SVR ENGINEERING COLLEGE

AYYALURUMETTA (V), NANDYAL, KURNOOL DT.

ANDHRA PRADESH – 518502

2020 – 2021 Ist Semester

COMPUTER NETWORKS

Prepared by

A.D.SIVARAMA KUMAR

Assistant Professor

For

III B.Tech (CSE)

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING

UNIT I

INTRODUCTION: Networks, Network types, Internet History, Standards and Administration, Protocol Layering, TCP/IP protocol suite, The OSI model

Introduction to Physical Layer: Data and Signals. Transmission Impairment, Data rate Limits, performance.

Transmission media: Introduction, Guided Media, unguided media,

Switching: Introduction Circuit Switched Networks, Packet Switching

NETWORKS:

A **network** is the interconnection of a set of devices capable of communication. In this definition, a device can be a **host** (or an *end system* as it is sometimes called) such as a large computer, desktop, laptop, workstation, cellular phone, or security system. A device in this definition can also be a **connecting device** such as a router, which connects the network to other networks, a switch, which connects devices together, a modem (modulator-demodulator), which changes the form of data. These devices in a network are connected using wired or wireless transmission media such as cable

Network Criteria:

A network must be able to meet a certain number of criteria. The most important of these are **performance, reliability, and security.**

Performance

Performance can be measured in many ways, including transit time and response time. Transit time is the amount of time required for a message to travel from one device to another. Response time is the elapsed time between an inquiry and a response. The performance of a network depends on a number of factors, including the number of users, the type of transmission medium, the capabilities of the connected hardware, and the efficiency of the software.

Performance is often evaluated by two networking metrics: **throughput** and **delay.** We often need more throughputs and less delay.

Reliability

In addition to accuracy of delivery, network **reliability** is measured by the frequency of failure, the time it takes a link to recover from a failure, and the network's robustness in a catastrophe.

Security

Network **security** issues include protecting data from unauthorized access, protecting data from damage and development, and implementing policies and procedures for recovery from breaches and data losses.

Physical Structures:

Type of Connection

A network is two or more devices connected through links. A link is a communications pathway that transfers data from one device to another. For visualization purposes, it is simplest to imagine any link as a line drawn between two points. For communication to occur, two devices must be connected in some way to the same link at the same time.

There are two possible types of connections: **point-to-point and multipoint.**

Point-to-Point

A **point-to-point connection** provides a dedicated link between two devices. The entire capacity of the link is reserved for transmission between those two devices. Most point-to-point connections use an actual length of wire or cable to connect the two ends

Multipoint

A **multipoint** (also called **multidrop**) **connection** is one in which more than two specific devices share a single link

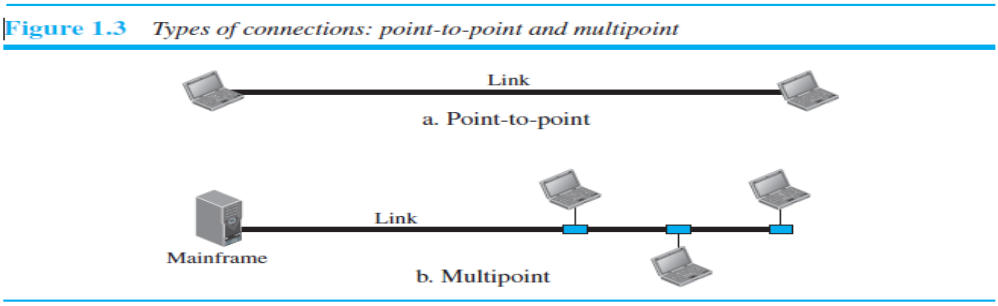


Figure 1.3 *Types of connections: point-to-point and multipoint*

Physical Topology

The term ***physical topology*** refers to the way in which a network is laid out physically. Two or more devices connect to a link; two or more links form a topology. The topology of a network is the geometric representation of the relationship of all the links and linking devices (usually called ***nodes***) to one another.

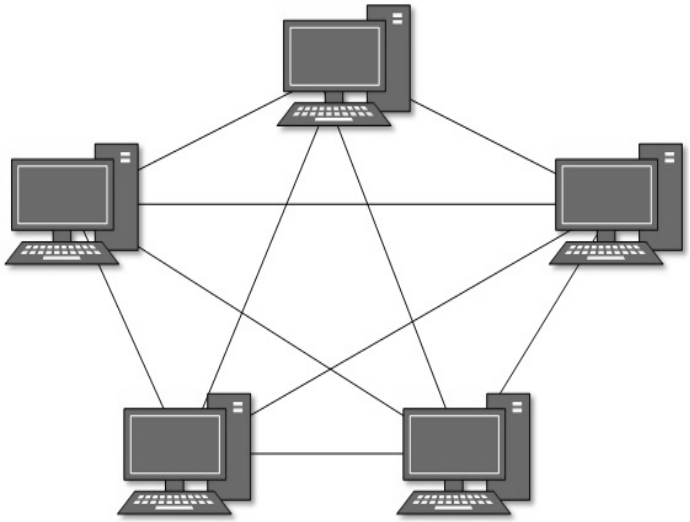
There are four basic topologies possible:

- ***Mesh Topology,***
- ***Star Topology,***
- ***Bus Topology, and***
- ***Ring Topology.***

A Network Topology is the arrangement with which computer systems or network devices are connected to each other. Topologies may define both physical and logical aspect of the network. Both logical and physical topologies could be same or different in a same network.

Mesh Topology

In this type of topology, a host is connected to one or multiple hosts. This topology has hosts in point-to-point connection with every other host or may also have hosts which are in point-to-point connection with few hosts only.



The main disadvantages of a mesh are related to the amount of cabling and the number of I/O ports required. First, because every device must be connected to every other device, installation and reconnection are difficult. Second, the sheer bulk of the wiring can be greater than the available space (in walls, ceilings, or floors) can accommodate.

Hosts in Mesh topology also work as relay for other hosts which do not have direct point-to-point links. Mesh technology comes into two types:

Full Mesh: All hosts have a point-to-point connection to every other host in the network. Thus for every new host n(n-1)/2 connections are required. It provides the most reliable network structure among all network topologies.

Partially Mesh: Not all hosts have point-to-point connection to every other host. Hosts connect to each other in some arbitrarily fashion. This topology exists where we need to provide reliability to some hosts out of all.

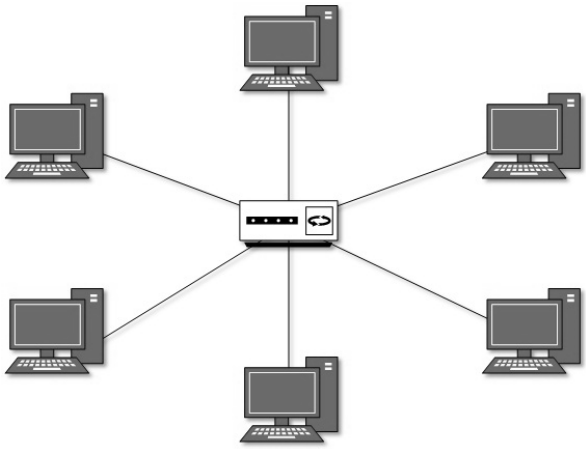
Star Topology

All hosts in Star topology are connected to a central device, known as hub device, using a point-to-point connection. That is, there exists a point to point connection between hosts and hub. The hub device can be any of the following:

Layer-1 device such as hub or repeater

Layer-2 device such as switch or bridge

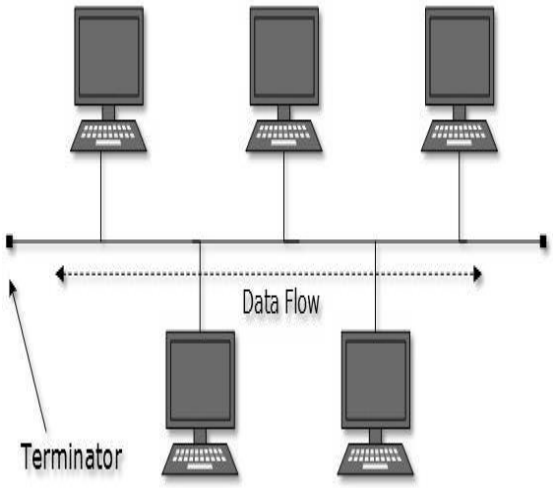
Layer-3 device such as router or gateway



As in Bus topology, hub acts as single point of failure. If hub fails, connectivity of all hosts to all other hosts fails. Every communication between hosts takes place through only the hub. Star topology is not expensive as to connect one more host, only one cable is required and configuration is simple.

Bus Topology

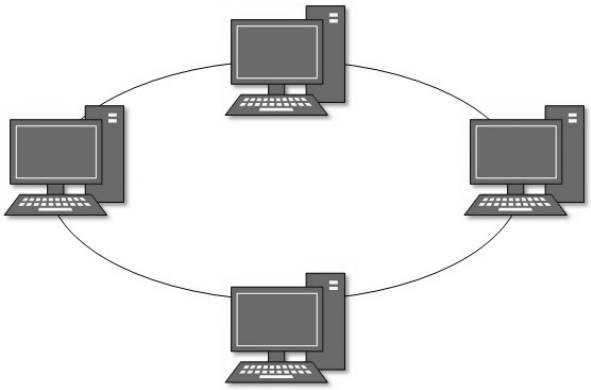
In case of Bus topology, all devices share single communication line or cable. Bus topology may have problem while multiple hosts sending data at the same time. Therefore, Bus topology either uses CSMA/CD technology or recognizes one host as Bus Master to solve the issue. It is one of the simple forms of networking where a failure of a device does not affect the other devices. But failure of the shared communication line can make all other devices stop functioning.



Both ends of the shared channel have line terminator. The data is sent in only one direction and as soon as it reaches the extreme end, the terminator removes the data from the line.

Ring Topology

In ring topology, each host machine connects to exactly two other machines, creating a circular network structure. When one host tries to communicate or send message to a host which is not adjacent to it, the data travels through all intermediate hosts. To connect one more host in the existing structure, the administrator may need only one more extra cable.



Failure of any host results in failure of the whole ring. Thus, every connection in the ring is a point of failure. There are methods which employ one more backup ring.Ring topology was prevalent when IBM introduced its local-area network, Token
Ring.

NETWORK TYPES

Different types of networks

Local Area Network

A local area network (LAN) is usually privately owned and connects some hosts in a single office, building, or campus. Depending on the needs of an organization, a LAN can be as simple as two PCs and a printer in someone's home office, or it can extend throughout a company and include audio and video devices. Each host in a LAN has an identifier, an address, that uniquely defines the host in the LAN. A packet sent by a host to another host carries both the source host's and the destination host's addresses.

All hosts in a network were connected through a common cable, which meant that a packet sent from one host to another was received by all hosts.

Local area networks, generally called LANs, are privately-owned networks within a single building or campus of up to a few kilometres 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. LANs are distinguished from other kinds of networks by three characteristics:

(1) Their size,

(2) Their transmission technology, and

(3) Their topology.

LANs are restricted in size, which means that the worst-case transmission time is bounded and known in advance. Knowing this bound makes it possible to use certain kinds of designs that would not otherwise be possible. It also simplifies network management.

LANs may use a transmission technology consisting of a cable to which all the machines are attached, like the telephone company party lines once used in rural areas. Traditional LANs run at speeds of 10 Mbps to 100 Mbps, have low delay (microseconds or nanoseconds), and make very few errors. Newer LANs operate at up to 10 Gbps Various topologies are possible for broadcast LANs.

Figure1 shows two of them. In a bus (i.e., a linear cable) network, at any instant at most one machine is the master and is allowed to transmit. All other machines are required to refrain from sending. An arbitration mechanism is needed to resolve conflicts when two or more machines want to transmit simultaneously. The arbitration mechanism may be centralized or distributed. IEEE 802.3, popularly called Ethernet, for example, is a bus-based broadcast network with decentralized control, usually operating at 10 Mbps to 10 Gbps. Computers on an Ethernet can transmit whenever they want to; if two or more packets collide, each computer just waits a random time and tries again later.

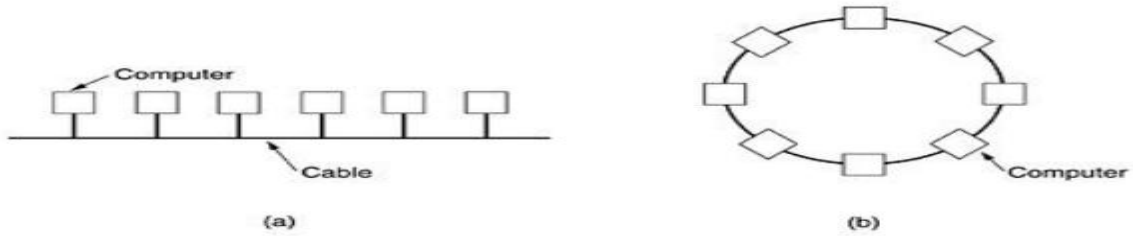
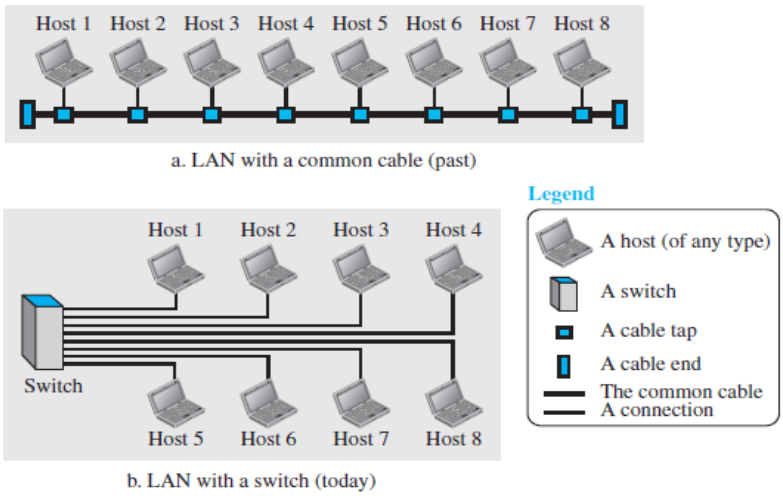


Fig.1: Two broadcast networks . (a) Bus. (b) Ring.

A second type of broadcast system is the ring. In a ring, each bit propagates around on its own, not waiting for the rest of the packet to which it belongs. Typically, each bit circumnavigates the entire ring in the time it takes to transmit a few bits, often before the complete packet has even been transmitted. As with all other broadcast systems, some rule is needed for arbitrating simultaneous accesses to the ring. Various methods, such as having the machines take turns, are in use. IEEE 802.5 (the IBM token ring), is a ring-based LAN operating at 4 and 16 Mbps. FDDI is another example of a ring network.

Figure 1.8 *An isolated LAN in the past and today*



Wide Area Network

A wide area network (WAN) is also an interconnection of devices capable of communication. However, there are some differences between a LAN and a WAN. A LAN is normally limited in size, spanning an office, a building, or a campus.

A WAN has a wider geographical span, spanning a town, a state, a country, or even the world. A LAN interconnects hosts. A WAN interconnects connecting devices such as switches, routers, or modems. A LAN is normally privately owned by the organization that uses it. A WAN is normally created and run by communication companies and leased by an organization that uses it.

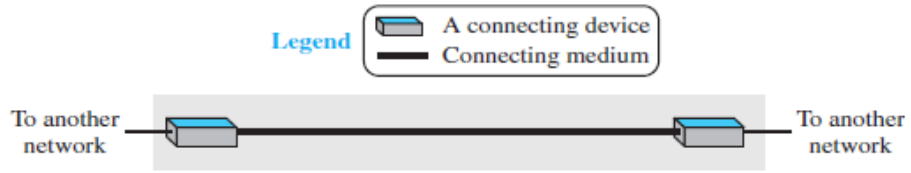
Two distinct examples of WANs today

- Point-to-point WANs and
- Switched WANs.

Point-to-Point WAN

A point-to-point WAN is a network that connects two communicating devices through a transmission media (cable or air).

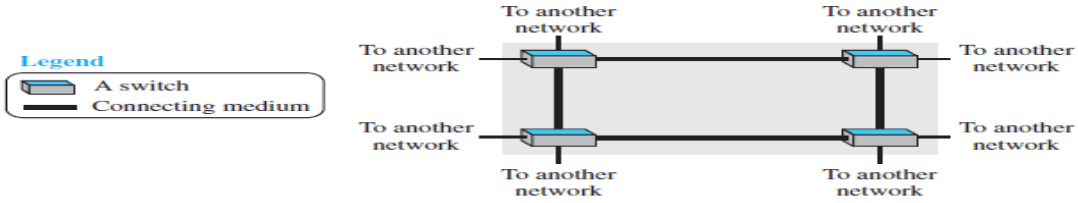
Figure 1.9 *A point-to-point WAN*



Switched WAN

A switched WAN is a network with more than two ends. A switched WAN, as we will see shortly, is used in the backbone of global communication today. We can say that a switched WAN is a combination of several point-to-point WANs that are connected by switches.

Figure 1.10 *A switched WAN*



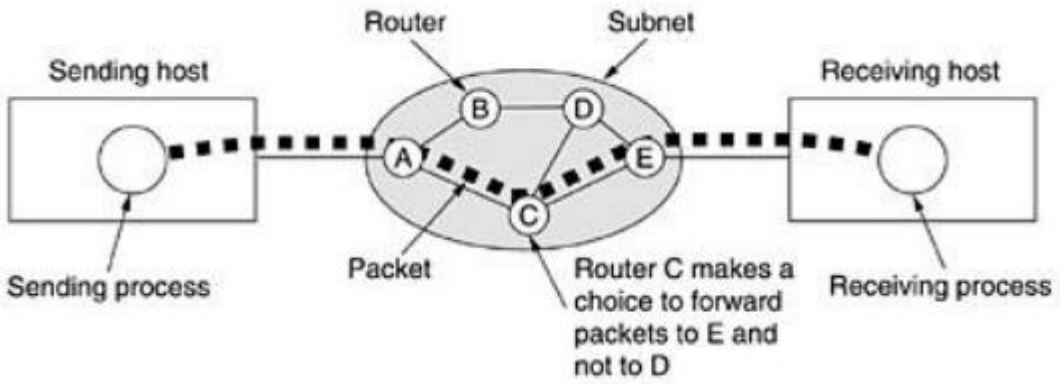


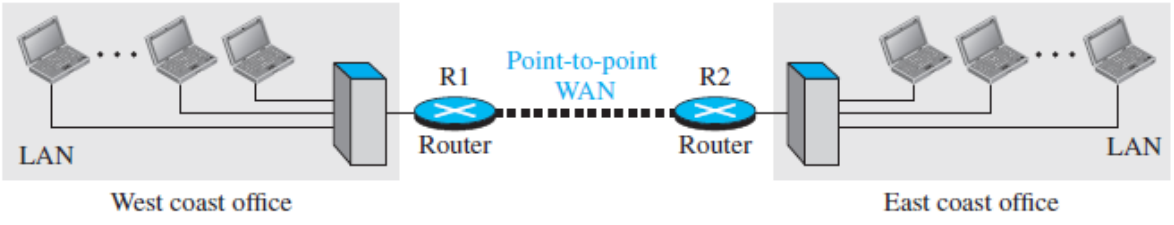
Fig. A stream of packets from sender to receiver.

In this figure, all the packets follow the route ACE, rather than ABDE or ACDE. In some networks all packets from a given message must follow the same route; in others each acked is routed separately. Of course, if ACE is the best route, all packets may be sent along it, even if each packet is individually routed.

Internetwork

It is very rare to see a LAN or a WAN in isolation; they are connected to one another. When two or more networks are connected, they make an internetwork, or internet. As an example, assume that an organization has two offices, one on the east coast and the other on the west coast. Each office has a LAN that allows all employees in the office to communicate with each other. To make the communication between employees at different offices possible, the management leases a point-to-point dedicated WAN from a service provider, such as a telephone company, and connects the two LANs. Now the company has an internetwork, or a private internet (with lowercase i). Communication between offices is now possible.

Figure 1.11 *An internetwork made of two LANs and one point-to-point WAN*



When a host in the west coast office sends a message to another host in the same office, the router blocks the message, but the switch directs the message to the destination. On the other hand, when a host on the west coast sends a message to a host on the east coast, router R1 routes the packet to router R2, and the packet reaches the destination.

Switching

An internet is a switched network in which a switch connects at least two links together. A switch needs to forward data from a network to another network when required.

The two most common types of switched networks are

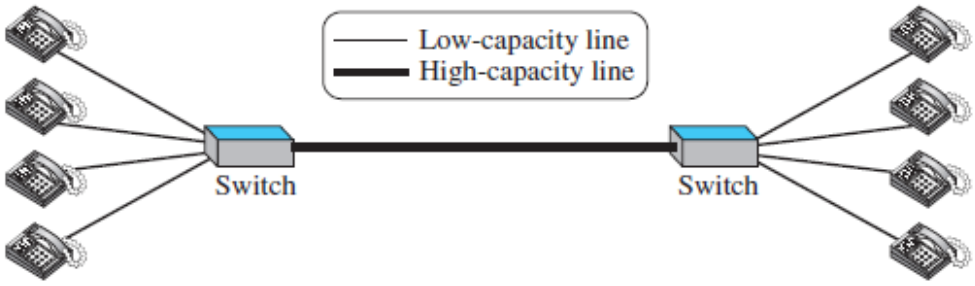
- **Circuit-switched and**
- **Packet-switched networks.**

Circuit-Switched Network

In a circuit-switched network, a dedicated connection, called a circuit, is always available between the two end systems; the switch can only make it active or inactive. Figure 1.13 shows a very simple switched network that connects four telephones to each end. We have used telephone sets instead of computers as an end system because circuit switching was very common in telephone networks in the past, although part of the telephone network today is a packet-switched network.

In Figure 1.13, the four telephones at each side are connected to a switch. The switch connects a telephone set at one side to a telephone set at the other side. The thick line connecting two switches is a high-capacity communication line that can handle four voice communications at the same time; the capacity can be shared between all pairs of telephone sets. The switches used in this example have forwarding tasks but no storing capability.

Figure 1.13 *A circuit-switched network*



Packet-Switched Network

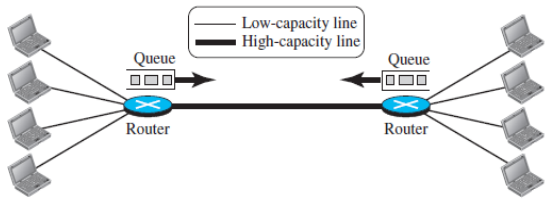
In a computer network, the communication between the two ends is done in blocks of data called packets. Instead of the continuous communication we see between two telephone sets when they are being used, we see the exchange of individual data packets between the two computers. This allows us to make the switches function for both storing and forwarding because a packet is an

independent entity that can be stored and sent later. Figure 1.14 shows a small packet-switched network that connects four computers at one site to four computers at the other site.

A router in a packet-switched network has a queue that can store and forward the packet. Now assume that the capacity of the thick line is only twice the capacity of the data line connecting the computers to the routers. If only two computers (one at each site) need to communicate with each other, there is no waiting for the packets.

However, if packets arrive at one router when the thick line is already working at its full capacity, the packets should be stored and forwarded in the order they arrived. The two simple examples show that a packet-switched network is more efficient than a circuits witched network, but the packets may encounter some delays.

Figure 1.14 *A packet-switched network*



The Internet

An internet (note the lowercase i) is two or more networks that can communicate with each other. The most notable internet is called the Internet (uppercase I), and is composed of thousands of interconnected networks.

The Internet is a global, interconnected computer network in which every computer connected and it can exchange data with any other connected computer.

Significance of an Internet are as follows:

It's the first mass medium that involves computers and uses digitized data.

It provides the potential for media convergence, the unification of all media.

It's transforming how we communicate, obtain information, learn, seek jobs, and maintain professional growth.

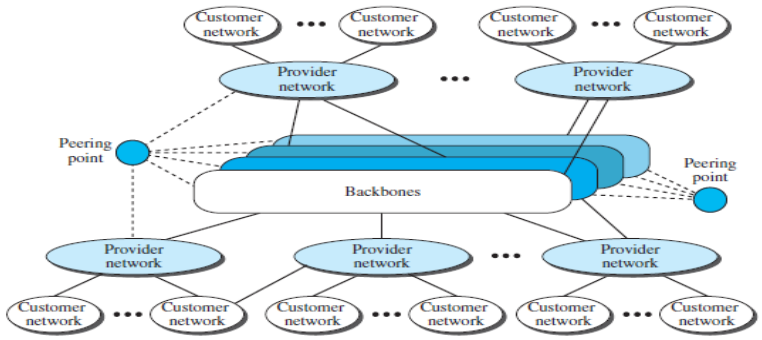
Businesses find it an indispensable tool for their needs.

The figure shows the Internet as several backbones, provider networks, and customer networks. At the top level, the backbones are large networks owned by some communication companies such as Sprint, Verizon (MCI), AT&T, and NTT. The backbone networks are connected through some complex switching systems, called peering points. At the second level, there are smaller networks, called provider networks that use the services of the backbones for a fee. The provider networks are connected to backbones and sometimes to other provider networks. The customer

networks are networks at the edge of the Internet that actually use the services provided by the Internet.

They pay fees to provider networks for receiving services. Backbones and provider networks are also called Internet Service Providers (ISPs). The backbones are often referred to as international ISPs. The provider networks are often referred to as national or regional ISPs.

Figure 1.15 *The Internet today*



Accessing the Internet

The Internet today is an internetwork that allows any user to become part of it. The user, however, needs to be physically connected to an ISP. The physical connection is normally done through a point-to-point WAN.

Using Telephone Networks

Today most residences and small businesses have telephone service, which means they are connected to a telephone network. Since most telephone networks have already connected themselves to the Internet, one option for residences and small businesses to connect to the Internet is to change the voice line between the residence or business and the telephone center to a point-to-point WAN. This can be done in two ways.

Dial-up service: The first solution is to add to the telephone line a modem that converts data to voice. The software installed on the computer dials the ISP and imitates making a telephone connection. Unfortunately, the dial-up service is very slow, and when the line is used for Internet connection, it cannot be used for telephone (voice) connection. It is only useful for small residences.

DSL Service: some telephone companies have upgraded their telephone lines to provide higher speed Internet services to residences or small businesses. The DSL service also allows the line to be used simultaneously for voice and data communication.

Using Cable Networks

More and more residents over the last two decades have begun using cable TV services instead of antennas to receive TV broadcasting. The cable companies have been upgrading their cable networks and connecting to the Internet. A residence or a small business can be connected to the

Internet by using this service. It provides a higher speed connection, but the speed varies depending on the number of neighbors that use the same cable.

Using Wireless Networks

Wireless connectivity has recently become increasingly popular. A household or a small business can use a combination of wireless and wired connections to access the Internet. With the growing wireless WAN access, a household or a small business can be connected to the Internet through a wireless WAN.

Direct Connection to the Internet

A large organization or a large corporation can itself become a local ISP and be connected to the Internet. This can be done if the organization or the corporation leases a high-speed WAN from a carrier provider and connects itself to a regional ISP. For example, a large university with several campuses can create an internetwork and then connect the internetwork to the Internet.

INTERNET HISTORY

An overview of the Internet, let us give a brief history of the Internet. This brief history makes it clear how the Internet has evolved from a private network to a global one in less than 40 years.

Early History

There were some communication networks, such as telegraph and telephone networks, before 1960. These networks were suitable for constant-rate communication at that time, which means that after a connection was made between two users, the encoded message (telegraphy) or voice (telephony) could be exchanged. A computer network, on the other hand, should be able to handle bursty data, which means data received at variable rates at different times. The world needed to wait for the packet-switched network to be invented.

Birth of Packet-Switched Networks

The theory of packet switching for bursty traffic was first presented by Leonard Kleinrock in **1961 at MIT**. At the same time, two other researchers, Paul Baran at Rand Institute and Donald Davies at National Physical Laboratory in England, published some papers about packet-switched networks.

ARPANET

In the mid-1960s, mainframe computers in research organizations were stand-alone devices. Computers from different manufacturers were unable to communicate with one another. The **Advanced Research Projects Agency** (**ARPA)** in the **Department of Defense (DOD)** was interested in finding a way to connect computers so that the researchers they funded could share their findings, thereby reducing costs and eliminating duplication of effort.

In 1967, at an **Association for Computing Machinery (ACM)** meeting, ARPA presented its ideas for the **Advanced Research Projects Agency Network (ARPANET),** a small network of connected computers. The idea was that each host computer (not necessarily from the same manufacturer) would be attached to a specialized computer, called an **interface message processor**

(IMP). The IMPs, in turn, would be connected to each other. Each IMP had to be able to communicate with other IMPs as well as with its own attached host.

By 1969, ARPANET was a reality. Four nodes, at the **University of California at Los Angeles (UCLA),** the **University of California at Santa Barbara (UCSB), Stanford Research Institute (SRI),** and the University of Utah, were connected via the IMPs to form a network. Software called the Network Control Protocol (NCP) provided communication between the hosts.

Birth of the Internet

In 1972, Vint Cerf and Bob Kahn, both of whom were part of the core ARPANET group, collaborated on what they called the Internetting Project. They wanted to link dissimilar networks so that a host on one network could communicate with a host on another. There were many problems to overcome: diverse packet sizes, diverse interfaces, and diverse transmission rates, as well as differing reliability requirements. Cerf and Kahn devised the idea of a device called a gateway to serve as the intermediary hardware to transfer data from one network to another.

TCP/IP

Cerf and Kahn's landmark 1973 paper outlined the protocols to achieve end-to-end delivery of data. This was a new version of NCP. This paper on transmission control protocol (TCP) included concepts such as encapsulation, the datagram, and the functions of a gateway. A radical idea was the transfer of responsibility for error correction from the IMP to the host machine. This ARPA Internet now became the focus of the communication effort. Around this time, responsibility for the ARPANET was handed over to the Defense Communication Agency (DCA).

In October 1977, an internet consisting of three different networks (ARPANET, packet radio, and packet satellite) was successfully demonstrated. Communication between networks was now possible.

Shortly thereafter, authorities made a decision to split TCP into two protocols: **Transmission Control Protocol (TCP) and Internet Protocol (IP)**. IP would handle datagram routing while TCP would be responsible for higher level functions such as segmentation,
reassembly, and error detection. The new combination became known as TCP/IP.

In 1983, authorities abolished the original ARPANET protocols, and TCP/IP became the official protocol for the ARPANET. Those who wanted to use the Internet to access a computer on a different network had to be running TCP/IP.

MILNET

In 1983, ARPANET split into two networks: Military Network (MILNET) for military users and ARPANET for nonmilitary users.

CSNET

Another milestone in Internet history was the creation of CSNET in 1981. Computer Science Network (CSNET) was a network sponsored by the National Science Foundation (NSF). The network was conceived by universities that were ineligible to join ARPANET due to an absence of ties to the

Department of Defense. CSNET was a less expensive network; there were no redundant links and the transmission rate was slower.

By the mid-1980s, most U.S. universities with computer science departments were part of CSNET. Other institutions and companies were also forming their own networks and using TCP/IP to interconnect. The term Internet, originally associated with government-funded connected networks, now referred to the connected networks using TCP/IP protocols.

World Wide Web

The 1990s saw the explosion of Internet applications due to the emergence of the World Wide Web (WWW). The Web was invented at CERN by Tim Berners-Lee. This invention has added the commercial applications to the Internet.

Multimedia

Recent developments in the multimedia applications such as voice over IP (telephony), video over IP (Skype), view sharing (YouTube), and television over IP (PPLive) has increased the number of users and the amount of time each user spends on the network.

