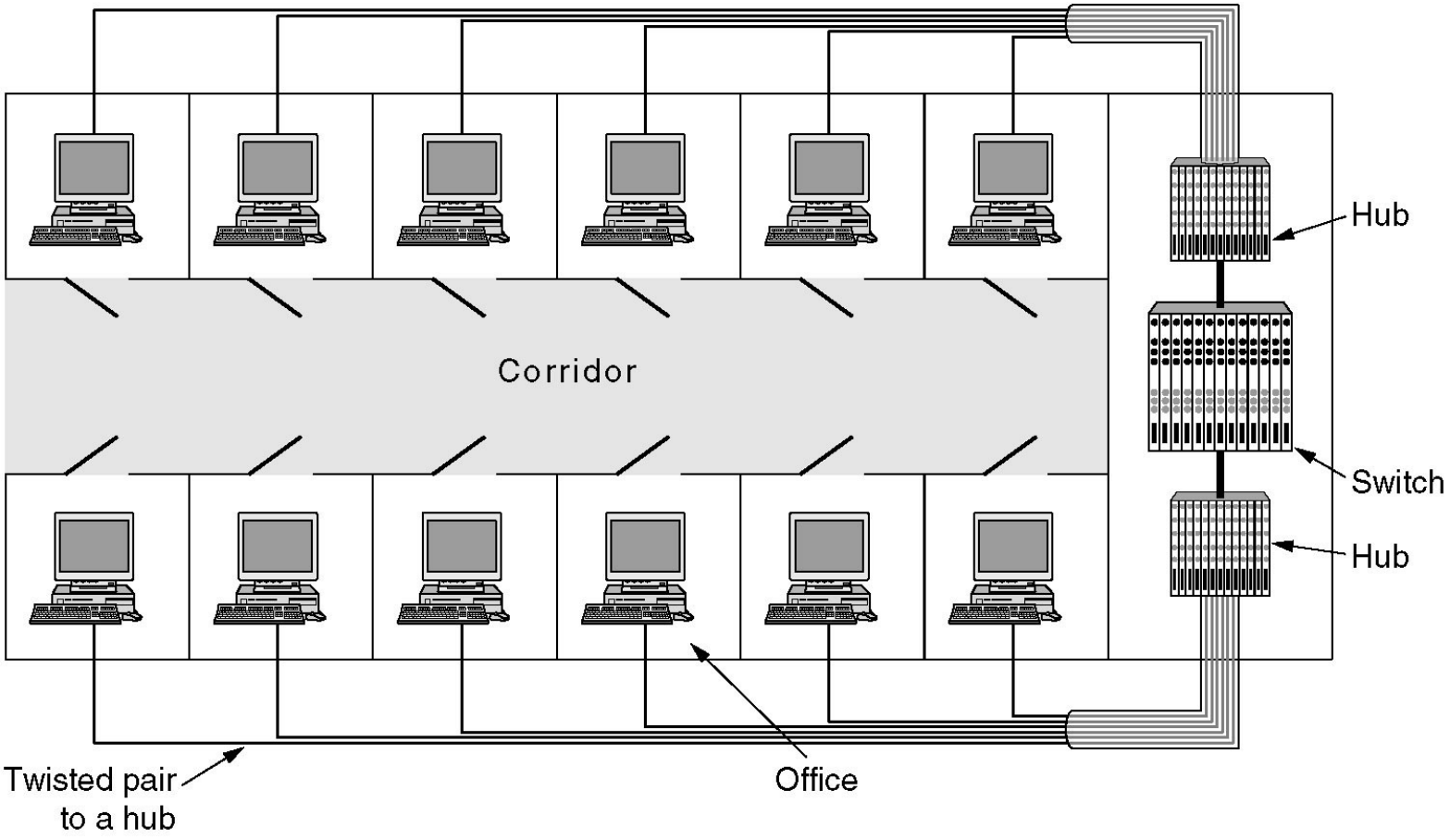
(b) Frames, packets, and headers.

Repeaters, Hubs, Bridges, Switches, Routers and Gateways (2)



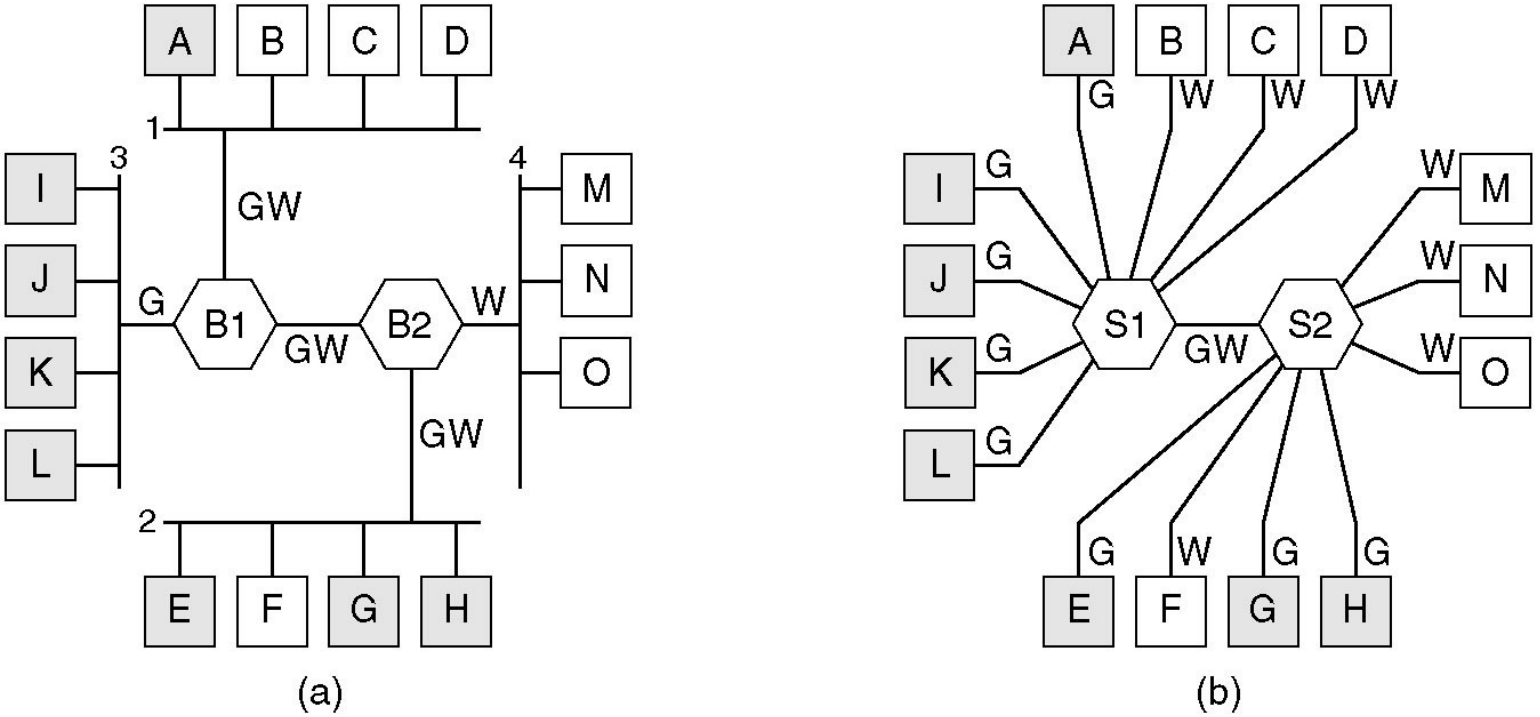
(a) A hub. (b) A bridge. (c) a switch.

Virtual LANs



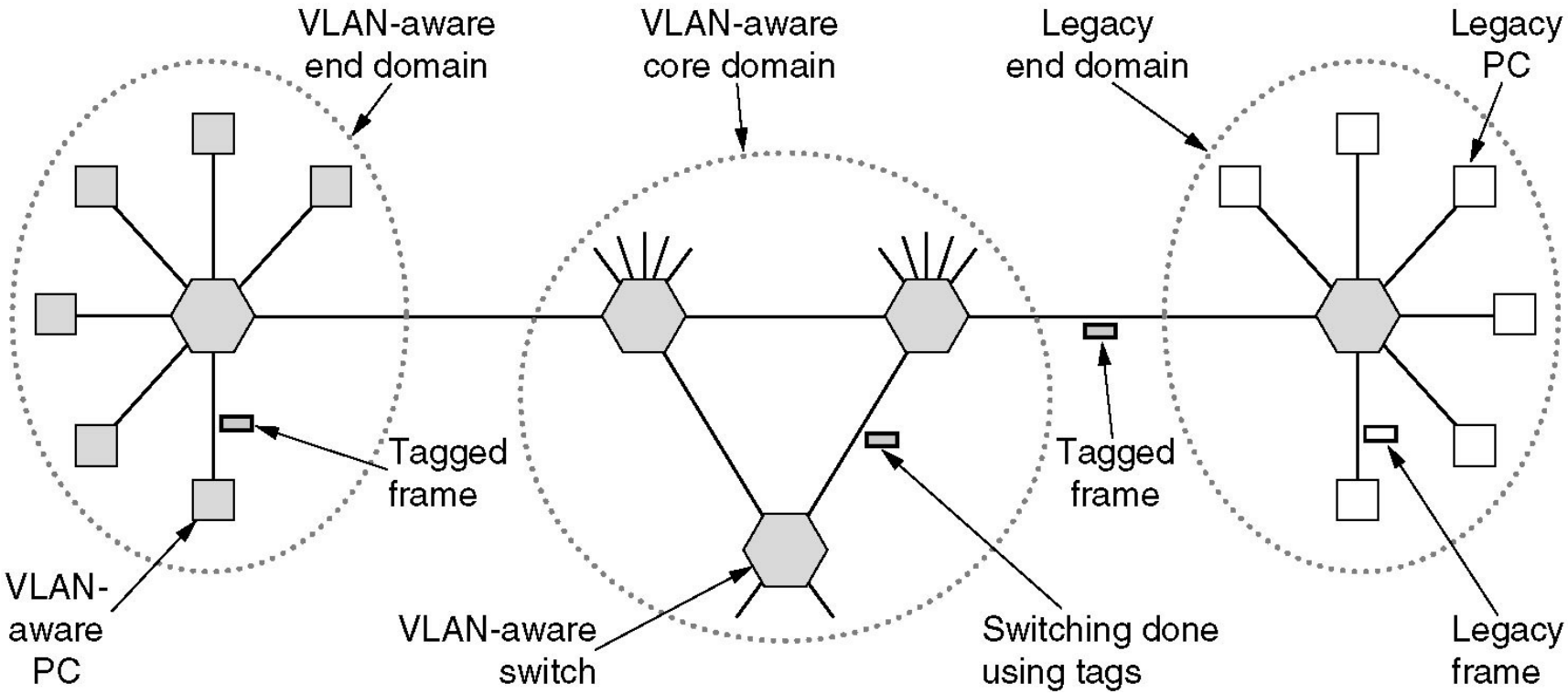
A building with centralized wiring using hubs and a switch.

Virtual LANs (2)



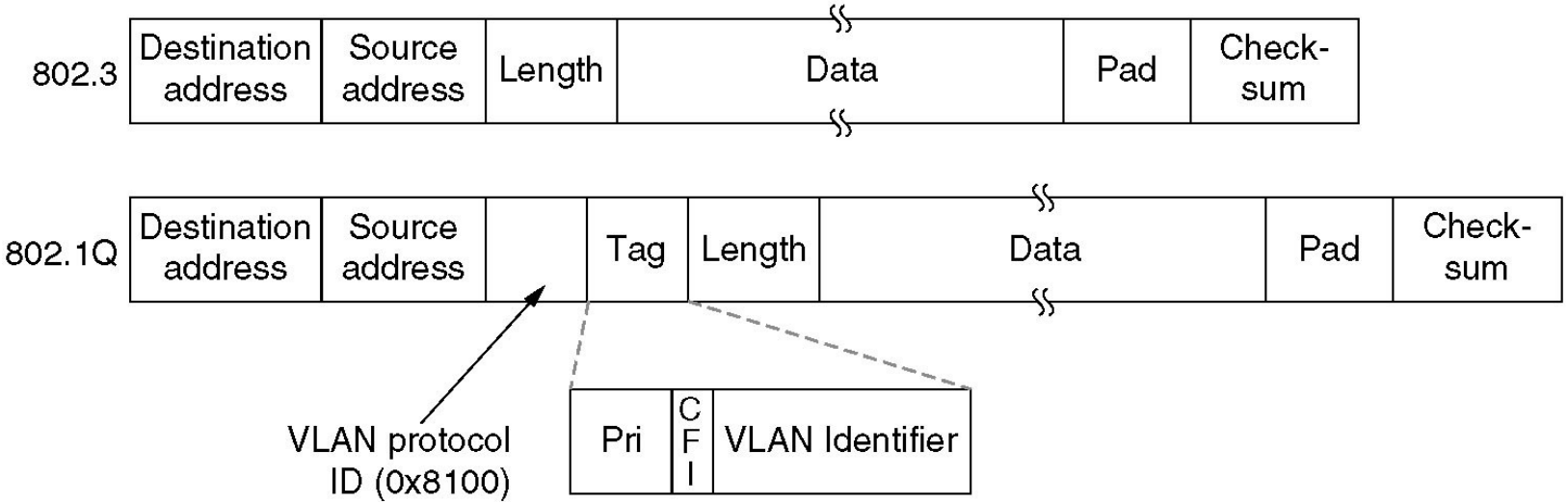
(a) Four physical LANs organized into two VLANs, gray and white, by two bridges. (b) The same 15 machines organized into two VLANs by switches.

The IEEE 802.1Q Standard



Transition from legacy Ethernet to VLAN-aware Ethernet. The shaded symbols are VLAN aware. The empty ones are not.

The IEEE 802.1Q Standard (2)



The 802.3 (legacy) and 802.1Q Ethernet frame formats.

Summary

Method	Description
FDM	Dedicate a frequency band to each station
WDM	A dynamic FDM scheme for fiber
TDM	Dedicate a time slot to each station
Pure ALOHA	Unsynchronized transmission at any instant
Slotted ALOHA	Random transmission in well-defined time slots
1-persistent CSMA	Standard carrier sense multiple access
Nonpersistent CSMA	Random delay when channel is sensed busy
P-persistent CSMA	CSMA, but with a probability of p of persisting
CSMA/CD	CSMA, but abort on detecting a collision
Bit map	Round robin scheduling using a bit map
Binary countdown	Highest numbered ready station goes next
Tree walk	Reduced contention by selective enabling
MACA, MACAW	Wireless LAN protocols
Ethernet	CSMA/CD with binary exponential backoff
FHSS	Frequency hopping spread spectrum
DSSS	Direct sequence spread spectrum
CSMA/CA	Carrier sense multiple access with collision avoidance

Channel allocation methods and systems for a common channel.

Module 4
Network layer: Network Layer Design Issues, Routing Algorithm – Optimality principle - Flooding - Distance vector routing – Link state routing –Multicast Routing – Congestion Control Algorithms – General principles – Congestion prevention policies – Choke packets – Random Early Detection- Quality of Service requirements- Buffering, Traffic shaping – Leaky bucket algorithm.

NETWORK LAYER

The network layer is the third level of the Open Systems Interconnection Model (OSI Model) and the layer that provides data routing paths for network communication. Data is transferred in the form of packets via logical network paths in an ordered format controlled by the network layer.

Logical connection setup, data forwarding, routing and delivery error reporting are the network layer's primary responsibilities.

NETWORK LAYER DESIGN ISSUES

To solve the problem of delivery through several links, the network layer was designed. The network layer is responsible for host-to-host delivery and for routing the packets through the routers. Network Layer Design Issues include the service provided to the transport layer and the internal design of the subnet.

Major issues that the designers of the network layer must deal with are

i. Store-and-Forward Packet Switching
ii. Services Provided to the Transport Layer
iii. Implementation of Connectionless Service
iv. Implementation of Connection-Oriented Service

i. Store-and-Forward Packet Switching

In Store-and Forwarding Packet Switching mechanism, a host transmits a packet to the nearest router, the packet is stored there until it has fully arrived so the checksum can be verified, then it is forwarded to the next router along the path until it reaches the destination host.

The environment of the network layer protocols is shown in figure 4.1. The major components of the system are the carrier's equipment (routers connected by transmission lines), shown inside the shaded oval, and the customers' equipment, shown outside the oval. Host H1 is directly connected to one of the carrier's routers, A, by a leased line. In contrast, H2 is on a LAN with a router, F, owned and operated by the customer. This router also has a leased line to the carrier's equipment. We have shown F as being outside the oval because it does not belong to the carrier, but in terms of construction, software, and protocols, it is probably no different from the carrier's routers

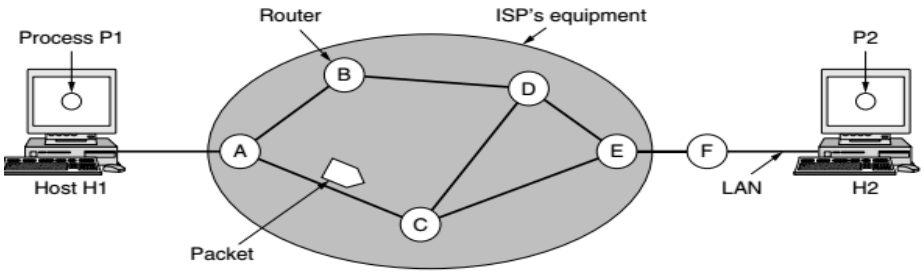


Fig. 4.1 The environment of the network layer protocols

In tore-and-forward packet switching, a host with a packet to send transmits it to the nearest router, either on its own LAN or over a point-to-point link to the carrier. The packet is stored there until it has fully arrived so the checksum can be verified. Then it is forwarded to the next router along the path until it reaches the destination host, where it is delivered.

ii. Services provided to the Transport Layer

The network layer provides services to the transport layer at the network layer/transport layer interface. The network layer services have been designed with the following goals in mind.

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

Two major services provided by the network layer to the transport layer are

- Connectionless service (Eg. Internet)
- Connection-Oriented Service (Eg. ATM Network)

iii. Implementation of Connectionless Service

- In connectionless service, packets are injected into the subnet individually and routed independently of each other.
- No advance setup is needed.
- The packets in connectionless service are called datagrams and the subnet is called a datagram subnet.

Working of datagram subnet

- Suppose that the process P1 in Fig. 4-2 has a long message for P2. It hands the message to the transport layer with instructions to deliver it to process P2 on host H2.
- The transport layer code runs on H1, prepends a transport header to the front of the of the message and hands the result to the network layer.

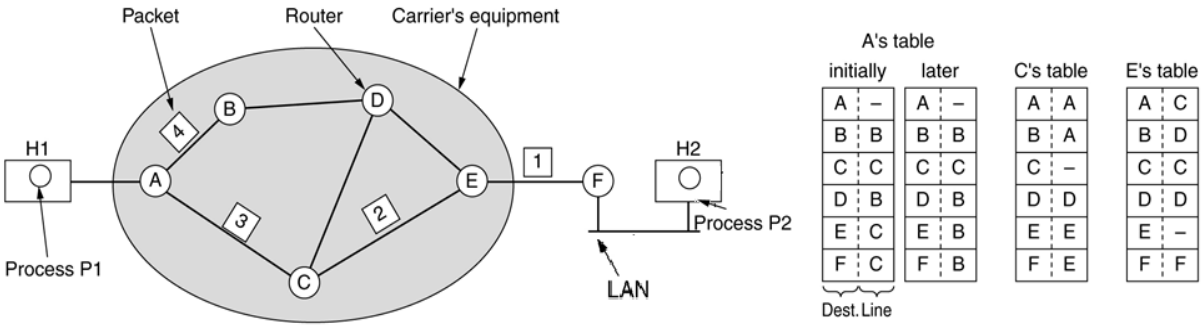


Figure 4-2. Routing within a datagram subnet.

- The transport layer code runs on H1, prepends a transport header to the front of the of the message and hands the result to the network layer.
- Assume that the message is four times longer than the maximum packet size, so the network layer has to break it into four packets, 1, 2, 3, and 4 and sends each of them in turn to router A using some point-to-point protocol.
- Every router has an internal table telling it where to send packets for each possible destination. Each table entry is a pair consisting of a destination and the outgoing line to use for that destination. Only directly-connected lines can be used.
- In Fig. 4-2, A has only two outgoing lines—to B and C—so every incoming packet must be sent to one of these routers
- As they arrived at A, packets 1, 2, and 3 were stored briefly (to verify their checksums). Then each was forwarded to C according to A's table as shown under the label "initially". Packet 1 was then forwarded to E and then to F. When it got to F, it was encapsulated in a data link layer frame and sent to H2 over the LAN. Packets 2 and 3 follow the same route.
- On the arrival of packet 4 to A it was sent to a different route (Router B) than that of the first three because A updated its routing table, as shown under the label "later". The algorithm that manages the tables and makes the routing decisions is called the routing algorithm.

iv. Implementation of Connection-Oriented Service

- In connection-oriented service, a path from the source router to the destination router must be established before any data packets can be sent. This connection is called a VC (virtual circuit) and the subnet is called a virtual-circuit subnet.
- In Virtual circuits, when a connection is established, a route from the source machine to the destination machine is chosen as part of the connection setup and stored in tables inside the routers.
- That route is used for all traffic flowing over the connection.
- When the connection is released, the virtual circuit is also terminated.
- With connection-oriented service, each packet carries a **flow label** (a virtual circuit identifier) that defines the virtual path the packet should follow
- As an example, consider the situation of Fig. 4-3. Here, host H1 has established connection 1 with host H2. It is remembered as the first entry in each of the routing tables. The first line of A's table says that if a packet bearing connection identifier 1 comes in from H1, it is

to be sent to router C and given connection identifier 1. Similarly, the first entry at C routes the packet to E, also with connection identifier 1.

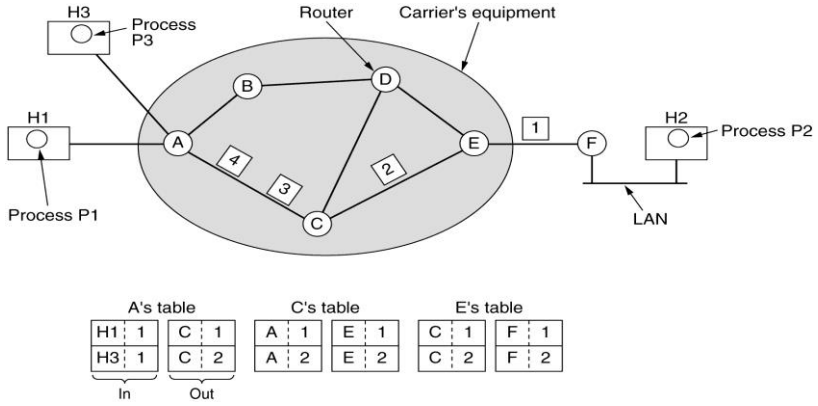


Figure 5-3. Routing within a virtual-circuit subnet.

- When H3 wants to establish a connection to H2. It chooses connection identifier 1 (because it is initiating the connection and this is its only connection) and tells the subnet to establish the virtual circuit.
- This leads to the second row in the tables. Note that we have a conflict here because although A can easily distinguish connection 1 packets from H1 from connection 1 packets from H3, C cannot do this.
- For this reason, A assigns a different connection identifier to the outgoing traffic for the second connection. Avoiding conflicts of this kind is why routers need the ability to replace connection identifiers in outgoing packets, called label switching.

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

Inside the subnet, several trade-offs exist between virtual circuits and datagrams.

- Between router memory space and bandwidth.

- Virtual circuits allow packets to contain circuit numbers instead of full destination addresses. A full destination address in every packet may represent a significant amount of overhead and wasted bandwidth. The price paid for using virtual circuits internally is the table space within the routers. Depending upon the relative cost of communication circuits versus router memory, one or the other may be cheaper.

- Setup time versus address parsing time.
  - Using virtual circuits requires a setup phase, which takes time and consumes resources. However, figuring out what to do with a data packet in a virtual-circuit subnet is easy: the router just uses the circuit number to index into a table to find out where the packet goes. In a datagram subnet, a more complicated lookup procedure is required to locate the entry for the destination.
- Amount of table space required in router memory
  - A datagram subnet needs to have an entry for every possible destination
  - A virtual-circuit subnet just needs an entry for each virtual circuit.
- Quality of Service & Congestion
  - Virtual circuits have advantages in guaranteeing quality of service and avoiding congestion within the subnet because resources (e.g., buffers, bandwidth, and CPU cycles) can be reserved in advance, when the connection is established.
  - With a datagram subnet, congestion avoidance is more difficult.
- Router crashes
  - If a router crashes and loses its memory in virtual circuits, even if it comes back up a second later, all the virtual circuits passing through it will have to be aborted.
  - If a datagram router goes down, only those users whose packets were queued in the router at the time will suffer, and maybe not even all those, depending upon whether they have already been acknowledged.

ROUTING ALGORITHMS

- The main function of the network layer is routing packets from the source machine to the destination machine. The routing algorithm is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on.
- For subnet uses datagrams internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time.
- If the subnet uses virtual circuits internally, routing decisions are made only when a new virtual circuit is being set up. Thereafter, data packets just follow the previously-established route. (session routing)

Routing and forwarding

- Routing makes decision of which routes to use. It is responsible for filling in and updating the routing tables.
- Forwarding handles each packet as it arrives, looking up the outgoing line to use for it in the routing tables.

Desirable properties in a routing algorithm

- **Correctness:** The routing should be done properly and correctly so that the packets may reach their proper destination.
- **Simplicity:** The routing should be done in a simple manner so that the overhead is as low as possible. With increasing complexity of the routing algorithms the overhead also increases.
- **Robustness:** Once a major network becomes operative, it may be expected to run continuously for years without any failures. The algorithms designed for routing should be robust enough to handle hardware and software failures and should be able to cope with changes in the topology and traffic without requiring all jobs in all hosts to be aborted and the network rebooted every time some router goes down.
- **Stability:** The routing algorithms should be stable under all possible circumstances.
- **Fairness:** Every node connected to the network should get a fair chance of transmitting their packets. This is generally done on a first come first serve basis.
- **Optimality:** The routing algorithms should be optimal in terms of throughput and minimizing mean packet delays. Here there is a trade-off and one has to choose depending on his suitability.

Conflict between fairness and optimality

In Fig. 5-4, suppose that there is enough traffic between A and A', between B and B', and between C and C' to saturate the horizontal links. To maximize the total flow, the X to X' traffic should be shut off altogether. Unfortunately, X and X' may not see it that way

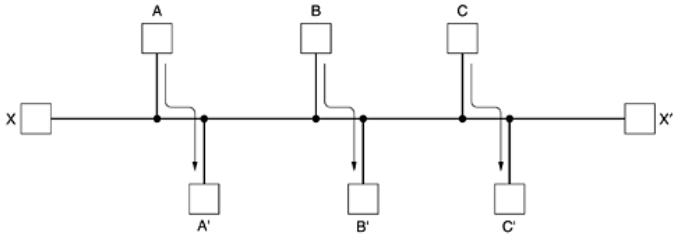
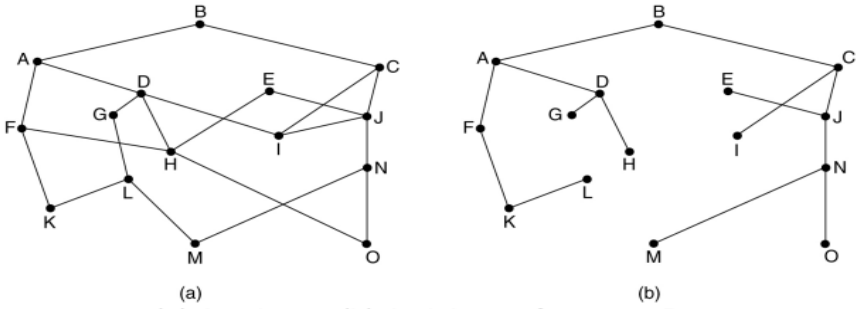


Figure 5-4. Conflict between fairness and optimality

The Optimality Principle

- A general statement can make about optimal routes without regard to network topology or traffic: It states that 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.

- **Sink tree:** Set of optimal routes from all sources to a given destination form a tree rooted at the destination; not necessarily unique; no loops. The goal of all routing algorithms is to discover and use the sink trees for all routers
- The optimality principle and the sink tree provide a benchmark against which other routing algorithms can be measured

Classification of Routing Algorithms

Routing algorithms can be grouped into two major classes - Non adaptive algorithms and Adaptive algorithms

Non adaptive algorithms (Static routing)

- Nonadaptive algorithms do not base their routing decisions on measurements or estimates of the current traffic and topology. Instead, the choice of the route to use is computed in advance, off-line, and downloaded to the routers when the network is booted.
- Eg: Shortest Path Routing and Flooding

Adaptive algorithms

- Adaptive algorithms, change their routing decisions to reflect changes in the topology, and usually the traffic as well.
- Adaptive algorithms differ in where they get their information (e.g., locally, from adjacent routers, or from all routers), when they change the routes (e.g., when the load changes or when the topology changes), and what metric is used for optimization (e.g., distance, number of hops, or estimated transit time).
- Eg. Distance Vector Routing and Link State Routing.

SHORTEST PATH ROUTING

Shortest path algorithm finds the shortest paths between routers(nodes) in a graph. The widely used shortest path algorithm is Dijkstra's shortest path algorithm.

- The idea is to build a graph of the subnet, with each node of the graph representing a router and each arc of the graph representing a communication line (often called a link).
- To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.

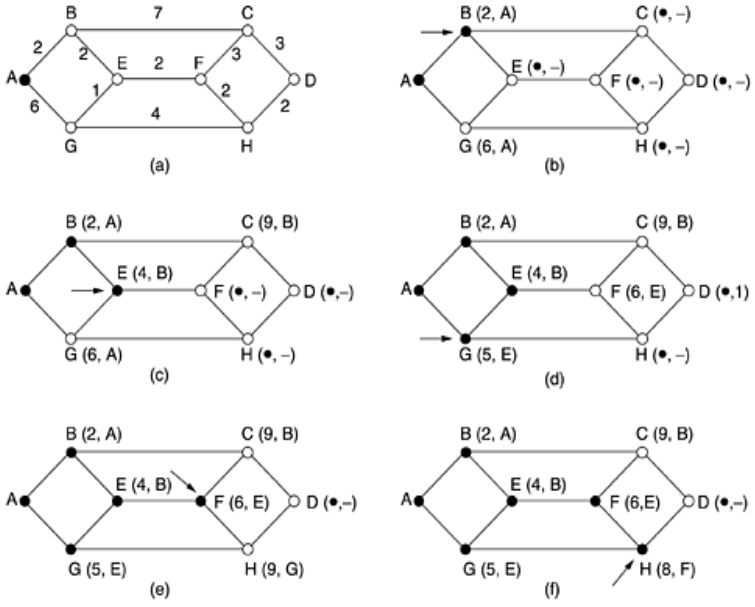


Figure 5-7. The first five steps used in computing the shortest path from A to D. The arrows indicate the working node

- The labels on the arcs could be computed as a function of the distance, bandwidth, average traffic, communication cost, mean queue length, measured delay, and other factors.
- By changing the weighting function, the algorithm would then compute the "shortest" path measured according to any one of a number of criteria or to a combination of criteria.
- An algorithm for computing the shortest path between two nodes of a graph is Dijkstra.

Working of Algorithm

- Fig. 5-7(a) contains a weighted, undirected graph, where the weights represent is distance.
- To find the shortest path from A to D, start out by marking node A as permanent, indicated by a filled-in circle.
- Examine, each of the nodes adjacent to A (the working node), relabeling each one with the distance to A.
- Whenever a node is relabeled, we also label it with the node from which the probe was made so that we can reconstruct the final path later. Having examined each of the nodes adjacent to A, we examine all the tentatively labeled nodes in the whole graph and make the one with the smallest label permanent, as shown in Fig. 5-7(b). This one becomes the new working node.
- We now start at B and examine all nodes adjacent to it. If the sum of the label on B and the distance from B to the node being considered is less than the label on that node, we have a shorter path, so the node is relabeled.
- After all the nodes adjacent to the working node have been inspected and the tentative labels changed if possible, the entire graph is searched for the tentatively-labeled node with the smallest value. This node is made permanent and becomes the working node for the next round. Figure 5-7 shows the first five steps of the algorithm.