Peer-to-Peer Applications

Peer-to-peer networking is also a new area of communication with a lot of potential.

STANDARDS AND ADMINISTRATION

The Internet and its protocol, we often see a reference to a standard or an administration entity.

Internet Standards

An Internet standard is a thoroughly tested specification that is useful to and adhered to by those who work with the Internet. It is a formalized regulation that must be followed. There is a strict procedure by which a specification attains Internet standard status. A specification begins as an Internet draft. An Internet draft is a working document (a work in progress) with no official status and a six-month lifetime. Upon recommendation from the Internet authorities, a draft may be published as a Request for Comment (RFC). Each RFC is edited, assigned a number, and made available to all interested parties. RFCs go through maturity levels and are categorized according to their requirement level.

Maturity Levels

An RFC, during its lifetime, falls into one of six maturity levels: proposed standard, draft standard, Internet standard, historic, experimental, and informational (see Figure 1.16).

- **Proposed Standard**. A proposed standard is a specification that is stable, well understood, and of sufficient interest to the Internet community. At this level, the specification is usually tested and implemented by several different groups.

❑ **Draft Standard.** A proposed standard is elevated to draft standard status after at least two successful independent and interoperable implementations. Barring difficulties, a draft standard, with modifications if specific problems are encountered, normally becomes an Internet standard.

❑ **Internet Standard.** A draft standard reaches Internet standard status after demonstrations of successful implementation.

❑ **Historic.** The historic RFCs are significant from a historical perspective. They either have been superseded by later specifications or have never passed the necessary maturity levels to become an Internet standard.

❑ **Experimental**. An RFC classified as experimental describes work related to an experimental situation that does not affect the operation of the Internet. Such an RFC should not be implemented in any functional Internet service.

❑ **Informational.** An RFC classified as informational contains general, historical, or tutorial information related to the Internet. It is usually written by someone in a non-Internet organization, such as a vendor.

Requirement Levels

RFCs are classified into five requirement levels: required, recommended, elective, limited use, and not recommended.

❑ **Required**. An RFC is labeled required if it must be implemented by all Internet systems to achieve minimum conformance. For example, IP and ICMP (Chapter 19) are required protocols.

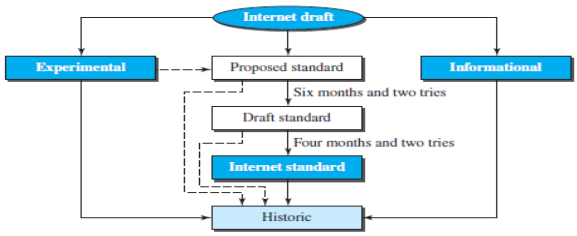
❑ **Recommended**. An RFC labeled recommended is not required for minimum conformance; it is recommended because of its usefulness. For example, FTP (Chapter 26) and TELNET (Chapter 26) are recommended protocols.

❑ **Elective**. An RFC labeled elective is not required and not recommended. However, a system can use it for its own benefit.

❑ **Limited Use.** An RFC labeled limited use should be used only in limited situations. Most of the experimental RFCs fall under this category.

❑ **Not Recommended**. An RFC labeled not recommended is inappropriate for general use. Normally a historic (deprecated) RFC may fall under this category.

Figure 1.16 *Maturity levels of an RFC*



Internet Administration

The Internet, with its roots primarily in the research domain, has evolved and gained a broader user base with significant commercial activity. Various groups that coordinate Internet issues have guided this growth and development. Appendix G gives the addresses, e-mail addresses, and telephone numbers for some of these groups. Figure 1.17 shows the general organization of Internet administration.

ISOC

The Internet Society (ISOC) is an international, nonprofit organization formed in 1992 to provide support for the Internet standards process. ISOC accomplishes this through maintaining and supporting other Internet administrative bodies such as IAB, IETF, IRTF, and IANA (see the following sections). ISOC also promotes research and other scholarly activities relating to the Internet.

IAB

The Internet Architecture Board (IAB) is the technical advisor to the ISOC. The main purposes of the IAB are to oversee the continuing development of the TCP/IP Protocol Suite and to serve in a technical advisory capacity to research members of the Internet community. IAB accomplishes this through its two primary components, the Internet Engineering Task Force (IETF) and the Internet Research Task Force (IRTF). Another responsibility of the IAB is the editorial management of the RFCs, described earlier. IAB is also the external liaison between the Internet and other standards organizations and forums.

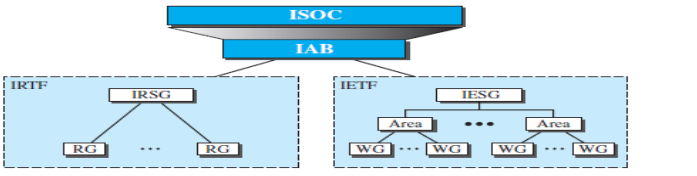
IETF

The Internet Engineering Task Force (IETF) is a forum of working groups managed by the Internet Engineering Steering Group (IESG). IETF is responsible for identifying operational problems and proposing solutions to these problems. IETF also develops and reviews specifications intended as Internet standards. The working groups are collected into areas, and each area concentrates on a specific topic. Currently nine areas have been defined. The areas include applications, protocols, routing, network management next generation (IPng), and security.

IRTF

The Internet Research Task Force (IRTF) is a forum of working groups managed by the Internet Research Steering Group (IRSG). IRTF focuses on long-term research topics related to Internet protocols, applications, architecture, and technology.

Figure 1.17 *Internet administration*



2.1 PROTOCOL LAYERING

In data communication and networking, a protocol defines the rules that both the sender and receiver and all intermediate devices need to follow to be able to communicate effectively. When communication is simple, we may need only one simple protocol. When the communication is complex, we may need to divide the task between different layers, in which case we need a protocol at each layer, or **protocol layering.**

Scenarios

Two simple scenarios to better understand the need for protocol layering.

First Scenario

In the first scenario, communication is so simple that it can occur in only one layer. Communication between first person and second person takes place in one layer, face to face. Even in this simple scenario, we can see that a set of rules needs to be followed.

First, Maria and Ann know that they should greet each other when they meet.

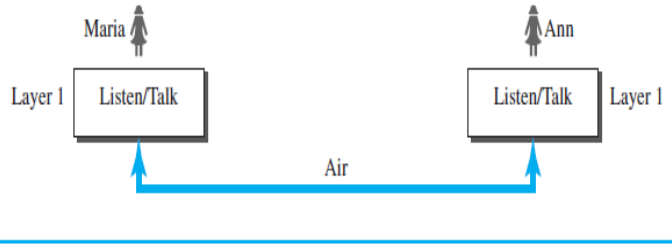
Second, they know that they should confine their vocabulary to the level of their friendship.

Third, each party knows that she should refrain from speaking when the other party is speaking.

Fourth, each party knows that the conversation should be a dialog, not a monolog: both should have the opportunity to talk about the issue.

Fifth, they should exchange some nice words when they leave.

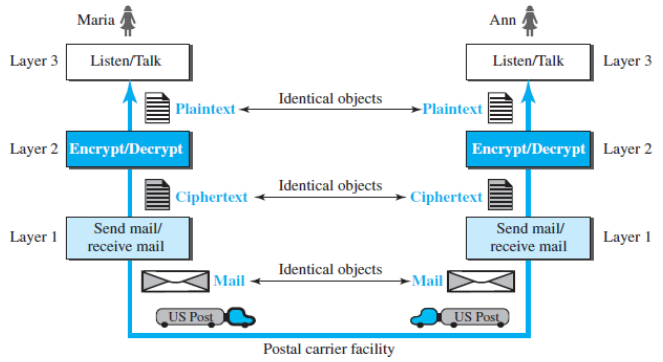
Figure 2.1 *A single-layer protocol*



Second Scenario

In the second scenario, we assume that Ann is offered a higher-level position in her company, but needs to move to another branch located in a city very far from Maria. The two friends want to continue their communication and exchange ideas because they have come up with an innovative project to start a new business when they both retire. They decide to continue their conversation using regular mail through the post office. They do not want their ideas to be revealed by other people if the letters are intercepted. They agree on an **encryption/decryption** technique. The sender of the letter encrypts it to make it unreadable by an intruder. The receiver of the letter decrypts it to get the original letter. We can say that the communication between Maria and Ann takes place in three layers.

Figure 2.2 *A three-layer protocol*



One of the advantages of protocol layering is that it allows us to separate the services from the implementation. A layer needs to be able to receive a set of services from the lower layer and to give the services to the upper layer

Another advantage of protocol layering, which cannot be seen in our simple examples but reveals itself when we discuss protocol layering in the Internet, is that communication does not always use only two end systems

Principles of Protocol Layering

Let us discuss two principles of protocol layering.

First Principle

The first principle dictates that if we want bidirectional communication, we need to make each layer so that it is able to perform two opposite tasks, one in each direction.

For example,

The third layer task is to listen (in one direction) and *talk* (in the other direction).

The second layer needs to be able to encrypt and decrypt.

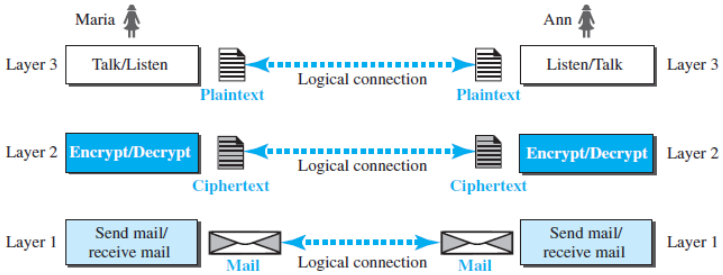
The first layer needs to send and receive mail.

Second Principle

The second principle that we need to follow in protocol layering is that the two objects under each layer at both sites should be identical. The object under layer 3 at both sites should be a plaintext letter. The object under layer 2 at both sites should be a ciphertext letter. The object under layer 1 at both sites should be a piece of mail.

Logical Connections

The two principles, we can think about logical connection between each layer as shown in Figure. This means that we have layer-to-layer communication. Maria and Ann can think that there is a logical (imaginary) connection at each layer through which they can send the object created from that layer.



2.2 TCP/IP PROTOCOL SUITE

The concept of protocol layering and the logical communication between layers. TCP/IP is a protocol suite (a set of protocols organized in different layers) used in the Internet today. It is a hierarchical protocol made up of interactive modules, each of which provides a specific functionality. The term ***hierarchical*** means that each upper level protocol is supported by the services provided by one or more lower level protocols. The original TCP/IP protocol suite was defined as four software layers built upon the hardware.

Layered Architecture

TCP/IP protocol suite are involved in communication between two hosts, we assume that we want to use the suite in a small internet made up of three LANs (links), each with a link-layer switch.

Figure 2.4 *Layers in the TCP/IP protocol suite*

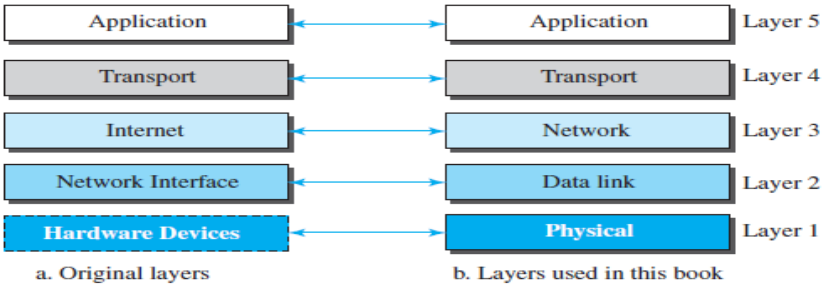
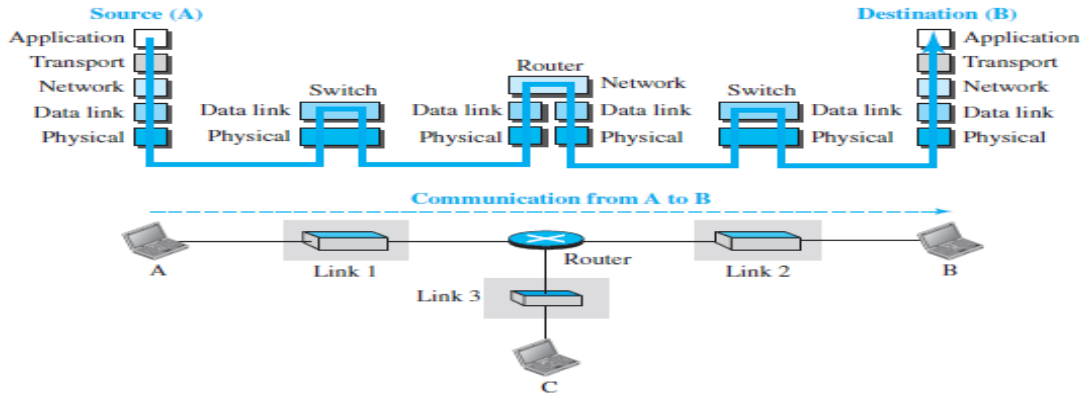


Figure 2.5 *Communication through an internet*



Let us assume that computer A communicates with computer B. As the figure shows, we have five communicating devices in this communication:

1. ***source host (computer A),***
2. ***the link-layer switch in link 1,***
3. ***the router,***
4. ***the link-layer switch in link 2, and***
5. ***the destination host (computer B).***

Each device is involved with a set of layers depending on the role of the device in the internet. The two hosts are involved in all five layers.

The **source host** needs to create a message in the application layer and send it down the layers so that it is physically sent to the destination host.

The destination host needs to receive the communication at the physical layer and then deliver it through the other layers to the application layer.

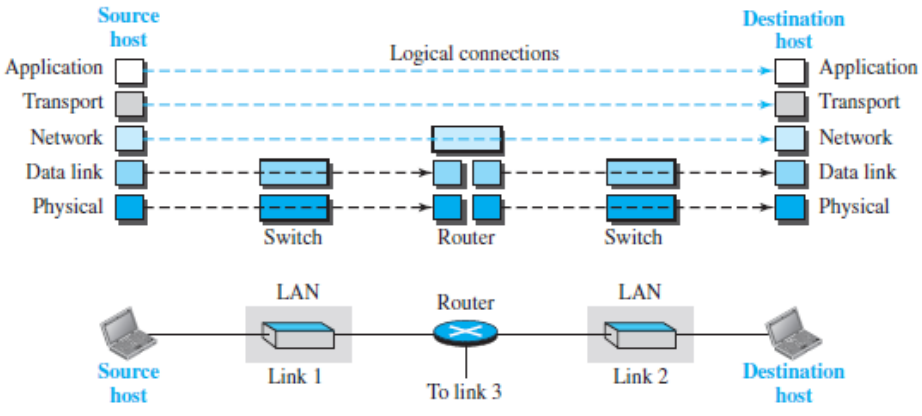
The **router** is involved in only three layers; there is no transport or application layer in a router as long as the router is used only for routing

A **router** is always involved in one network layer, it is involved in n combinations of link and physical layers in which n is the number of links the router is connected to. The reason is that each link may use its own data-link or physical protocol. The **router** is involved in only three layers there is no transport or application layer in a router as long as the router is used only for routing.

Layers in the TCP/IP Protocol Suite

The functions and duties of layers in the TCP/IP protocol suite. To better understand the duties of each layer, we need to think about the logical connections between layers. Figure 2.6 shows logical connections in our simple internet.

Figure 2.6 *Logical connections between layers of the TCP/IP protocol suite*



Using logical connections makes it easier for us to think about the duty of each layer. As the figure shows, the duty of the application, transport, and network layers is end-to-end. However, the duty of the data-link and physical layers is hop-to-hop, in which a hop is a host or router.

Description of Each Layer

The Physical Layer:

Function: Defines electrical and mechanical standards. deals with timing interfaces.

The physical layer is concerned with transmitting raw bits over a communication channel. The design issues have to do with making sure that when one side sends a 1 bit, it is received by the other side as a 1 bit, not as a 0 bit.

The Data Link Layer:

Function: Framing error detection and flow control.

The main task of the data link layer is to transform a raw transmission facility into a line that appears free of undetected transmission errors to the network layer. It accomplishes this task by having the sender break up the input data into data frames (typically a few hundred or a few thousand bytes) and transmits the frames sequentially. If the service is reliable, the receiver confirms correct receipt of each frame by sending back an acknowledgement frame. Another issue that arises in the data link layer (and most of the higher layers as well) is how to keep a fast transmitter from drowning a slow receiver in data. Some traffic regulation mechanism is often needed to let the transmitter know how much buffer space the receiver has at the moment. Frequently, this flow regulation and the error handling are integrated.

The Network Layer:

Function: Routing Qos and congestion control.

The network layer controls the operation of the subnet. A key design issue is determining how packets are routed from source to destination. Routes can be based on static tables that are "wired into" the network and rarely changed. They can also be determined at the start of each conversation, for example, a terminal session (e.g., a login to a remote machine). Finally, they can be highly dynamic, being determined anew for each packet, to reflect the current network load. If too many packets are present in the subnet at the same time, they will get in one another's way, forming bottlenecks. The control of such congestion also belongs to the network layer. More generally, the quality of service provided (delay, transit time, jitter, etc.) is also a network layer issue. When a packet has to travel from one network to another to get to its destination, many problems can arise. The addressing used by the second network may be different from the first one. The second one may not accept the packet at all because it is too large. The protocols may differ, and so on. It is up to the network layer to overcome all these problems to allow heterogeneous networks to be interconnected. In broadcast networks, the routing problem is simple, so the network layer is often thin or even nonexistent.

The Transport Layer:

Function: Connection management, flow control and error flow.

The basic function of the transport layer is to accept data from above, split it up into smaller units if need be, pass these to the network layer, and ensure that the pieces all arrive correctly at the

other end. Furthermore, all this must be done efficiently and in a way that isolates the upper layers from the inevitable changes in the hardware technology. The transport layer also determines what type of service to provide to the session layer, and, ultimately, to the users of the network. The most popular type of transport connection is an error-free point-to-point channel that delivers messages or bytes in the order in which they were sent. However, other possible kinds of transport service are the transporting of isolated messages, with no guarantee about the order of delivery, and the broadcasting of messages to multiple destinations. The type of service is determined when the connection is established.

The transport layer is a true end-to-end layer, all the way from the source to the destination. In other words, a program on the source machine carries on a conversation with a similar program on the destination machine, using the message headers and control messages. In the lower layers, the protocols are between each machine and its immediate neighbors, and not between the ultimate source and destination machines, which may be separated by many routers.

The Session Layer:

Function: Session management, token management, dialog control and synchronization.

The session layer allows users on different machines to establish sessions between them. Sessions offer various services, including dialog control (keeping track of whose turn it is to transmit), token management (preventing two parties from attempting the same critical operation at the same time), and synchronization (check pointing long transmissions to allow them to continue from where they were after a crash).

The Presentation Layer:

Function: Encoding and decoding,encryption and decryption ,compression and decompression

The presentation layer is concerned with the syntax and semantics of the information transmitted. In order to make it possible for computers with different data representations to communicate, the data structures to be exchanged can be defined in an abstract way, along with a standard encoding to be used "on the wire." The presentation layer manages these abstract data structures and allows higher-level data structures (e.g., banking records), to be defined and exchanged.

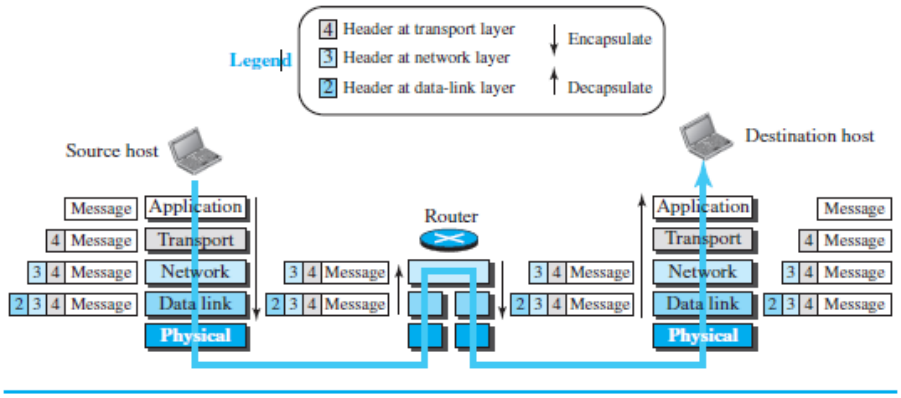
The Application Layer:

The application layer contains a variety of protocols that are commonly needed by users. One widely-used application protocol is HTTP (Hypertext Transfer Protocol), which is the basis for the World Wide Web. When a browser wants a Web page, it sends the name of the page it wants to the server using HTTP. The server then sends the page back. Other application protocols are used for file transfer, electronic mail, and network news.

Encapsulation and Decapsulation

One of the important concepts in protocol layering in the Internet is encapsulation/decapsulation. Figure 2.8 shows this concept for the small internet

Figure 2.8 *Encapsulation/Decapsulation*



Encapsulation at the Source Host

At the source, we have only encapsulation.

1. At the application layer, the data to be exchanged is referred to as a *message*. A message normally does not contain any header or trailer. The message is passed to the transport layer.

2. The transport layer takes the message as the payload, the load that the transport layer should take care of. It adds the transport layer header to the payload, which contains the identifiers of the source and destination application programs that want to communicate plus some more information that is needed for the end-to-end delivery of the message, such as information needed for **flow, error control, or congestion control.** The result is the transport-layer packet, which is called the *segment* (in TCP) and the *user datagram* (in UDP). The transport layer then passes the packet to the network layer.

3. The network layer takes the transport-layer packet as data or payload and adds its own header to the payload. The header contains the addresses of the source and destination hosts and some more information used for **error checking of the header, fragmentation information**, and so on. The result is the network-layer packet, called a ***datagram***. The network layer then passes the packet to the data-link layer.

4. The data-link layer takes the network-layer packet as data or payload and adds its own header, which contains the link-layer addresses of the host or the next hop (the router). The result is the link-layer packet, which is called a *frame*. The frame is passed to the physical layer for transmission.

Decapsulation and Encapsulation at the Router

At the router, we have both decapsulation and encapsulation because the router is connected to two or more links.

1. After the set of bits are delivered to the data-link layer, this layer decapsulates the datagram from the frame and passes it to the network layer.

2. The network layer only inspects the source and destination addresses in the datagram header and consults its forwarding table to find the next hop to which the datagram is to be delivered. The contents of the datagram should not be changed by the network layer in the router unless there is a need to fragment the datagram if it is too big to be passed through the next link. The datagram is then passed to the data-link layer of the next link.

3. The data-link layer of the next link encapsulates the datagram in a frame and passes it to the physical layer for transmission.

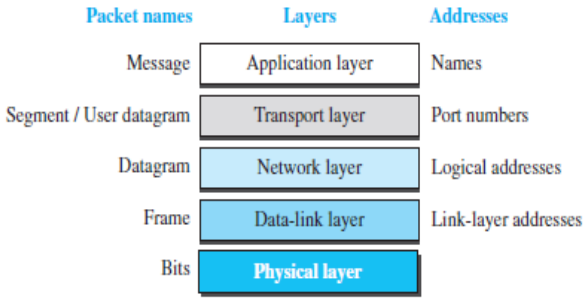
Addressing

Any communication that involves two parties needs two addresses:

1. ***source address and***
2. ***Destination address.***

TO communicate from one system to another system we need five pairs of addresses, one pair per layer, we normally have only four because the physical layer does not need addresses; the unit of data exchange at the physical layer is a bit, which definitely cannot have an address

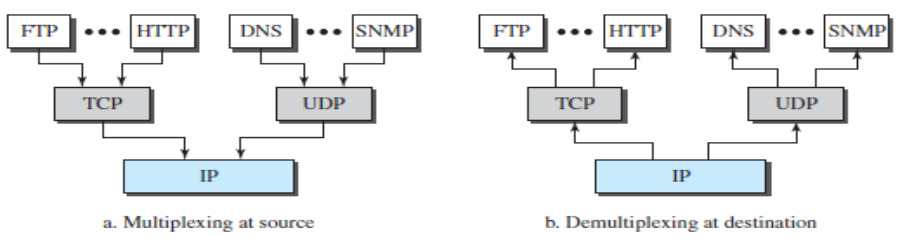
Figure 2.9 *Addressing in the TCP/IP protocol suite*



Multiplexing and Demultiplexing

Since the TCP/IP protocol suite uses several protocols at some layers, we can say that we have multiplexing at the source and demultiplexing at the destination. Multiplexing in this case means that a protocol at a layer can encapsulate a packet from several next-higher layer protocols (one at a time). Demultiplexing means that a protocol can decapsulate and deliver a packet to several next-higher layer protocols (one at a time). Figure 2.10 shows the concept of multiplexing and demultiplexing at the three upper layers.

Figure 2.10 *Multiplexing and demultiplexing*



At the transport layer, either UDP or TCP can accept a message from several application-layer protocols.

At the network layer, IP can accept a segment from TCP or a user datagram from UDP. IP can also accept a packet from other protocols such as ICMP, IGMP, and so on.

At the data-link layer, a frame may carry the payload coming from IP or other protocols such as ARP.

2.3 The OSI Reference Model:

The OSI model (minus the physical medium) is shown in Fig. This model is based on a proposal developed by the International Standards Organization (ISO) as a first step toward international standardization of the protocols used in the various layers (Day and Zimmermann, 1983). It was revised in 1995(Day, 1995). The model is called the ISO-OSI (Open Systems Interconnection) Reference Model because it deals with connecting open systems—that is, systems that are open for communication with other systems. The OSI model has 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.

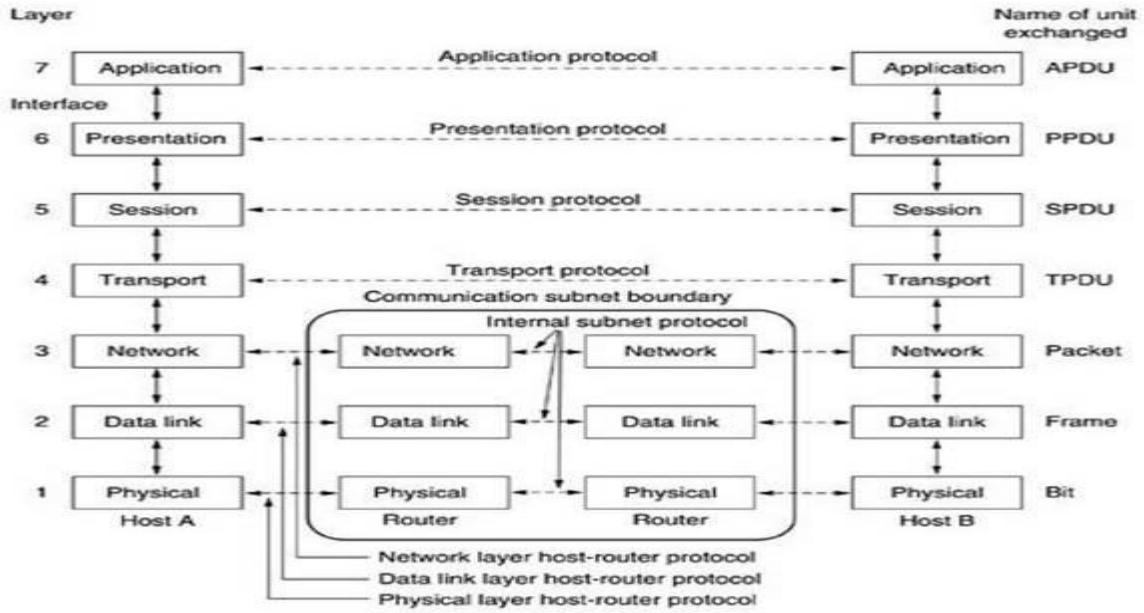
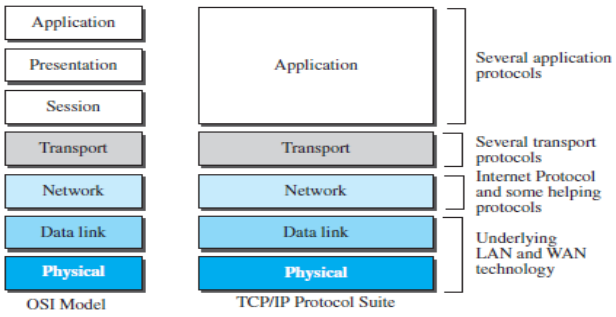


Fig. The OSI reference model.

OSI versus TCP/IP

When we compare the two models, we find that two layers, session and presentation, are missing from the TCP/IP protocol suite. These two layers were not added to the TCP/IP protocol suite after the publication of the OSI model. The application layer in the suite is usually considered to be the combination of three layers in the OSI model, as shown in Figure 2.12.

Figure 2.12 *TCP/IP and OSI model*



Two reasons were mentioned for this decision.

First, TCP/IP has more than one transport-layer protocol. Some of the functionalities of the session layer are available in some of the transport-layer protocols.

Second, the application layer is not only one piece of software. Many applications can be developed at this layer. If some of the functionalities mentioned in the session and presentation layers are needed for a particular application, they can be included in the development of that piece of software.

Problems of the TCP/IP Reference Mode:

First, the model does not clearly distinguish the concepts of service, interface, and protocol. Good software engineering practice requires differentiating between the specification and the implementation, something that OSI does very carefully, and TCP/IP does not. Consequently, the TCP/IP model is not much of a guide for designing new networks using new technologies.

Second, the TCP/IP model is not at all general and is poorly suited to describing any protocol stack other than TCP/IP. Trying to use the TCP/IP model to describe Bluetooth, for example, is completely impossible.

Third, the host-to-network layer is not really a layer at all in the normal sense of the term as used in the context of layered protocols. It is an interface (between the network and data link layers). The distinction between an interface and a layer is crucial, and one should not be sloppy about it.

Fourth, the TCP/IP model does not distinguish (or even mention) the physical and data link layers. These are completely different. The physical layer has to do with the transmission characteristics of copper wire, fiber optics, and wireless communication. The data link layer's job is to delimit the start and end of frames and get them from one side to the other with the desired degree of reliability. A proper model should include both as separate layers. The TCP/IP model does not do this.

DATA AND SIGNALS

Online service provides communications between two persons in five different levels.