FLOODING

- In flooding, every incoming packet is sent out on every outgoing line except the one it arrived on.
- Flooding generates vast numbers of duplicate packets.

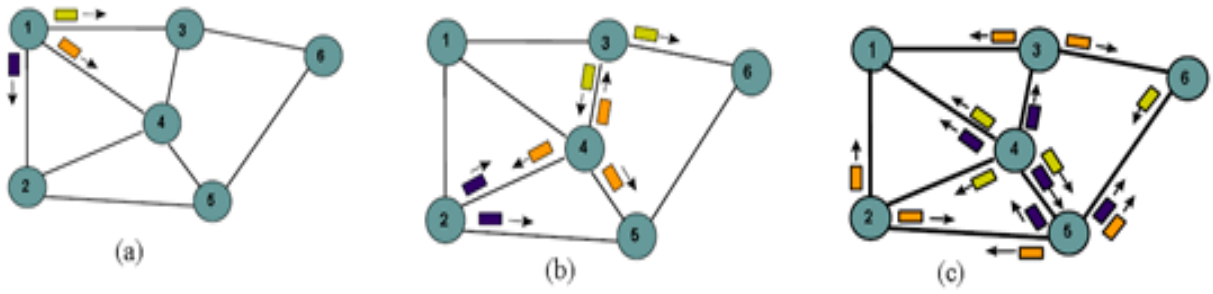


Figure: Flooding is initiated from Node 1: (a) Hop 1 transmissions (b) Hop 2 transmissions (c) Hop 3 transmissions

- To avoid a packet keep in a network forever, add a hop counter in the header of each packet, which is decremented at each hop, with the packet being discarded when the counter reaches zero.

- Ideally, the hop counter should be initialized to the length of the path from source to destination.
- An alternative technique for controlling the flood is to keep track of which packets have been flooded, to avoid sending them out a second time. Source router put a sequence number in each packet it receives from its hosts. Each router then needs a list per source router telling which sequence numbers originating at that source have already been seen. If an incoming packet is on the list, it is not flooded.
- To prevent the list from growing without bound, each list should be augmented by a counter, k, meaning that all sequence numbers through k have been seen. When a packet comes in, it is easy to check if the packet is a duplicate; if so, it is discarded. Furthermore, the full list below k is not needed, since k effectively summarizes it.
- A variation of flooding that is slightly more practical is **selective flooding**. In this algorithm the routers do not send every incoming packet out on every line, only on those lines that are going approximately in the right direction.
- Flooding is not practical in most applications, but it does have some uses.
    - Military applications, where large numbers of routers may be blown to bits at any instant, the tremendous robustness of flooding is highly desirable.
    - In distributed database applications, it is sometimes necessary to update all the databases concurrently, in which case flooding can be useful.
    - In wireless networks, all messages transmitted by a station can be received by all other stations within its radio range, which is, in fact, flooding, and some algorithms utilize this property.
    - A metric against which other routing algorithms can be compared.

DISTANCE VECTOR ROUTING

- It is also called **Bellman-Ford routing algorithm**.
- Distance vector routing operates by having each router maintain a table (i.e. a vector) containing one entry for, each router in the subnet. This entry contains two parts: the preferred outgoing line to use for that destination and an estimate of the time or distance to that destination.
- The distance vector routing algorithm passes periodic copies of a routing table from router to router. These regular updates between routers communicate topology changes.
- These tables are updated by exchanging information with the neighbors.
- The router is assumed to know the "distance" to each of its neighbors. If the metric is hops, the distance is just one hop. Metric may be delay time.
- Once every T msec each router sends to each neighbor a list of its estimated delays to each destination. It also receives a similar list from each neighbor.
- Imagine that one of these tables has just come in from neighbor X, with Xi being X's estimate of how long it takes to get to router i. If the router knows that the delay to X is m msec, it also knows that it can reach router i via X in Xi + m msec. By performing this calculation for each neighbor, a router can find out which estimate seems the best and use that estimate and the corresponding line in its new routing table. Note that the old routing table is not used in the calculation.

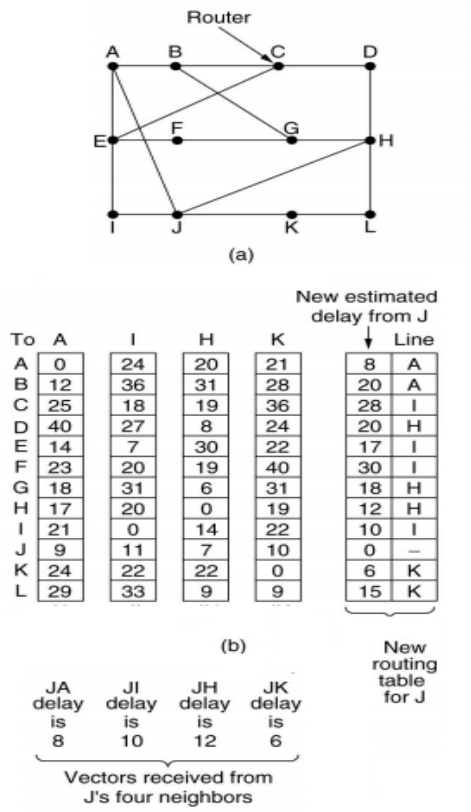


Fig. 5-9: Updating process is illustrated. Part (a) shows a subnet. The first four columns of part (b) show the delay vectors received from the neighbors of router J (Input from A, I, H, K) and the new routing table for J.

- In Fig. 5-9(b)- A claims to have a 12-msec delay to B, a 25-msec delay to C, a 40-msec delay to D, etc.
- Suppose that J has measured or estimated its delay to its neighbors, A, I, H, and K as 8, 10, 12, and 6 msec, respectively.

How J computes its new route to router G.

- It knows that it can get to A in 8 msec, and A claims to be able to get to G in 18 msec, so J knows it can count on a delay of 26 msec to G if it forwards packets bound for G to A.
- Similarly, it computes the delay to G via I, H, and K as 41 (31 + 10), 18 (6 + 12), and 37 (31 + 6) msec, respectively.
- The best of these values is 18, so it makes an entry in its routing table that the delay to G is 18 msec and that the route to use is via H.
- The same calculation is performed for all the other destinations, with the new routing table shown in the last column of the figure.

Advantage

- Their chief advantage is the simplicity of this algorithm.

Disadvantage

- The primary drawback of this algorithm is its vulnerability to the 'Count-to-Infinity' problem. Many partial solutions have been proposed but none works under all circumstances.
- It does not take into account link bandwidth.
- It takes longer time for convergence as network size grows.

Count to infinity problem

- Counting to infinity is just another name for a routing loop. It is an important issue in Distance Vector Routing.
- In distance vector routing, routing loops usually occur when an interface goes down.
- It can also occur when two routers send updates to each other at the same time.
- Distance vector routing reacts rapidly to good news, but leisurely to bad news.

- Consider a router whose best route to destination X is large. If on the next exchange neighbor A suddenly reports a short delay to X, the router just switches over to using the line to A to send traffic to X. In one vector exchange, the good news is processed.
- Consider the five-node (linear) subnet of Fig. 5-10, where the delay metric is the number of hops. Suppose A is down initially and all the other routers know this and they have all recorded the delay to A as infinity.
- Fig. 5-10 (a) shows how fast good news propagates. When A comes up, the other routers learn about it via the vector exchanges. At the time of the first exchange, B learns that its left neighbor has zero delay to A. B now makes an entry in its routing table that A is one hop away to the left. All the other routers still think that A is down. At this point, the routing table entries for A are as shown in the second row of Fig. 5-10(a). On the next exchange, C learns that B has a path of length 1 to A, so it updates its routing table to indicate a path of length 2, but D and E do not hear the good news until later. Clearly, the good news is spreading at the rate of one hop per exchange. In a subnet whose longest path is of length N hops, within N exchanges everyone will know about newly-revived lines and routers.

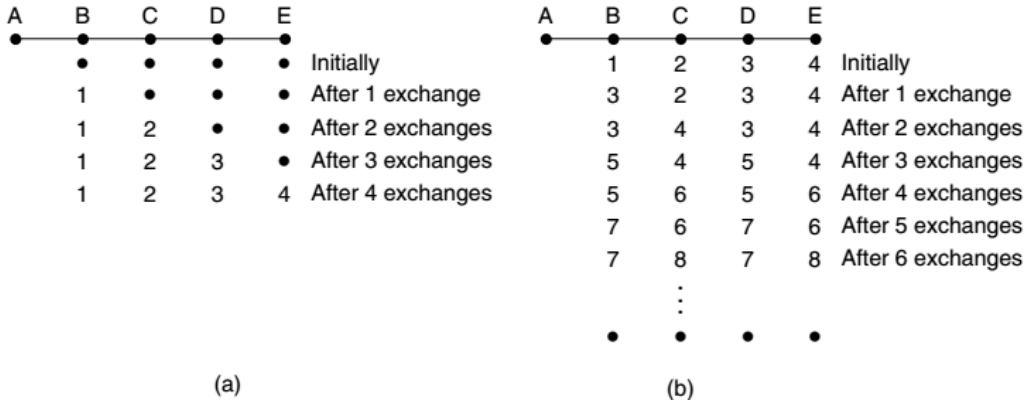


Fig. 5-10 (a) shows how fast good news propagates (b) Count to Infinity Problem

- In Fig. 5-10(b), all the lines and routers are initially up. Routers B, C, D, and E have distances to A of 1, 2, 3, and 4, respectively. Suddenly A goes down, or alternatively, the line between A and B is cut.
- At the first packet exchange, B does not hear anything from A. C informs A that it have a path to A of length 2. For all B knows, C might have another line with separate paths to A of length 2. As a result, B thinks it can reach A via C, with a path length of 3. D and E do not update their entries for A on the first exchange.
- On the second exchange, C notices that each of its neighbors claims to have a path to A of length 3. It picks one of them at random and makes its new distance to A 4, as shown in the third row of Fig. 5-10(b).
- Subsequent exchanges produce the history shown in the rest of Fig. 5-10(b).
- From this figure, it should be clear why bad news travels slowly: no router ever has a value more than one higher than the minimum of all its neighbors. Gradually, all routers work their way up to infinity, but the number of exchanges required depends on the numerical value used for infinity.

- For this reason, it is wise to set infinity to the longest path plus 1. If the metric is time delay, there is no well-defined upper bound, so a high value is needed to prevent a path with a long delay from being treated as down.
- There have been a few attempts to solve it (such as split horizon with poisoned reverse), but none of these work well in general.
- The core of the problem is that when X tells Y that it has a path somewhere, Y has no way of knowing whether it itself is on the path.

LINK STATE ROUTING

Link-state routers exchange messages to allow each router to learn the entire network topology. Based on this learned topology, each router is then able to compute its routing table by using a shortest path computation. For link-state routing, a network is modelled as a directed weighted graph. Each router is a node, and the links between routers are the edges in the graph. A positive weight is associated to each directed edge and routers use the shortest path to reach each destination.

The idea behind link state routing is simple and can be stated as five parts. Each router must do the following:

i. Discover its neighbors and learn their network addresses.
ii. Measure the delay or cost to each of its neighbors.
iii. Construct a packet telling all it has just learned.
iv. Send this packet to all other routers.
v. Compute the shortest path to every other router.

In effect, the complete topology and all delays are experimentally measured and distributed to every router. Then Dijkstra's algorithm can be run to find the shortest path to every other router.

i. Learning about the Neighbors

- It accomplishes by sending a special HELLO packet on each point-to-point line.
- The router on the other end is send back a reply telling who it is.
- These names must be globally unique because when a distant router later hears that three routers are all connected to F, it is essential that it can determine whether all three mean the same F.
- When two or more routers are connected by a LAN, consider it as a node itself.

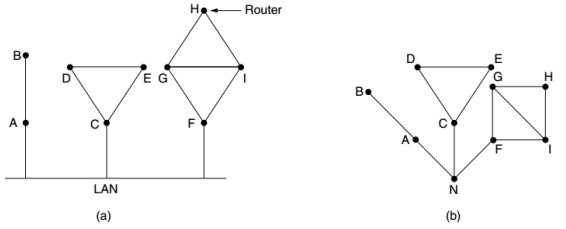


Figure 5-11. (a) Nine routers and a LAN. (b) A graph model of (a).

In Fig. 5-11(b), introduced a new, artificial node, N, to which A, C, and F are connected. The fact that it is possible to go from A to C on the LAN is represented by the path ANC here

ii. Measuring Line Cost

- The link state routing algorithm requires each router to know (or estimate of), the delay to each of its neighbors
- The most direct way to determine this delay is to send over the line a special ECHO packet that the other side is required to send back immediately.
- By measuring the round-trip time and dividing it by two, the sending router can get a reasonable estimate of the delay.

iii. Building Link State Packets

- Each router builds a packet containing the required data it collected from neighbors.
- The packet starts with the identity of the sender, followed by a sequence number and age and a list of neighbors. For each neighbor, the delay to that neighbor is given.
- An example subnet is given in Fig. 5-12(a) with delays shown as labels on the lines. The corresponding link state packets for all six routers are shown in Fig. 5-12(b).

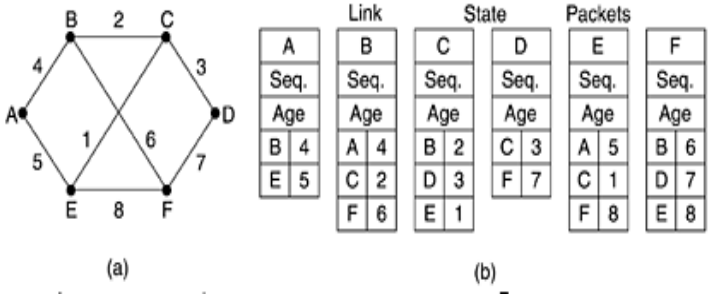


Figure 5-12. (a) A subnet. (b) The link state packets for this subnet

iv. Distributing the Link State Packets

- Use flooding to distribute the link state packets.
- To keep the flood in check, each packet contains a sequence number that is incremented for each new packet sent.
- Routers keep track of all the (source router, sequence) pairs they see.
- When a new link state packet comes in, it is checked against the list of packets already seen. If it is new, it is forwarded on all lines except the one it arrived on. If it is a duplicate, it is discarded.
- If a packet with a sequence number lower than the highest one seen so far ever arrives, it is rejected as being obsolete since the router has more recent data.

Problems

- If the sequence numbers wrap around. The solution here is to use a 32-bit sequence number. With one link state packet per second, it would take 137 years to wrap around.
- If a router ever crashes, it will lose track of its sequence number. If it starts again at 0, the next packet will be rejected as a duplicate.
- If a sequence number is ever corrupted and 65,540 is received instead of 4 (a 1-bit error), packets 5 through 65,540 will be rejected as obsolete, since the current sequence number is thought to be 65,540.

- The solution to all these problems is to include the age of each packet after the sequence number and decrement it once per second. When the age hits zero, the information from that router is discarded
- The data structure used by router B for the subnet shown in Fig. 5-12(a) is depicted in Fig. 5-13.

			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	

Figure 5-13. The packet buffer for router B in Fig. 5-12

- Each row here corresponds to a recently-arrived, but as yet not fully-processed, link state packet.
- The table records where the packet originated, its sequence number and age, and the data.
- In addition, there are send and acknowledgement flags for each of B's three lines (to A, C, and F, respectively).
- The send flags mean that the packet must be sent on the indicated line.
- The acknowledgement flags mean that it must be acknowledged there

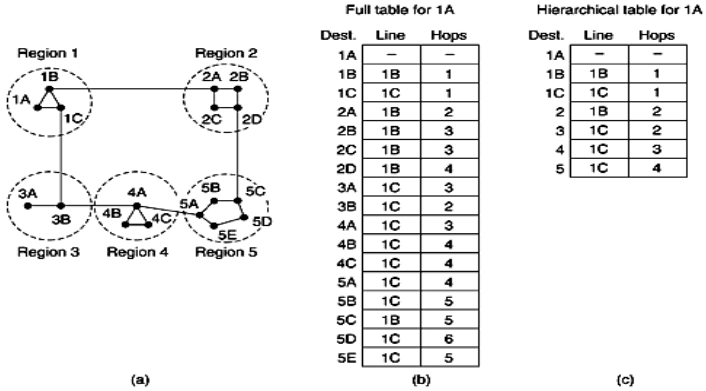
v. Computing the New Routes

- Once a router has accumulated a full set of link state packets, it can construct the entire subnet graph because every link is represented.
- Every link is, in fact, represented twice, once for each direction. The two values can be averaged or used separately.
- Dijkstra's algorithm is used locally to construct the shortest path to all possible destinations.
- The results of this algorithm can be installed in the routing tables, and normal operation resumed.
- Link state routing is widely used in actual networks. The OSPF protocol, which is widely used in the Internet, uses a link state algorithm.

HIERARCHICAL ROUTING

- As networks grow in size, the router routing tables grow proportionally.
- Router memory consumed by ever-increasing tables, more CPU time is needed to scan them, more bandwidth is needed to send status reports
- So the routing will have to be done hierarchically, as it is in the telephone network.
- When hierarchical routing is used, the routers are divided into regions, with each router knowing all the details about how to route packets to destinations within its own region, but knowing nothing about the internal structure of other regions.
- When different networks are interconnected, consider each one as a separate region in order to free the routers in one network from having to know the topological structure of the other ones.

- Figure 5-14 gives an example of routing in a two-level hierarchy with five regions. The full routing table for router 1A has 17 entries, as shown in Fig. 5-14(b). When routing is done hierarchically, as in Fig. 5-14(c), there are entries for all the local routers as before, but all other regions have been condensed into a single router, so all traffic for region 2 goes via the 1B -2A line, but the rest of the remote traffic goes via the 1C -3B line.
- Hierarchical routing has reduced the table from 17 to 7 entries. As the ratio of the number of regions to the number of routers per region grows, the savings in table space increase.



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

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

Figure 5-14. Hierarchical routing

- **Disadvantage:** Routes are not optimal. For example, the best route from 1A to 5C is via region 2, but with hierarchical routing all traffic to region 5 goes via region 3, because that is better for most destinations in region 5.

BROADCAST ROUTING

Broadcasting refers to transmitting a packet that will be received by every device on the network. For example, a service distributing weather reports, stock market updates, or live radio programs.

Methods for Broadcasting

a. The source to simply send a distinct packet to each destination.
b. Flooding
c. Multidestination routing
d. Spanning tree based algorithm
e. Reverse path forwarding

a. The source to simply send a distinct packet to each destination

- One broadcasting method that requires no special features from the subnet is for the source to simply send a distinct packet to each destination.
- Not only is the method wasteful of bandwidth, but it also requires the source to have a complete list of all destinations.

b. Flooding

- Flooding is used especially if none of the methods described below are applicable.
- The problem with flooding as a broadcast technique is the same problem it has as a point-to-point routing algorithm: it generates too many packets and consumes too much bandwidth.

c. Multidestination routing

- If this method, each packet contains either a list of destinations or a bit map indicating the desired destinations.
- When a packet arrives at a router, the router checks all the destinations to determine the set of output lines that will be needed. (An output line is needed if it is the best route to at least one of the destinations.)
- The router generates a new copy of the packet for each output line to be used and includes in each packet only those destinations that are to use the line.
- In effect, the destination set is partitioned among the output lines. After a sufficient number of hops, each packet will carry only one destination and can be treated as a normal packet.
- Multidestination routing is like separately addressed packets, except that when several packets must follow the same route, one of them pays full fare and the rest ride free.

d. Spanning tree based algorithm

- Using a spanning tree (basically a sink tree) for the router initiating the broadcast.
- A spanning tree is a subset of the subnet that includes all the routers but contains no loops. If each router knows which of its lines belong to the spanning tree, it can copy an incoming broadcast packet onto all the spanning tree lines except the one it arrived on.
- This method makes excellent use of bandwidth, generating the absolute minimum number of packets necessary to do the job.
- The only problem is that each router must have knowledge of some spanning tree for the method to be applicable. Sometimes this information is available (e.g., with link state routing) but sometimes it is not (e.g., with distance vector routing).

e. Reverse path forwarding

- When a broadcast packet arrives at a router, the router checks to see if the packet arrived on the line that is normally used for sending packets to the source of the broadcast.
- If so, there is an excellent chance that the broadcast packet itself followed the best route from the router and is therefore the first copy to arrive at the router. This being the case, the router forwards copies of it onto all lines except the one it arrived on.
- If, the broadcast packet arrived on a line other than the preferred one for reaching the source, the packet is discarded as a likely duplicate.

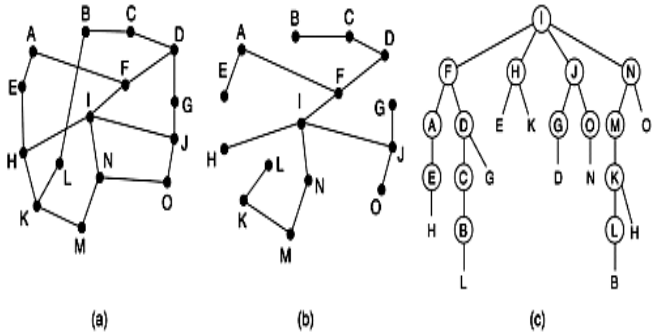


Figure 5-16. Reverse path forwarding. (a) A subnet. (b) A sink tree. (c) The tree built by reverse path forwarding

- The advantage of reverse path forwarding is that it is both reasonably efficient and easy to implement. It does not require routers to know about spanning trees, nor does it have the overhead of a destination list or bit map in each broadcast packet as does multidestination addressing. It does not require any special mechanism to stop the process, as flooding does.

MULTICAST ROUTING

- Multicast Routing are used to distribute data (for example, audio/video streaming broadcasts) to multiple recipients.
- Using multicast, a source can send a single copy of data to a single multicast address, which is then distributed to an entire group of recipients.
- Sending a message to a group is called multicasting, and its routing algorithm is called multicast routing.
- Multicasting requires group management to create and destroy groups, and to allow processes to join and leave groups.
- To do multicast routing, each router computes a spanning tree covering all other routers. For example, in Fig. 5-17(a) we have two groups, 1 and 2. Some routers are attached to hosts that belong to one or both of these groups, as indicated in the figure. A spanning tree for the leftmost router is shown in Fig. 5-17(b).

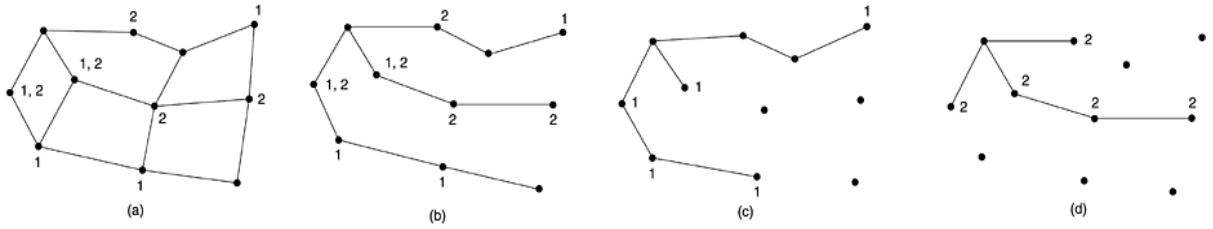


Figure 5-17. (a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.

- When a process sends a multicast packet to a group, the first router examines its spanning tree and removes all lines that do not lead to group members (pruning). In Fig. 5-17(c) shows the pruned spanning tree for group 1. Similarly, Fig. 5-17(d) shows the pruned spanning tree for group 2. Multicast packets are forwarded only along the appropriate spanning tree.
- One disadvantage of this algorithm is that it scales poorly to large networks. Suppose that a network has n groups, each with an average of m members. For each group, m pruned spanning trees must be stored, for a total of mn trees. When many large groups exist, considerable storage is needed to store all the trees.
- An alternative design uses core-based trees. Here, a single spanning tree per group is computed, with the root (the core) near the middle of the group. To send a multicast message, a host sends it to the core, which then does the multicast along the spanning tree. This tree will not be optimal for all sources, the reduction in storage costs from m trees to one tree per group is a major saving.

CONGESTION CONTROL ALGORITHMS

Congestion is a situation in Communication Networks in which too many packets are present in a part of the subnet, performance degrades. Congestion in a network may occur when the load on the network (i.e. the number of packets sent to the network) is greater than the capacity of the network (i.e. the number of packets a network can handle.)

When the number of packets dumped into the subnet by the hosts is within its carrying capacity, they are all delivered (except for a few that are afflicted with transmission errors) and the number delivered is proportional to the number sent.

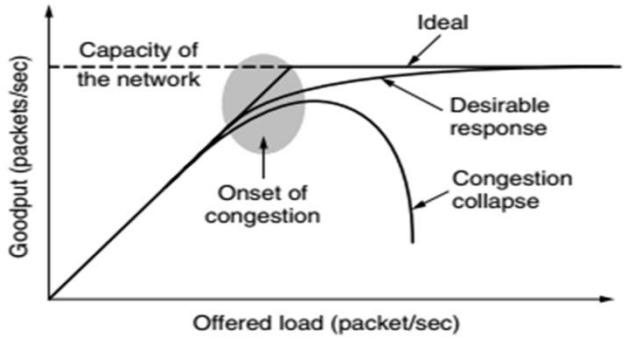


Figure 5-18. When too much traffic is offered, congestion sets in and performance degrades sharply.

As traffic increases too far, the routers are no longer able to cope and they begin losing packets. This tends to make matters worse. At very high trafffic, performance collapses completely and almost no packets are delivered.

Cause of Congestion:

The various causes of congestion in a subnet are:

1. The input traffic rate exceeds the capacity of the output lines. If suddenly, a stream of packet start arriving on three or four input lines and all need the same output line. In this case, a queue will be built up. If there is insufficient memory to hold all the packets, the packet will be lost. Increasing the memory to unlimited size does not solve the problem. This is because, by the time packets reach front of the queue, they have already timed out (as they waited the queue). When timer goes off source transmits duplicate packet that are also added to the queue. Thus same packets are added again and again, increasing the load all the way to the destination.
2. The routers are too slow to perform bookkeeping tasks (queuing buffers, updating tables, etc.)
3. The routers' buffer is too limited.
4. Congestion in a subnet can occur if the processors are slow. Slow speed CPU at routers will perform the routine tasks such as queuing buffers, updating table etc slowly. As a result of this, queues are built up even though there is excess line capacity.
5. Congestion is also caused by slow links. This problem will be solved when high speed links are used. But it is not always the case. Sometimes increase in link bandwidth can further deteriorate the congestion problem as higher speed links may make the network more unbalanced. Congestion can make itself worse. If a route!" does not have free buffers, it start ignoring/discarding the newly arriving packets. When these packets are discarded, the sender may retransmit them after the timer goes off. Such packets are transmitted by the sender again and again until the source gets the acknowledgement of these packets. Therefore multiple transmissions of packets will force the congestion to take place at the sending end.

General Principles of Congestion Control

On control theory point of view divide all solutions into two groups: open loop and closed loop.

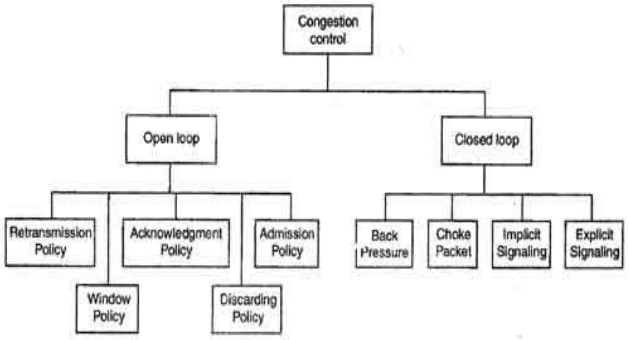
Open loop solutions

- Open loop solutions attempt to solve the problem by good design
- In this method, policies are used to prevent the congestion before it happens.
- Congestion control is handled either by the source or by the destination.
- Deciding when to accept new traffic
- Deciding when to discard packets and which ones, and
- Making scheduling decisions at various points in the network.
- All of these decisions are without regard to the current state of the network

Closed loop

- Closed loop congestion control mechanisms try to remove the congestion after it happens.
- Closed loop solutions are based on the concept of a feedback loop
- ***This approach has three parts when applied to congestion control:***

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



Types of Congestion Control Methods

Congestion Prevention Policies

Open loop systems are designed to minimize congestion in the first place, rather than letting it happen and reacting after the fact. They try to achieve their goal by using appropriate policies at various levels; data link, network, and transport policies that can affect congestion.

Layer	Policies
Transport	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy • Timeout determination
Network	• Virtual circuits versus datagram inside the subnet • Packet queueing and service policy • Packet discard policy • Routing algorithm • Packet lifetime management
Data link	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy

APPROACHES TO CONGESTION CONTROL

Congestion Control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened. Congestion control mechanisms are divided into two categories, one category prevents the congestion from happening and the other category removes congestion after it has taken place.

Congestion Control in Virtual-Circuit Subnets

i. Admission control

- Once congestion has been signaled, no more virtual circuits are set up until the problem has gone away. Thus, attempts to set up new transport layer connections fail.
- In a telephone system, when a switch gets overloaded, it also practices admission control, by not giving dial tones.
- This approach is crude, but it is simple and easy to carry out
- Admission control can also be combined with traffic-aware routing by considering routes around traffic hotspots as part of the setup procedure. For example, consider the network illustrated in Fig. 5-20(a), in which two routers are congested, as indicated.

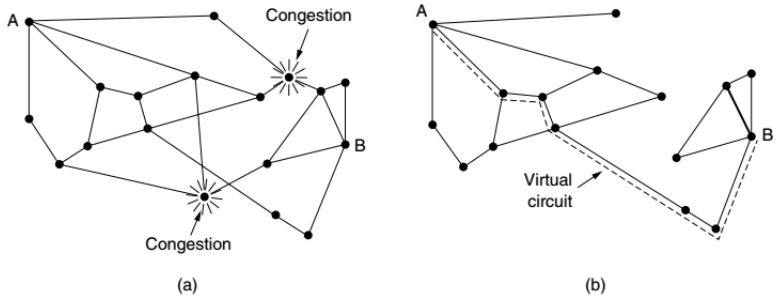


Figure 5-20. (a) A congested network. (b) The portion of the network that is not congested. A virtual circuit from *A* to *B* is also shown.

- Suppose that a host attached to router A wants to set up a connection to a host attached to router B. Normally, this connection would pass through one of the congested routers. To avoid this situation, we can redraw the network as shown in Fig. 5-20(b), omitting the congested routers and all of their lines. The dashed line shows a possible route for the virtual circuit that avoids the congested routers.

ii. Negotiating for an Agreement between the Host and Subnet when a Virtual Circuit is set up

- This agreement normally specifies the volume and shape of the traffic, quality of service (QoS) required, and other parameters.
- To keep its part of the agreement, the subnet will reserve resources along path when the circuit is set up.
- These resources can include table and buffer space in the routers and bandwidth in the lines.
- The drawback of this method that it tends to waste resources. For example, if six virtual circuits that might use 1 Mbps all pass through the same physical 6-Mbps line, the line has to marked as full, even though it may rarely happen that all six virtual circuits are transmitting at the same time.

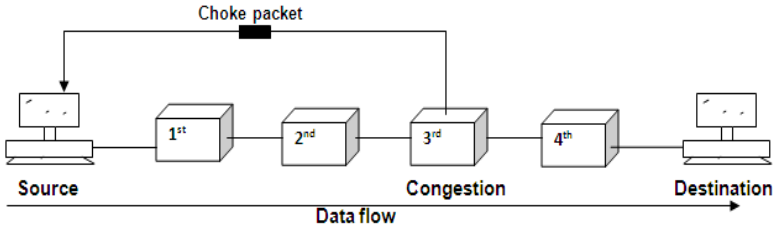
Congestion Control in Datagram Subnets

i. The Warning Bit

- A special bit in the packet header is set by the router to warn the source when congestion is detected.
- When the packet arrived at its destination, the transport entity copied the bit into the next acknowledgement sent back to the source. The source then cut back on traffic.
- As long as the router was in the warning state, it continued to set the warning bit, which meant that the source continued to get acknowledgements with it set.
- The source monitored the fraction of acknowledgements with the bit set and adjusted its transmission rate accordingly. As long as the warning bits continued to flow in, the source continued to decrease its transmission rate. When they slowed to a trickle, it increased its transmission rate.
- Note that since every router along the path could set the warning bit, traffic increased only when no router was in trouble.

ii. Choke Packets

- **Choke packets** are used in both virtual circuit and datagram subnets.
- A specialized packet that is used for flow control along a network.
- A router detects congestion by measuring the percentage of buffers in use, line utilization and average queue lengths.
- When it detects congestion, it sends choke packets across the network to all the data sources associated with the congestion.
- The original packet is tagged (a header bit is turned on) so that it will not generate any more choke packets farther along the path and is then forwarded in the usual way.
- In choke packet method, congested node sends a warning directly to the source station i.e. the intermediate nodes through which the packet has traveled are not warned.