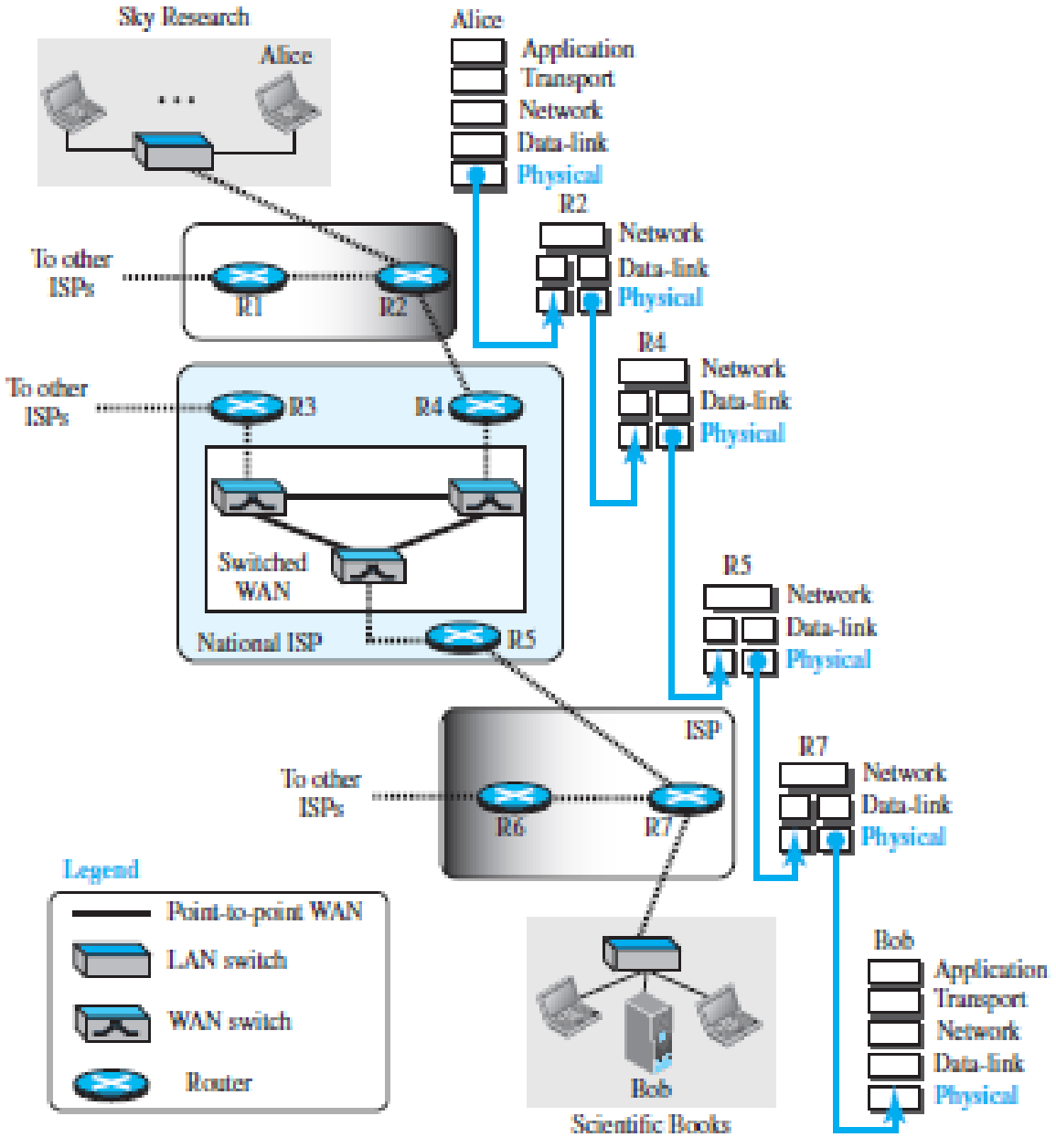
The Communication at application, transport, network, or data-link is ***logical***;

The communication at the physical layer is ***physical.***

The below picture represents host-to-router, router-to-router, and router-to-host, but the switches are also involved in the physical communication.

Although Alice and Bob need to exchange *data*, communication at the physical layer means exchanging *signals*. Data need to be transmitted and received, but the media have to change data to signals. Both data and the signals that represent them can be either **analog** or **digital** in form.

Figure 3.1 *Communication at the physical layer*



Analog and Digital Data

To communicated systems Data can be two types

1. Analog or
2. Digital.

Analog data: A**nalog data** refers to information that is continuous.

For example,

An analog clock that has hour, minute, and second hands gives information in a continuous form; the movements of the hands are continuous. Analog data, such as the sounds made by a human voice, take on continuous values. When someone speaks, an analog wave is created in the air. This can be captured by a microphone and converted to an analog signal or sampled and converted to a digital signal.

Digital data: **Digital data** refers to information that has discrete states. A digital clock that reports the hours and the minutes will change suddenly from 8:05 to 8:06. Digital data take on discrete values.

For example,

Data are stored in computer memory in the form of 0s and 1s. They can be converted to a digital signal or modulated into an analog signal for transmission across a medium.

Analog and Digital Signals

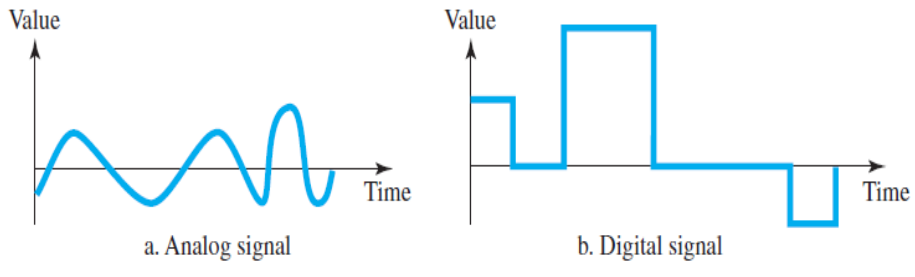
To communicated systems Signals can be two types. **signals** can be either **analog or digital**.

An **analog signal** has infinitely many levels of intensity over a period of time. As the wave moves from value *A* to value *B*, it passes through and includes an infinite number of values along its path.

A **digital signal,** on the other hand, can have only a limited number of defined values. Although each value can be any number, it is often as simple as 1 and 0. The simplest way to show signals is by plotting them on a pair of perpendicular axes. The vertical axis represents the value or strength of a signal. The horizontal axis represents time.

Figure 3.2 illustrates an analog signal and a digital signal. The curve representing the **analog signal** passes through an infinite number of points. The vertical lines of the **digital signal**, however, demonstrate the sudden jump that the signal makes from value to value.

Figure 3.2 *Comparison of analog and digital signals*



a. Analog signal b. Digital signal

Periodic and Non-periodic

Both analog and digital signals can take one of two forms: *periodic* or *nonperiodic*

A **periodic signal** completes a pattern within a measurable time frame, called a **period,** and repeats that pattern over subsequent identical periods. The completion of one full pattern is called a **cycle.**

A **non-periodic signal** changes without exhibiting a pattern or cycle that repeats over time.

Both analog and digital signals can be periodic or non-periodic.

In data communications, we commonly use periodic analog signals and non-periodic digital signals.

PERIODIC ANALOG SIGNALS

Periodic analog signals can be classified as **simple** or **composite**.

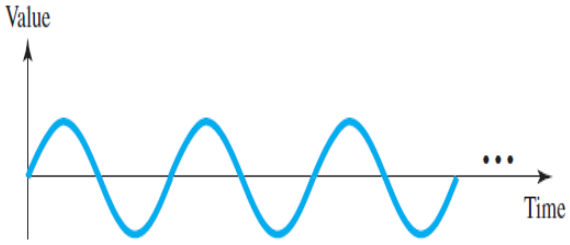
A **simple** periodic analog signal, a **sine wave,** cannot be decomposed into simpler signals.

A **composite** periodic analog signal is composed of multiple sine waves.

Sine Wave

The sine wave is the most fundamental form of a periodic analog signal. When we visualize it as a simple oscillating curve, its change over the course of a cycle is smooth and consistent, a continuous, rolling flow. Figure 3.3 shows a sine wave. Each cycle consists of a single arc above the time axis followed by a single arc below it.

Figure 3.3 *A sine wave*



A sine wave can be represented by three parameters:

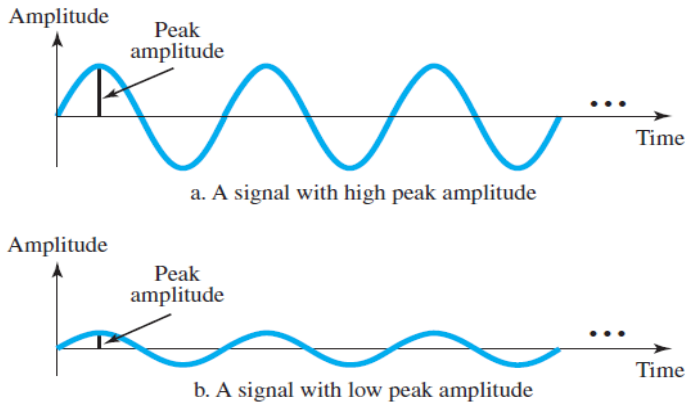
1. The *peak amplitude*,
2. The *frequency* ,and
3. The *phase*.

These three parameters fully describe a sine wave.

Peak Amplitude

The **peak amplitude** of a signal is the absolute value of its highest intensity, proportional to the energy it carries. For electric signals, peak amplitude is normally measured in *volts*. Figure 3.4 shows two signals and their peak amplitudes.

Figure 3.4 *Two signals with the same phase and frequency, but different amplitudes*



Period and Frequency

Period refers to the amount of time, in seconds, a signal needs to complete 1 cycle.

Frequency refers to the number of periods in 1 s.

Note that period and frequency are just one characteristic defined in two ways. Period is the inverse of frequency, and frequency is the inverse of period, as the following formulas show.

$$f = \frac{1}{T} \quad \text{and} \quad T = \frac{1}{f}$$

Figure 3.5 shows two signals and their frequencies. Period is formally expressed in seconds. Frequency is formally expressed in **Hertz (Hz),** which is cycle per second. Units of period and frequency are shown in Table 3.1.

Figure 3.5 *Two signals with the same amplitude and phase, but different frequencies*

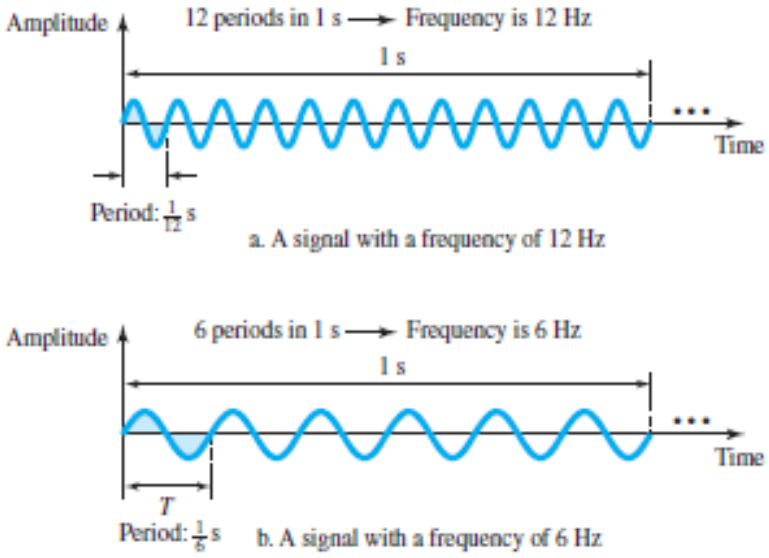


Table 3.1 *Units of period and frequency*

Period		Frequency	
Unit	*Equivalent*	*Unit*	*Equivalent*
Seconds (s)	1 s	Hertz (Hz)	1 Hz
Milliseconds (ms)	10^{-3} s	Kilohertz (kHz)	10^{3} Hz
Microseconds (μs)	10^{-6} s	Megahertz (MHz)	10^{6} Hz
Nanoseconds (ns)	10^{-9} s	Gigahertz (GHz)	10^{9} Hz
Picoseconds (ps)	10^{-12} s	Terahertz (THz)	10^{12} Hz

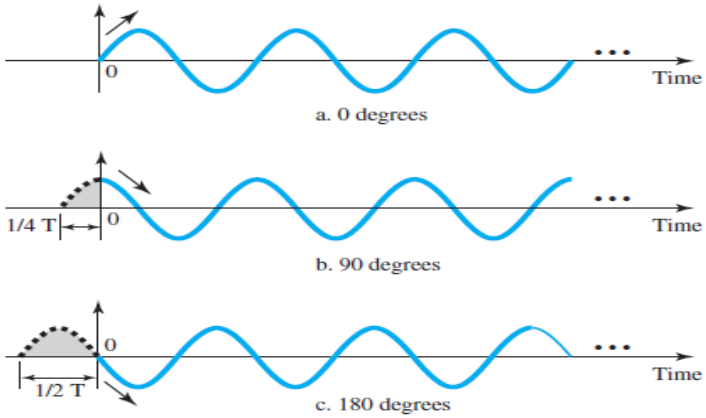
Phase

The term **phase,** or phase shift, describes the position of the waveform relative to time 0. If we think of the wave as something that can be shifted backward or forward along the time axis, phase describes the amount of that shift. It indicates the status of the first cycle.

Phase is measured in degrees or radians.

- A phase shift of 360º corresponds to a shift of a complete period.
- A phase shift of 180° corresponds to a shift of one-half of a period.
- A phase shift of 90º corresponds to a shift of one-quarter of a period.

Figure 3.6 *Three sine waves with the same amplitude and frequency, but different phases*



We can say that

a. A sine wave with a phase of 0° starts at time 0 with a zero amplitude. The amplitude is increasing.

b. A sine wave with a phase of 90° starts at time 0 with a peak amplitude. The amplitude is decreasing.

c. A sine wave with a phase of 180° starts at time 0 with a zero amplitude. The amplitude is decreasing.

Another way to look at the phase is in terms of shift or offset. We can say that

a. A sine wave with a phase of 0° is not shifted.

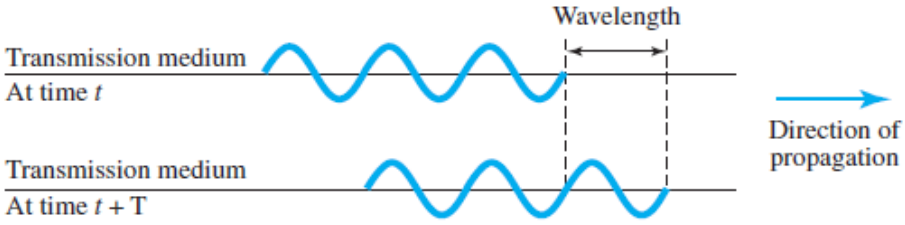
b. A sine wave with a phase of 90° is shifted to the left by ¼ cycles. However, note that the signal does not really exist before time 0.

c. A sine wave with a phase of 180° is shifted to the left by ½ cycles. However, note that the signal does not really exist before time 0.

Wavelength

Wavelength is another characteristic of a signal traveling through a transmission medium. Wavelength binds the period or the frequency of a simple sine wave to the **propagation speed** of the medium.

Figure 3.7 *Wavelength and period*



While the frequency of a signal is independent of the medium, the wavelength depends on both the frequency and the medium. Wavelength is a property of any type of signal. In data communications, we often use wavelength to describe the transmission of light in an optical fiber. The wavelength is the distance a simple signal can travel in one period.

Wavelength can be calculated if one is given the propagation speed (the speed of light) and the period of the signal. However, since period and frequency are related to each other, if we represent wavelength by λ, propagation speed by *c* (speed of light), and frequency by *f*, we get

$$\text{Wavelength} = (\text{propagation speed}) \times \text{period} = \frac{\text{propagation speed}}{\text{frequency}}$$

$$\lambda = \frac{c}{f}$$

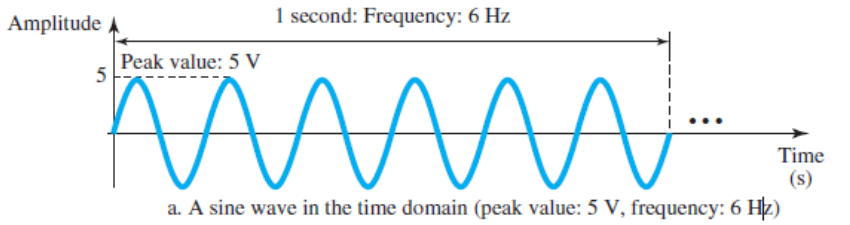
The propagation speed of electromagnetic signals depends on the medium and on the frequency of the signal.

Time and Frequency Domains

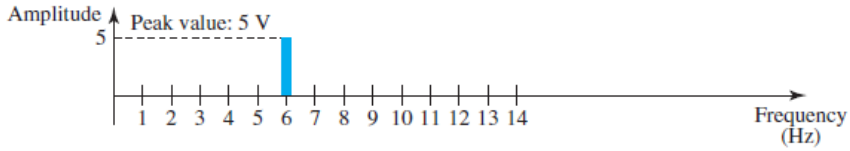
A sine wave is comprehensively defined by its amplitude, frequency, and phase. We have been showing a sine wave by using what is called a **time-domain** plot. The time-domain plot shows changes in signal amplitude with respect to time (it is an amplitude-versus-time plot). Phase is not explicitly shown on a **time-domain** plot.

To show the relationship between amplitude and frequency, we can use what is called a **frequency-domain** plot. A frequency-domain plot is concerned with only the peak value and the frequency. Changes of amplitude during one period are not shown. Figure 3.8 shows a signal in both the time and frequency domains.

Figure 3.8 *The time-domain and frequency-domain plots of a sine wave*



a. A sine wave in the time domain (peak value: 5 V, frequency: 6 Hz)



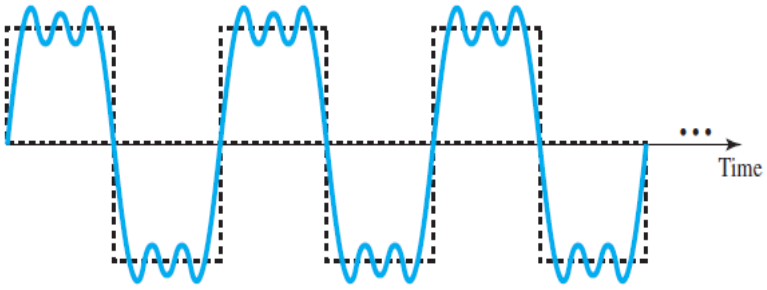
b. The same sine wave in the frequency domain (peak value: 5 V, frequency: 6 Hz)

Composite Signals

Simple sine waves have many applications in daily life. We can send a single sine wave to carry electric energy from one place to another.

Composite signal is made of many simple sine waves. A composite signal can be periodic or non-periodic. A periodic composite signal can be decomposed into a series of simple sine waves with discrete frequencies. Frequencies that have integer values (1, 2, 3, and so on). A non-periodic composite signal can be decomposed into a combination of an infinite number of simple sine waves with continuous frequencies, frequencies that have real values.

Figure 3.10 *A composite periodic signal*



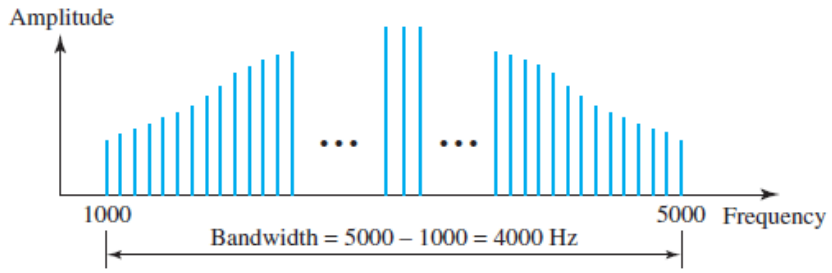
Bandwidth

The range of frequencies contained in a composite signal is its **bandwidth.** The bandwidth is normally a difference between two numbers.

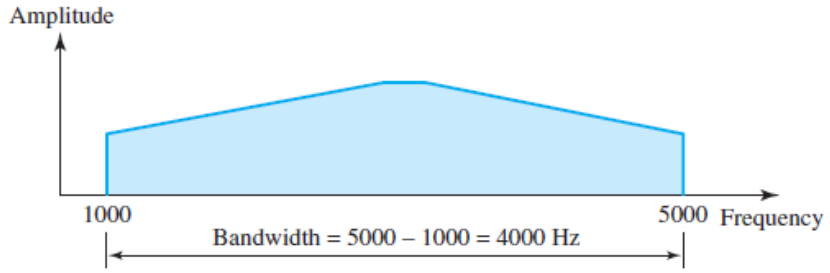
For example,

If a composite signal contains frequencies between 1000 and 5000, its bandwidth is 5000 − 1000, or 4000. Figure 3.13 shows the concept of bandwidth. The figure depicts two composite signals, one periodic and the other non-periodic. The bandwidth of the periodic signal contains all integer frequencies between 1000 and 5000 (1000, 1001, 1002 . . .). The bandwidth of the non-periodic signals has the same range, but the frequencies are continuous.

Figure 3.13 *The bandwidth of periodic and nonperiodic composite signals*



a. Bandwidth of a periodic signal

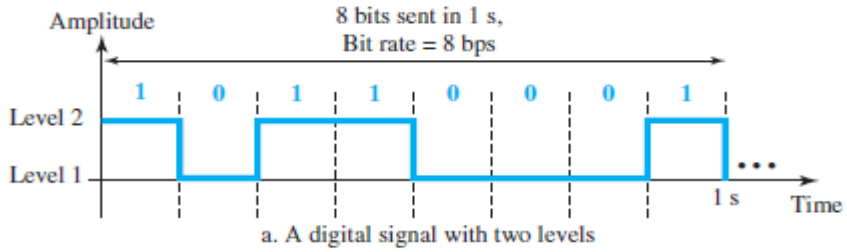


b. Bandwidth of a nonperiodic signal

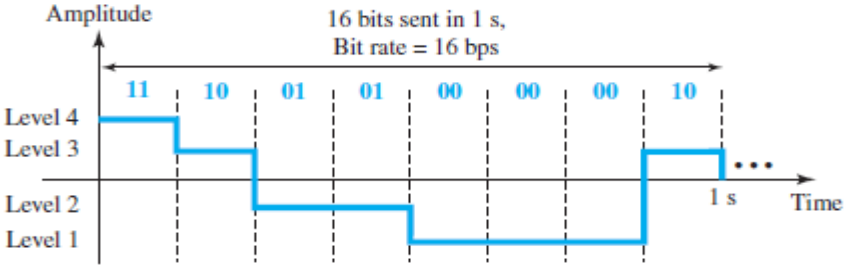
DIGITAL SIGNALS

An analog signal, information can also be represented by a digital signal. For example, a 1 can be encoded as a positive voltage and a 0 as zero voltage. A digital signal can have more than two levels. In this case, we can send more than 1 bit for each level. Figure 3.17 shows two signals, one with two levels and the other with four. We send 1 bit per level in part a of the figure and 2 bits per level in part b of the figure. In general, if a signal has *L* levels, each level needs log2 *L* bits. For this reason, we can send log24 = 2 bits in part b.

Figure 3.17 *Two digital signals: one with two signal levels and the other with four signal levels*



a. A digital signal with two levels



b. A digital signal with four levels

Bit Length

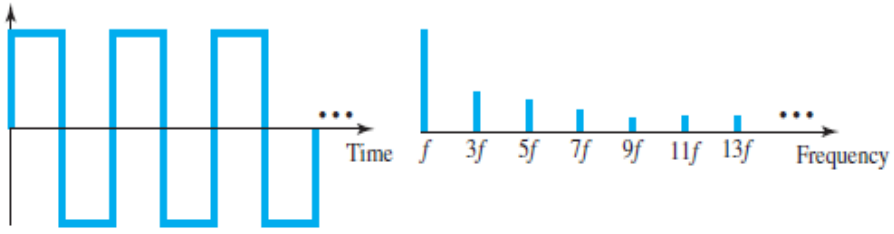
The concept of the wavelength for an **analog signal**: the distance one cycle occupies on the transmission medium. We can define something similar for a digital signal: the bit length. The **bit length** is the distance one bit occupies on the transmission medium.

Digital Signal as a Composite Analog Signal

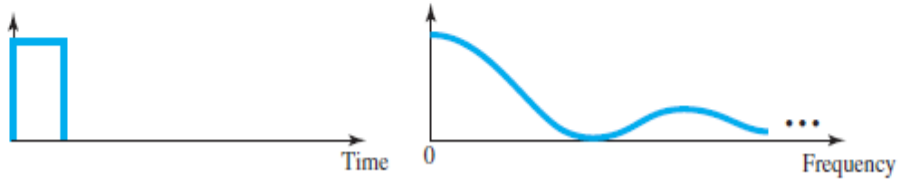
A digital signal is a **composite analog signal**. A digital signal, in the time domain, comprises connected **vertical** and **horizontal** line segments. A vertical line in the time domain means a frequency of infinity (sudden change in time). A horizontal line in the time domain means a frequency of zero (no change in time).

Fourier analysis can be used to decompose a **digital signal**. If the digital signal is **periodic**, which is rare in data communications, the decomposed signal has a frequency domain representation with an infinite bandwidth and discrete frequencies. If the digital signal is **non-periodic**, the decomposed signal still has an infinite bandwidth, but the frequencies are continuous. Figure 3.18 shows a periodic and a non-periodic digital signal and their bandwidths.

Figure 3.18 *The time and frequency domains of periodic and nonperiodic digital signals*



a. Time and frequency domains of periodic digital signal



b. Time and frequency domains of nonperiodic digital signal

Transmission of Digital Signals

Transmit of digital signals by using one of two different approaches:

- *Baseband transmission or*
- *Broadband transmission (using modulation).*

Baseband Transmission

Baseband transmission means sending a digital signal over a channel without changing the digital signal to an analog signal. Figure 3.19 shows **baseband** transmission.

Baseband transmission requires that we have a **low-pass channel,** a channel with a bandwidth that starts from zero. This is the case if we have a dedicated medium with a bandwidth constituting only one channel.

For example, the entire bandwidth of a cable connecting two computers is one single channel. As another example, we may connect several computers to a bus, but not allow more than two stations to communicate at a time. Again we have a low-pass channel, and we can use it for baseband communication.

Figure 3.20 shows two low-pass channels:

One with a **narrow** bandwidth and the other with a **wide** bandwidth. We need to remember that a low-pass channel with infinite bandwidth is ideal, but we cannot have such a channel in real life.

Figure 3.19 *Baseband transmission*

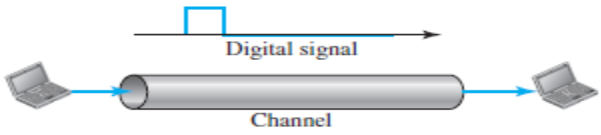
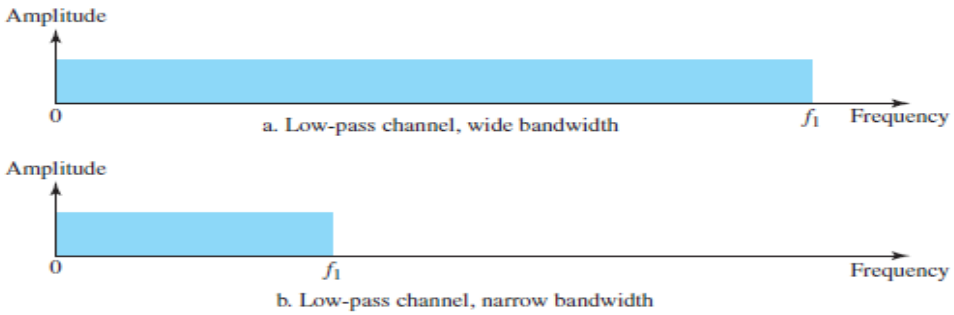


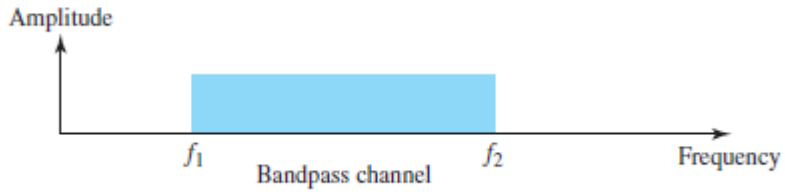
Figure 3.20 *Bandwidths of two low-pass channels*



Broadband Transmission (Using Modulation)

Broadband transmission or modulation means changing the digital signal to an analog signal for transmission. Modulation allows us to use a **band-pass channel**. A channel with a bandwidth that does not start from zero. This type of channel is more available than a low-pass channel. Figure 3.24 shows a band-pass channel

Figure 3.24 *Bandwidth of a bandpass channel*



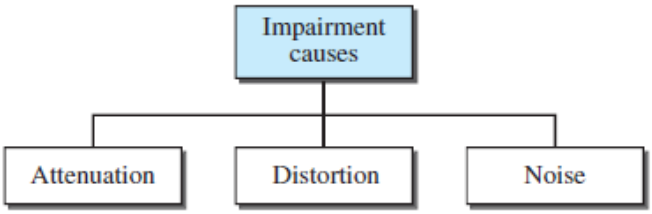
TRANSMISSION IMPAIRMENT

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

Three causes of impairment are

- ***attenuation,***
- ***distortion, and***
- ***noise.***

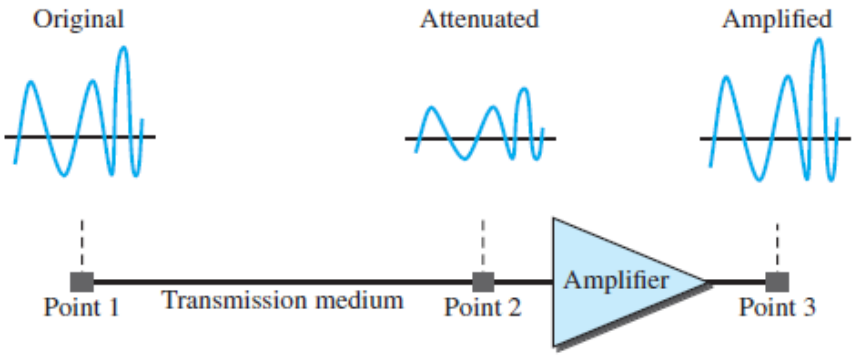
Figure 3.26 *Causes of impairment*



Attenuation

Attenuation means a loss of energy. When a signal travels through a medium, either simple or composite, it loses some of its energy in overcoming the resistance of the medium. That is why a wire carrying electric signals gets warm. Some of the electrical energy in the signal is converted to heat. To compensate for this loss, amplifiers are used to amplify the signal.

Figure 3.27 *Attenuation*



Decibel

The **decibel (dB)** measures the relative strengths of two signals or one signal at two different points. Note that the decibel is negative if a signal is attenuated and positive if a signal is amplified.

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

Variables $P1$ and $P2$ are the powers of a signal at points 1 and 2, respectively. In this case, because power is proportional to the square of the voltage, the formula is dB = 20 $\log_{10}$ ($V2/V1$). In this text, we express dB in terms of power.

Example 3.26

Suppose a signal travels through a transmission medium and its power is reduced to one-half. This means that $P_2 = \frac{1}{2} P_1$. In this case, the attenuation (loss of power) can be calculated as

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

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

Example 3.27

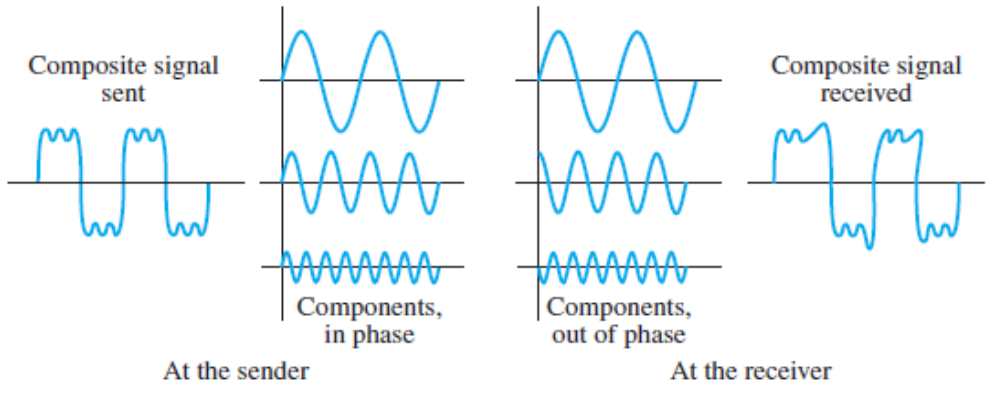
A signal travels through an amplifier, and its power is increased 10 times. This means that $P_2 = 10P_1$. In this case, the amplification (gain of power) can be calculated as

$$10 \log_{10} \frac{P_2}{P_1} = 10 \log_{10} \frac{10P_1}{P_1} = 10 \log_{10} 10 = 10(1) = 10 \text{ dB}$$

Distortion

Distortion means that the signal changes its form or shape. Distortion can occur in a composite signal made of different frequencies. Each signal component has its own propagation speed through a medium and, therefore, its own delay in arriving at the final destination. Differences in delay may create a difference in phase if the delay is not exactly the same as the period duration. In other words, signal components at the receiver have phases different from what they had at the sender. The shape of the composite signal is therefore not the same.

Figure 3.29 *Distortion*



Noise

Noise is another cause of impairment. Several types of noise, such as **thermal noise, induced noise, crosstalk, and impulse noise,** may corrupt the signal.

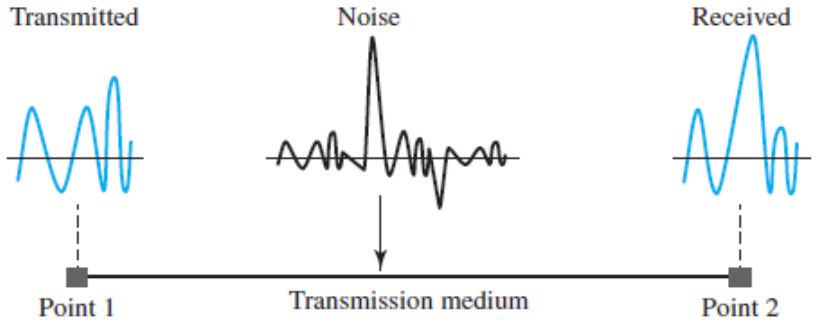
Thermal noise is the random motion of electrons in a wire, which creates an extra signal not originally sent by the transmitter.

Induced noise comes from sources such as motors and appliances. These devices act as a sending antenna, and the transmission medium acts as the receiving antenna.

Crosstalk is the effect of one wire on the other. One wire acts as a sending antenna and the other as the receiving antenna.

Impulse noise is a spike (a signal with high energy in a very short time) that comes from power lines, lightning, and so on. Figure 3.30 shows the effect of noise on a signal.

Figure 3.30 *Noise*



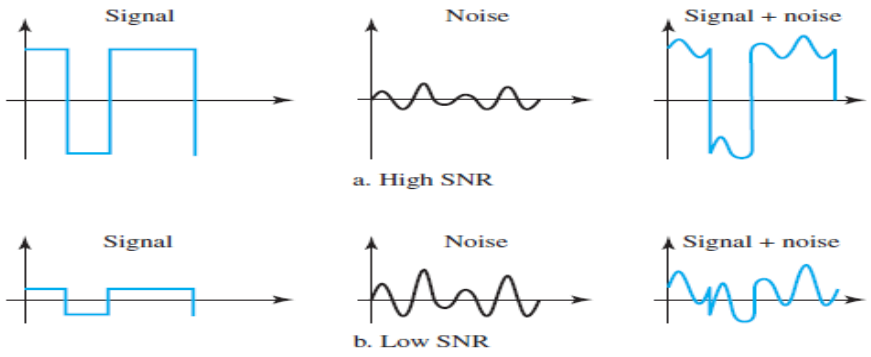
Signal-to-Noise Ratio (SNR)

To find the theoretical bit rate limit, we need to know the ratio of the signal power to the noise power. The **signal-to-noise ratio** is defined as

$$\text{SNR} = \frac{\text{average signal power}}{\text{average noise power}}$$

We need to consider the average signal power and the average noise power because these may change with time.

Figure 3.31 *Two cases of SNR: a high SNR and a low SNR*



SNR is actually the ratio of what is wanted (signal) to what is not wanted (noise). A **high SNR** means the signal is less corrupted by noise, a **low SNR** means the signal is more corrupted by noise. Because SNR is the ratio of two powers, it is often described in decibel units, SNRdB, defined as:

$$\text{SNR}_{\text{dB}} = 10 \log_{10} \text{SNR}$$

DATA RATE LIMITS

A very important consideration in data communications is how fast we can send data, in bits per second, over a channel. Data rate depends on three factors:

1. The bandwidth available

2. The level of the signals we use

3. The quality of the channel (the level of noise)

Two theoretical formulas were developed to calculate the data rate:

- one by Nyquist for a noiseless channel,
- another by Shannon for a noisy channel.

Noiseless Channel: Nyquist Bit Rate

For a noiseless channel, the **Nyquist bit rate** formula defines the theoretical maximum bit rate

$$\text{BitRate} = 2 \times \text{bandwidth} \times \log_2 L$$

In this formula, bandwidth is the bandwidth of the channel, L is the number of signal levels used to represent data, and Bit-Rate is the bit rate in bits per second.

According to the formula, we might think that, given a specific bandwidth, we can have any bit rate we want by increasing the number of signal levels. When we increase the number of signal levels, we impose a burden on the receiver. If the number of levels in a signal is just 2, the receiver can easily distinguish between a 0 and a 1. If the level of a signal is 64, the receiver must be very sophisticated to distinguish between 64 different levels. In other words, increasing the levels of a signal reduces the reliability of the system.

Example 3.34

Consider a noiseless channel with a bandwidth of 3000 Hz transmitting a signal with two signal levels. The maximum bit rate can be calculated as

$$\text{BitRate} = 2 \times 3000 \times \log_2 2 = 6000 \text{ bps}$$

Example 3.35

Consider the same noiseless channel transmitting a signal with four signal levels (for each level, we send 2 bits). The maximum bit rate can be calculated as

$$\text{BitRate} = 2 \times 3000 \times \log_2 4 = 12{,}000 \text{ bps}$$

Noisy Channel: Shannon Capacity

The channel is always noisy. In 1944, Claude Shannon introduced a formula, called the **Shannon capacity,** to determine the theoretical highest data rate for a noisy channel:

$$\text{Capacity} = \text{bandwidth} \times \log_2(1 + \text{SNR})$$

In this formula, bandwidth is the bandwidth of the channel, SNR is the signal-to noise ratio, and capacity is the capacity of the channel in bits per second. Note that in the Shannon formula there is no indication of the signal level, which means that no matter how many levels we have, we cannot achieve a data rate higher than the capacity of the channel. In other words, the formula defines a characteristic of the channel, not the method of transmission.

Example 3.37

Consider an extremely noisy channel in which the value of the signal-to-noise ratio is almost zero. In other words, the noise is so strong that the signal is faint. For this channel the capacity C is calculated as

$$C = B \log_2 (1 + \text{SNR}) = B \log_2(1 + 0) = B \log_2 1 = B \times 0 = 0$$

This means that the capacity of this channel is zero regardless of the bandwidth. In other words, we cannot receive any data through this channel.

Example 3.38

We can calculate the theoretical highest bit rate of a regular telephone line. A telephone line normally has a bandwidth of 3000 Hz (300 to 3300 Hz) assigned for data communications. The signal-to-noise ratio is usually 3162. For this channel the capacity is calculated as

$$C = B \log_2 (1 + \text{SNR}) = 3000 \log_2(1 + 3162) = 3000 \times 11.62 = 34{,}860 \text{ bps}$$

Using Both Limits

To use both methods to find the limits and signal levels. Let us show this with an example.

Example 3.41

We have a channel with a 1-MHz bandwidth. The SNR for this channel is 63. What are the appropriate bit rate and signal level?

Solution

First, we use the Shannon formula to find the upper limit.

$$C = B \log_2(1 + \text{SNR}) = 10^6 \log_2(1 + 63) = 10^6 \log_2 64 = 6 \text{ Mbps}$$

The Shannon formula gives us 6 Mbps, the upper limit. For better performance we choose something lower, 4 Mbps, for example. Then we use the Nyquist formula to find the number of signal levels.

$$4 \text{ Mbps} = 2 \times 1 \text{ MHz} \times \log_2 L \longrightarrow L = 4$$

PERFORMANCE

One important issue in networking is the performance of the network—how good is it? We discuss quality of service, an overall measurement of network performance. **Performance** of the system may be in several ways

- *Bandwidth*
- *Throughput*
- *Latency*
- *Bandwidth delay product*
- *Jitter*

1.Bandwidth

One characteristic that measures network performance is bandwidth. However, the term can be used in two different contexts with two different measuring values: **bandwidth in hertz** and **bandwidth in bits per second**.

Bandwidth in Hertz

Bandwidth in hertz is the range of frequencies contained in a composite signal or the range of frequencies a channel can pass. **For example**, we can say the bandwidth of a subscriber telephone line is 4 kHz.

Bandwidth in Bits per Seconds

The term *bandwidth* can also refer to the number of bits per second that a channel, a link, or even a network can transmit. For example, one can say the bandwidth of a Fast Ethernet network (or the links in this network) is a maximum of 100 Mbps. This means that this network can send 100 Mbps.

Relationship

There is an explicit relationship between the bandwidth in hertz and bandwidth in bits per second. Basically, an increase in bandwidth in hertz means an increase in bandwidth in bits per second. The relationship depends on whether we have baseband transmission or transmission with modulation.

Example 1

The bandwidth of a subscriber line is 4 kHz for voice or data. The bandwidth of this line for data transmission can be up to 56,000 bps using a sophisticated modem to change the digital signal to analog.

Example 2

If the telephone company improves the quality of the line and increases the bandwidth to 8 kHz, we can send 112,000 bps by using the same technology as mentioned in Example 3.42.

2.Throughput

The **throughput** is a measure of how fast we can actually send data through a network. Although, at first glance, bandwidth in bits per second and throughput seem the same, they are different. A link may have a bandwidth of *B* bps, but we can only send *T* bps through this link with *T* always less than *B*. In other words, the bandwidth is a potential measurement of a link. **The throughput** is an actual measurement of how fast we can send data.

For example, we may have a link with a bandwidth of 1 Mbps, but the devices connected to the end of the link may handle only 200 kbps. This means that we cannot send more than 200 kbps through this link.

Example:

A network with bandwidth of 10 Mbps can pass only an average of 12,000 frames per minute with each frame carrying an average of 10,000 bits. What is the throughput of this network?

Solution

We can calculate the throughput as:

Throughput = (12,000 * 10,000) / 60 = 2 Mbps

3.Latency (Delay)

The **latency** or delay defines how long it takes for an entire message to completely arrive at the destination from the time the first bit is sent out from the source. We can say that latency is made of four components: **propagation time, transmission time, queuing time** and **processing delay**.

Latency = propagation time + transmission time + queuing time + processing delay

Propagation Time

Propagation time measures the time required for a bit to travel from the source to the destination. The propagation time is calculated by dividing the distance by the propagation speed.

Propagation time = **Distance / (Propagation Speed)**

The propagation speed of electromagnetic signals depends on the medium and on the frequency of the signal. For example, in a vacuum, light is propagated with a speed of 3×10^8 m/s. It is lower in air; it is much lower in cable.

Example

What is the propagation time if the distance between the two points is 12,000 km? Assume the propagation speed to be 2.4 × 108 m/s in cable.

Solution

We can calculate the propagation time as

Propagation time = (12,000 * 10,000) / (2.4 * 2^8) = 50 ms

The example shows that a bit can go over the Atlantic Ocean in only 50 ms if there is a direct cable between the source and the destination.

Transmission Time

In data communications we send a message in the form of bits. The first bit may take a time equal to the propagation time to reach its destination; the last bit also may take the same amount of time. There is a time between the first bit leaving the sender and the last bit arriving at the receiver. The first bit leaves earlier and arrives earlier; the last bit leaves later and arrives later. The **transmission time** of a message depends on the size of the message and the bandwidth of the channel.

Transmission time = (Message size) / Bandwidth

Example

What are the propagation time and the transmission time for a 2.5-KB (kilobyte) message (an email) if the bandwidth of the network is 1 Gbps? Assume that the distance between the sender and the receiver is 12,000 km and that light travels at 2.4×10^8 m/s.

Solution

We can calculate the propagation and transmission time as

Propagation time = (12,000 * 1000) / ($2.4 * 10^8$) = 50 ms

Transmission time = (2500 * 8) / 10^9 = 0.020 ms

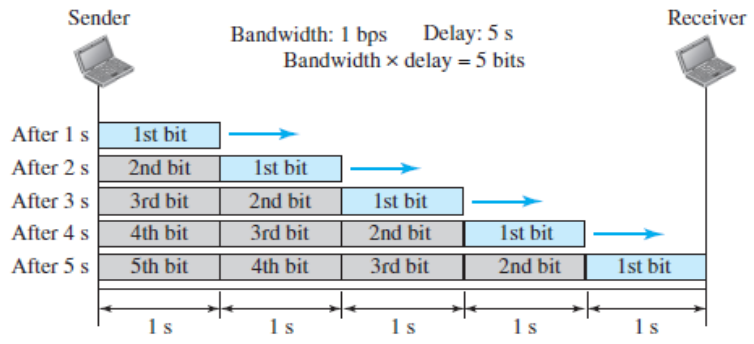
Queuing Time

The third component in latency is the **queuing time**, the time needed for each intermediate or end device to hold the message before it can be processed. The queuing time is not a fixed factor; it changes with the load imposed on the network. When there is heavy traffic on the network, the queuing time increases. An intermediate device, such as a router, queues the arrived messages and processes them one by one. If there are many messages, each message will have to wait.

4. Bandwidth-Delay Product:

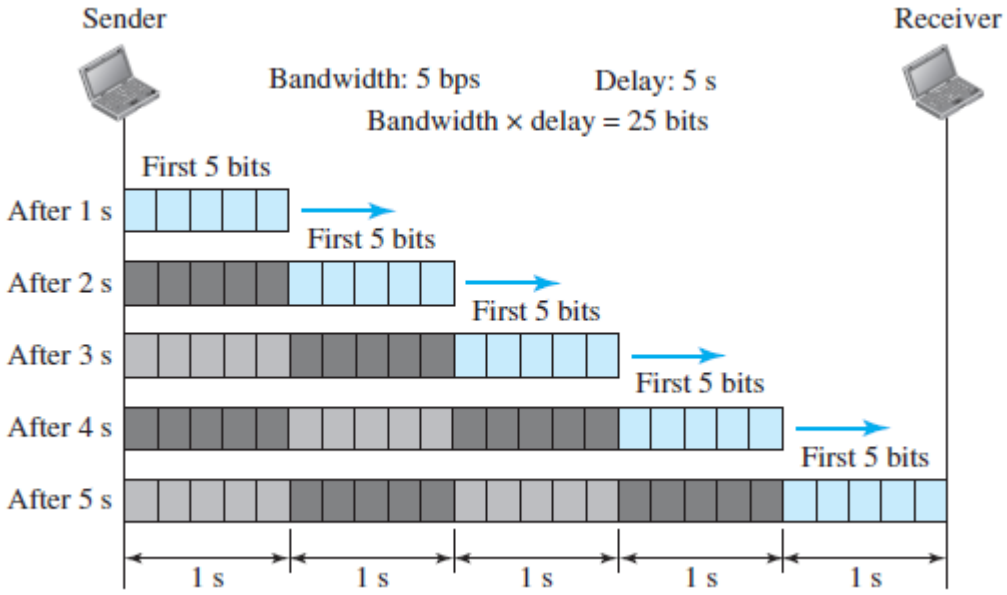
Bandwidth and delay are two performance metrics of a link. However, as we will see in this chapter and future chapters, what is very important in data communications is the product of the two, the bandwidth-delay product.

Case 1:



Let us assume that we have a link with a bandwidth of 1 bps (unrealistic, but good for demonstration purposes). We also assume that the delay of the link is 5 s (also unrealistic). We want to see what the bandwidth-delay product means in this case. Looking at the figure, we can say that this product 1 × 5 is the maximum number of bits that can fill the link. There can be no more than 5 bits at any time on the link.

Case 2. Now assume we have a bandwidth of 5 bps. Figure shows that there can be maximum 5 × 5 = 25 bits on the line. The reason is that, at each second, there are 5 bits on the line; the duration of each bit is 0.20 s.



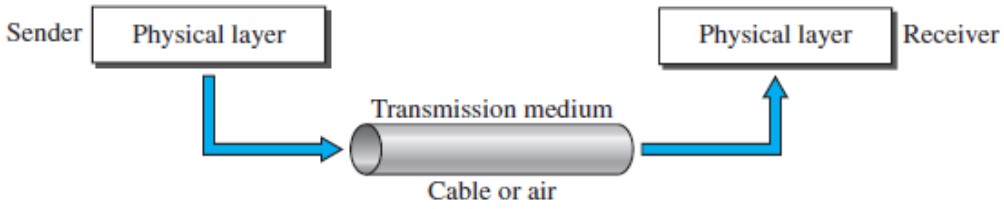
5. Jitter

Another performance issue that is related to delay is **jitter.** We can roughly say that jitter is a problem if different packets of data encounter different delays and the application using the data at the receiver site is time-sensitive (audio and video data, for example). If the delay for the first packet is 20 ms, for the second is 45 ms, and for the third is 40 ms, then the real-time application that uses the packets endures jitter.

Transmission Media

Transmission media are actually located below the physical layer and are directly controlled by the physical layer. The purpose of the **physical layer** is to transport a raw bit stream from one machine to another. Various physical media can be used for the actual transmission. Each one has its own niche in terms of bandwidth, delay, cost, and ease of installation and maintenance.

Figure 7.1 *Transmission medium and physical layer*



Transmission media belongs to layer zero. Figure 7.1 shows the position of transmission media in relation to the **physical layer**. A **transmission medium** can be broadly defined as anything that can carry information from a source to a destination.

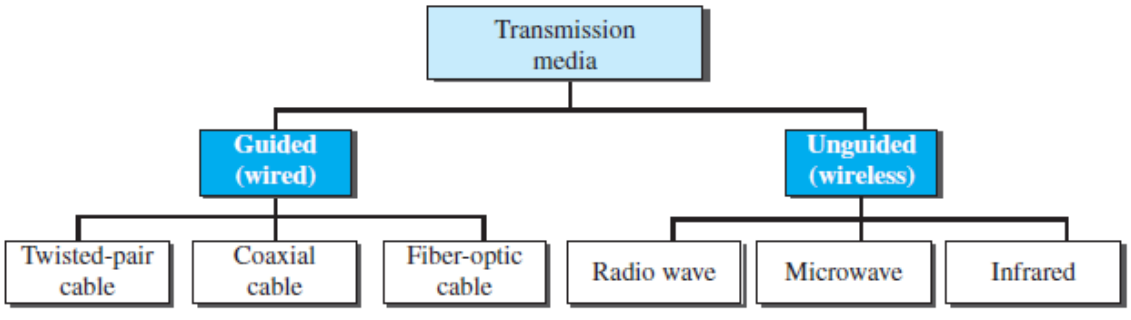
For example,

- The transmission medium for two people having a conversation is the air. The air can also be used to convey the message in a smoke signal or semaphore.
- The transmission medium might be a mail carrier.
- In data communications the definition of the information and the transmission medium is more specific.
- The transmission medium is usually free space, metallic cable, or fiber-optic cable. The information is usually a signal that is the result of a conversion of data from another form.
- The communication was, unreliable due to the poor quality of the wires. The lines were often noisy and the technology was unsophisticated.
- For a long way. Better metallic media have been invented (twisted-pair and coaxial cables

In telecommunications, transmission media can be divided into **two** broad categories:

Guided: Guided media include twisted-pair cable, coaxial cable, and fiber-optic cable.

Unguided: Unguided medium is free space.

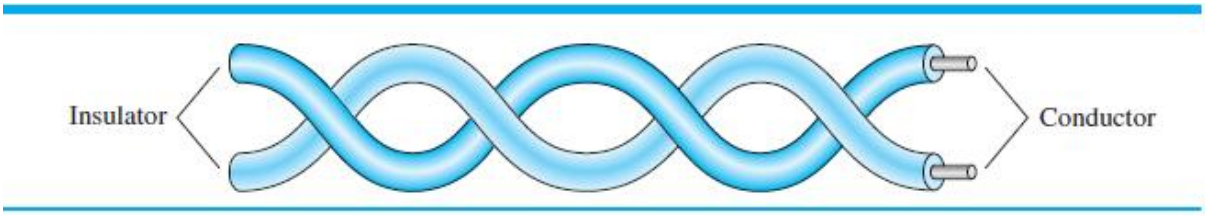


GUIDED MEDIA

Guided media, which are those that provide a conduit from one device to another, include **twisted-pair cable, coaxial cable,** and **fiber-optic cable.** A signal traveling along any of these media is directed and contained by the physical limits of the medium. **Twisted-pair and coaxial cable** use metallic (copper) conductors that accept and transport signals in the form of electric current. **Optical fiber** is a cable that accepts and transports signals in the form of light.

Twisted-Pair Cable

Twisted pair is the oldest and most common transmission media. A twisted pair consists of two insulated copper wires, typically about 1 mm thick. The wires are twisted together in a helical form, just like a DNA molecule.



One of the wires is used to carry signals to the receiver, and the other is used only as a ground reference. The receiver uses the difference between the two. the signal sent by the sender on one of the wires, interference (noise) and crosstalk may affect both wires and create unwanted signals.

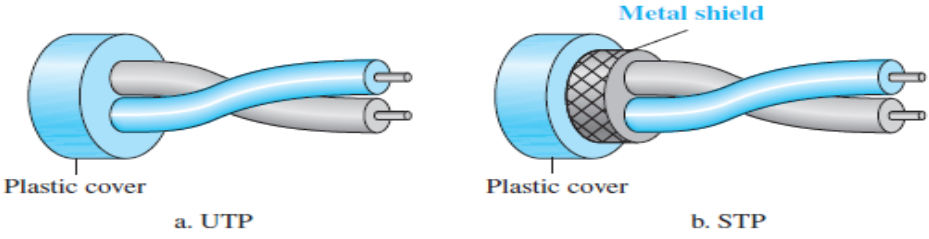
If the two wires are parallel, the effect of these unwanted signals is not the same in both wires because they are at different locations relative to the noise or crosstalk sources (e.g., one is closer and the other is farther). This results in a difference at the receiver.

Unshielded Versus Shielded Twisted-Pair Cable

The most common twisted-pair cable used in communications is referred to as ***unshielded twisted-pair*** **(UTP)** IBM has also produced a version of twisted-pair cable for its use, called ***shielded twisted-pair*** **(STP).** STP cable 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. Figure 7.4 shows the difference between UTP and STP.

Figure 7.4 *UTP and STP cables*



Categories

The Electronic Industries Association (EIA) has developed standards to classify unshielded twisted-pair cable into seven categories. Categories are determined by cable quality, with 1 as the lowest and 7 as the highest. Each EIA category is suitable for specific uses. Table 7.1 shows these categories.

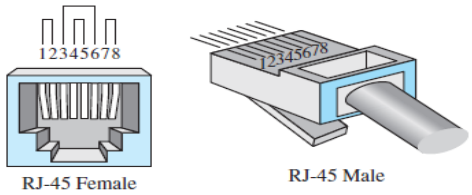
Table 7.1 *Categories of unshielded twisted-pair cables*

Category	*Specification*	*Data Rate (Mbps)*	*Use*
1	Unshielded twisted-pair used in telephone	< 0.1	Telephone
2	Unshielded twisted-pair originally used in T lines	2	T-1 lines
3	Improved CAT 2 used in LANs	10	LANs
4	Improved CAT 3 used in Token Ring networks	20	LANs
5	Cable wire is normally 24 AWG with a jacket and outside sheath	100	LANs
5E	An extension to category 5 that includes extra features to minimize the crosstalk and electromagnetic interference	125	LANs
6	A new category with matched components coming from the same manufacturer. The cable must be tested at a 200-Mbps data rate.	200	LANs
7	Sometimes called *SSTP (shielded screen twisted-pair)*. Each pair is individually wrapped in a helical metallic foil followed by a metallic foil shield in addition to the outside sheath. The shield decreases the effect of crosstalk and increases the data rate.	600	LANs

Connectors

The most common UTP connector is **RJ45** (RJ stands for registered jack), as shown in Figure 7.5. The RJ45 is a keyed connector, meaning the connector can be inserted in only one way.

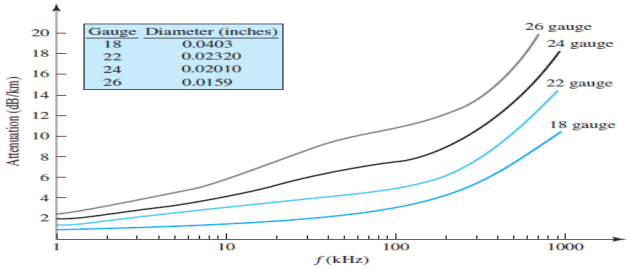
Figure 7.5 *UTP connector*



Performance

One way to measure the performance of twisted-pair cable is to compare attenuation versus frequency and distance. A twisted-pair cable can pass a wide range of frequencies. However, Figure 7.6 shows that with increasing frequency, the attenuation, measured in decibels per kilometer (dB/km), sharply increases with frequencies above 100 kHz. Note that ***gauge*** is a measure of the thickness of the wire.

Figure 7.6 *UTP performance*



Coaxial Cables:

Coaxial cable is a common transmission medium. It has better shielding than twisted pairs, so it can pan longer distances at higher speeds. 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 other kind, 75-ohm cable, is commonly used for analog transmission and cable television but is becoming more important with the advent of Internet over cable. This distinction is based on historical, rather than technical, factors (e.g., early dipole antennas had an impedance of 300 ohms, and it was easy to use existing 4:1 impedance matching transformers).

A coaxial cable consists of a stiff copper wire as the core, surrounded by an insulating material. The insulator is encased by a cylindrical conductor, often as a closely-woven braided mesh. The outer conductor is covered in a protective plastic sheath. A cutaway view of a coaxial cable is shown in Fig.

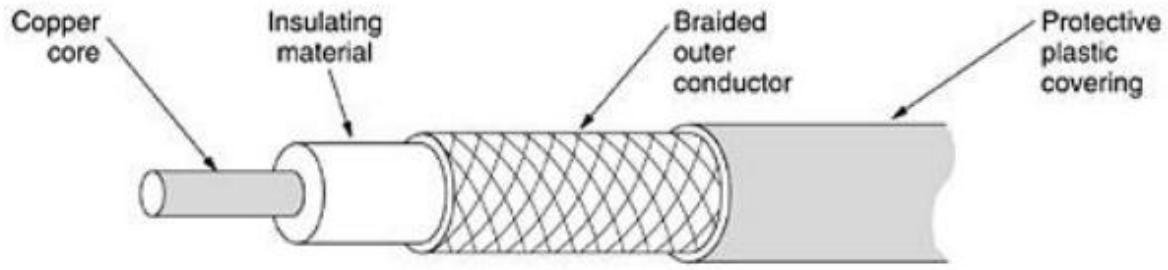


Fig.: A coaxial cable.

The construction and shielding of the coaxial cable give it a good combination of high bandwidth and excellent noise immunity. The bandwidth possible depends on the cable quality, length, and signal-to-noise ratio of the data signal. Modern cables have a bandwidth of close to 1 GHz. Coaxial cables used to be widely used within the telephone system for long-distance lines but have now largely been replaced by fiber optics on long-haul routes. Coax is still widely used for cable television and metropolitan area networks.

Coaxial Cable Standards

Coaxial cables are categorized by their **Radio Government (RG)** ratings. Each RG number denotes a unique set of physical specifications, including the wire gauge of the inner conductor, the thickness and type of the inner insulator, the construction of the shield, and the size and type of the outer casing. Each cable defined by an RG rating is adapted for a specialized function, as shown in Table 7.2.

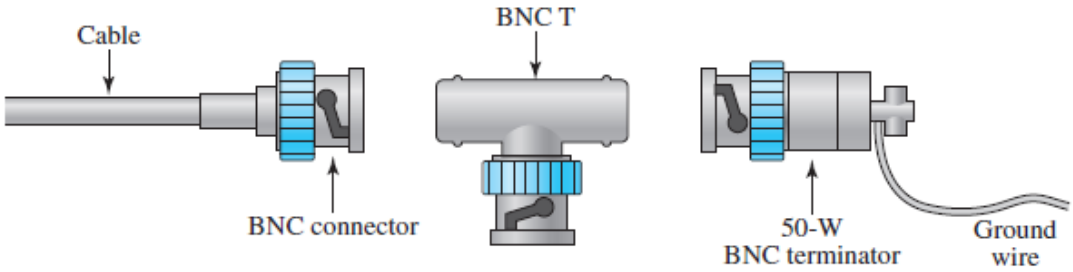
Table 7.2 *Categories of coaxial cables*

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

Coaxial Cable Connectors

To connect coaxial cable to devices, we need coaxial connectors. The most common type of connector used today is the **Bayonet Neill-Concelman (BNC)** connector. Figure 7.8 shows three popular types of these connectors: the BNC connector, the BNC T connector, and the BNC terminator.

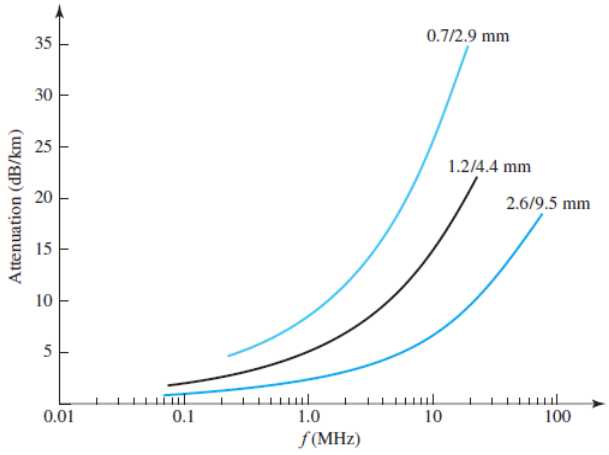
Figure 7.8 *BNC connectors*



Performance

We can measure the performance of a coaxial cable. We notice in Figure 7.9 that the attenuation is much higher in coaxial cable than in twisted-pair cable. In other words, although coaxial cable has a much higher bandwidth, the signal weakens rapidly and requires the frequent use of repeaters.

Figure 7.9 *Coaxial cable performance*



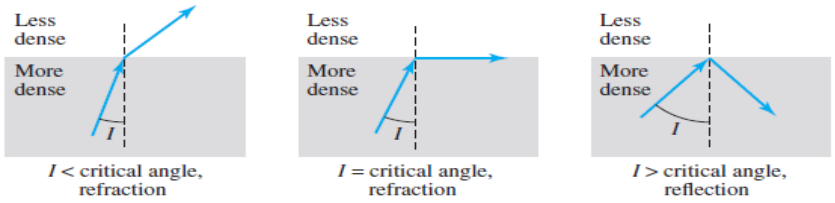
Cable TV networks (see Chapter 14) also use coaxial cables. Another common application of coaxial cable is in traditional Ethernet LANs. Because of its high bandwidth, and consequently high data rate, coaxial cable was chosen for digital transmission in early Ethernet LANs. The **10Base-2**, or Thin Ethernet, uses **RG-58** coaxial cable with BNC connectors to transmit data at 10 Mbps with a range of 185 m. The **10Base5**, or Thick Ethernet, uses **RG-11** (thick coaxial cable) to transmit 10 Mbps with a range of 5000 m. Thick Ethernet has specialized connectors.

Fiber-Optic Cable

A fiber-optic cable is made of glass or plastic and transmits signals in the form of light. To understand optical fiber, we first need to explore several aspects of the nature of light. Light travels in a straight line as long as it is moving through a single uniform substance.

If a ray of light traveling through one substance suddenly enters another substance (of a different density), the ray changes direction. Figure 7.10 shows how a ray of light changes direction when going from a denser to a less dense substance.

Figure 7.10 *Bending of light ray*

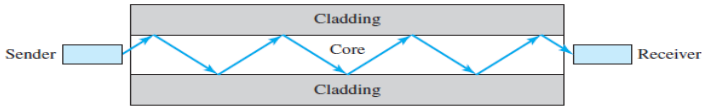


If the **angle of incidence** *I* (the angle the ray makes with the line perpendicular to the interface between the two substances) is less than the **critical angle,** the ray **refracts** and moves closer to the surface. If the angle of incidence is equal to the critical angle, the light bends along the interface.