- When the source host gets the choke packet, it is required to reduce the traffic sent to the specified destination by X percent. Since other packets aimed at the same destination are probably already under way and will generate yet more choke packets, the host should ignore choke packets referring to that destination for a fixed time interval. After that period has expired, the host listens for more choke packets for another interval. If one arrives, the line is still congested, so the host reduces the flow still more and begins ignoring choke packets again.
- If no choke packets arrive during the listening period, the host may increase the flow again. The feedback implicit in this protocol can help prevent congestion yet not throttle any flow unless trouble occurs.

iii. Hop-by-Hop Choke Packets

- At high speeds or over long distances, sending a choke packet to the source hosts does not work well because the reaction is so slow.
- An alternative approach is to have the choke packet take effect at every hop it passes through, as shown in the sequence of Fig. 5-22(b). Here, as soon as the choke packet reaches F, F is required to reduce the flow to D.
- Doing so will require F to devote more buffers to the flow, since the source is still sending away at full blast, but it gives D immediate relief.
- In the next step, the choke packet reaches E, which tells E to reduce the flow to F. This action puts a greater demand on E's buffers but gives F immediate relief.
- Finally, the choke packet reaches A and the flow genuinely slows down.
- The net effect of this hop-by-hop scheme is to provide quick relief at the point of congestion at the price of using up more buffers upstream. In this way, congestion can be nipped in the bud without losing any packets.

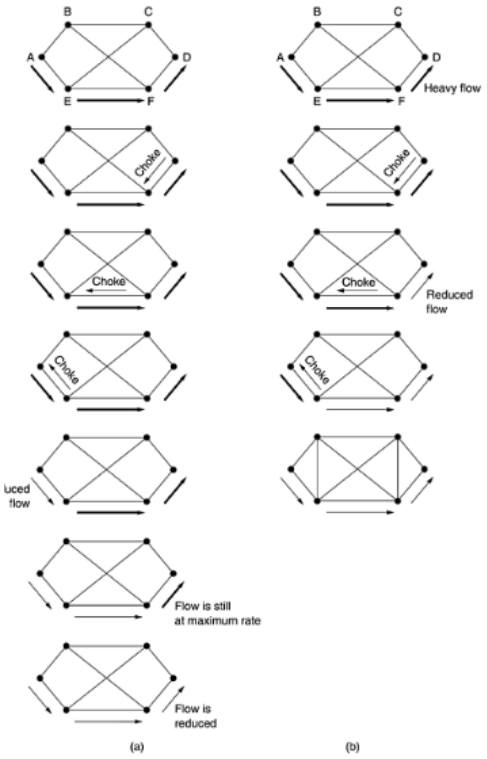


Figure 5-22. (a) A choke packet that affects only the source. (b) A choke packet that affects each hop it passes through

iv. Load Shedding

- The basic idea of **load shedding** scheme is that when routers have a lot of packets that they cannot handle, they just throw them away.
- The term comes from the world of electrical power generation, where it refers to the practice of utilities intentionally blacking out certain areas to save the entire grid from collapsing on hot summer days when the demand for electricity greatly exceeds the supply.
- A router drowning in packets can just pick packets at random to drop, may depend on the applications running.
- For file transfer, an old packet is worth more than a new one because dropping packet 6 and keeping packets 7 through 10 will cause a gap at the receiver that may force packets 6 through 10 to be retransmitted (if the receiver routinely discards out-of-order packets). In a 12-packet file, dropping 6 may require 7 through 12 to be retransmitted, whereas dropping 10 may require only 10 through 12 to be retransmitted. In contrast, for multimedia, a new packet is more important than an old one. If transmitting a document containing ASCII text and pictures. Losing a line of pixels in some image is far less damaging than losing a line of readable text.

- To implement an intelligent discard policy, applications must mark their packets in priority classes to indicate how important they are. If they do this, then when packets have to be discarded, routers can first drop packets from the lowest class, then the next lowest class, and so on.

v. Random Early Detection

- Random early detection (RED is a queueing discipline for a network scheduler suited for congestion avoidance
- RED monitors the average queue size and drops (or marks when used in conjunction with ECN) packets based on statistical probabilities.
- If the buffer is almost empty, then all incoming packets are accepted.
- As the queue grows, the probability for dropping an incoming packet grows too.
- When the buffer is full, the probability has reached 1 and all incoming packets are dropped.

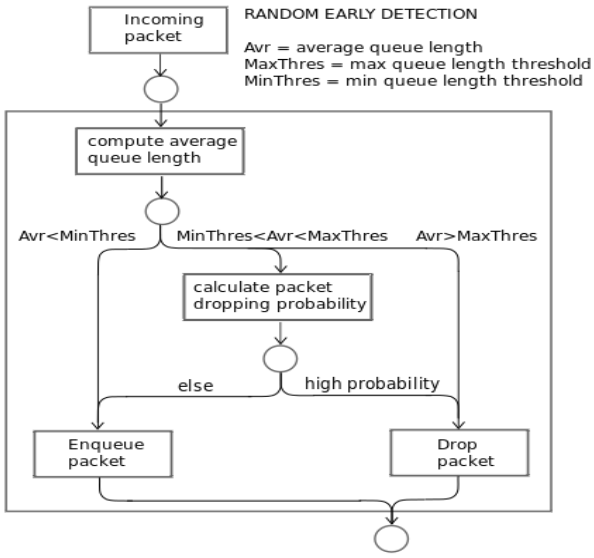


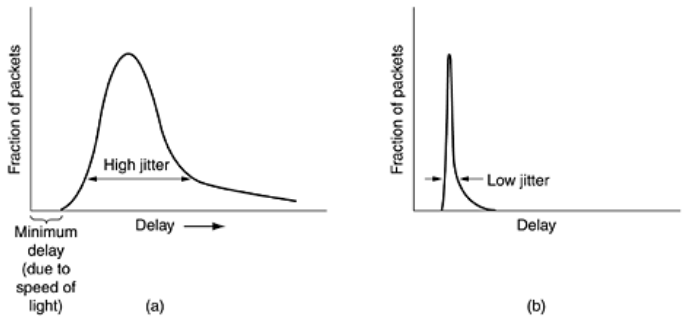
Fig. 5.23 Random Early Detection

- RED is more fair than tail drop, in the sense that it does not possess a bias against bursty traffic that uses only a small portion of the bandwidth.
- The more a host transmits, the more likely it is that its packets are dropped as the probability of a host's packet being dropped is proportional to the amount of data it has in a queue.
- Early detection helps avoid TCP global synchronization.

vi. Jitter Control

- Jitter is the amount of variation (i.e., standard deviation) in the packet arrival time.
- For applications such as audio and video streaming, it does not matter much if the packets take 20 msec or 30 msec to be delivered, as long as the transit time is constant.

(a) High jitter. (b) Low jitter.



- The jitter can be bounded by computing the expected transit time for each hop along the path. When a packet arrives at a router, the router checks to see how much the packet is behind or ahead of its schedule. The information is stored in the packet and updated at each hop.
- If the packet is ahead of schedule, it is held long enough to get it back on schedule.
- If the packet is behind schedule, the router tries to get it out quickly. The algorithm for determining which of several packets competing for an output line should go next can always choose the packet furthest behind in its schedule.

QUALITY OF SERVICE REQUIREMENTS

With the growth of multimedia networking, attempts at guaranteeing quality of service through network and protocol design are needed.

- A stream of packets from a source to a destination is called a flow.
- In a connection-oriented network, all the packets belonging to a flow follow the same route; in a connectionless network, they may follow different routes.
- The needs of each flow can be characterized by four primary parameters: reliability, delay, jitter, and bandwidth.
- Together these determine the QoS (Quality of Service) the flow requires.

Common applications and the stringency of their requirements are listed below.

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

- The first four applications have stringent requirements on reliability. No bits may be delivered incorrectly. This goal is usually achieved by checksumming each packet and verifying the checksum at the destination. If a packet is damaged in transit, it is not acknowledged and will be retransmitted eventually. This strategy gives high reliability.
- The four final (audio/video) applications can tolerate errors, so no checksums are computed or verified.
- File transfer applications, including e-mail and video, are not delay sensitive. If all packets are delayed uniformly by a few seconds, no harm is done.
- Interactive applications, such as Web surfing and remote login, are more delay sensitive. Real-time applications, such as telephony and videoconferencing have strict delay requirements. If all the words in a telephone call are each delayed by exactly 2.000

seconds, the users will find the connection unacceptable. On the other hand, playing audio or video files from a server does not require low delay.

- The first three applications are not sensitive to the packets arriving with irregular time intervals between them. Remote login is somewhat sensitive to that, since characters on the screen will appear in little bursts if the connection suffers much jitter.
- Video and especially audio are extremely sensitive to jitter. If a user is watching a video over the network and the frames are all delayed by exactly 2.000 seconds, no harm is done. But if the transmission time varies randomly between 1 and 2 seconds, the result will be terrible. For audio, a jitter of even a few milliseconds is clearly audible.

Techniques for Achieving Good Quality of Service

No single technique provides efficient, dependable QoS in an optimum way. Instead, a variety of techniques have been developed, with practical solutions often combining multiple techniques. Following are some of the techniques system designers use to achieve QoS.

i. Overprovisioning
ii. Buffering
iii. Traffic Shaping
iv. The Leaky Bucket Algorithm
v. The Token Bucket Algorithm
vi. Resource Reservation
vii. Admission Control
viii. Proportional Routing
ix. Packet Scheduling

Buffering

- Flows can be buffered on the receiving side before being delivered.
- It smooths out the jitter.
- For audio and video on demand, jitter is the main problem, so this technique helps a lot.

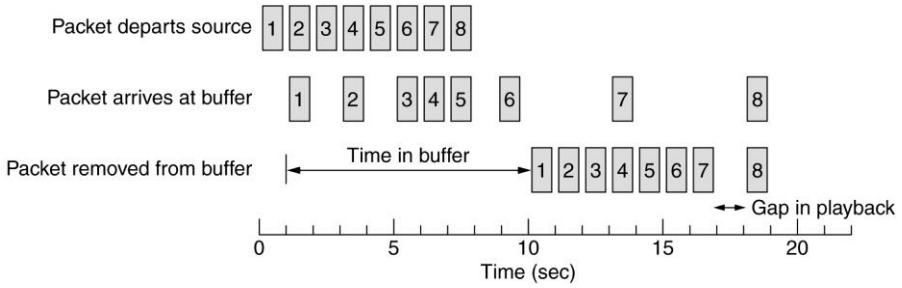


Figure 5-25. Smoothing the output stream by buffering packets

- In Figure 5-25, At t = 10 sec, playback begins. At this time, packets 1 through 6 have been buffered so that they can be removed from the buffer at uniform intervals for smooth play. Unfortunately, packet 8 has been delayed so much that it is not available when its play slot

comes up, so playback must stop until it arrives, creating an annoying gap in the music or movie.

- This problem can be alleviated by delaying the starting time even more, although doing so also requires a larger buffer. Commercial Web sites that contain streaming audio or video all use players that buffer for about 10 seconds before starting to play.

Traffic Shaping

- Traffic in data networks is bursty. It typically arrives at nonuniform rates as the traffic rate varies. e.g., videoconferencing with compression), browsing a new Web page, and computers switch between tasks.
- Traffic Shaping smooth out the traffic on the server side, rather than on the client side
- Traffic shaping is about regulating the average rate (and burstiness) of data transmission. When a connection is set up, the user and the subnet (i.e., the customer and the carrier) agree on a certain traffic pattern (i.e., shape) for that circuit. This is called a service level agreement.
- As long as the customer fulfills her part of the bargain and only sends packets according to the agreed-on contract, the carrier promises to deliver them all in a timely fashion. Traffic shaping reduces congestion and thus helps the carrier live up to its promise. Such agreements are not so important for file transfers but are of great importance for real-time data, such as audio and video connections, which have stringent quality-of-service requirements.
- Monitoring a traffic flow is called **traffic policing**.

The Leaky Bucket Algorithm

- It is a single-server queueing system with constant service time.
- Imagine a bucket with a small hole in the bottom, as illustrated in Fig. 5-26(a).
- No matter the rate at which water enters the bucket, the outflow is at a constant rate, r, when there is any water in the bucket and zero when the bucket is empty.
- Also, once the bucket is full, any additional water entering it spills over the sides and is lost (i.e., does not appear in the output stream under the hole).
- The same idea can be applied to packets, as shown in Fig. 5-26(b). Conceptually, each host is connected to the network by an interface containing a leaky bucket, that is, a finite internal queue.
- If a packet arrives at the queue when it is full, the packet is discarded.
- This arrangement can be built into the hardware interface or simulated by the host operating system.

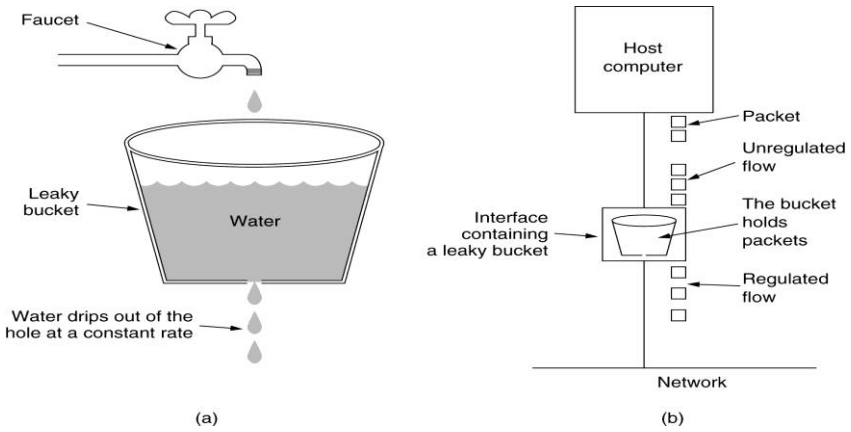


Figure 5-26. (a) A leaky bucket with water. (b) A leaky bucket with packets

- Implementing the original leaky bucket algorithm is easy. The leaky bucket consists of a finite queue. When a packet arrives, if there is room on the queue it is appended to the queue; otherwise, it is discarded. At every clock tick, one packet is transmitted.
- The byte-counting leaky bucket is implemented almost the same way. At each tick, a counter is initialized to n. If the first packet on the queue has fewer bytes than the current value of the counter, it is transmitted, and the counter is decremented by that number of bytes. Additional packets may also be sent, as long as the counter is high enough. When the counter drops below the length of the next packet on the queue, transmission stops until the next tick, at which time the residual byte count is reset and the flow can continue.
- In Fig. 5-27(a) we see the input to the leaky bucket running at 25 MB/sec for 40 msec. In Fig. 5-27(b) we see the output draining out at a uniform rate of 2 MB/sec for 500 msec.
- The leaky bucket algorithm enforces a rigid output pattern at the average rate, no matter how bursty the traffic is.
- The leaky bucket algorithm does not allow idle hosts to save up permission to send large bursts later

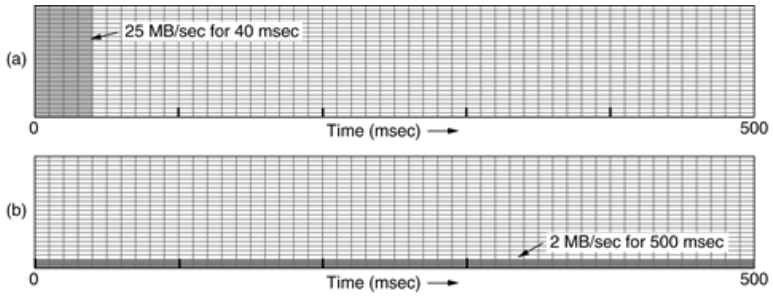


Figure 5-27. (a) Input to a leaky bucket. (b) Output from a leaky bucket

The Token Bucket Algorithm

- Token Bucket Algorithm allows the output to speed up somewhat when large bursts arrive, to avoid loses data.
- In this algorithm, the leaky bucket holds tokens, generated by a clock at the rate of one token every ΔT sec.