If the angle is greater than the critical angle, the ray **reflects** (makes a turn) and travels again in the denser substance. Note that the critical angle is a property of the substance, and its value differs from one substance to another.

Optical fibers use reflection to guide light through a channel. A glass or plastic **core** is surrounded by a **cladding** of less dense glass or plastic. The difference in density of the two materials must be such that a beam of light moving through the core is reflected off the cladding instead of being refracted into it. See Figure 7.11.

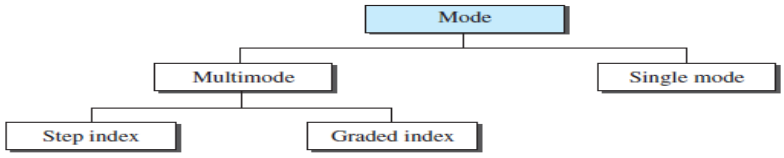
Figure 7.11 *Optical fiber*



Propagation Modes

Current technology supports two modes (multimode and single mode) for propagating light along optical channels, each requiring fiber with different physical characteristics. Multimode can be implemented in two forms: step-index or graded-index (see Figure 7.12).

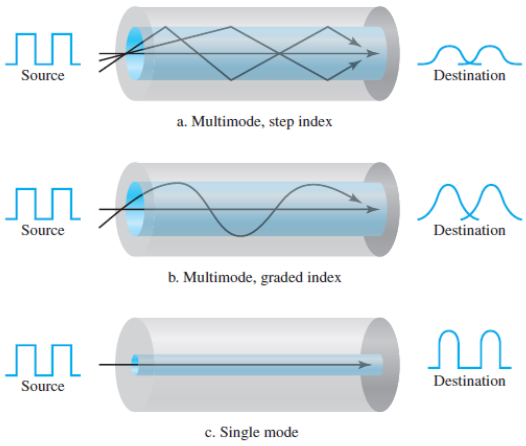
Figure 7.12 *Propagation modes*



Multimode

Multimode is so named because multiple beams from a light source move through the core in different paths. How these beams move within the cable depends on the structure of the core, as shown in Figure 7.13.

Figure 7.13 *Modes*



In **multimode step-index fiber,** the density of the core remains constant from the center to the edges. A beam of light moves through this constant density in a straight line until it reaches the interface of the core and the cladding. At the interface, there is an abrupt change due to a lower density; this alters the angle of the beam's motion. The term *step-index* refers to the suddenness of this change, which contributes to the distortion of the signal as it passes through the fiber.

A second type of fiber, called **multimode graded-index fiber,** decreases this distortion of the signal through the cable. The word *index* here refers to the index of refraction. As we saw above, the index of refraction is related to density. A graded-index fiber, therefore, is one with varying densities. Density is highest at the center of the core and decreases gradually to its lowest at the edge. Figure 7.13 shows the impact of this variable density on the propagation of light beams.

Single-Mode

Single-mode uses step-index fiber and a highly focused source of light that limits beams to a small range of angles, all close to the horizontal. The **single-mode fiber** itself is manufactured with a much smaller diameter than that of multimode fiber, and with substantially lower density (index of refraction). The decrease in density results in a critical angle that is close enough to 90° to make the propagation of beams almost horizontal. In this case, propagation of different beams is almost identical, and delays are negligible. All the beams arrive at the destination "together" and can be recombined with little distortion to the signal (see Figure 7.13).

Fiber Sizes

Optical fibers are defined by the ratio of the diameter of their core to the diameter of their cladding, both expressed in micrometers. The common sizes are shown in Table 7.3. Note that the last size listed is for single-mode only.

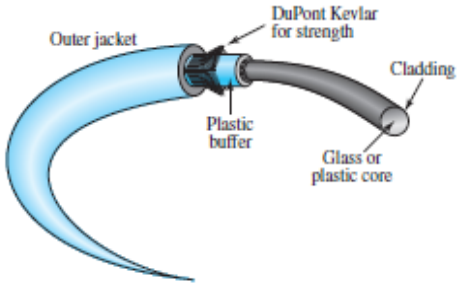
Table 7.3 *Fiber types*

Type	*Core (μm)*	*Cladding (μm)*	*Mode*
50/125	50.0	125	Multimode, graded index
62.5/125	62.5	125	Multimode, graded index
100/125	100.0	125	Multimode, graded index
7/125	7.0	125	Single mode

Cable Composition

Figure 7.14 shows the composition of a typical fiber-optic cable. The outer jacket is made of either PVC or Teflon. Inside the jacket are Kevlar strands to strengthen the cable. Kevlar is a strong material used in the fabrication of bulletproof vests. Below the Kevlar is another plastic coating to cushion the fiber. The fiber is at the center of the cable, and it consists of cladding and core.

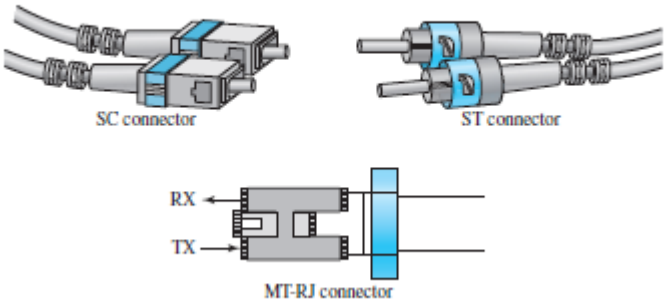
Figure 7.14 *Fiber construction*



Fiber-Optic Cable Connectors

There are three types of connectors for fiber-optic cables, as shown in Figure 7.15. The **subscriber channel (SC) connector** is used for cable TV. It uses a push/pull locking system. The **straight-tip (ST) connector** is used for connecting cable to networking devices. It uses a bayonet locking system and is more reliable than SC. **MT-RJ** is a connector that is the same size as RJ45.

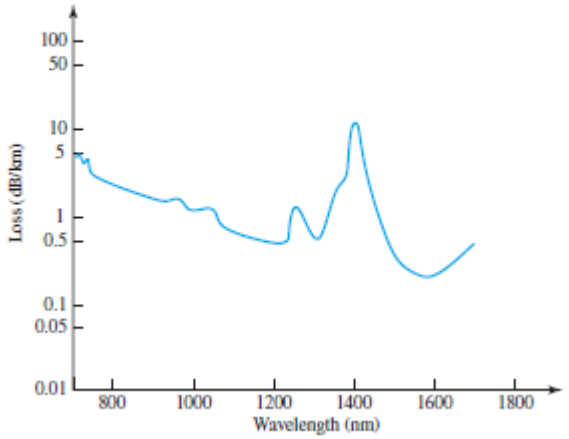
Figure 7.15 *Fiber-optic cable connectors*



Performance

The plot of attenuation versus wavelength in Figure 7.16 shows a very interesting phenomenon in fiber-optic cable. Attenuation is flatter than in the case of twisted-pair cable and coaxial cable. The performance is such that we need fewer (actually one tenth as many) repeaters when we use fiber-optic cable.

Figure 7.16 *Optical fiber performance*



Advantages and Disadvantages of Optical Fiber

Advantages

Fiber-optic cable has several advantages over metallic cable (twisted-pair or coaxial).

Higher bandwidth. Fiber-optic cable can support dramatically higher bandwidths (and hence data rates) than either twisted-pair or coaxial cable. Currently, data rates and bandwidth utilization over fiber-optic cable are limited not by the medium but by the signal generation and reception technology available.

Less signal attenuation. Fiber-optic transmission distance is significantly greater than that of other guided media. A signal can run for 50 km without requiring regeneration. We need repeaters every 5 km for coaxial or twisted-pair cable.

Immunity to electromagnetic interference. Electromagnetic noise cannot affect fiber-optic cables.

Resistance to corrosive materials. Glass is more resistant to corrosive materials than copper.

Light weight. Fiber-optic cables are much lighter than copper cables.

Greater immunity to tapping. Fiber-optic cables are more immune to tapping than copper cables. Copper cables create antenna effects that can easily be tapped.

Disadvantages

There are some disadvantages in the use of optical fiber.

Installation and maintenance. Fiber-optic cable is a relatively new technology. Its installation and maintenance require expertise that is not yet available everywhere.

Unidirectional light propagation. Propagation of light is unidirectional. If we need bidirectional communication, two fibers are needed.

Cost. The cable and the interfaces are relatively more expensive than those of other guided media. If the demand for bandwidth is not high, often the use of optical fiber cannot be justified.

UNGUIDED MEDIA: WIRELESS

Wireless transmission helps users to access information anywhere and anytime. Mobile users needs to access data for their computing devices, such as laptop, notebook, shirt pocket, palmtop etc. for these users, wireless communication will be help to fulfill their needs. Some people believe that the future holds only two kinds of communication: fiber and wireless. All fixed (i.e., non-mobile) computers, telephones, faxes, and so on will use fiber, and all mobile ones will use wireless. Wireless has advantages for even fixed devices in some circumstances.

For example, if running a fiber to a building is difficult due to the terrain (mountains, jungles, swamps, etc.), wireless may be better.

Unguided medium transport electromagnetic waves without using a physical conductor. This type of communication is often referred to as ***wireless communication*.** Signals are normally broadcast through free space and thus are available to anyone who has a device capable of receiving them.

Figure 7.17 shows the part of the electromagnetic spectrum, ranging from 3 kHz to 900 THz, used for wireless communication.

Unguided signals can travel from the source to the destination in several ways:

Ground propagation: In **ground propagation: R**adio waves travel through the lowest portion of the atmosphere, hugging the earth.

Sky propagation: In **sky propagation,** higher-frequency radio waves radiate upward into the ionosphere (the layer of atmosphere where particles exist as ions) where they are reflected back to earth. This type of transmission allows for greater distances with lower output power.

Line-of-sight propagation: In **line-of-sight propagation,** very high-frequency signals are transmitted in straight lines directly from antenna to antenna. Antennas must be directional, facing each other and either tall enough or close enough together not to be affected by the curvature of the earth. Line-of sight propagation is tricky because radio transmissions cannot be completely focused.

Figure 7.17 *Electromagnetic spectrum for wireless communication*

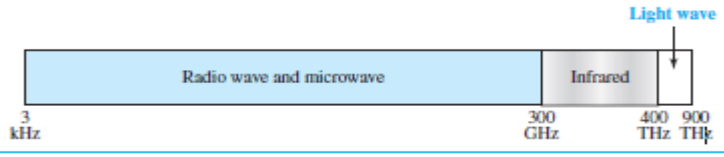
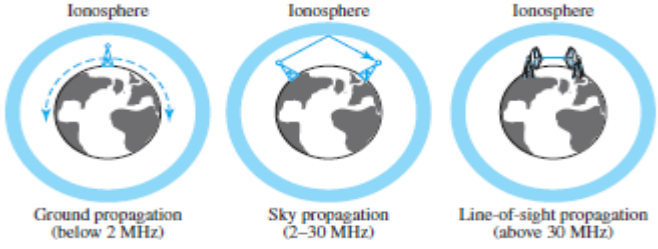


Figure 7.18 *Propagation methods*



Radio Transmission

Radio waves has the electromagnetic spectrum in the range of 3 Hz to 300 GHz. Radio waves are easy to generate, can travel long distances, and can penetrate buildings easily, so they are widely used for communication, both indoors and outdoors. Radio waves also are Omni directional, meaning that they travel in all directions from the source, so the transmitter and receiver do not have to be carefully aligned physically.

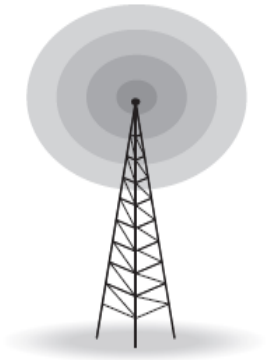
1) The properties of radio waves are frequency dependent.
2) At low frequencies, radio waves pass through obstacles without any problem.
3) At high frequencies, radio waves tend to travel in straight lines and bounce off obstacles.
4) They are also absorbed by rain. At all frequencies, radio waves are subject to interference from motors and other electrical equipment.
5) Radio systems are subject to multipath fading and interference.
6) The radio waves can travel long distances, but interference between users is a problem.

The range of various frequency bands are classified according to wavelength. Low frequency (LF) spans in the range of 30 KHz to 300 KHz, which corresponds to a wavelength of 1 KM to 10 KM.

Applications of Radio waves:

1) Cellular communications- analog cellular telephone system.
2) Wireless LANs- (WLAN).
3) Point to point and point to multipoint radio systems.

Figure 7.19 *Omnidirectional antenna*



Microwave Transmission:

Electromagnetic waves having frequencies between 1 and 300 GHz are called microwaves. Microwaves are unidirectional. When an antenna transmits microwaves, they can be narrowly focused. This means that the sending and receiving antennas need to be aligned. The unidirectional property has an obvious advantage.

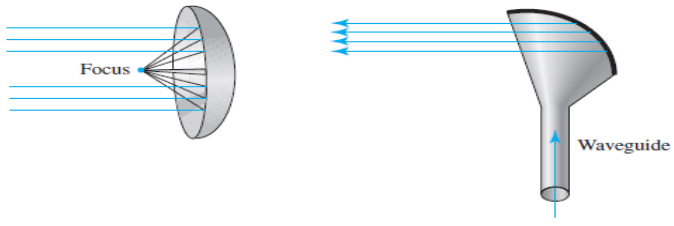
A pair of antennas can be aligned without interfering with another pair of aligned antennas. The following describes some characteristics of microwave propagation:

- Microwave propagation is line-of-sight. Since the towers with the mounted antennas need to be in direct sight of each other, towers that are far apart need to be very tall. The curvature of the earth as well as other blocking obstacles do not allow two short towers to communicate by using microwaves. Repeaters are often needed for long distance communication.
- Very high-frequency microwaves cannot penetrate walls. This characteristic can be a disadvantage if receivers are inside buildings.
- The microwave band is relatively wide, almost 299 GHz. Therefore wider sub-bands can be assigned, and a high data rate is possible.
- Use of certain portions of the band requires permission from authorities.

Unidirectional Antenna

Microwaves need **unidirectional antennas** that send out signals in one direction. Two types of antennas are used for microwave communications: the parabolic dish and the horn

Figure 7.20 *Unidirectional antennas*



Properties of microwaves:

1. microwaves travel in a straight line
2. microwaves do not pass through buildings
3. microwave communication is so widely used for long-distance telephone communication, mobile phones, television distribution
4. Microwaves are weather and frequency dependent.
5. Microwaves are easily absorbed by rain.
6. Some waves may be refracted off low-lying atmospheric layers and may take slightly longer to arrive than the direct waves. The delayed waves may arrive out of phase with the direct wave and thus cancel the signal. This effect is called multipath fading.

Advantages of microwaves:

1. Microwave is also relatively inexpensive.
2. Cheaper than leasing the telephone company's fiber.

Applications of microwaves:

1. Point to point wireless systems use microwave frequencies.
2. Digital microwave transmission system has been deployed to provide long distance communication.
3. Microwaves are used in satellite communication.

Infrared and Millimeter Waves:

Infrared waves, with frequencies from 300 GHz to 400 THz (wavelengths from 1 mm to 770 nm), can be used for short-range communication. Infrared waves, having high frequencies, cannot penetrate walls.

This advantageous characteristic prevents interference between one system and another; a short-range communication system in one room cannot be affected by another system in the next room. Unguided infrared and millimeter waves are widely used for short-range communication.

The remote controls used on televisions, VCRs, and stereos all use infrared communication. They are relatively directional, cheap, and easy to build. In general, as we go from long-wave radio toward visible light, the waves behave more and more like light and less and less like radio. Development is very high speed LANs. Infrared communication operates in the region from 850 mm to 900 mm.

Disadvantages:

1. A major drawback: they do not pass through solid objects (try standing between your remote control and your television and see if it still works).
2. Infrared communication has a limited use on the desktop, for example, connecting notebook computers and printers is not used majorly.

Light wave Transmission:

A more modern application is to connect the LANs in two buildings via lasers mounted on their rooftops. Coherent optical signaling using lasers is inherently unidirectional, so each building needs its own laser and its own photo detector. This scheme offers very high bandwidth and very low cost

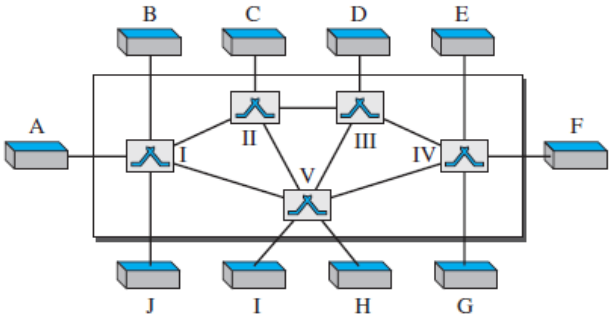
A disadvantage is that laser beams cannot penetrate rain or thick fog, but they normally work well on sunny days. The lenses are put into the system to defocus the beam slightly.

Switching

A network is a set of connected devices. Whenever we have multiple devices, we have the problem of how to connect them to make one-to-one communication possible. One solution is to make a point-to-point connection between each pair of devices (a mesh topology) or between a central device and every other device (a star topology).

A better solution is **switching.** A switched network consists of a series of interlinked nodes, called ***switches*.** Switches are devices capable of creating temporary connections between two or more devices linked to the switch. In a switched network, some of these nodes are connected to the end systems

Figure 8.1 *Switched network*

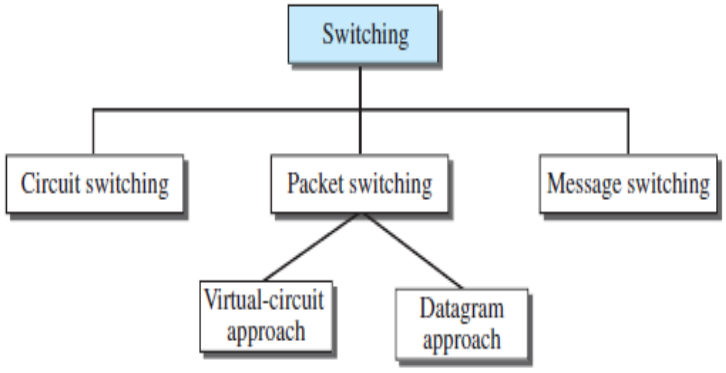


In the above figure, the **end systems** (communicating devices) are labeled A, B, C, D, and so on, and the switches are labeled I, II, III, IV, and V. Each switch is connected to multiple links.

Three Methods of Switching

The three methods of switching have been discussed: **circuit switching**, **packet switching**, and **message switching**. **Packet switching** can further be divided into two subcategories they are **virtual-circuit approach** and **datagram approach**

Figure 8.2 *Taxonomy of switched networks*



Switching and TCP/IP Layers

Switching can happen at several layers of the TCP/IP protocol suite.

Switching at Physical Layer

At the physical layer, we can have only circuit switching. There are no packets exchanged at the physical layer. The switches at the physical layer allow signals to travel in one path or another.

Switching at Data-Link Layer

At the data-link layer, we can have packet switching. However, the term *packet* in this case means *frames* or *cells*. Packet switching at the data-link layer is normally done using a virtual-circuit approach.

Switching at Network Layer

At the network layer, we can have packet switching. In this case, either a virtual-circuit approach or a datagram approach can be used. Currently the Internet uses a datagram approach, but the tendency is to move to a virtual-circuit approach.

Switching at Application Layer

At the application layer, we can have only **message switching**. The communication at the application layer occurs by exchanging messages. Conceptually, we can say that communication using e-mail is a kind of **message-switched communication**, but we do not see any network that actually can be called **a message-switched network**.

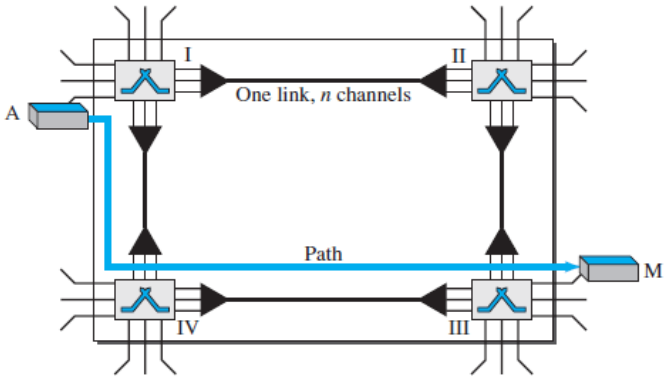
CIRCUIT-SWITCHED NETWORKS

A **circuit-switched network** consists of a set of switches connected by physical links. A connection between two stations is a dedicated path made of one or more links.

Each connection uses only one dedicated channel on each link. Each link is normally divided into *n* channels by using FDM or TDM.

Figure 8.3 shows a trivial circuit-switched network with four switches and four links. Each link is divided into *n* (*n* is 3 in the figure) channels by using FDM or TDM.

Figure 8.3 *A trivial circuit-switched network*



We have explicitly shown the multiplexing symbols to emphasize the division of the link into channels even though multiplexing can be implicitly included in the switch fabric.

The end systems, such as computers or telephones, are directly connected to a switch. We have shown only two end systems for simplicity. When end system A needs to communicate with end system M, system A needs to request a connection to M that must be accepted by all switches as well as by M itself. This is called the **setup phase. A** circuit (channel) is reserved on each link, and the combination of circuits or channels defines the dedicated path. After the dedicated path made of connected circuits (channels) is established, the **data-transfer phase** can take place. After all data have been transferred, the circuits are torn down.

We need to emphasize several points here:

- Circuit switching takes place at the physical layer.
- Before starting communication, the stations must make a reservation for the resources to be used during the communication. These resources, such as channels (bandwidth in FDM and time slots in TDM), switch buffers, switch processing time, and switch input/output ports, must remain dedicated during the entire duration of data transfer until the **teardown phase.**
- Data transferred between the two stations are not packetized (physical layer transfer of the signal). The data are a continuous flow sent by the source station and received by the destination station, although there may be periods of silence.
- There is no addressing involved during data transfer. The switches route the data based on their occupied band (FDM) or time slot (TDM). Of course, there is end-to-end addressing used during the setup phase, as we will see shortly.

Three Phases

The actual communication in a circuit-switched network requires three phases:

- ***connection***
- ***setup,***
- ***data transfer, and***
- ***Connection teardown.***

Setup Phase

Before the two parties (or multiple parties in a conference call) can communicate, a dedicated circuit (combination of channels in links) needs to be established. The end systems are normally connected through dedicated lines to the switches, so connection setup means creating dedicated channels between the switches.

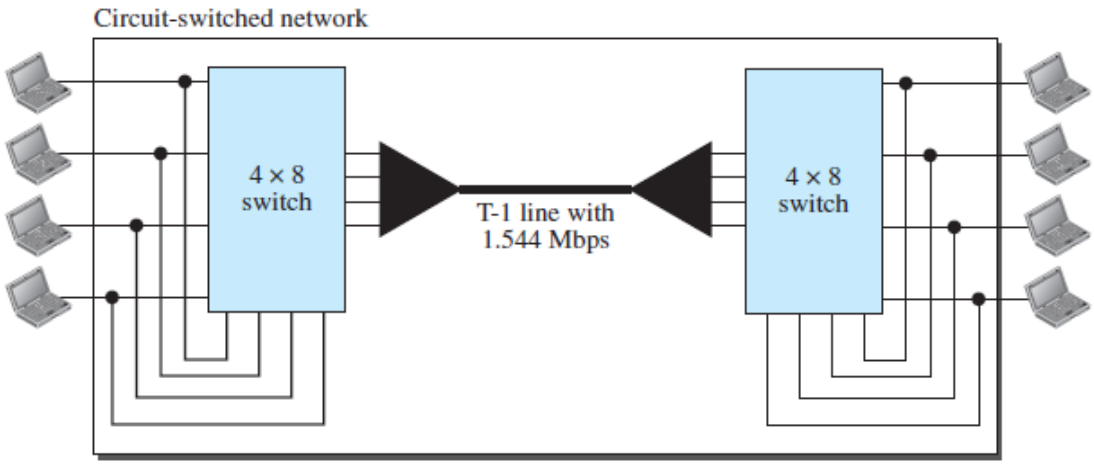
For example, in Figure 8.3, when system A needs to connect to system M, it sends a setup request that includes the address of system M, to switch I. Switch I finds a channel between itself and switch IV that can be dedicated for this purpose. Switch I then sends the request to switch IV, which

finds a dedicated channel between itself and switch III. Switch III informs system M of system A's intention at this time.

In the next step to making a connection, an acknowledgment from system M needs to be sent in the opposite direction to system A. Only after system A receives this acknowledgment is the connection established. Note that end-to-end addressing is required for creating a connection between the two end systems.

For example, the addresses of the computers assigned by the administrator in a TDM network, or telephone numbers in an FDM network.

Figure 8.5 *Circuit-switched network used in Example 8.2*



Data-Transfer Phase

After the establishment of the dedicated circuit (channels), the two parties can transfer data.

Teardown Phase

When one of the parties needs to disconnect, a signal is sent to each switch to release the resources.

Efficiency

It can be argued that circuit-switched networks are not as efficient as the other two types of networks because resources are allocated during the entire duration of the connection. These resources are unavailable to other connections.

Delay

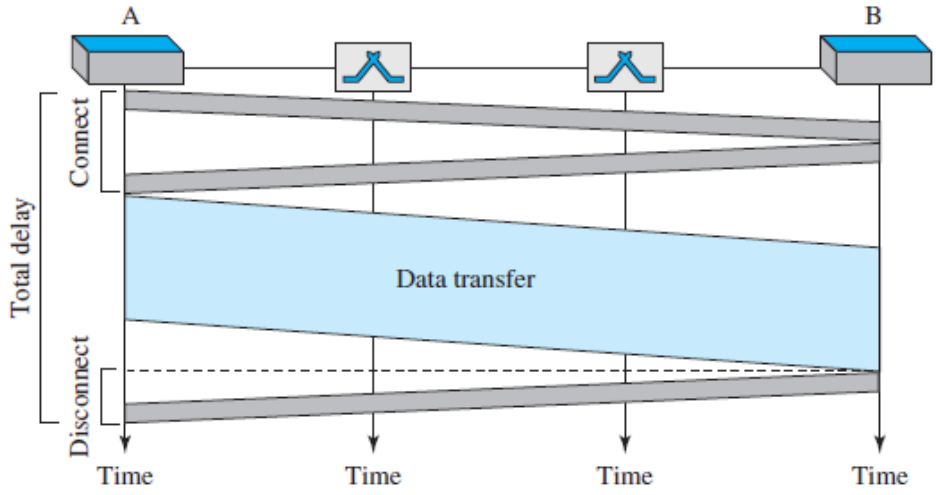
A circuit-switched network normally has low efficiency, the delay in this type of network is minimal. During data transfer the data are not delayed at each switch; the resources are allocated for the duration of the connection. Figure 8.6 shows the idea of delay in a circuit-switched network when only two switches are involved. As Figure 8.6 shows, there is no waiting time at each switch. The total delay is due to the time needed to create the connection, transfer data, and disconnect the circuit.

- The delay caused by the setup is the sum of four parts:
- the propagation time of the source computer request (slope of the first gray box),

- the request signal transfer time (height of the first gray box),
- the propagation time of the acknowledgment from the destination computer (slope of the second gray box), and
- The signal transfer time of the acknowledgment (height of the second gray box).

The delay due to data transfer is the sum of two parts: the propagation time (slope of the colored box) and data transfer time (height of the colored box), which can be very long. The third box shows the time needed to tear down the circuit. We have shown the case in which the receiver requests disconnection, which creates the maximum delay.

Figure 8.6 *Delay in a circuit-switched network*



PACKET SWITCHING

In data communications, we need to send messages from one end system to another. If the message is going to pass through a **packet-switched network**, it needs to be divided into packets of **fixed or variable size**. The size of the packet is determined by the network and the governing protocol.

In packet switching, there is no resource allocation for a packet. This means that there is no reserved bandwidth on the links, and there is no scheduled processing time for each packet. Resources are allocated on demand. The allocation is done on a first-come, first-served basis. When a switch receives a packet, no matter what the source or destination is, the packet must wait if there are other packets being processed.

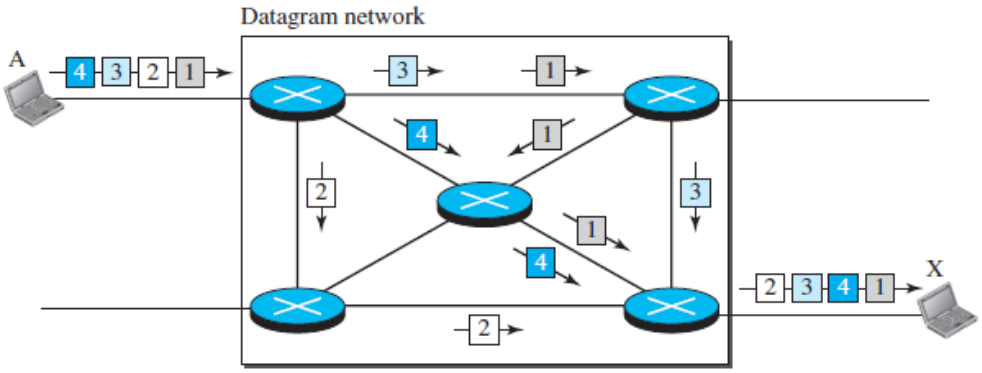
We can have two types of packet-switched networks**: datagram networks and virtual circuit networks.**

Datagram Networks

In a **datagram network**, each packet is treated independently of all others. Even if a packet is part of a multi-packet transmission, the network treats it as though it existed alone. Packets in this approach are referred to as ***datagram's***. Datagram switching is normally done at the network layer.

Figure 8.7 shows how the datagram approach is used to deliver four packets from station A to station X. The switches in a datagram network are traditionally referred to as routers. That is why we use a different symbol for the switches in the figure.

Figure 8.7 *A datagram network with four switches (routers)*



In this example, all four packets (or datagram's) belong to the same message, but may travel different paths to reach their destination. This is so because the links may be involved in carrying packets from other sources and do not have the necessary bandwidth available to carry all the packets from A to X.

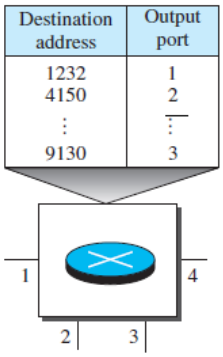
This approach can cause the datagram's of a transmission to arrive at their destination out of order with different delays between the packets. Packets may also be lost or dropped because of a lack of resources.

The datagram networks are sometimes referred to as *connectionless networks*. The term *connectionless* here means that the switch (packet switch) does not keep information about the connection state. There are no setup or teardown phases. Each packet is treated the same by a switch regardless of its source or destination.

Routing Table

The routing tables are dynamic and are updated periodically. The destination addresses and the corresponding forwarding output ports are recorded in the tables. Figure 8.8 shows the routing table for a switch.

Figure 8.8 *Routing table in a datagram network*



Destination Address

Every packet in a datagram network carries a header that contains, among other information, the destination address of the packet. When the switch receives the packet, this destination address is examined; the routing table is consulted to find the corresponding port through which the packet should be forwarded.

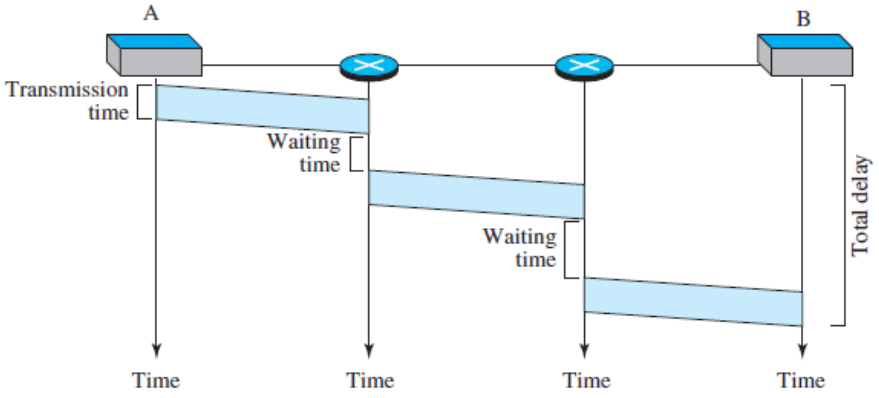
Efficiency

The efficiency of a datagram network is better than that of a circuit-switched network; resources are allocated only when there are packets to be transferred. If a source sends a packet and there is a delay of a few minutes before another packet can be sent, the resources can be reallocated during these minutes for other packets from other sources.

Delay

There may be greater delay in a datagram network than in a virtual-circuit network. Although there are no setup and teardown phases, each packet may experience a wait at a switch before it is forwarded. In addition, since not all packets in a message necessarily travel through the same switches, the delay is not uniform for the packets of a message. Figure 8.9 gives an example of delay in a datagram network for one packet.

Figure 8.9 *Delay in a datagram network*



Virtual-Circuit Networks

A **virtual-circuit network** is a cross between a circuit-switched network and a datagram network. It has some characteristics of both.

1. As in a circuit-switched network, there are setup and teardown phases in addition to the data transfer phase.

2. Resources can be allocated during the setup phase, as in a circuit-switched network, or on demand, as in a datagram network.

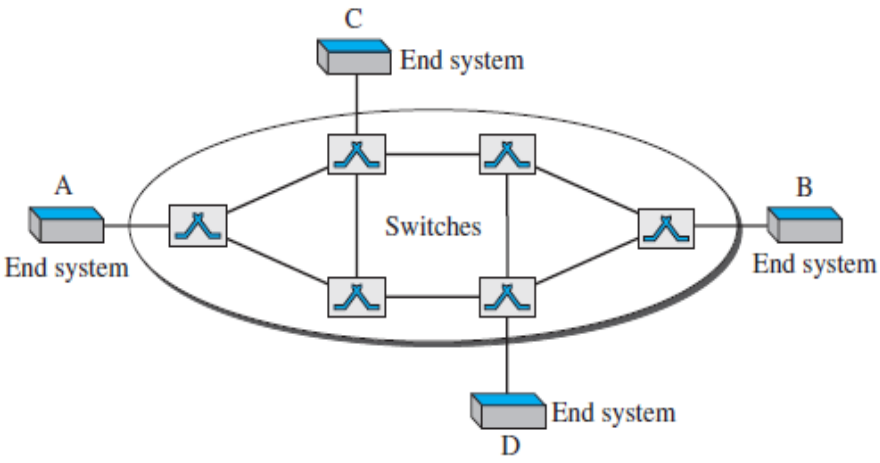
3. As in a datagram network, data are packetized and each packet carries an address in the header.

4. As in a circuit-switched network, all packets follow the same path established during the connection.

5. A virtual-circuit network is normally implemented in the data-link layer, while a circuit-switched network is implemented in the physical layer and a datagram network in the network layer. But this may change in the future.

Figure 8.10 is an example of a virtual-circuit network. The network has switches that allow traffic from sources to destinations. A source or destination can be a computer, packet switch, bridge, or any other device that connects other networks.

Figure 8.10 *Virtual-circuit network*



Addressing

In a virtual-circuit network, two types of addressing are involved: **global and local** (virtual-circuit identifier).

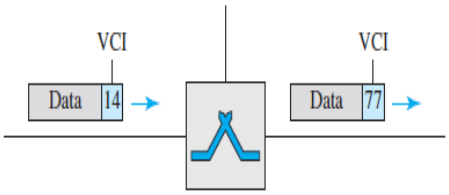
Global Addressing

A source or a destination needs to have a **global address** an address that can be unique in the scope of the network or internationally if the network is part of an international network.

Virtual-Circuit Identifier

The identifier that is actually used for data transfer is called the ***virtual-circuit identifier* (VCI)** or the ***label*.** A VCI, unlike a global address, is a small number that has only switch scope; it is used by a frame between two switches. When a frame arrives at a switch, it has a VCI; when it leaves, it has a different VCI. Figure 8.11 shows how the VCI in a data frame changes from one switch to another. Note that a VCI does not need to be a large number since each switch can use its own unique set of VCIs.

Figure 8.11 *Virtual-circuit identifier*



Three Phases

As in a circuit-switched network, a source and destination need to go through three phases in a virtual-circuit network: setup, data transfer, and teardown.

- In the setup phase, the source and destination use their global addresses to help switches make table entries for the connection.
- In the teardown phase, the source and destination inform the switches to delete the corresponding entry.
- Data transfer occurs between these two phases.

UNIT II:

Introduction to Data Link Layer: *Introduction, Link Layer addressing.*

Error detection and Correction: *Cyclic codes, Checksum, Forward error Correction.*

Data link Control: *DLC services, Data link layer protocols, HDLC, Point to Point Protocol.*

Media Access Control: *Random Access, Controlled Access, Channelization.*

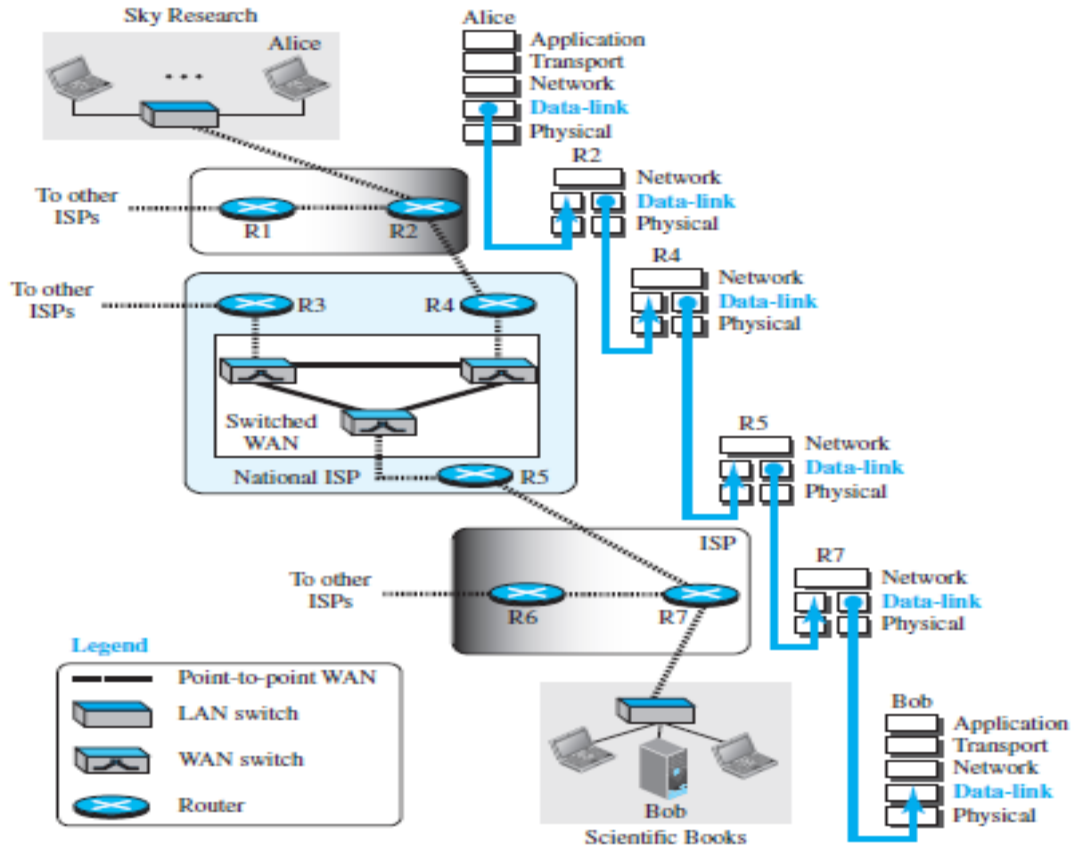
Connecting Devices and Virtual LAN's: *Connecting Devices.*

Introduction to Data Link Layer:

INTRODUCTION

The Internet is a combination of networks glued together by connecting devices (routers or switches). If a packet is to travel from a host to another host, it needs to pass through these networks. Communication at the data-link layer is made up of five separate logical connections between the data-link layers in the path.

Figure 9.1 *Communication at the data-link layer*



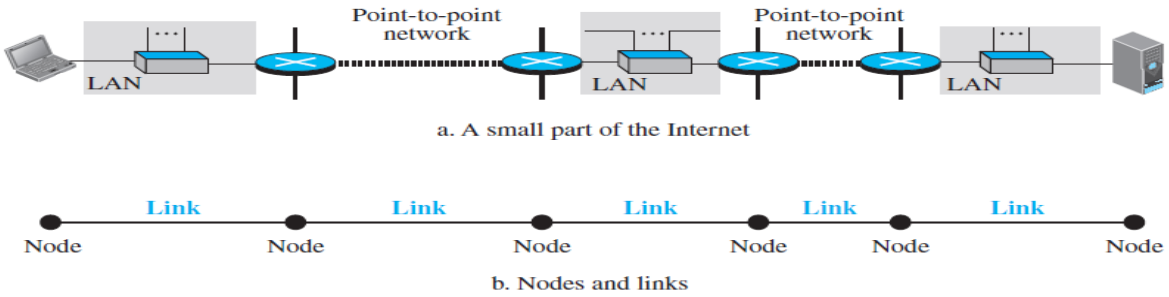
The data-link layer at Alice's computer communicates with the data-link layer at router R2. The data-link layer at router R2 communicates with the data-link layer at router R4, the data-link layer at router R7 communicates with the data-link layer at Bob's computer. Only one data-link layer is involved at the source or the destination, but two data-link layers are involved at each router.

1) Nodes and Links

Communication at the data-link layer is node-to-node. A data unit from one point in the Internet needs to pass through many networks (LANs and WANs) to reach another point. Theses LANs and WANs are connected by routers. It is customary to refer to the two end hosts and the routers as ***nodes*** and the networks in between as ***links.***

Figure 9.2 is a simple representation of links and nodes when the path of the data unit is only six nodes.

Figure 9.2 *Nodes and Links*



The first node is the source host; the last node is the destination host. The other four nodes are four routers. The first, the third, and the fifth links represent the three LANs; the second and the fourth links represent the two WANs.

2) Services

The data-link layer is located between the physical and the network layers. The data link layer provides services to the network layer; it receives services from the physical layer. Let us discuss services provided by the data-link layer.

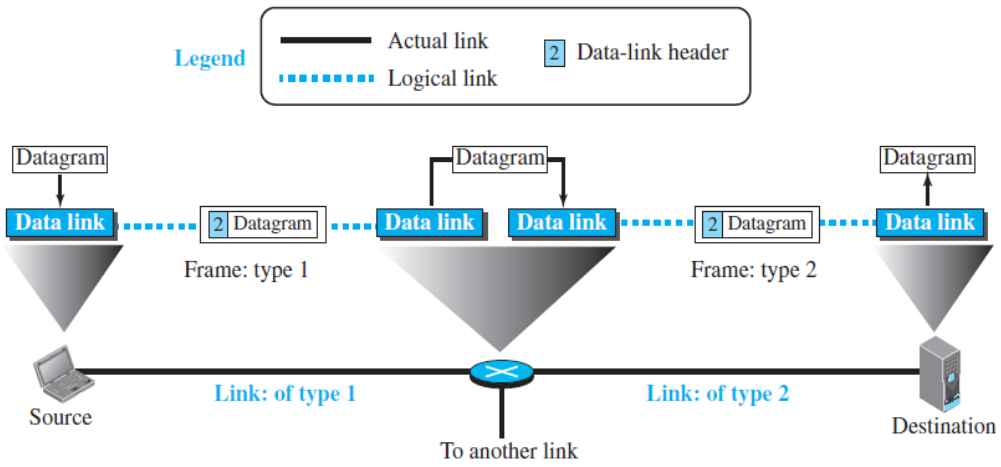
The main scope of the data-link layer is node-to-node. When a packet is travelling in the Internet, the data-link layer of a node (host or router) is responsible for delivering a datagram to the next node in the path.

The data-link layer of the sending node needs to encapsulate the datagram received from the network in a frame, and the data-link layer of the receiving node needs to decapsulate the datagram from the frame.

The data-link layer of the source host needs only to encapsulate, the data-link layer of the destination host needs to decapsulate, but each intermediate node needs to both encapsulate and decapsulate.

Figure 9.3 shows the encapsulation and decapsulation at the data-link layer. The datagram received by the data-link layer of the source host is encapsulated in a frame. The frame is logically transported from the source host to the router. The frame is decapsulated at the data-link layer of the router and encapsulated at another frame. The new frame is logically transported from the router to the destination host.

Figure 9.3 *A communication with only three nodes*



Various types of services provided by data link layer is

- ***Framing***
- ***Flow Control***
- ***Error Control***
- ***Congestion Control***

Framing

The first service provided by the data-link layer is **framing**. The data-link layer at each node needs to encapsulate the datagram (packet received from the network layer) in a **frame** before sending it to the next node.

The node also needs to decapsulate the datagram from the frame received on the logical channel. Although we have shown only a header for a frame.

A frame may have both a header and a trailer. Different data-link layers have different formats for framing.

Flow Control

Whenever we have a producer and a consumer, we need to think about flow control. If the producer produces items that cannot be consumed, accumulation of items occurs.

The sending data-link layer at the end of a link is a producer of frames; the receiving data-link layer at the other end of a link is a consumer. If the rate of produced frames is higher than the rate of consumed frames, frames at the receiving end need to be buffered while waiting to be consumed (processed).

The first choice is to let the receiving data-link layer drop the frames if its buffer is full. The second choice is to let the receiving data-link layer send a feedback to the sending data-link layer to ask it to stop or slow down. Different data-link-layer protocols use different strategies for flow control.

Error Control

A frame in a data-link layer needs to be changed to bits, transformed to electromagnetic signals, and transmitted through the transmission media. At the receiving node, electromagnetic signals are received, transformed to bits, and put together to create a frame. Electromagnetic signals are susceptible to error; a frame is susceptible to error. The error needs first to be detected. After detection, it needs to be either corrected at the receiver node or discarded and retransmitted by the sending node.

Congestion Control

A link may be congested with frames, which may result in frame loss; most data-link-layer protocols do not directly use a congestion control to alleviate congestion, although some wide-area networks do. In general, congestion control is considered an issue in the network layer or the transport layer because of its end-to-end nature.

3) Two Categories of Links

Two nodes are physically connected by a transmission medium such as cable or air, we need to remember that the data-link layer controls how the medium is used. We can have a data-link layer that uses the whole capacity of the medium. We can also have a data-link layer that uses only part of the capacity of the link. Two various categories of links is :

A ***point-to-point link:*** In a point-to-point link, the link is dedicated to the two devices

A ***broadcast link***: in a broadcast link, the link is shared between several pairs of devices.

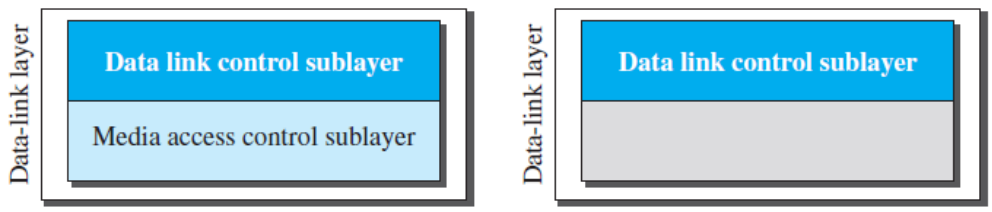
For example: When two friends use the traditional home phones to chat, they are using a point-to-point link.

4) Two Sub-layers

The services provided by the link layer, we can divide the data-link layer into two sub-layers:

- **Data link control (DLC):** The data link control sub-layer deals with all issues common to both point-to-point and broadcast links.
- **Media access control (MAC):** The media access control sub-layer deals only with issues specific to broadcast links.

Figure 9.4 *Dividing the data-link layer into two sublayers*



a. Data-link layer of a broadcast link

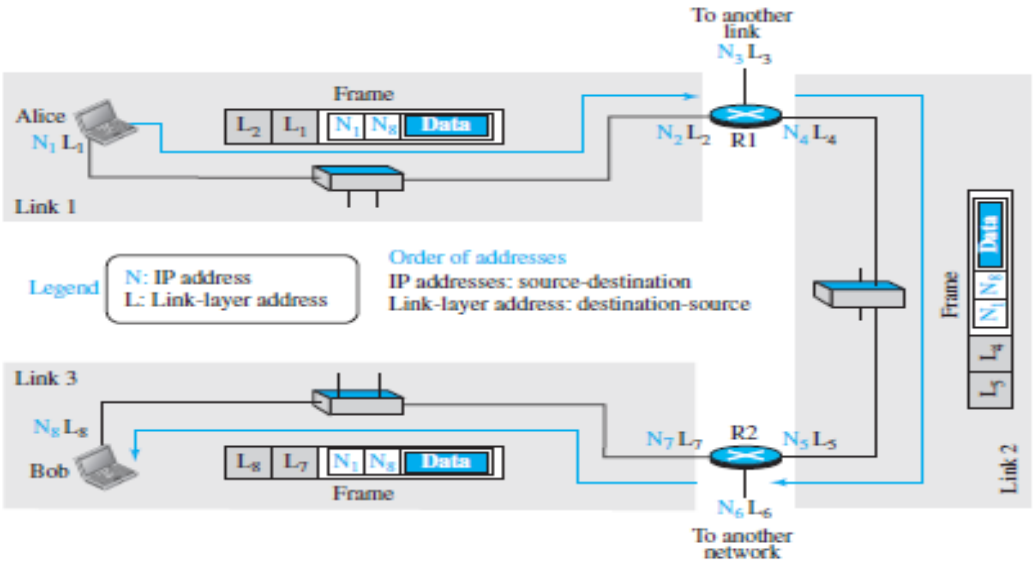
b. Data-link layer of a point-to-point link

LINK-LAYER ADDRESSING

- The source and destination IP addresses define the two ends but cannot define which links the datagram should pass through.
- The IP addresses in a datagram should not be changed. If the destination IP address in a datagram changes, the packet never reaches its destination
- If the source IP address in a datagram changes, the destination host or a router can never communicate with the source if a response needs to be sent back or an error needs to be reported back to the source.
- Addressing mechanism in a connectionless internetwork is the link-layer addresses of the two nodes.
    - **A *link-layer address* is sometimes called a *link address*.**
    - **A *physical address*, and sometimes a *MAC address*.**

Since a link is controlled at the data-link layer, the addresses need to belong to the data-link layer. When a datagram passes from the network layer to the data-link layer, the datagram will be encapsulated in a frame and two data-link addresses are added to the frame header. These two addresses are changed every time the frame moves from one link to another.

Figure 9.5 *IP addresses and link-layer addresses in a small internet*



In the above figure **three links and two routers**. For each host, we have shown two addresses, the IP addresses (N) and the link-layer addresses (L).

In the figure the three frames, one in each link. Each frame carries the same datagram with the same source and destination addresses (**N1** and **N8**), but the link-layer addresses of the frame change from link to link. In link 1, the link-layer addresses are L1 and L2. In link 2, they are L4 and L5. In link 3, they are L7 and L8.

For IP addresses, the source address comes before the destination address; for link-layer addresses, the destination address comes before the source. The datagram's and frames are designed in this way, and we follow the design. We may raise several questions:

- If the IP address of a router does not appear in any datagram sent from a source to a destination, why do we need to assign IP addresses to routers? The answer is that in some protocols a router may act as a sender or receiver of a datagram.
- Why do we need more than one IP address in a router, one for each interface? The answer is that an interface is a connection of a router to a link.
- How are the source and destination IP addresses in a packet determined? The answer is that the host should know its own IP address, which becomes the source IP address in the packet.
- How are the source and destination link-layer addresses determined for each link? Again, each hop (router or host) should know its own link-layer address. The destination link-layer address is determined by using the Address Resolution Protocol.

Three Types of addresses

Some link-layer protocols define three types of addresses: unicast, multicast, and broadcast.

Unicast Address

Each host or each interface of a router is assigned a unicast address. Unicasting means one-to-one communication. A frame with a unicast address destination is destined only for one entity in the link.

Example

The unicast link-layer addresses in the most common LAN, Ethernet, are 48 bits (six bytes) that are presented as 12 hexadecimal digits separated by colons; for example, the following is a link-layer address of a computer.

A3:34:45:11:92:F1

Multicast Address

Some link-layer protocols define multicast addresses. Multicasting means one-to-many communication. However, the jurisdiction is local (inside the link).

Example

The multicast link-layer addresses in the most common LAN, Ethernet, are 48 bits (six bytes) that are presented as 12 hexadecimal digits separated by colons. The second digit, however, needs to be an even number in hexadecimal. The following shows a multicast address:

A2:34:45:11:92:F1

Broadcast Address

Some link-layer protocols define a broadcast address. Broadcasting means one-to-all communication. A frame with a destination broadcast address is sent to all entities in the link.

Example :

The broadcast link-layer addresses in the most common LAN, Ethernet, are 48 bits, all 1s, that are presented as 12 hexadecimal digits separated by colons. The following shows a broadcast address:

FF:FF:FF:FF:FF:FF

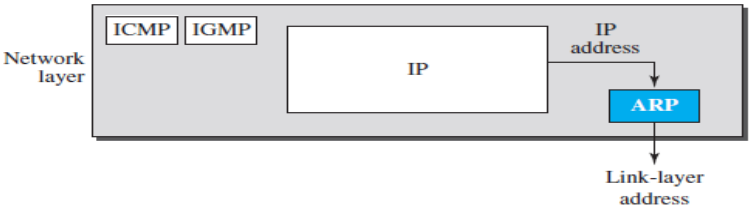
Address Resolution Protocol (ARP)

A node has an IP datagram to send to another node in a link, it has the IP address of the receiving node. The source host knows the IP address of the default router. Each router except the last one in the path gets the IP address of the next router by using its forwarding table. The last router knows the IP address of the destination host.

The IP address of the next node is not helpful in moving a frame through a link; we need the link-layer address of the next node. This is the time when the **Address Resolution Protocol (ARP)** becomes helpful. The ARP protocol is one of the auxiliary protocols defined in the network layer.

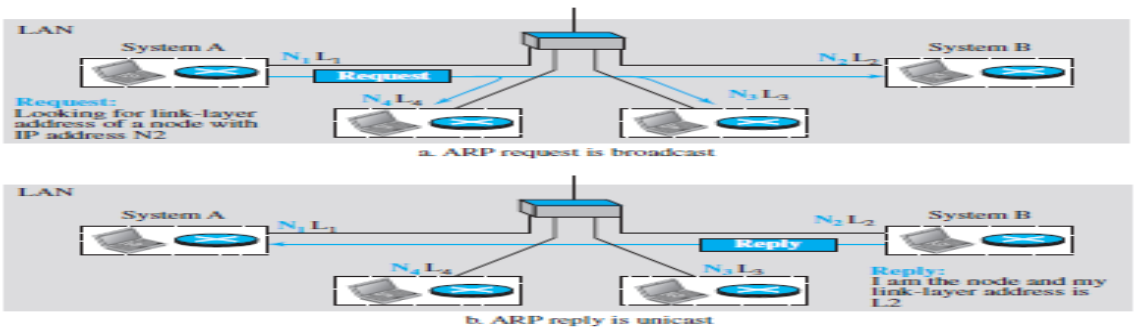
ARP accepts an IP address from the IP protocol, maps the address to the corresponding link-layer address, and passes it to the data-link layer.

Figure 9.6 *Position of ARP in TCP/IP protocol suite*



Anytime a host or a router needs to find the link-layer address of another host or router in its network, it sends an ARP request packet. The packet includes the link-layer and IP addresses of the sender and the IP address of the receiver. Because the sender does not know the link-layer address of the receiver, the query is broadcast over the link using the link-layer broadcast address.

Figure 9.7 *ARP operation*



Every host or router on the network receives and processes the ARP request packet, but only the intended recipient recognizes its IP address and sends back an ARP response packet. The response packet contains the recipient's IP and link-layer addresses. The packet is unicast directly to the node that sent the request packet.

In Figure 9.7a, the system on the left (A) has a packet that needs to be delivered to another system (B) with IP address **N2**. System A needs to pass the packet to its data-link layer for the actual delivery, but it does not know the physical address of the recipient. It uses the services of ARP by asking the ARP protocol to send a broadcast ARP request packet to ask for the physical address of a system with an IP address of **N2**.

This packet is received by every system on the physical network, but only system B will answer it, as shown in Figure 9.7b. System B sends an ARP reply packet that includes its physical address. Now system A can send all the packets it has for this destination using the physical address it received.

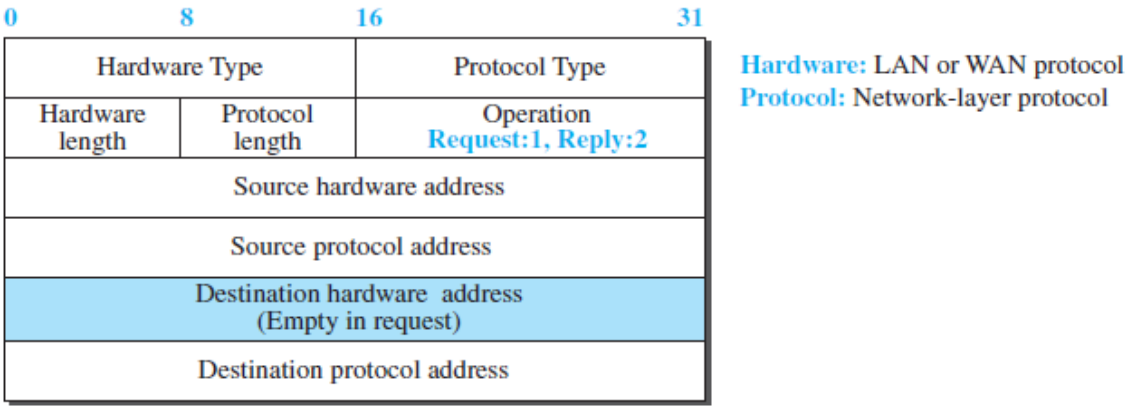
Packet Format

The format of an ARP packet. The names of the fields are self explanatory.

- The *hardware type* field defines the type of the link-layer protocol.
- Ethernet is given the type 1.
- The *protocol type* field defines the network-layer protocol:
- IPv4 protocol is (0800)16.
- The source hardware and source protocol addresses are variable-length fields defining the link-layer and network-layer addresses of the sender.
- The destination hardware address and destination protocol address fields define the receiver link-layer and network-layer addresses.

An ARP packet is encapsulated directly into a data-link frame. The frame needs to have a field to show that the payload belongs to the ARP and not to the network-layer datagram.

Figure 9.8 *ARP packet*



Error detection and Correction: *Cyclic codes, Checksum, Forward error Correction.*

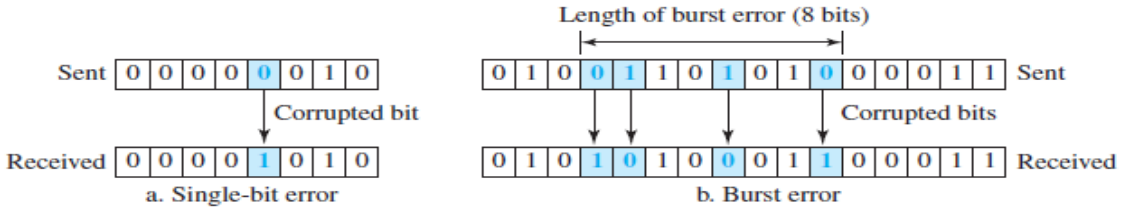
INTRODUCTION

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.

- The term ***single-bit error*** means that only 1 bit of a given data unit (such as a byte, character, or packet) is changed from 1 to 0 or from 0 to 1.
- 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 10.1 *Single-bit and burst error*



A burst error is more likely to occur than a single-bit error because the duration of the noise signal is normally longer than the duration of 1 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 second can affect 10 bits; if we are sending data at 1 Mbps, the same noise can affect 10,000 bits.

Redundancy

The central concept in detecting or correcting errors is **redundancy**. To be able to detect or correct errors, we need to send some extra bits with our data. These redundant bits are added by the sender and removed by the receiver. Their presence allows the receiver to detect or correct corrupted bits.

Detection versus Correction

The correction of errors is more difficult than the detection.

- In **error detection**, we are only looking to see if any error has occurred. The answer is a simple yes or no. A single-bit error is the same for us as a burst error.
- In **error correction**, we need to know the exact number of bits that are corrupted and, more importantly, their location in the message. The number of errors and the size of the message are important factors. If we need to correct a single error in an 8-bit data unit, we need to consider eight possible error locations; if we need to correct two errors in a data unit of the same size, we need to consider 28 (permutation of 8 by 2) possibilities. You can imagine the receiver's difficulty in finding 10 errors in a data unit of 1000 bits.

Coding

Redundancy is achieved through various coding schemes. The sender adds redundant bits through a process that creates a relationship between the redundant bits and the actual data bits. The receiver checks the relationships between the two sets of bits to detect errors. The ratio of redundant bits to data bits and the robustness of the process are important factors in any coding scheme.

We can divide coding schemes into two broad categories:

- ***block coding*** *and*
- ***convolution coding***.

BLOCK CODING

In block coding, we divide our message into blocks, each of *k* bits, called ***datawords*.** We add *r* redundant bits to each block to make the length $n = k + r$. The resulting *n*-bit blocks are called ***codewords*.** How the extra *r* bits are chosen or calculated is something we will discuss later. For the moment, it is important to know that we have a set of datawords, each of size *k*, and a set of codewords, each of size of *n*. With *k* bits, we can create a combination of 2^k datawords; with *n* bits, we can create a combination of 2^n codewords. Since $n > k$, the number of possible codewords is larger than the number of possible datawords. The block coding process is one-to-one; the same dataword is always encoded as the same codeword. This means that we have $2^n - 2^k$ codewords that are not used. We call these codewords invalid or illegal.

If the receiver receives an invalid codeword, this indicates that the data was corrupted during transmission.

Error Detection

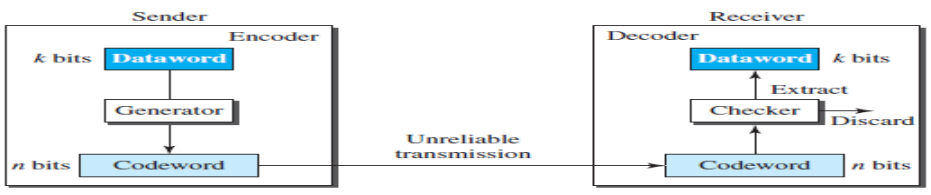
How can errors be detected by using block coding? If the following two conditions are met, the receiver can detect a change in the original codeword.

1. The receiver has (or can find) a list of valid codewords.

2. The original codeword has changed to an invalid one.

Figure 10.2 shows the role of block coding in error detection. The sender creates codewords out of datawords by using a generator that applies the rules and procedures of encoding Each codeword sent to the receiver may change during transmission. If the received codeword is the same as one of the valid codewords, the word is accepted; the corresponding dataword is extracted for use. If the received codeword is not valid, it is discarded. However, if the codeword is corrupted during transmission but the received word still matches a valid codeword, the error remains undetected.