- In Fig. 5-28(a) we see a bucket holding three tokens, with five packets waiting to be transmitted. For a packet to be transmitted, it must capture and destroy one token. In Fig. 5-34(b) we see that three of the five packets have gotten through, but the other two are stuck waiting for two more tokens to be generated.

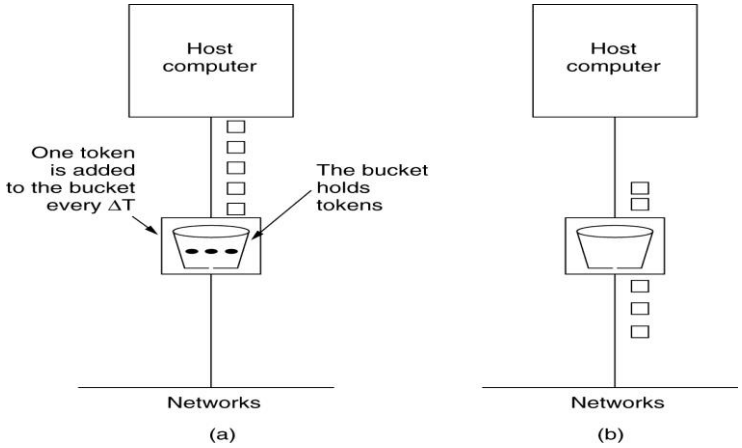
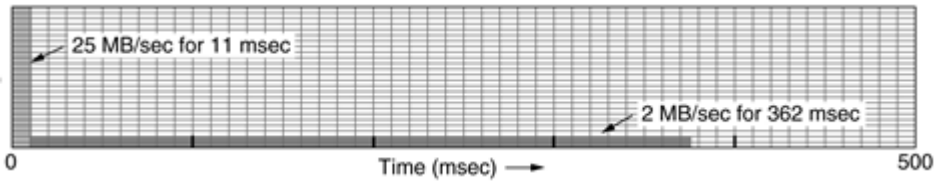


Figure 5-28. The token bucket algorithm. (a) Before. (b) After.

- The token bucket algorithm allow saving, up to the maximum size of the bucket, n. This property means that bursts of up to n packets can be sent at once.
- Token bucket algorithm throws away tokens (i.e., transmission capacity) when the bucket fills up but never discards packets.
- A minor variant is possible, in which each token represents the right to send not one packet, but k bytes. A packet can only be transmitted if enough tokens are available to cover its length in bytes. Fractional tokens are kept for future use.
- The implementation of the basic token bucket algorithm is just a variable that counts tokens. The counter is incremented by one every ΔT and decremented by one whenever a packet is sent. When the counter hits zero, no packets may be sent. In the byte-count variant, the counter is incremented by k bytes every ΔT and decremented by the length of each packet sent.
- Token bucket allow bursts, up to a regulated maximum length. Figure below shows, a token bucket with a capacity of 250 KB. Tokens arrive at a rate allowing output at 2 MB/sec. Assuming the token bucket is full when the 1-MB burst arrives, the bucket can drain at the full 25 MB/sec for about 11 msec. Then it has to cut back to 2 MB/sec until the entire input burst has been sent.



TRANSPORT PROTOCOLS - UDP

The Internet has two main protocols in the transport layer, **a connectionless protocol** and a **connection-oriented** one. The protocols complement each other. The connectionless protocol is **UDP.** It does almost nothing beyond sending packets between applications, letting applications build their own protocols on top as needed.

The connection-oriented protocol is **TCP.** It does almost everything. It makes connections and adds reliability with retransmissions, along with flow control and congestion control, all on behalf of the

applications that use it. Since UDP is a transport layer protocol that typically runs in the operating system and protocols that use UDP typically run in user s pace, these uses might be considered applications.

INTROUCTION TO UDP

- The Internet protocol suite supports a connectionless transport protocol called UDP (User Datagram Protocol). UDP provides a way for applications to send encapsulated IP datagrams without having to establish a connection.
- UDP transmits segments consisting of an 8-byte header followed by the pay-load. The two ports serve to identify the end-points within the source and destination machines.
- When a UDP packet arrives, its payload is handed to the process attached to the destination port. This attachment occurs when the BIND primitive. Without the port fields, the transport layer would not know what to do with each incoming packet. With them, it delivers the embedded segment to the correct application.

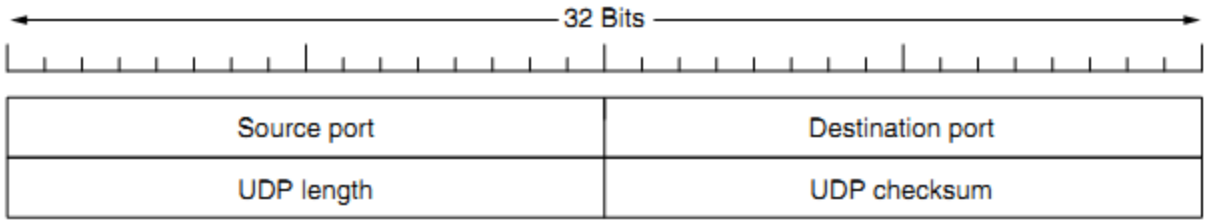


Fig 4.9: The UDP header

Source port, destination port: Identifies the end points within the source and destination machines.
UDP length: Includes 8-byte header and the data
UDP checksum: Includes the UDP header, the UDP data padded out to an even number of bytes if need be. It is an optional field

REMOTE PROCEDURE CALL

- In a certain sense, sending a message to a remote host and getting a reply back is like making a function call in a programming language. This is to arrange request-reply interactions on networks to be cast in the form of procedure calls.
- For example, just imagine a procedure named *get IP address* (*host name*) that works by sending a UDP packet to a DNS server and waiting or the reply, timing out and trying again if one is not forthcoming quickly enough. In this way, all the details of networking can be hidden from the programmer.

- RPC is used to call remote programs using the procedural call. When a process on machine 1 calls a procedure on machine 2, the calling process on 1 is suspended and execution of the called procedure takes place on 2.

- Information can be transported from the caller to the callee in the parameters and can come back in the procedure result. No message passing is visible to the application programmer. This technique is known as **RPC** (**Remote Procedure Call**) and has become the basis for many networking applications.

Traditionally, the calling procedure is known as the **client** and the called procedure is known as the **server.**

- In the simplest form, to call a remote procedure, the client program must be bound with a small library procedure, called the **client stub**,that represents the server procedure in the client's address space. Similarly, the server is bound with a procedure called the **server stub**. These procedures hide the fact that the procedure call from the client to the server is not local.

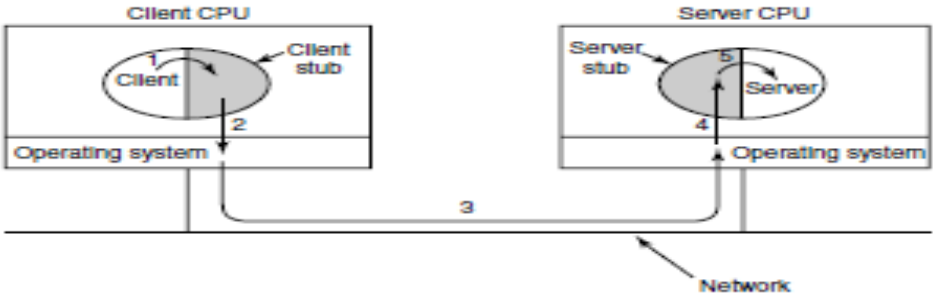


Fig 4.10: Steps in making a RPC

Step 1 is the client calling the client stub. This call is a local procedure call, with the parameters pushed onto the stack in the normal way.
Step 2 is the client stub packing the parameters into a message and making a system call to send the message. Packing the parameters is called **marshaling**.
Step 3 is the operating system sending the message from the client machine to the server machine.
Step 4 is the operating system passing the incoming packet to the server stub.
Step 5 is the server stub calling the server procedure with the **unmarshaled** parameters. The reply traces the same path in the other direction.

The key item to note here is that the client procedure, written by the user, just makes a normal (i.e., local) procedure call to the client stub, which has the same name as the server procedure. Since the client procedure and client stub are in the same address space, the parameters are passed in the usual way.

Similarly, the server procedure is called by a procedure in its address space with the parameters it expects. To the server procedure, nothing is unusual. In this way, instead of I/O being done on sockets, network communication is done by faking a normal procedure call. With RPC, passing pointers is impossible because the client and server are in different address spaces.

TCP (TRANSMISSION CONTROL PROTOCOL)

It was specifically designed to provide a reliable end-to end byte stream over an unreliable network. It was designed to adapt dynamically to properties of the inter network and to be robust in the face of many kinds of failures.

Each machine supporting TCP has a TCP transport entity, which accepts user data streams from local processes, breaks them up into pieces not exceeding 64kbytes and sends each piece as a separate IP datagram. When these datagrams arrive at a machine, they are given to TCP entity, which reconstructs the original byte streams. It is up to TCP to time out and retransmits them as needed, also to reassemble datagrams into messages in proper sequence.

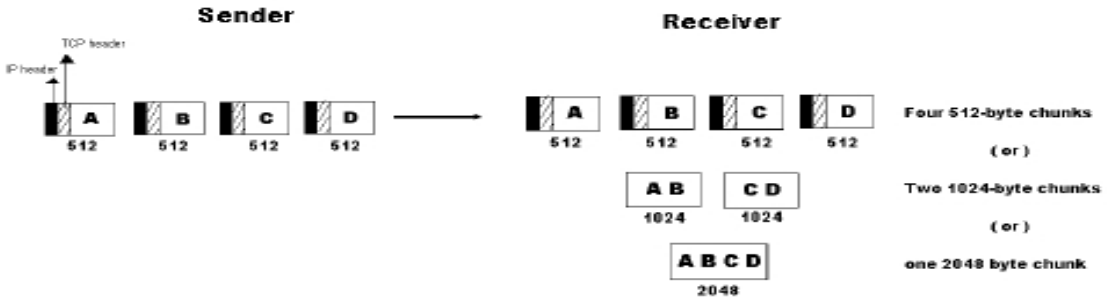
The different issues to be considered are:

1. The TCP Service Model
2. The TCP Protocol
3. The TCP Segment Header
4. The Connection Management
5. TCP Transmission Policy
6. TCP Congestion Control
7. TCP Timer Management.

The TCP Service Model

- TCP service is obtained by having both the sender and receiver create end points called **SOCKETS**
- Each socket has a socket number(address)consisting of the IP address of the host, called a **"PORT"** (= TSAP)
- To obtain TCP service a connection must be explicitly established between a socket on the sending machine and a socket on the receiving machine
- All TCP connections are full duplex and point to point i.e., multicasting or broadcasting is not supported.
- A TCP connection is a byte stream, not a message stream i.e., the data is delivered as chunks

E.g.: 4 * 512 bytes of data is to be transmitted.



Sockets:

A socket may be used for multiple connections at the same time. In other words, 2 or more connections may terminate at same socket. Connections are identified by socket identifiers at same socket. Connections are identified by socket identifiers at both ends. Some of the sockets are listed below:

Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt 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

Ports: Port numbers below 256 are called Well- known ports and are reserved for standard services.
***Eg*:**

PORT-21	To establish a connection to a host to transfer a file using FTP
PORT-23	To establish a remote login session using TELNET

The TCP Protocol

- A key feature of TCP, and one which dominates the protocol design, is that every byte on a TCP connection has its own 32-bit sequence number.
- When the Internet began, the lines between routers were mostly 56-kbps leased lines, so a host blasting away at full speed took over 1 week to cycle through the sequence numbers.
- The basic protocol used by TCP entities is the **sliding window protocol**.
- When a sender transmits a segment, it also starts a timer.
- When the segment arrives at the destination, the receiving TCP entity sends back a segment (with data if any exist, otherwise without data) bearing an acknowledgement number equal to the next sequence number it expects to receive.
- If the sender's timer goes off before the acknowledgement is received, the sender transmits the segment again.

The TCP Segment Header

Every segment begins with a fixed-format, 20-byte header. The fixed header may be followed by header options. After the options, if any, up to 65,535 - 20 - 20 = 65,495 data bytes may follow, where the first 20 refer to the IP header and the second to the TCP header. Segments without any data are legal and are commonly used for acknowledgements and control messages.

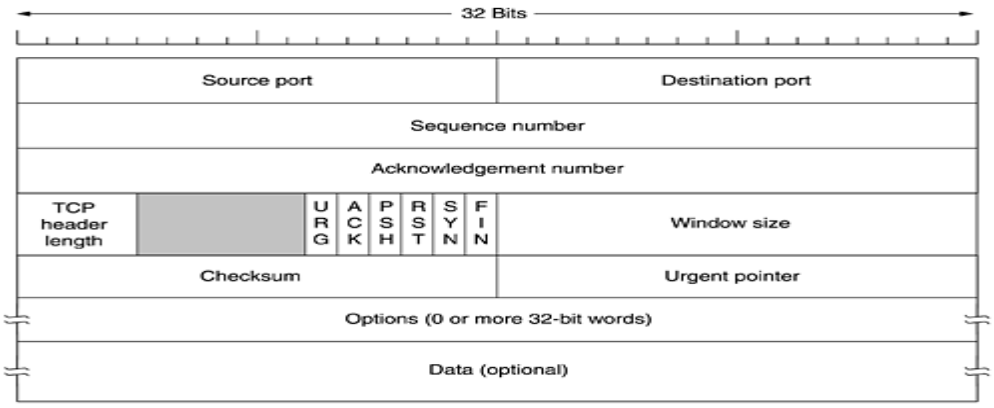


Fig 4.11: The TCP Header

Source Port, Destination Port : Identify local end points of the connections

Sequence number: Specifies the sequence number of the segment

Acknowledgement Number: Specifies the next byte expected.

TCP header length: Tells how many 32-bit words are contained in TCP header

URG: It is set to 1 if URGENT pointer is in use, which indicates start of urgent data.

ACK: It is set to 1 to indicate that the acknowledgement number is valid.

PSH: Indicates pushed data

RST: It is used to reset a connection that has become confused due to reject an invalid segment or refuse an attempt to open a connection.

FIN: Used to release a connection.

SYN: Used to establish connections.

TCP Connection Establishment

To establish a connection, one side, say, the server, passively waits for an incoming connection by executing the LISTEN and ACCEPT primitives, either specifying a specific source or nobody in particular.

The other side, say, the client, executes a CONNECT primitive, specifying the IP address and port to which it wants to connect, the maximum TCP segment size it is willing to accept, and optionally some user data (e.g., a password).

The CONNECT primitive sends a TCP segment with the *SYN* bit on and *ACK* bit off and waits for a response.

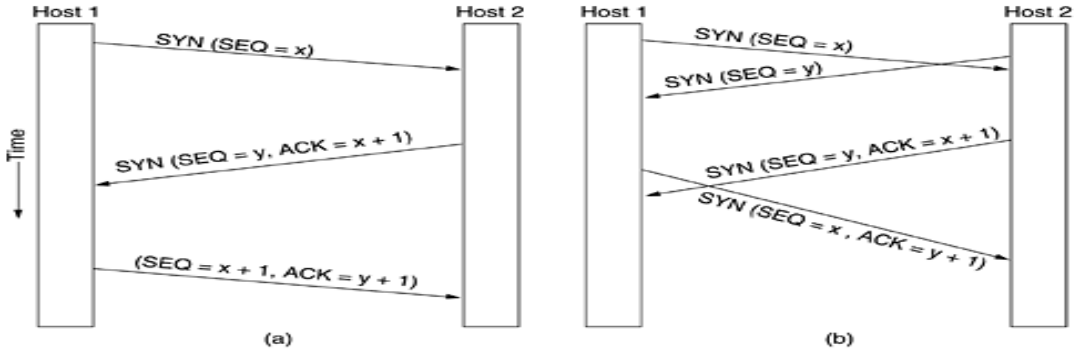


Fig 4.12: a) TCP Connection establishment in the normal case b) Call Collision

TCP Connection Release

- Although TCP connections are full duplex, to understand how connections are released it is best to think of them as a pair of simplex connections.
- Each simplex connection is released independently of its sibling. To release a connection, either party can send a TCP segment with the *FIN* bit set, which means that it has no more data to transmit.
- When the *FIN* is acknowledged, that direction is shut down for new data. Data may continue to flow indefinitely in the other direction, however.
- When both directions have been shut down, the connection is released.
- Normally, four TCP segments are needed to release a connection, one *FIN* and one *ACK* for each direction. However, it is possible for the first *ACK* and the second *FIN* to be contained in the same segment, reducing the total count to three.