Figure 10.2 *Process of error detection in block coding*



Example 10.1

Let us assume that $k = 2$ and $n = 3$. Table 10.1 shows the list of datawords and codewords. Later, we will see how to derive a codeword from a dataword.

Table 10.1 *A code for error detection in Example 10.1*

Dataword	*Codeword*	*Dataword*	*Codeword*
00	000	10	101
01	011	11	110

Assume the sender encodes the dataword 01 as 011 and sends it to the receiver. Consider the following cases:

1. The receiver receives 011. It is a valid codeword. The receiver extracts the dataword 01 from it.

2. The codeword is corrupted during transmission, and 111 is received (the leftmost bit is corrupted). This is not a valid codeword and is discarded.

3. The codeword is corrupted during transmission, and 000 is received (the right two bits are corrupted). This is a valid codeword. The receiver incorrectly extracts the dataword 00. Two corrupted bits have made the error undetectable.

Hamming Distance

The **Hamming distance** between two words (of the same size) is the number of differences between the corresponding bits. We show the Hamming distance between two words x and y as $\boldsymbol{d}(x, y)$. We may wonder why Hamming distance is important for error detection. The reason is that the Hamming distance between the received codeword and the sent codeword is the number of bits that are corrupted during transmission.

For example,

if the codeword 00000 is sent and 01101 is received, 3 bits are in error and the Hamming distance between the two is $\boldsymbol{d}(00000, 01101) = 3$. In other words, if the Hamming distance between the sent and the received codeword is not zero, the codeword has been corrupted during transmission. The Hamming distance can easily be found if we apply the XOR operation () on the two words and count the number of 1s in the result.

Example 10.2

Let us find the Hamming distance between two pairs of words.

1. The Hamming distance $\boldsymbol{d}(000, 011)$ is 2 because (000 011) is 011 (two 1s).

2. The Hamming distance $\boldsymbol{d}(10101, 11110)$ is 3 because (10101 11110) is 01011 (three 1s).

Minimum Hamming Distance for Error Detection

In a set of codewords, the **minimum Hamming distance** is the smallest Hamming distance between all possible pairs of codewords. Now let us find the minimum Hamming distance in a code if we want to be able to detect up to s errors. If s errors occur during transmission, the Hamming distance between the sent codeword and received codeword is s. If our system is to detect up to s

errors, the minimum distance between the valid codes must be (s + 1), so that the received codeword does not match a valid codeword.

In other words, if the minimum distance between all valid codewords is (s + 1), the received codeword cannot be erroneously mistaken for another codeword. The error will be detected. We need to clarify a point here: Although a code with $d$min = s + 1 may be able to detect more than s errors in some special cases, only s or fewer errors are guaranteed to be detected.

Let us assume that the sent codeword x is at the center of a circle with radius s. All received codewords that are created by 0 to s errors are points inside the circle or on the perimeter of the circle. All other valid codewords must be outside the circle, as shown in Figure 10.3. This means that $d$min must be an integer greater than s or $d$min = s + 1.

CYCLIC CODES

Cyclic codes are special linear block codes with one extra property. In a **cyclic code,** if a codeword is cyclically shifted (rotated), the result is another codeword.

For example,

If 1011000 is a codeword and we cyclically left-shift, then 0110001 is also a codeword. In this case, if we call the bits in the first word $a0$ to $a6$, and the bits in the second word $b0$ to $b6$, we can shift the bits by using the following:

$$b_1 = a_0 \qquad b_2 = a_1 \qquad b_3 = a_2 \qquad b_4 = a_3 \qquad b_5 = a_4 \qquad b_6 = a_5 \qquad b_0 = a_6$$

In the rightmost equation, the last bit of the first word is wrapped around and becomes the first bit of the second word.

Cyclic Redundancy Check

Cyclic codes called the **cyclic redundancy check (CRC)**, which is used in networks such as LANs and WANs.

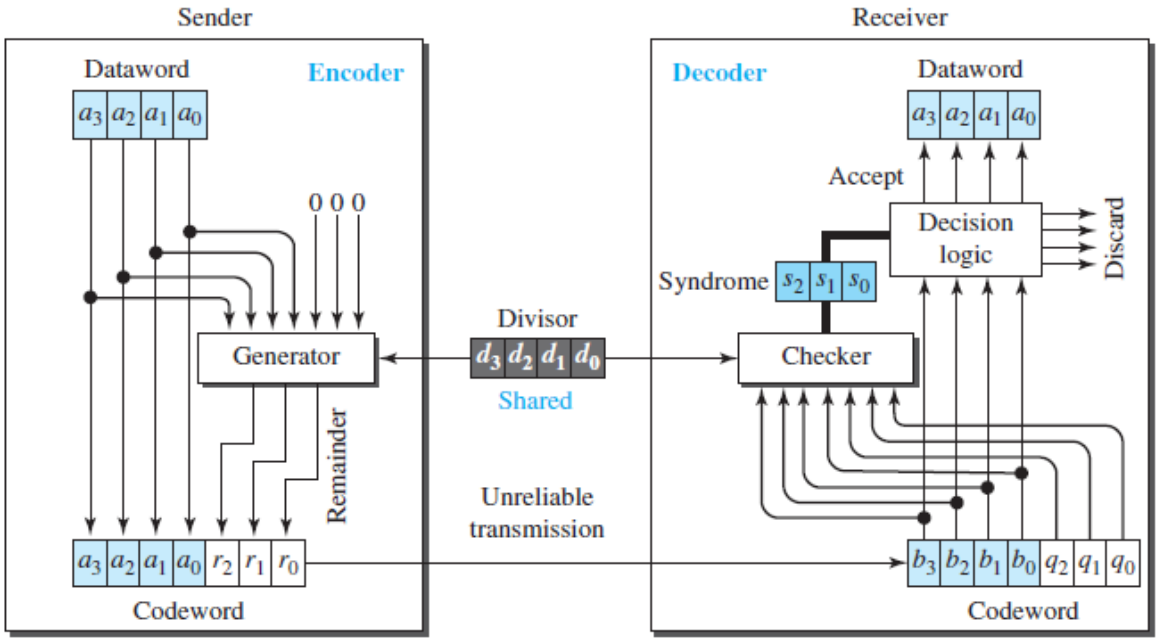
Table 10.3 shows an example of a CRC code. We can see both the linear and cyclic properties of this code.

Table 10.3 *A CRC code with C(7, 4)*

Dataword	*Codeword*	*Dataword*	*Codeword*
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

Figure 10.5 shows one possible design for the encoder and decoder.

Figure 10.5 *CRC encoder and decoder*



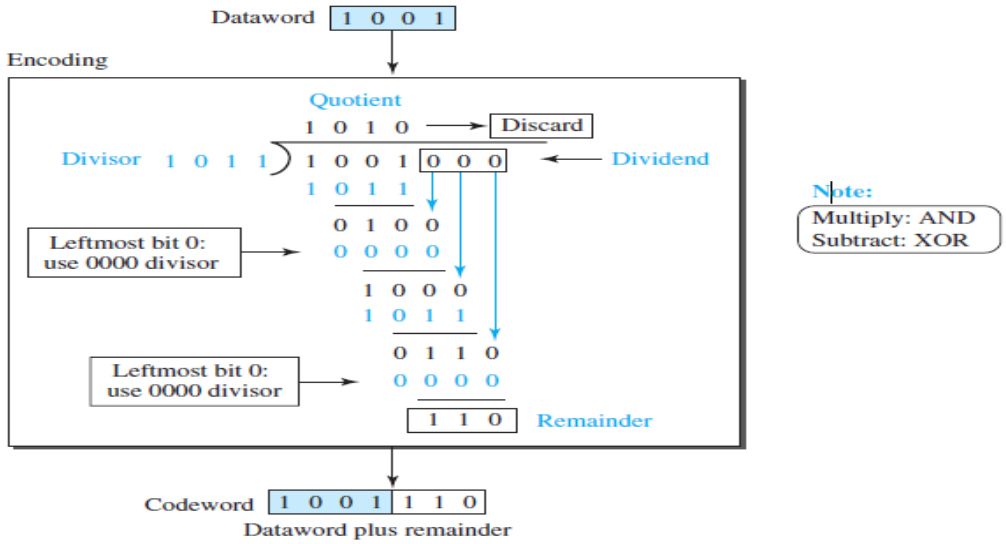
In the encoder, the dataword has *k* bits (4 here); the codeword has *n* bits (7 here). The size of the dataword is augmented by adding *n* - *k* (3 here) 0s to the right-hand side of the word. The *n*-bit result is fed into the generator. The generator uses a divisor of size $n - k + 1$ (4 here), predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder (*r*2*r*1*r*0) is appended to the dataword to create the codeword.

The decoder receives the codeword (possibly corrupted in transition). A copy of all *n* bits is fed to the checker, which is a replica of the generator. The remainder produced by the checker is a syndrome of *n*- *k* (3 here) bits, which is fed to the decision logic analyzer. The analyzer has a simple function. If the syndrome bits are all 0s, the 4 leftmost bits of the codeword are accepted as the dataword (interpreted as no error); otherwise, the 4 bits are discarded (error).

Encoder

Let us take a closer look at the encoder. The encoder takes a dataword and augments it with $n - k$ number of 0s. It then divides the augmented dataword by the divisor, as shown in Figure 10.6.

Figure 10.6 *Division in CRC encoder*



The process of modulo-2 binary division is the same as the familiar division process we use for decimal numbers. However, addition and subtraction in this case are the same; we use the XOR operation to do both.

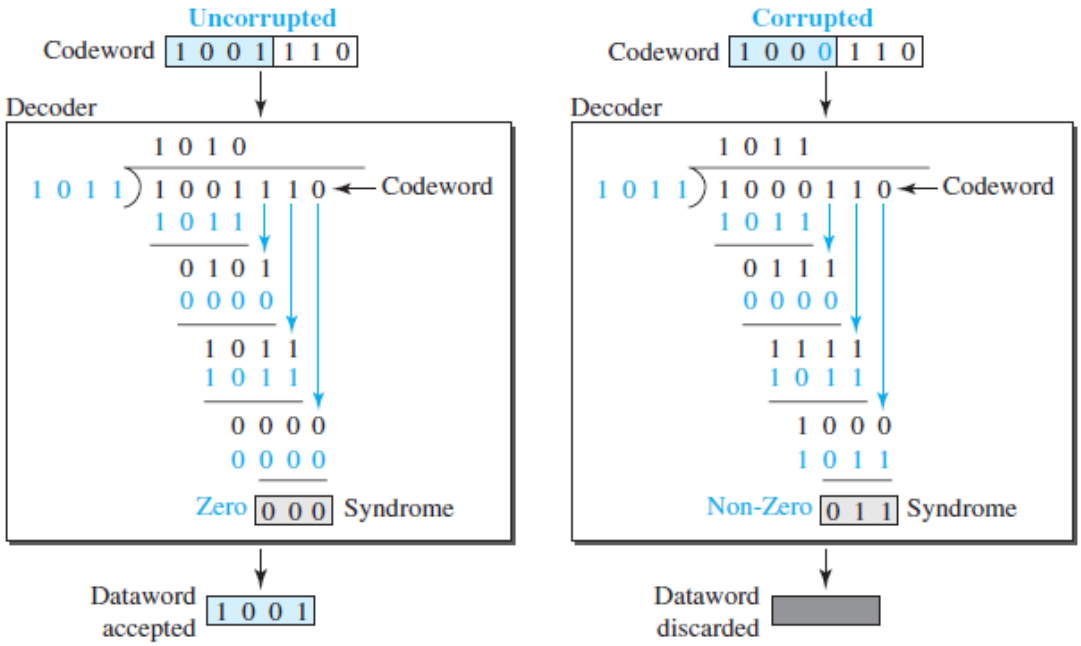
As in decimal division, the process is done step by step. In each step, a copy of the divisor is XORed with the 4 bits of the dividend. The result of the XOR operation (remainder) is 3 bits (in this case), which is used for the next step after 1 extra bit is pulled down to make it 4 bits long. There is one important point we need to remember in this type of division. If the leftmost bit of the dividend (or the part used in each step) is 0, the step cannot use the regular divisor; we need to use an all-0s divisor. When there are no bits left to pull down, we have a result. The 3-bit remainder forms the **check bits** ($r2$, $r1$, and $r0$). They are appended to the dataword to create the codeword.

Decoder

The codeword can change during transmission. The decoder does the same division process as the encoder. The remainder of the division is the syndrome. If the syndrome is all 0s, there is no error with a high probability; the dataword is separated from the received codeword and accepted. Otherwise, everything is discarded.

Figure 10.7 shows two cases: The left-hand figure shows the value of the syndrome when no error has occurred; the syndrome is 000. The right-hand part of the figure shows the case in which there is a single error. The syndrome is not all 0s (it is 011).

Figure 10.7 *Division in the CRC decoder for two cases*



Divisor

We may be wondering how the divisor 1011 is chosen. This depends on the expectation we have from the code.

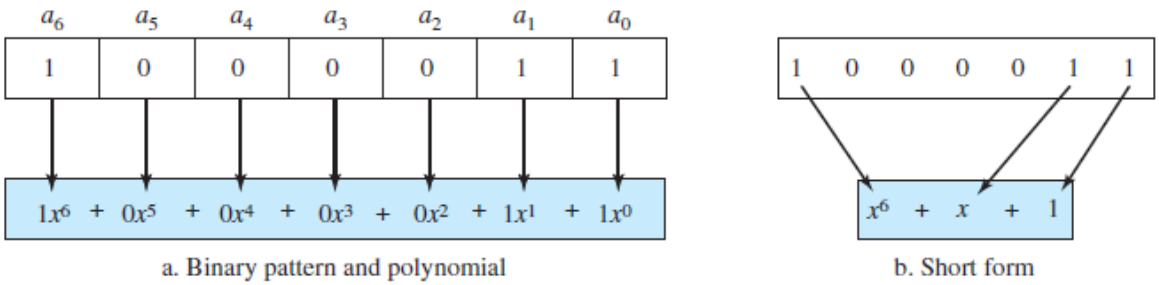
Polynomials

To understand cyclic codes and how they can be analyzed is to represent them as polynomials.

A pattern of 0s and 1s can be represented as a **polynomial** with coefficients of 0 and 1. The power of each term shows the position of the bit; the coefficient shows the value of the bit.

Figure 10.8 shows a binary pattern and its polynomial representation. In **Figure 10.8a** we show how to translate a binary pattern into a polynomial; in **Figure 10.8b** we show how the polynomial can be shortened by removing all terms with zero coefficients and replacing $x1$ by x and $x0$ by 1.

Figure 10.8 *A polynomial to represent a binary word*



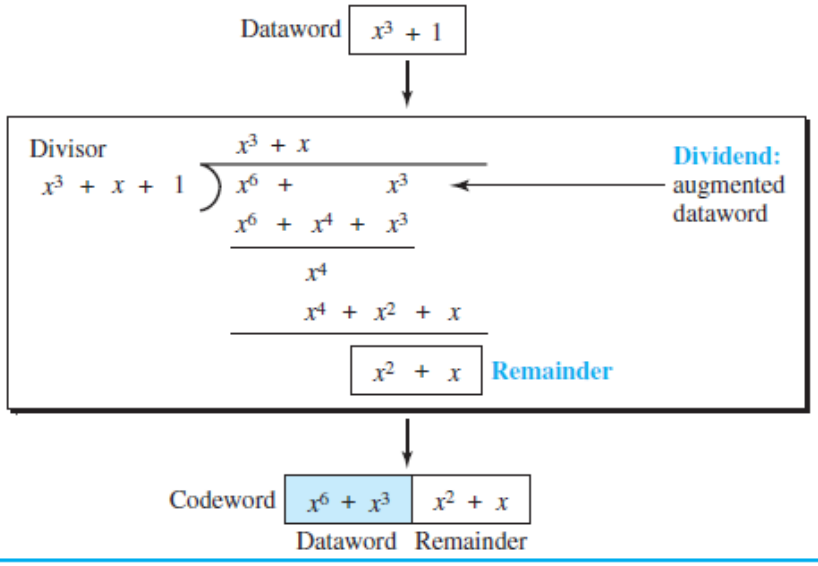
Degree of a Polynomial

The degree of a polynomial is the highest power in the polynomial. For example, the degree of the polynomial $x6 + x + 1$ is 6. Note that the degree of a polynomial is 1 less than the number of bits in the pattern. The bit pattern in this case has 7 bits.

Cyclic Code Encoder Using Polynomials

We show the creation of a codeword from a dataword. Figure 10.9 is the polynomial version of Figure 10.6. We can see that the process is shorter. The dataword 1001 is represented as $x3 + 1$. The divisor 1011 is represented as $x3 + x + 1$. To find the augmented dataword, we have left-shifted the dataword 3 bits (multiplying by $x3$). The result is $x6 + x3$. Division is straightforward. We divide the first term of the dividend, $x6$, by the first term of the divisor, $x3$. The first term of the quotient is then $x6/x3$, or $x3$. Then we multiply $x3$ by the divisor and subtract (according to our previous definition of subtraction) the result from the dividend. The result is $x4$, with a degree greater than the divisor's degree; we continue to divide until the degree of the remainder is less than the degree of the divisor.

Figure 10.9 *CRC division using polynomials*



It can be seen that the polynomial representation can easily simplify the operation of division in this case, because the two steps involving all-0s divisors are not needed here. (Of course, one could argue that the all-0s divisor step can also be eliminated in binary division.) In a polynomial representation, the divisor is normally referred to as the ***generator polynomial*** $t(x)$.

Cyclic Code Analysis

We can analyze a cyclic code to find its capabilities by using polynomials. We define the following, where $f(x)$ is a polynomial with binary coefficients.

Dataword: $d(x)$ **Codeword: $c(x)$** **Generator: $g(x)$** **Syndrome: $s(x)$** **Error: $e(x)$**

If $s(x)$ is not zero, then one or more bits is corrupted. However, if $s(x)$ is zero, either no bit is corrupted or the decoder failed to detect any errors. (Note that ¦ means divide).

In a cyclic code,

1. If $s(x)$ ¦ 0, one or more bits is corrupted.

2. If $s(x) = 0$, either

a. No bit is corrupted, or

b. Some bits are corrupted, but the decoder failed to detect them.

In our analysis we want to find the criteria that must be imposed on the generator, $g(x)$ to detect the type of error we especially want to be detected. Let us first find the relationship among the sent codeword, error, received codeword, and the generator. We can say

$$\text{Received codeword} = c(x) + e(x)$$

In other words, the received codeword is the sum of the sent codeword and the error. The receiver divides the received codeword by $g(x)$ to get the syndrome. We can write this as

$$\frac{\text{Received codeword}}{g(x)} = \frac{c(x)}{g(x)} + \frac{e(x)}{g(x)}$$

The first term at the right-hand side of the equality has a remainder of zero (according to the definition of codeword). So the syndrome is actually the remainder of the second term on the right-hand side. If this term does not have a remainder (syndrome = 0), either $e(x)$ is 0 or $e(x)$ is divisible by $g(x)$. We do not have to worry about the first case (there is no error); the second case is very important. Those errors that are divisible by $g(x)$ are not caught.

CHECKSUM

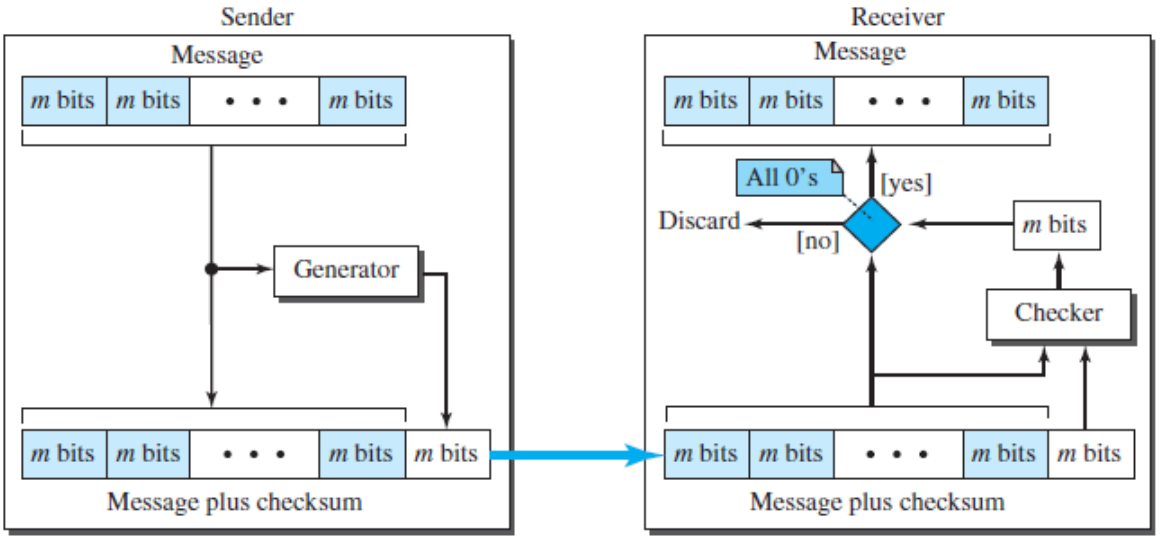
Checksum is an error-detecting technique that can be applied to a message of any length. In the Internet, the checksum technique is mostly used at the network and transport layer rather than the data-link layer. However, to make our discussion of error detecting techniques complete.

At the source, the message is first divided into *m*-bit units. The generator then creates an extra *m*-bit unit called the ***checksum,*** which is sent with the message.

At the destination, the checker creates a new checksum from the combination of the message and sent checksum. If the new checksum is all 0s, the message is accepted; otherwise, the message is discarded .

The checksum unit is not necessarily added at the end of the message; it can be inserted in the middle of the message.

Figure 10.15 *Checksum*



Concept

The idea of the traditional checksum is simple. We show this using a simple example.

Example 10.11

Suppose the message is a list of five 4-bit numbers that we want to send to a destination. In addition to sending these numbers, we send the sum of the numbers.

For example:

If the set of numbers is (7, 11, 12, 0, 6), we send (7, 11, 12, 0, 6, **36**), where 36 is the sum of the original numbers. The receiver adds the five numbers and compares the result with the sum. If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum. Otherwise, there is an error somewhere and the message is not accepted.

One's Complement Addition

The previous example has one major drawback. Each number can be written as a 4-bit word (each is less than 15) except for the sum. One solution is to use **one's complement** arithmetic. In this arithmetic, we can represent unsigned numbers between 0 and 2^m - 1 using only *m* bits. If the number has more than *m* bits, the extra leftmost bits need to be added to the *m* rightmost bits (wrapping).

Example 10.12

In the previous example, the decimal number 36 in binary is $(100100)_2$. To change it to a 4-bit number we add the extra leftmost bit to the right four bits as shown below.

$$(10)_2 + (0100)_2 = (0110)_2 \rightarrow (6)_{10}$$

Instead of sending 36 as the sum, we can send 6 as the sum (7, 11, 12, 0, 6, **6**). The receiver can add the first five numbers in one's complement arithmetic. If the result is 6, the numbers are accepted; otherwise, they are rejected.

Checksum

We can make the job of the receiver easier if we send the complement of the sum, the checksum. In one's complement arithmetic, the complement of a number is found by completing all bits (changing all 1s to 0s and all 0s to 1s). This is the same as subtracting the number from 2^m - 1.

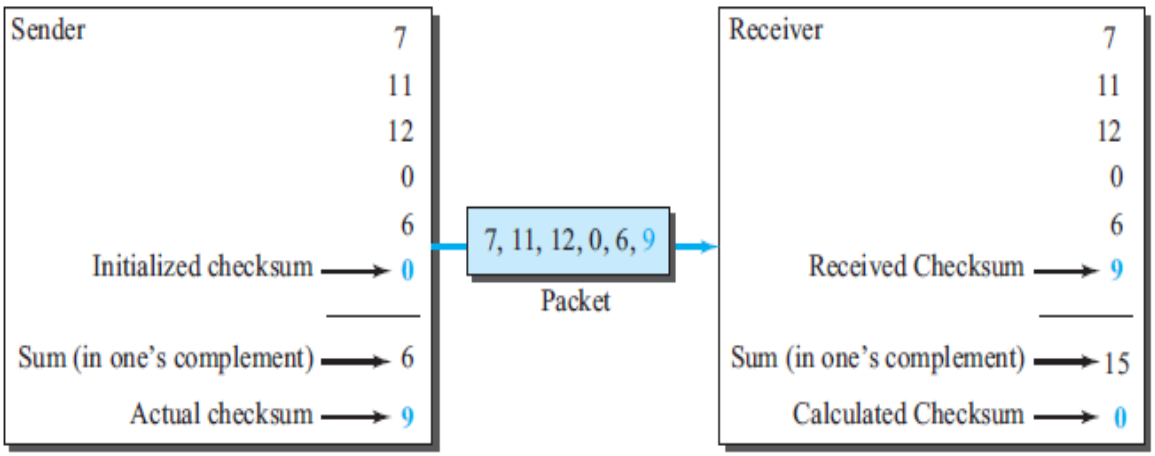
In one's complement arithmetic, we have two 0s: one positive and one negative, which are complements of each other. The positive zero has all *m* bits set to 0; the negative zero has all bits set to 1 (it is 2^m- 1). If we add a number with its complement, we get a negative zero (a number with all bits set to 1). When the receiver adds all five numbers (including the checksum), it gets a negative zero. The receiver can complement the result again to get a positive zero.

Example 10.13

Let us use the idea of the checksum in Example 10.12. The sender adds all five numbers in one's complement to get the sum = 6. The sender then complements the result to get the checksum = 9, which is 15 − 6. Note that 6 = (0110)2 and 9 = (1001)2; they are complements of each other. The sender sends the five data numbers and the checksum (7, 11, 12, 0, 6, **9**). If there is no corruption in transmission, the receiver receives (7, 11, 12, 0, 6, **9**) and adds them in one's complement to get 15. The sender complements 15 to get 0. This shows that data have not been corrupted.

Figure 10.16 shows the process.

Figure 10.16 *Example 10.13*



Internet Checksum

Traditionally, the Internet has used a 16-bit checksum. The sender and the receiver follow the steps depicted in Table 10.5. The sender or the receiver uses five steps.

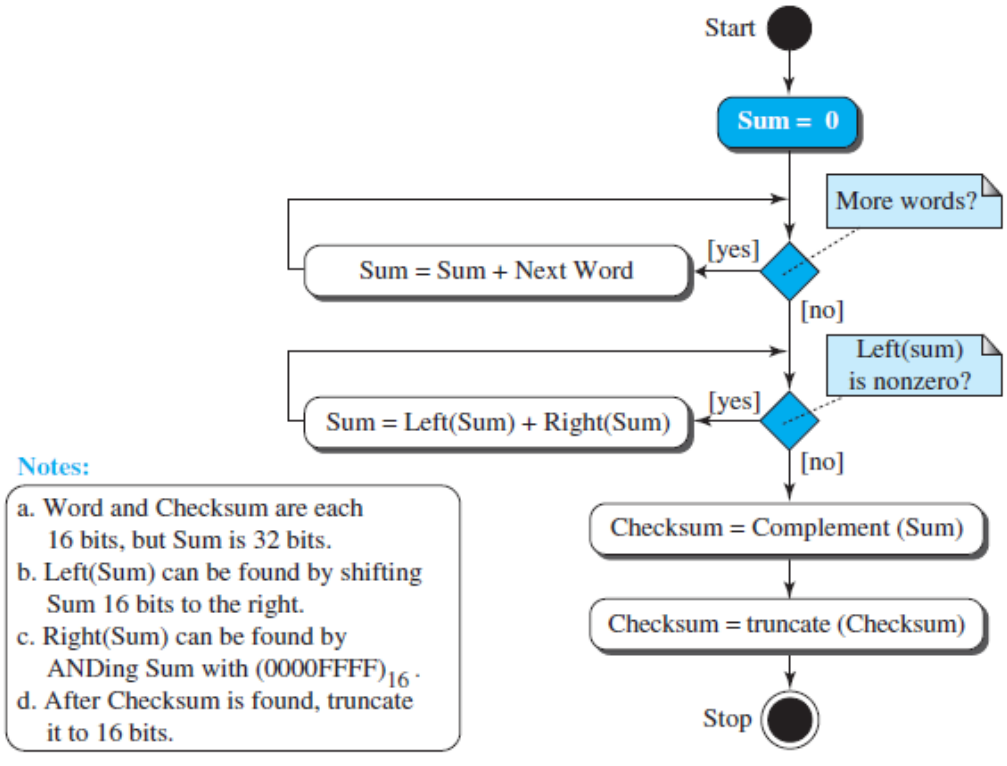
Table 10.5 *Procedure to calculate the traditional checksum*

Sender	*Receiver*
1. The message is divided into 16-bit words.	1. The message and the checksum are received.
2. The value of the checksum word is initially set to zero.	2. The message is divided into 16-bit words.
3. All words including the checksum are added using one's complement addition.	3. All words are added using one's complement addition.
4. The sum is complemented and becomes the checksum.	4. The sum is complemented and becomes the new checksum.
5. The checksum is sent with the data.	5. If the value of the checksum is 0, the message is accepted; otherwise, it is rejected.

Algorithm

We can use the flow diagram of Figure 10.17 to show the algorithm for calculation of the checksum. A program in any language can easily be written based on the algorithm. Note that the first loop just calculates the sum of the data units in two's complement; the second loop wraps the extra bits created from the two's complement calculation to simulate the calculations in one's complement. This is needed because almost all computers today do calculation in two's complement.

Figure 10.17 *Algorithm to calculate a traditional checksum*



Performance

The traditional checksum uses a small number of bits (16) to detect errors in a message of any size (sometimes thousands of bits). However, it is not as strong as the CRC in error-checking capability.

For example: If the value of one word is incremented and the value of another word is decremented by the same amount, the two errors cannot be detected because the sum and checksum remain the same. Also, if the values of several words are incremented but the sum and the checksum do not change, the errors are not detected. Fletcher and Adler have proposed some weighted checksums that eliminate the first problem. However, the tendency in the Internet, particularly in designing new protocols, is to replace the checksum with a CRC.

FORWARD ERROR CORRECTION

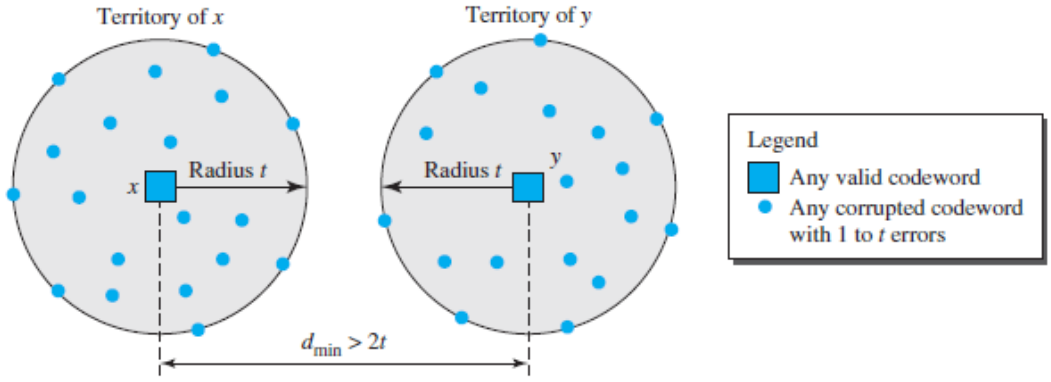
Retransmission of corrupted and lost packets is not useful for real-time multimedia transmission because it creates an unacceptable delay in reproducing: we need to wait until the lost or corrupted packet is resent. We need to correct the error or reproduce the packet immediately. Several schemes have been designed and used in this case that are collectively referred to as **forward error correction** (**FEC**) techniques.