TCP Connection Management Modeling

The steps required establishing and release connections can be represented in a finite state machine with the 11 states listed in Fig. 4.13. In each state, certain events are legal. When a legal event happens, some action may be taken. If some other event happens, an error is reported.

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

Figure 4.13. The states used in the TCP connection management finite state machine.

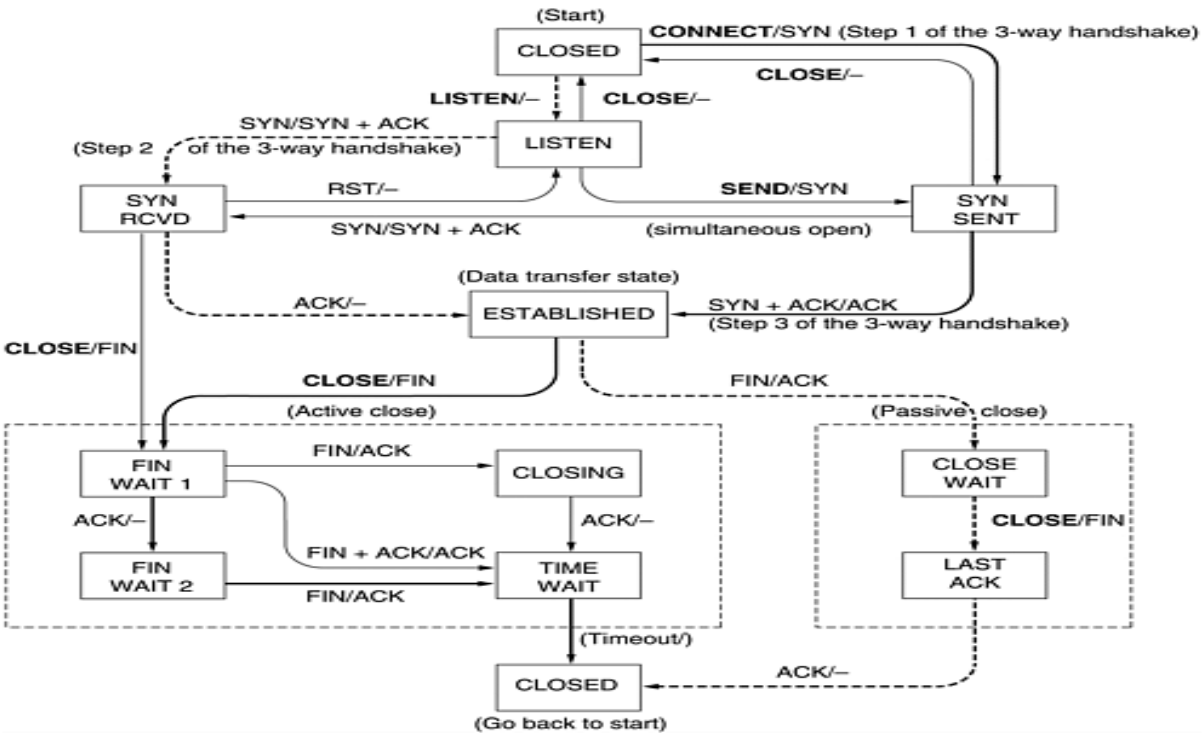


Figure 4.14 - TCP connection management finite state machine.

TCP Connection management from server's point of view:

1. The server does a **LISTEN** and settles down to see who turns up.
2. When a **SYN** comes in, the server acknowledges it and goes to the **SYNRCVD** state
3. When the servers **SYN** is itself acknowledged the 3-way handshake is complete and server goes to the **ESTABLISHED** state. Data transfer can now occur.
4. When the client has had enough, it does a close, which causes a **FIN** to arrive at the server [dashed box marked passive close].
5. The server is then signaled.
6. When it too, does a **CLOSE**, a **FIN** is sent to the client.
7. When the client's acknowledgement shows up, the server releases the connection and deletes the connection record.

TCP Transmission Policy

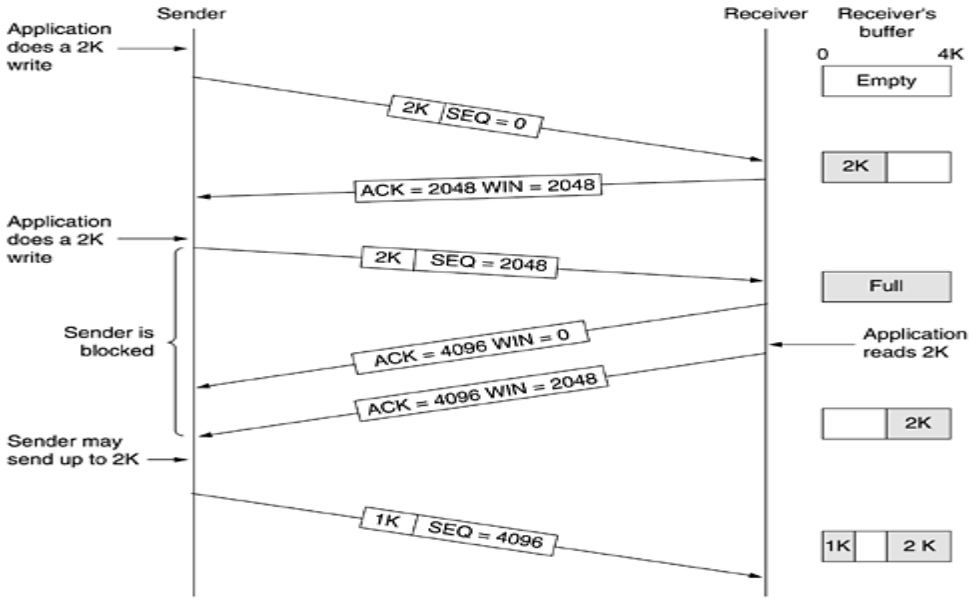
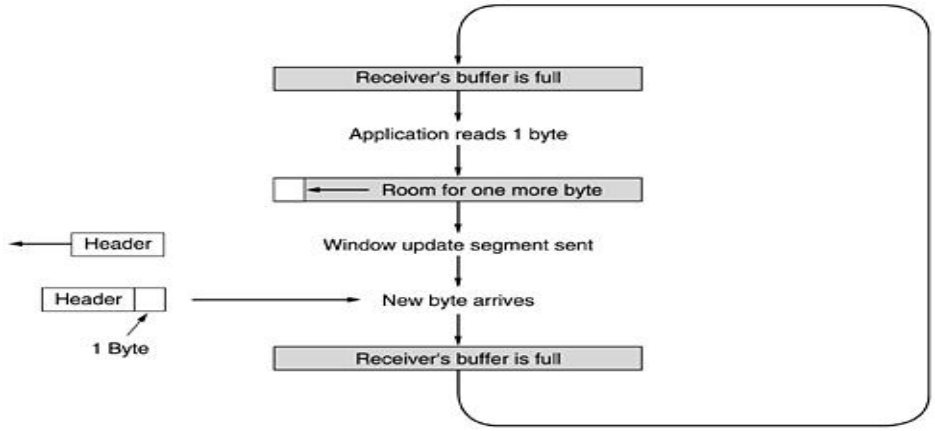


Figure 4.15 - Window management in TCP.

1. In the above example, the receiver has 4096-byte buffer.
2. If the sender transmits a 2048-byte segment that is correctly received, the receiver will acknowledge the segment.
3. Now the receiver will advertise a window of 2048 as it has only 2048 of buffer space, now.
4. Now the sender transmits another 2048 bytes which are acknowledged, but the advertised window is'0'.
5. The sender must stop until the application process on the receiving host has removed some data from the buffer, at which time TCP can advertise a layer window.

SILLY WINDOW SYNDROME:

This is one of the problems that ruin the TCP performance, which occurs when data are passed to the sending TCP entity in large blocks, but an interactive application on the receiving side reads 1 byte at a time.



- Initially the TCP buffer on the receiving side is full and the sender knows this(win=0)
- Then the interactive application reads 1 character from tcp stream.
- Now, the receiving TCP sends a window update to the sender saying that it is all right to send 1 byte.
- The sender obligates and sends 1 byte.
- The buffer is now full, and so the receiver acknowledges the 1 byte segment but sets window to zero. This behavior can go on forever.

TCP CONGESTION CONTROL:

TCP does to try to prevent the congestion from occurring in the first place in the following way:

When a connection is established, a suitable window size is chosen and the receiver specifies a window based on its buffer size. If the sender sticks to this window size, problems will not occur due to buffer overflow at the receiving end. But they may still occur due to internal congestion within the network. Let's see this problem occurs.

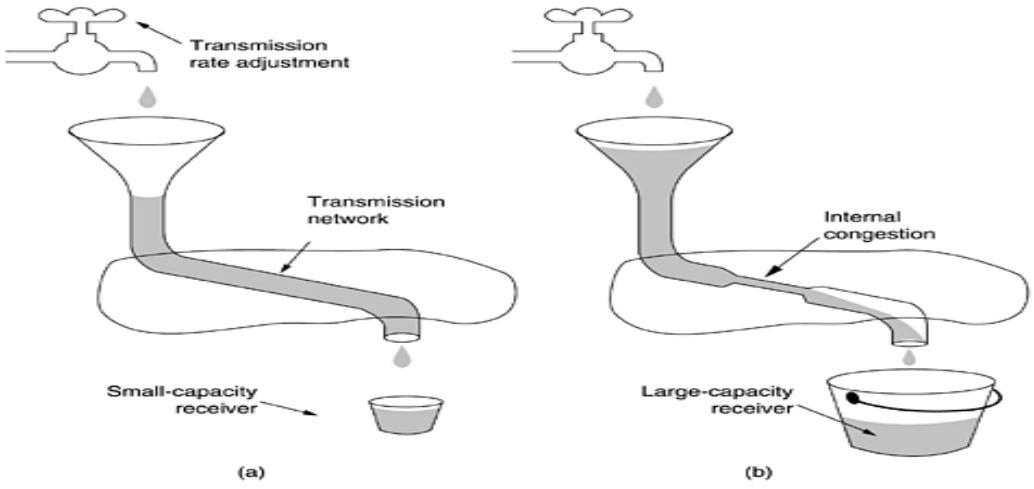


Figure 4.16. (a) A fast network feeding a low-capacity receiver. (b) A slow network feeding a high-capacity receiver.

In fig (a): We see a thick pipe leading to a small- capacity receiver. As long as the sender does not send more water than the bucket can contain, no water will be lost.

In fig (b): The limiting factor is not the bucket capacity, but the internal carrying capacity of the n/w. if too much water comes in too fast, it will backup and some will be lost.

- When a connection is established, the sender initializes the congestion window to the size of the max segment in use our connection.
- It then sends one max segment .if this max segment is acknowledged before the timer goes off, it adds one segment s worth of bytes to the congestion window to make it two maximum size segments and sends 2 segments.
- As each of these segments is acknowledged, the congestion window is increased by one max segment size.
- When the congestion window is 'n' segments, if all 'n' are acknowledged on time, the congestion window is increased by the byte count corresponding to 'n' segments.
- The congestion window keeps growing exponentially until either a time out occurs or the receiver's window is reached.
- The internet congestion control algorithm uses a third parameter, the **"threshold"** in addition to receiver and congestion windows.

Different congestion control algorithms used by TCP are:

- RTT variance Estimation.
- Exponential RTO back-off } Re-transmission Timer Management
- Karn's Algorithm
- Slow Start
- Dynamic window sizing on congestion
- Fast Retransmit } Window Management
- Fast Recovery

TCP TIMER MANAGEMENT:

TCP uses 3 kinds of timers:

1. Retransmission timer
2. Persistence timer
3. Keep-Alive timer.

1. Retransmission timer: When a segment is sent, a timer is started. If the segment is acknowledged before the timer expires, the timer is stopped. If on the other hand, the timer goes off before the acknowledgement comes in, the segment is retransmitted and the timer is started again. The algorithm that constantly adjusts the time-out interval, based on continuous measurements of n/w performance was proposed by JACOBSON and works as follows:

- for each connection, TCP maintains a variable RTT, that is the best current estimate of the round trip time to the destination inn question.
- When a segment is sent, a timer is started, both to see how long the acknowledgement takes and to trigger a retransmission if it takes too long.
- If the acknowledgement gets back before the timer expires, TCP measures how long the measurements took say M
- It then updates RTT according to the formula

$$\mathbf{RTT = \alpha RTT + (1-\alpha) M}$$

Where α = a smoothing factor that determines how much weight is given to the old value. Typically, $\alpha = 7/8$

Retransmission timeout is calculated as

$$\mathbf{D = \alpha D + (1-\alpha) | RTT-M |}$$

Where D = another smoothed variable, Mean RTT = expected acknowledgement value
M = observed acknowledgement value

$$\mathbf{Timeout = RTT+(4*D)}$$

2. Persistence timer:

It is designed to prevent the following deadlock:

- The receiver sends an acknowledgement with a window size of '0' telling the sender to wait later, the receiver updates the window, but the packet with the update is lost now both the sender and receiver are waiting for each other to do something
- when the persistence timer goes off, the sender transmits a probe to the receiver the response to the probe gives the window size
- if it is still zero, the persistence timer is set again and the cycle repeats
- if it is non zero, data can now be sent

3. Keep-Alive timer: When a connection has been idle for a long time, this timer may go off to cause one side to check if other side is still there. If it fails to respond, the connection is terminated.

APPLICATION LAYER

DOMAIN NAME SYSTEM

This is primarily used for mapping host and e-mail destinations to IP addresses but can also be used other purposes. DNS is defined in RFCs 1034 and 1035.

Working:-

- To map a name onto an IP address, an application program calls a library procedure called Resolver, passing it the name as a parameter.
- The resolver sends a UDP packet to a local DNS server, which then looks up the name and returns the IP address to the resolver, which then returns it to the caller.
- Armed with the IP address, the program can then establish a TCP connection with the destination, or send it UDP packets.

1. **The DNS name space.**
2. **Resource Records.**
3. **Name Servers.**

1. THE DNS NAME SPACE:

The Internet is divided into several hundred top level domains, where each domain covers many hosts. Each domain is partitioned into sub domains, and these are further partitioned as so on. All these domains can be represented by a tree, in which the leaves represent domains that have no sub domains. A leaf domain may contain a single host, or it may represent a company and contains thousands of hosts. Each domain is named by the path upward from it to the root. The components are separated by periods (pronounced "dot")

Eg: Sun Microsystems Engg. Department = eng.sun.com.

The top domain comes in 2 flavours:-

- **Generic:** com(commercial), edu(educational instructions), mil(the U.S armed forces, government), int (certain international organizations), net(network providers), org (non profit organizations).

- **Country:** include 1 entry for every country. Domain names can be either absolute (ends with a period e.g. eng.sum.com) or relative (doesn't end with a period). Domain names are case sensitive and the component names can be up to 63 characters long and full path names must not exceed 255 characters.

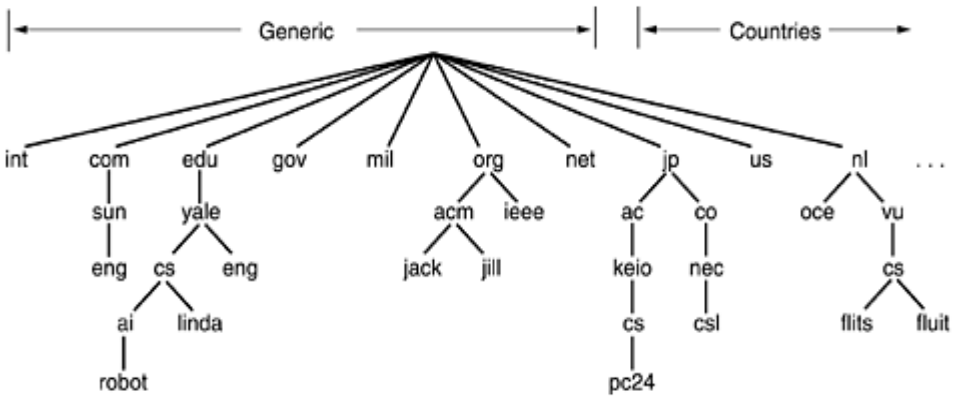


Figure 5-1. A portion of the Internet domain name space.

Insertions of a domain into the tree can be done in 2 days:-
• Under a generic domain (Eg: cs.yale.edu)
• Under the domain of their country (E.g: cs.yale.ct.us)

2. RESOURCE RECORDS:

Every domain can have a sent of resource records associated with it. For a single host, the most common resource record is just its IP address. When a resolver gives a domain name to DNS, it gets both the resource records associated with that name i.e., the real function of DNS is to map domain names into resource records.
A resource record is a 5-tuple and its format is as follows:

Domain	Name	Time to live	Type	Class	Value

Domain _name : Tells the domain to which this record applies.
Time- to- live : Gives an identification of how stable the record is (High Stable = 86400 i.e. no. of seconds /day) (High Volatile = 1 min)
Type: Tells what kind of record this is.
Class: It is IN for the internet information and codes for non internet information
Value: This field can be a number a domain name or an ASCII string

Type	Meaning	Value
SOA	Start of Authority	Parameters for this zone
A	IP address of a host	32-Bit integer
MX	Mail exchange	Priority, domain willing to accept e-mail
NS	Name Server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
HINFO	Host description	CPU and OS in ASCII
TXT	Text	Uninterpreted ASCII text

3. NAME SERVERS:

It contains the entire database and responds to all queries about it. DNS name space is divided up into non-overlapping zones, in which each zone contains some part of the tree and also contains name servers holding the authoritative information about that zone.

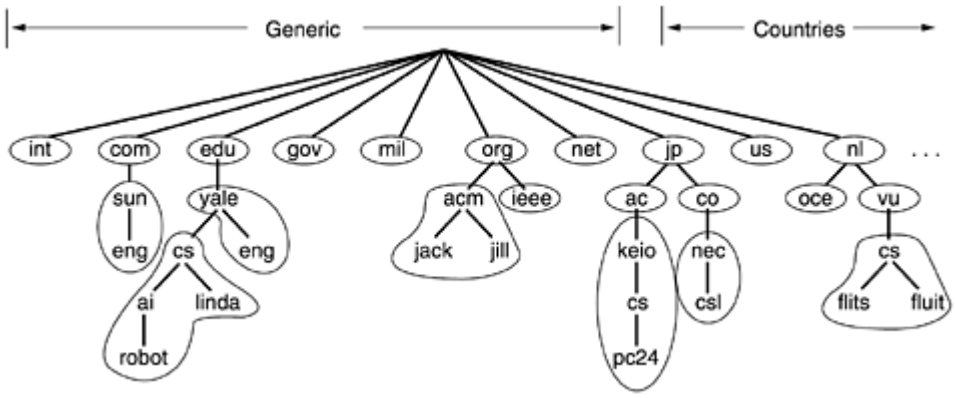


Figure 5-2. Part of the DNS name space showing the division into zones.

When a resolver has a query about a domain name, it passes the query to one of the local name servers:

1. If the domain being sought falls under the jurisdiction of name server, it returns the authoritative resource records (that comes from the authority that manages the record,and is always correct).
2. If the domain is remote and no information about the requested domain is available locally the name server sends a query message to the top level name server for the domain requested.

E.g.: A resolver of flits.cs.vle.nl wants to know the IP address of the host Linda.cs.yale.edu

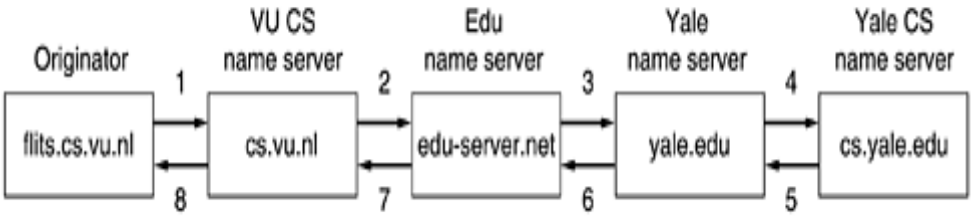


Figure 5-3. How a resolver looks up a remote name in eight steps.

Step 1: Resolver sends a query containing domain name sought the type and the class to local name server, cs.vu.nl.

Step 2: Suppose local name server knows nothing about it, it asks few others nearby name servers. If none of them know, it sends a UDP packet to the server for edu-server.net.

Step 3: This server knows nothing about Linda.cs.yale.edu or cs.yale.edu and so it forwards the request to the name server for yale.edu.

Step 4: This one forwards the request to cs.yale.edu which must have authoritative resource records.

Step 5 to 8: The resource record requested works its way back in steps 5-8 This query method is known as **Recursive Query**

3. When a query cannot be satisfied locally, the query fails but the name of the next server along the line to try is returned.

ELECTRONIC MAIL

1. ARCHITECTURE AND SERVICES:

E-mail systems consist of two subsystems. They are:-

(1). User Agents, which allow people to read and send e-mail

(2). Message Transfer Agents, which move messages from source to destination

E-mail systems support 5 basic functions:-

a. Composition
b. Transfer
c. Reporting
d. Displaying
e. Disposition

(a). Composition: It refers to the process of creating messages and answers. Any text editor is used for body of the message. While the system itself can provide assistance with addressing and numerous header fields attached to each message.

(b). Reporting: It has to do with telling the originator what happened to the message that is, whether it was delivered, rejected (or) lost.

(c). Transfer: It refers to moving messages from originator to the recipient.

(d). Displaying: Incoming messages are to be displayed so that people can read their email.

(e). Disposition: It concerns what the recipient dose with the message after receiving it. Possibilities include throwing it away before reading (or) after reading, saving it and so on.

Most systems allow users to create **mailboxes** to store incoming e-mail. Commands are needed to create and destroy mailboxes, inspect the contents of mailboxes, insert and delete messages from mailboxes, and so on.

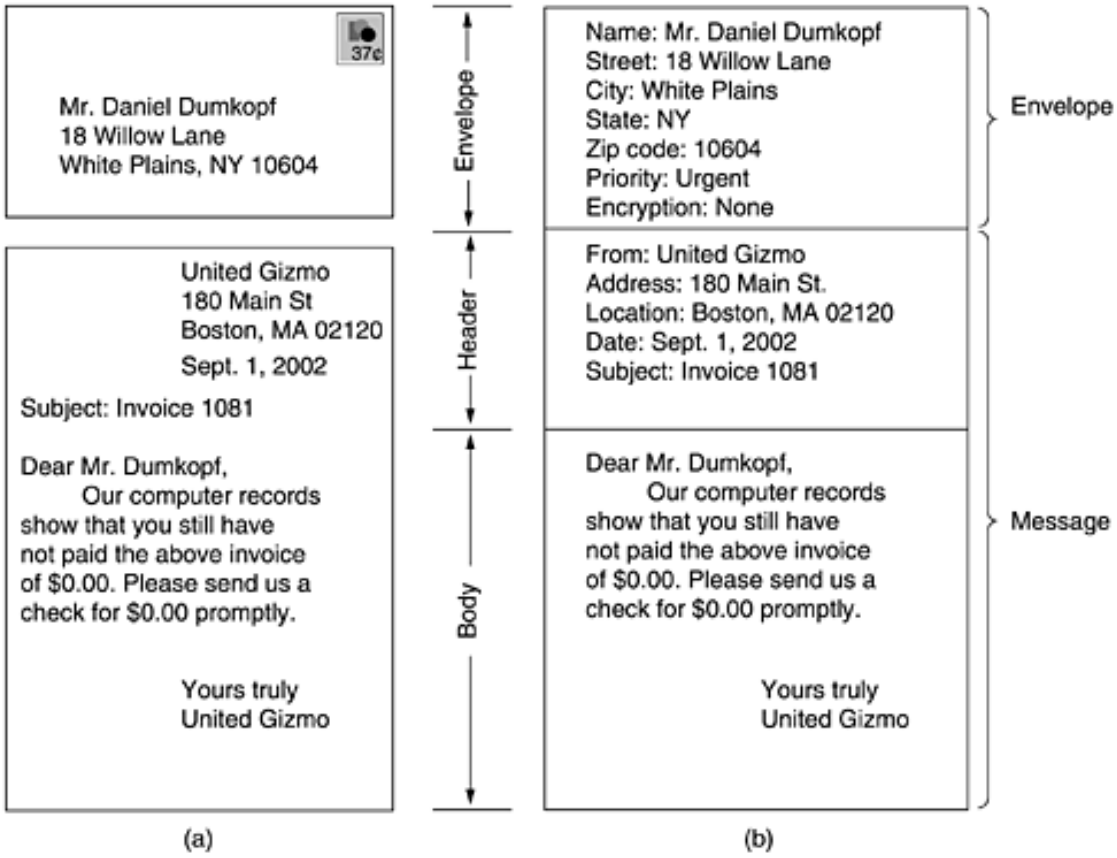


Figure 5-4: Envelopes and messages. (a) Paper mail. (b) Electronic mail.

(1) THE USER AGENT

A user agent is normally a program (sometimes called a mail reader) that accepts a variety of commands for composing, receiving, and replying to messages, as well as for manipulating mailboxes.

SENDING E-MAIL

To send an e-mail message, a user must provide the message, the destination address, and possibly some other parameters. The message can be produced with a free-standing text editor, a word processing program, or possibly with a specialized text editor built into the user agent. The destination address must be in a format that the user agent can deal with. Many user agents expect addresses of the form *user@dns-address*.

READING E-MAIL

When a user agent is started up, it looks at the user's mailbox for incoming e-mail before displaying anything on the screen. Then it may announce the number of messages in the mailbox or display a one-line summary of each one and wait for a command.

(2) MESSAGE FORMATS

RFC 822

Messages consist of a primitive envelope (described in RFC 821), some number of header fields, a blank line, and then the message body. Each header field (logically) consists of a single line of ASCII text containing the field name, a colon, and, for most fields, a value.

Header	Meaning
To:	E-mail address(es) of primary recipient(s)
Cc:	E-mail address(es) of secondary recipient(s)
Bcc:	E-mail address(es) for blind carbon copies
From:	Person or people who created the message
Sender:	E-mail address of the actual sender
Received:	Line added by each transfer agent along the route
Return-Path:	Can be used to identify a path back to the sender

Figure 5-5: RFC 822 header fields related to message transport

MIME — The Multipurpose Internet Mail Extensions

RFC 822 specified the headers but left the content entirely up to the users. Nowadays, on the worldwide Internet, this approach is no longer adequate. The problems include sending and receiving

1. Messages in languages with accents (e.g., French and German).
2. Messages in non-Latin alphabets (e.g., Hebrew and Russian).
3. Messages in languages without alphabets (e.g., Chinese and Japanese).
4. Messages not containing text at all (e.g., audio or images).

A solution was proposed in RFC 1341 called **MIME** (**Multipurpose Internet Mail Extensions**)

The basic idea of MIME is to continue to use the RFC 822 format, but to add structure to the message body and define encoding rules for non-ASCII messages. By not deviating from RFC 822, MIME messages can be sent using the existing mail programs and protocols. All that has to be changed are the sending and receiving programs, which users can do for themselves.

Header	Meaning
MIME-Version:	Identifies the MIME version
Content-Description:	Human-readable string telling what is in the message
Content-Id:	Unique identifier
Content-Transfer-Encoding:	How the body is wrapped for transmission
Content-Type:	Type and format of the content

Figure 5-6: RFC 822 headers added by MIME

MESSAGE TRANSFER

The message transfer system is concerned with relaying messages from the originator to the recipient. The simplest way to do this is to establish a transport connection from the source machine to the destination machine and then just transfer the message.

SMTP—THE SIMPLE MAIL TRANSFER PROTOCOL

SMTP is a simple ASCII protocol. After establishing the TCP connection to port 25, the sending machine, operating as the client, waits for the receiving machine, operating as the server, to talk first. The server starts by sending a line of text giving its identity and telling whether it is prepared to receive mail. If it is not, the client releases the connection and tries again later.

Even though the SMTP protocol is completely well defined, a **few problems** can still arise.

One problem relates to message length. Some older implementations cannot handle messages exceeding 64 KB.

Another problem relates to timeouts. If the client and server have different timeouts, one of them may give up while the other is still busy, unexpectedly terminating the connection.

Finally, in rare situations, infinite mailstorms can be triggered.

For example, if host 1 holds mailing list *A* and host 2 holds mailing list *B* and each list contains an entry for the other one, then a message sent to either list could generate a never-ending amount of e-mail traffic unless somebody checks for it.

FINAL DELIVERY

With the advent of people who access the Internet by calling their ISP over a modem, it breaks down.

One solution is to have a message transfer agent on an ISP machine accept e-mail for its customers and store it in their mailboxes on an ISP machine. Since this agent can be on-line all the time, e-mail can be sent to it 24 hours a day.

POP3

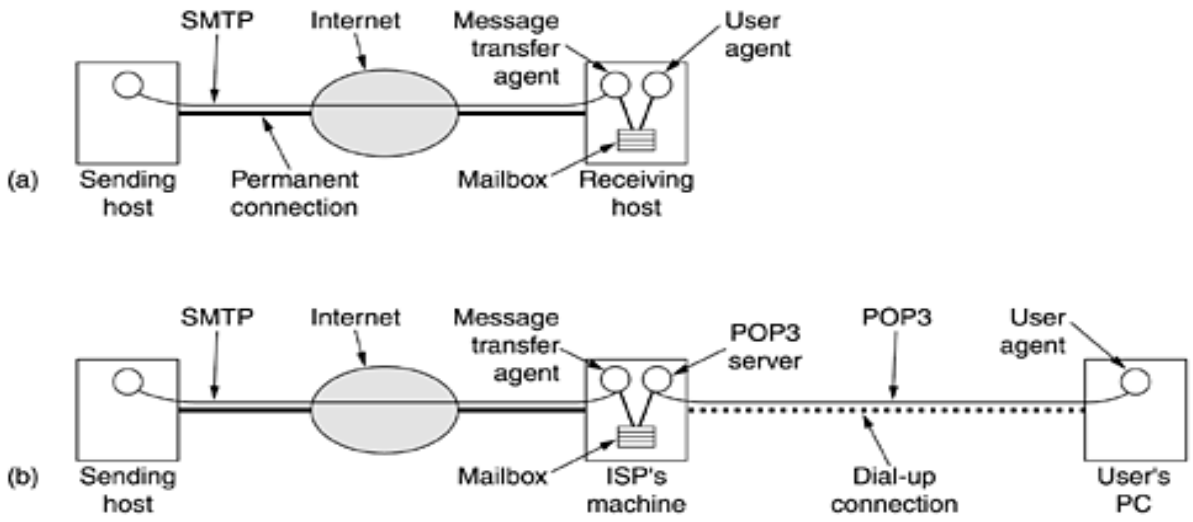


Figure:5-7

(a) Sending and reading mail when the receiver has a permanent Internet connection and the user agent runs on the same machine as the message transfer agent.

(b) Reading e-mail when the receiver has a dial-up connection to an ISP

POP3 begins when the user starts the mail reader. The mail reader calls up the ISP (unless there is already a connection) and establishes a TCP connection with the message transfer agent at port 110. Once the connection has been established, the POP3 protocol goes through three states in sequence:

1. Authorization.
2. Transactions.
3. Update.

The authorization state deals with having the user log in.

The transaction state deals with the user collecting the e-mails and marking them for deletion from the mailbox.

The update state actually causes the e-mails to be deleted.

IMAP (**Internet Message Access Protocol**).

POP3 normally downloads all stored messages at each contact, the result is that the user's e-mail quickly gets spread over multiple machines, more or less at random; some of them not even the user's.

This disadvantage gave rise to an alternative final delivery protocol, **IMAP** (**Internet Message Access Protocol**).

IMAP assumes that all the e-mail will remain on the server indefinitely in multiple mailboxes. IMAP provides extensive mechanisms for reading messages or even parts of messages, a feature useful when using a slow modem to read the text part of a multipart message with large audio and video attachments.