Using Hamming Distance

For error detection, we definitely need more distance. It can be shown that to detect *t* errors, we need to have *d*min = 2*t* + 1. In other words, if we want to correct 10 bits in a packet, we need to make the minimum hamming distance 21 bits, which means a lot of redundant bits need to be sent with the data.

Example, consider the famous BCH code. In this code, if data is 99 bits, we need to send 255 bits (extra 156 bits) to correct just 23 possible bit errors. Most of the time we cannot afford such a redundancy. We give some examples of how to calculate the required bits in the practice set. Figure 10.20 shows the geometrical representation of this concept.

Figure 10.20 *Hamming distance for error correction*



Using XOR

Another recommendation is to use the property of the exclusive OR operation as shown below.

$$R = P_1 \oplus P_2 \oplus \ldots \oplus P_i \oplus \ldots \oplus P_N \quad \rightarrow \quad P_i = P_1 \oplus P_2 \oplus \ldots \oplus R \oplus \ldots \oplus P_N$$

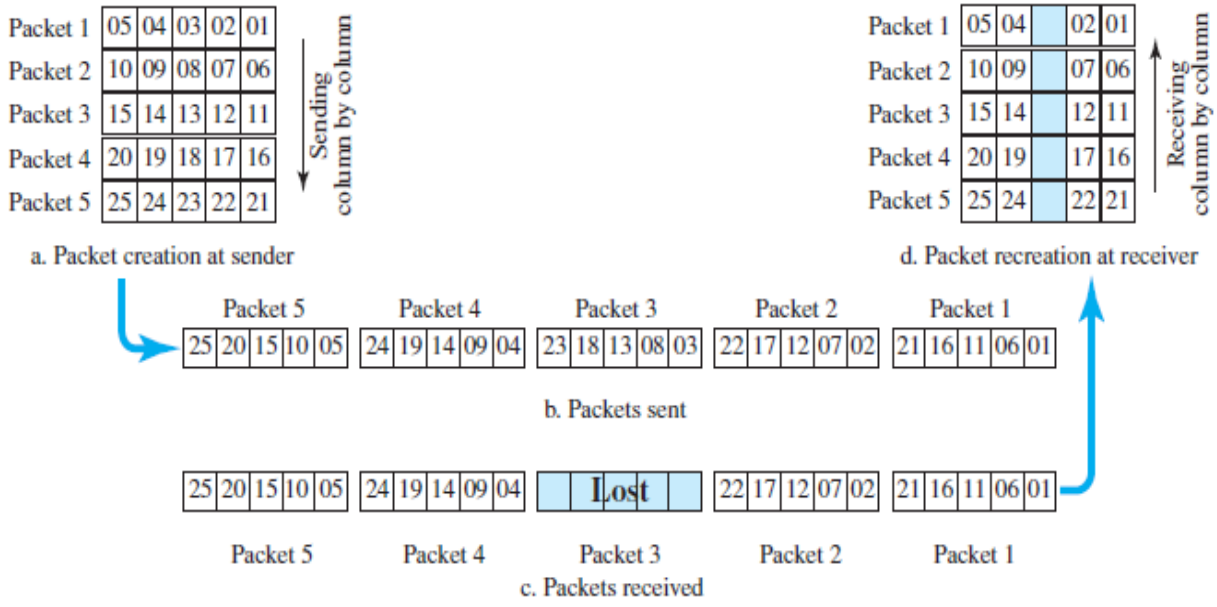
If we apply the exclusive OR operation on *N* data items (P1 to P*N*), we can recreate any of the data items by exclusive-ORing all of the items, replacing the one to be created by the result of the previous operation (R). This means that we can divide a packet into *N* chunks, create the exclusive OR of all the chunks and send *N* + 1 chunks. If any chunk is lost or corrupted, it can be created at the receiver site. Now the question is what should the value of *N* be. If *N* = 4, it means that we need to send 25 percent extra data and be able to correct the data if only one out of four chunks is lost.

Chunk Interleaving

Another way to achieve FEC in multimedia is to allow some small chunks to be missing at the receiver. We cannot afford to let all the chunks belonging to the same packet be missing; however, we can afford to let one chunk be missing in each packet.

Figure 10.21 shows that we can divide each packet into 5 chunks (normally the number is much larger). We can then create data chunk by chunk (horizontally), but combine the chunks into packets vertically. In this case, each packet sent carries a chunk from several original packets. If the packet is lost, we miss only one chunk in each packet, which is normally acceptable in multimedia communication.

Figure 10.21 *Interleaving*



Combining Hamming Distance and Interleaving

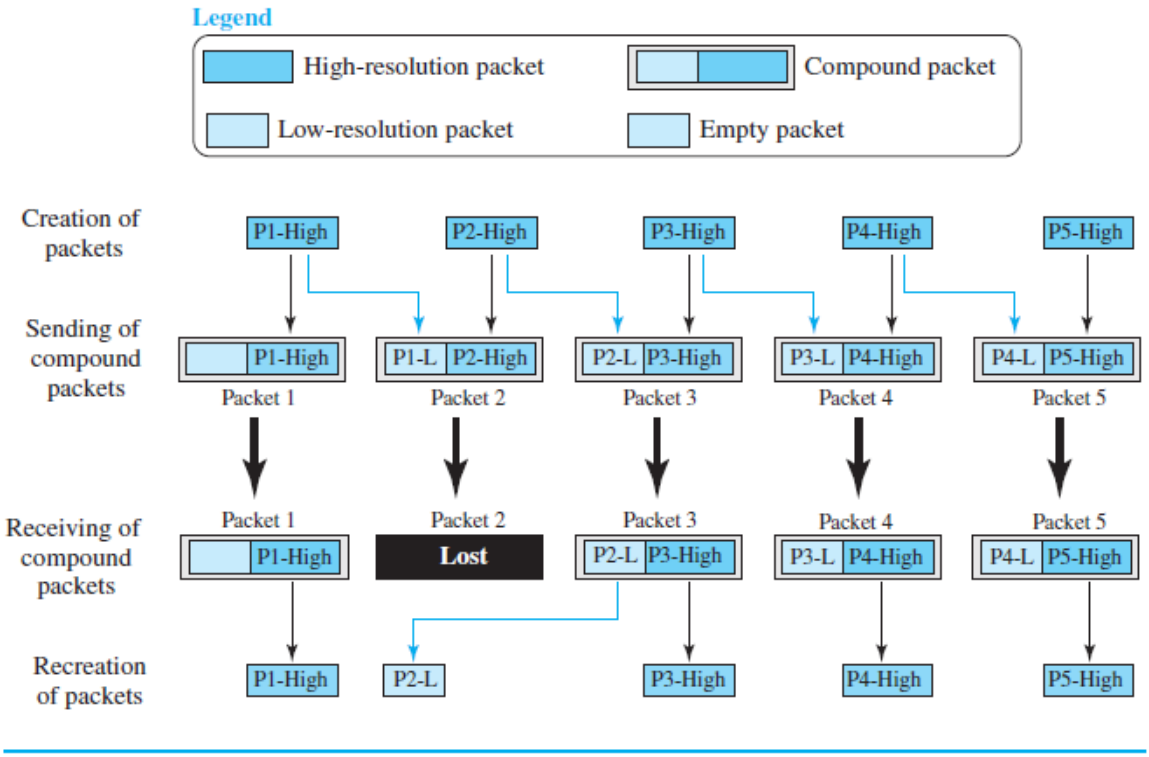
Hamming distance and interleaving can be combined. We can first create *n*-bit packets that can correct *t*-bit errors. Then we interleave *m* rows and send the bits column by column. In this way, we can automatically correct burst errors up to $m \times t$-bit errors.

Compounding High- and Low-Resolution Packets

Still another solution is to create a duplicate of each packet with a low-resolution redundancy and combine the redundant version with the next packet.

For example, we can create four low-resolution packets out of five high-resolution packets and send them as shown in Figure 10.22. If a packet is lost, we can use the low-resolution version from the next packet. Note that the low-resolution section in the first packet is empty. In this method, if the last packet is lost, it cannot be recovered, but we use the low-resolution version of a packet if the lost packet is not the last one. The audio and video reproduction does not have the same quality, but the lack of quality is not recognized most of the time.

Figure 10.22 *Compounding high- and low-resolution packets*



Data link Control: *DLC services, Data link layer protocols, HDLC, Point to Point Protocol.*

DLC services**:**

The **data link control (DLC)** deals with procedures for communication between two adjacent nodes. **DLC** performs node-to-node communication whether link is Local Area Network or in Broadcast.

The main functions of Data link control are

- *framing* and
- *flow control and*
- *Error control*.

1.1 Framing:

- Data transmission in the physical layer means moving bits in the form of a signal from the source to the destination.
- The physical layer provides bit synchronization to ensure that the sender and receiver use the same bit durations and timing.
- The data-link layer needs to pack bits into frames, so that each frame is distinguishable from another.
- ***Framing*** in the data-link layer separates a message from one source to a destination by adding a sender address and a destination address.
- The destination address defines where the packet is to go.
- The sender address helps the recipient acknowledge the receipt.
- The whole message could be packed in one frame. One reason is that a frame can be very large, making flow and error control very inefficient.
- When a message is carried in one very large frame, even a single-bit error would require the retransmission of the whole frame.
- When a message is divided into smaller frames, a single-bit error affects only that small frame.

Frame Size

Frame sizes can have two various types fixed **size or variable size**.

In **fixed-size framing**, there is no need for defining the boundaries of the frames; the size itself can be used as a delimiter. An example of this type of framing is the ATM WAN, which uses frames of fixed size called cells.

In **variable-size framing** which are used local area networks, we need a way to define the end of one frame and the beginning of the next. **Variable-size framing** can have two approaches they are:

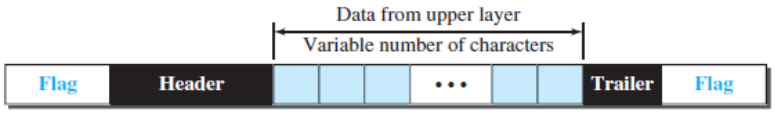
A character-oriented approach *and*

A bit-oriented approach.

Character-Oriented Framing:

- In *character-oriented (or byte-oriented) framing*, data to be carried are 8-bit characters from a coding system such as ASCII.
- The header, which normally carries the source and destination addresses and other control information,
- The trailer, which carries error detection redundant bits, is also multiples of 8 bits. To separate one frame from the next, an 8-bit (1-byte) **flag** is added at the beginning and the end of a frame.
- The flag, composed of protocol-dependent special characters, signals the start or end of a frame. Figure 11.1 shows the format of a frame in a character-oriented protocol.

Figure 11.1 *A frame in a character-oriented protocol*

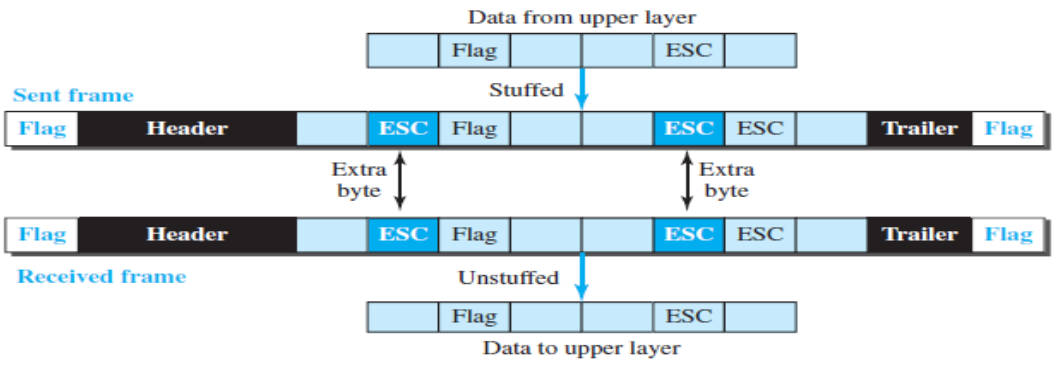


Character-oriented framing was popular when only text was exchanged by the data-link layers. The flag could be selected to be any character not used for text communication. Other types of information such as graphs, audio, and video, any character used for the flag could also be part of the information. If this happens, the receiver, when it encounters this pattern in the middle of the data, thinks it has reached the end of the frame.

To fix this problem, a byte-stuffing strategy was added to character-oriented framing.

In **byte stuffing** (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag. The data section is stuffed with an extra byte. This byte is usually called the ***escape character (ESC)*** and has a predefined bit pattern. Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not as a delimiting flag.

Figure 11.2 *Byte stuffing and unstuffing*



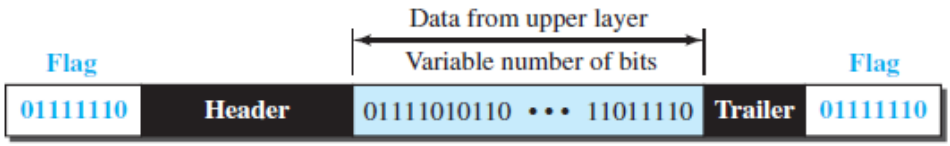
Byte stuffing by the escape character allows the presence of the flag in the data section of the frame, but it creates another problem. What happens if the text contains one or more escape characters followed by a byte with the same pattern as the flag? The receiver removes the escape character, but keeps the next byte, which is incorrectly interpreted as the end of the frame. To solve this problem, the escape characters that are part of the text must also be marked by another escape character. In other words, if the escape character is part of the text, an extra one is added to show that the second one is part of the text.

Character-oriented protocols present another problem in data communications. The universal coding systems in use today, such as Unicode, have 16-bit and 32-bit characters that conflict with 8-bit characters.

Bit-Oriented Framing

In *bit-oriented framing*, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on. In addition to headers (and possible trailers), we still need a delimiter to separate one frame from the other. Most protocols use a special 8-bit pattern flag, 01111110, as the delimiter to define the beginning and the end of the frame, as shown in Figure 11.3.

Figure 11.3 *A frame in a bit-oriented protocol*



This flag can create the same type of problem we saw in the character-oriented protocols. That is, if the flag pattern appears in the data, we need to somehow inform the receiver that this is not the end of the frame. We do this by stuffing 1 single bit (instead of 1 byte) to prevent the pattern from looking like a flag. The strategy is called **bit stuffing.** In bit stuffing, if a 0 and five consecutive 1 bits are encountered, an extra 0 is added. This extra stuffed bit is eventually removed from the data by the receiver. Note that the extra bit is added after one 0 followed by five 1s regardless of the value of the next bit. This guarantees that the flag field sequence does not inadvertently appear in the frame.

Figure 11.4 *Bit stuffing and unstuffing*

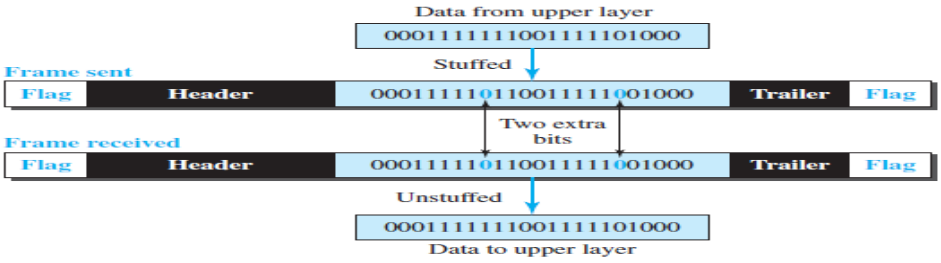


Figure 11.4 shows bit stuffing at the sender and bit removal at the receiver. Note that even if we have a 0 after five 1s, we still stuff a 0. The 0 will be removed by the receiver. This means that if the flaglike pattern 01111110 appears in the data, it will change to 011111010 (stuffed) and is not mistaken for a flag by the receiver. The real flag 01111110 is not stuffed by the sender and is recognized by the receiver.

1.2 Flow and Error Control

One of the responsibilities of the data-link control sub-layer is flow and error control at the data-link layer.

Flow Control

In communication at the data-link layer, we are dealing with four entities:

- **Network and data-link layers at the sending node and**
- **Network and data-link layers at the receiving node.**

The figure shows that the data-link layer at the sending node tries to push frames toward the data-link layer at the receiving node. If the receiving node cannot process and deliver the packet to its network at the same rate that the frames arrive, it becomes overwhelmed with frames. Flow control in this case can be feedback from the receiving node to the sending node to stop or slow down pushing frames.

Buffers

Flow control can be implemented in several ways; one of the solutions is normally to use **two *buffers***:

- One at the sending data-link layer and the other at the receiving data-link layer.
- A buffer is a set of memory locations that can hold packets at the sender and receiver.
- The flow control communication can occur by sending signals from the consumer to the producer. When the buffer of the receiving data-link layer is full, it informs the sending data-link layer to stop pushing frames.

Error Control

Error control at the data-link layer is normally very simple and implemented using one of the following two methods. In both methods, a CRC is added to the frame header by the sender and checked by the receiver.

In the first method, if the frame is corrupted, it is silently discarded; if it is not corrupted, the packet is delivered to the network layer. This method is used mostly in wired LANs such as Ethernet.

In the second method, if the frame is corrupted, it is silently discarded; if it is not corrupted, an acknowledgment is sent (for the purpose of both flow and error control) to the sender.

Combination of Flow and Error Control

Flow and error control can be combined. In a simple situation, the acknowledgment that is sent for flow control can also be used for error control to tell the sender the packet has arrived uncorrupted. The lack of acknowledgment means that there is a problem in the sent frame.

A frame that carries an acknowledgment is normally called an *ACK* to distinguish it from the data frame.

1.3 Connectionless and Connection-Oriented

A DLC protocol can be either connectionless or connection-oriented.

Connectionless Protocol

In a connectionless protocol, frames are sent from one node to the next without any relationship between the frames; each frame is independent. Note that the term *connectionless* here does not mean that there is no physical connection (transmission medium) between the nodes; it means that there is no ***connection*** between frames. The frames are not numbered and there is no sense of ordering. Most of the data-link protocols for **LANs** are connectionless protocols.

Connection-Oriented Protocol

In a connection-oriented protocol, a logical connection should first be established between the two nodes (setup phase). After all frames that are somehow related to each other are transmitted (transfer phase), the logical connection is terminated (teardown phase).

In this type of communication, the frames are numbered and sent in order. If they are not received in order, the receiver needs to wait until all frames belonging to the same set are received and then deliver them in order to the network layer.

Connection-oriented protocols are rare in wired LANs, but we can see them in some point-to-point protocols, some wireless LANs, and some WANs.

2 DATA-LINK LAYER PROTOCOLS

Four protocols have been defined for the data-link layer to deal with flow and error control:

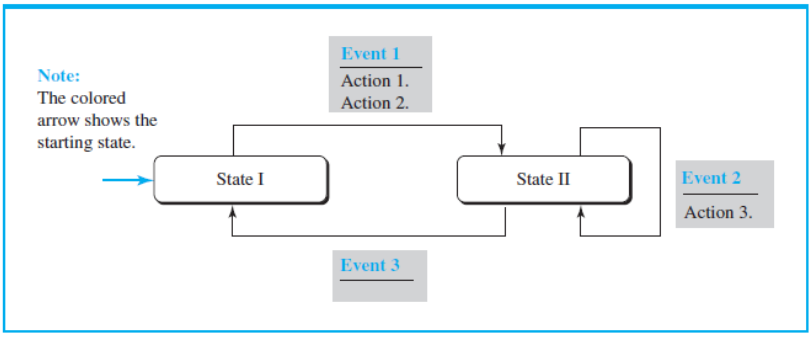
- ***Simple,***
- ***Stop-and-Wait,***
- ***Go-Back-N, and (not using in DLC)***
- ***Selective-Repeat. (not using in DLC)***

The behavior of a data-link-layer protocol can be better shown as a **finite state machine (FSM).** An FSM is thought of as a machine with a finite number of states. The machine is always in one of the states until an *event* occurs. Each event is associated with two reactions:

- ***Defining the list (possibly empty) of actions to be performed*** *and*
- ***Determining the next state (which can be the same as the current state).***

One of the states must be defined as the initial state, the state in which the machine starts when it turns on.

Figure 11.6 *Connectionless and connection-oriented service represented as FSMs*



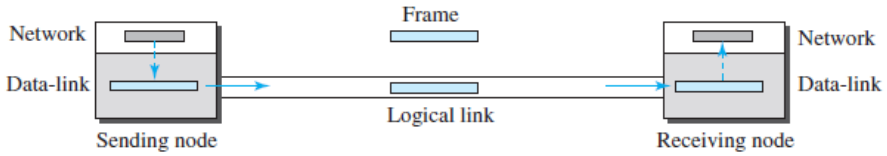
In Figure 11.6, we show an example of a machine using FSM. We have used rounded-corner rectangles to show states, colored text to show events, and regular black text to show actions. A horizontal line is used to separate the event from the actions, although later we replace the horizontal line with a slash. The arrow shows the movement to the next state.

The figure shows a machine with three states. There are only three possible events and three possible actions. The machine starts in state I. If event 1 occurs, the machine performs actions 1 and 2 and moves to state II. When the machine is in state II, two events may occur. If event 1 occurs, the machine performs action 3 and remains in the same state, state II. If event 3 occurs, the machine performs no action, but move to state I.

2.1 Simple Protocol

Our first protocol is a **simple protocol** with neither flow nor error control. We assume that the receiver can immediately handle any frame it receives. The receiver can never be overwhelmed with incoming frames. Figure 11.7 shows the layout for this protocol.

Figure 11.7 *Simple protocol*



The data-link layer at the sender gets a packet from its network layer, makes a frame out of it, and sends the frame. The data-link layer at the receiver receives a frame from the link, extracts the packet from the frame, and delivers the packet to its network layer. The data-link layers of the sender and receiver provide transmission services for their network layers.

FSMs

The sender site should not send a frame until its network layer has a message to send. The receiver site cannot deliver a message to its network layer until a frame arrives. We can show these requirements using two FSMs.

Each FSM has only one state, the *ready state*.

- The sending machine remains in the ready state until a request comes from the process in the network layer. When this event occurs, the sending machine encapsulates the message in a frame and sends it to the receiving machine.
- The receiving machine remains in the ready state until a frame arrives from the sending machine. When this event occurs, the receiving machine decapsulates the message out of the frame and delivers it to the process at the network layer. Figure 11.8 shows the FSMs for the simple protocol.

Figure 11.8 *FSMs for the simple protocol*

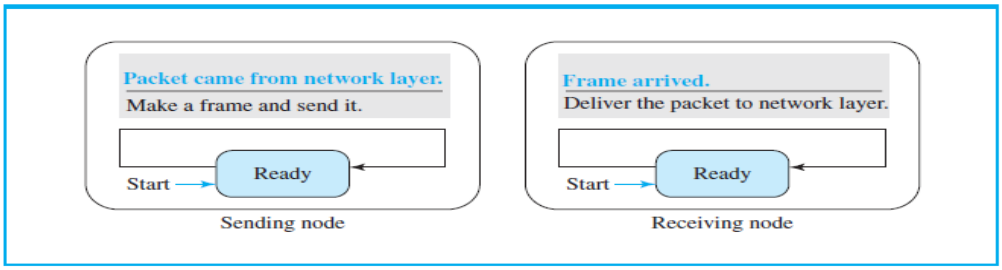
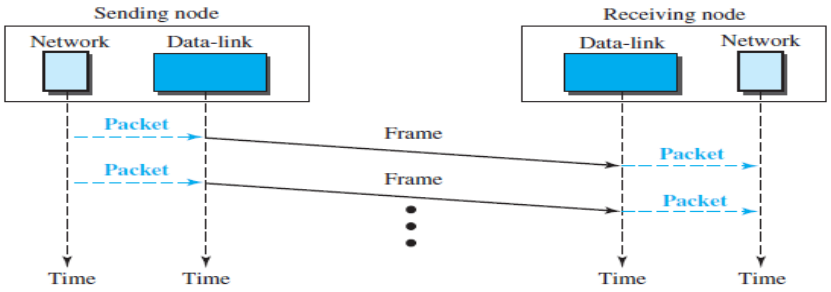


Figure 11.9 shows an example of communication using this protocol. It is very simple. The sender sends frames one after another without even thinking about the receiver.

Figure 11.9 *Flow diagram for Example 11.2*



2.2 Stop-and-Wait Protocol

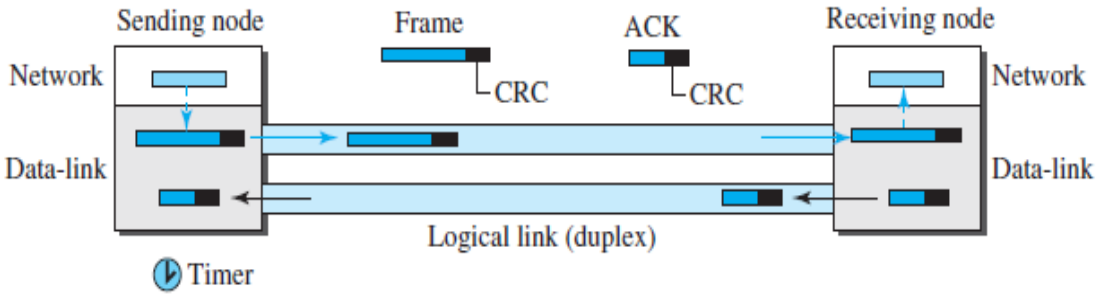
Our second protocol is called the **Stop-and-Wait protocol,** which uses both flow and error control. In this protocol, the sender sends one frame at a time and waits for an acknowledgment before sending the next one. To detect corrupted frames, we need to add a CRC to each data frame.

- When a frame arrives at the receiver site, it is checked. If its CRC is incorrect, the frame is corrupted and silently discarded. The silence of the receiver is a signal for the sender that a frame was either corrupted or lost.
- Every time the sender sends a frame, it starts a timer. If an acknowledgment arrives before the timer expires, the timer is stopped and the sender sends the next frame (if it has one to send).
- If the timer expires, the sender resends the previous frame, assuming that the frame was either lost or corrupted. This means that the sender needs to keep a copy of the frame until its

acknowledgment arrives. When the corresponding acknowledgment arrives, the sender discards the copy and sends the next frame if it is ready.

Figure 11.10 shows the outline for the Stop-and-Wait protocol. Note that only one frame and one acknowledgment can be in the channels at any time.

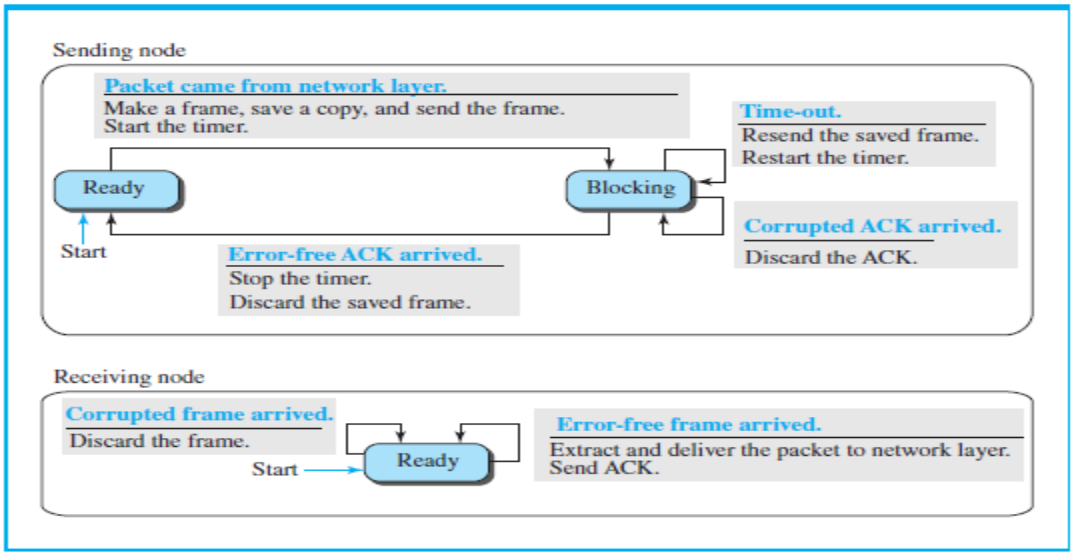
Figure 11.10 *Stop-and-Wait protocol*



FSMs

Figure 11.11 shows the FSMs for our primitive Stop-and-Wait protocol.

Figure 11.11 *FSM for the Stop-and-Wait protocol*



We describe the sender and receiver states below.

Sender States

The sender is initially in the ready state, but it can move between the ready and blocking state.

Ready State.:

When the sender is in this state, it is only waiting for a packet from the network layer. If a packet comes from the network layer, the sender creates a frame, saves a copy of the frame, starts the only timer and sends the frame. The sender then moves to the blocking state.

Blocking State:

When the sender is in this state, three events can occur:

a. If a time-out occurs, the sender resends the saved copy of the frame and restarts the timer.

b. If a corrupted ACK arrives, it is discarded.

c. If an error-free ACK arrives, the sender stops the timer and discards the saved copy of the frame. It then moves to the ready state.

Receiver

The receiver is always in the *ready* state. Two events may occur:

a. If an error-free frame arrives, the message in the frame is delivered to the network layer and an ACK is sent.

b. If a corrupted frame arrives, the frame is discarded.

2.3 Piggybacking

The two protocols we discussed in this section are designed for unidirectional communication, in which data is flowing only in one direction although the acknowledgment may travel in the other direction. Protocols have been designed in the past to allow data to flow in both directions. However, to make the communication more efficient, the data in one direction is piggybacked with the acknowledgment in the other direction.

Example: when node A is sending data to node B, Node A also acknowledges the data received from node B. Because piggybacking makes communication at the data-link layer more complicated, it is not a common practice.

3.HDLC

High-level Data Link Control (HDLC) is a bit-oriented protocol for communication over point-to-point and multipoint links.

3.1 Configurations and Transfer Modes

HDLC provides two common transfer modes that can be used in different configurations:

- ***Normal response mode (NRM) and***
- ***Asynchronous balanced mode (ABM).***

In ***normal response mode (NRM),*** the station configuration is unbalanced. We have one primary station and multiple secondary stations. A *primary station* can send commands; a *secondary station* can only respond. The NRM is used for both point-to-point and multipoint links, as shown in Figure 11.14.

In ***Asynchronous balanced Mode*** **(ABM),** the configuration is balanced. The link is point-to-point, and each station can function as a primary and a secondary (acting as peers), as shown in Figure 11.15. This is the common mode today.

Figure 11.14 *Normal response mode*

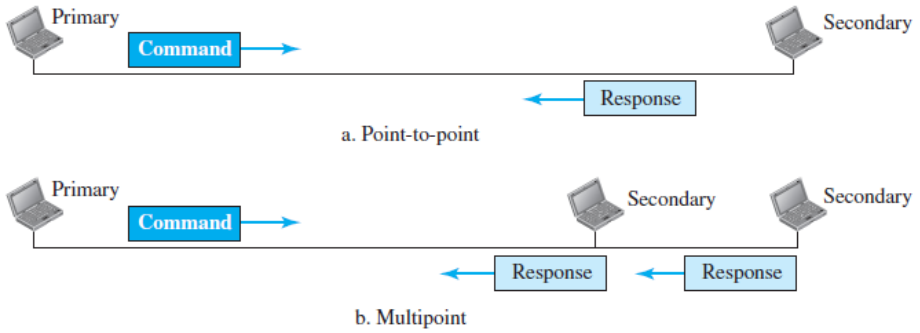


Figure 11.15 *Asynchronous balanced mode*

3.2 Framing

HDLC defines three types of frames:

Each type of frame serves as an envelope for the transmission of a different type of message.

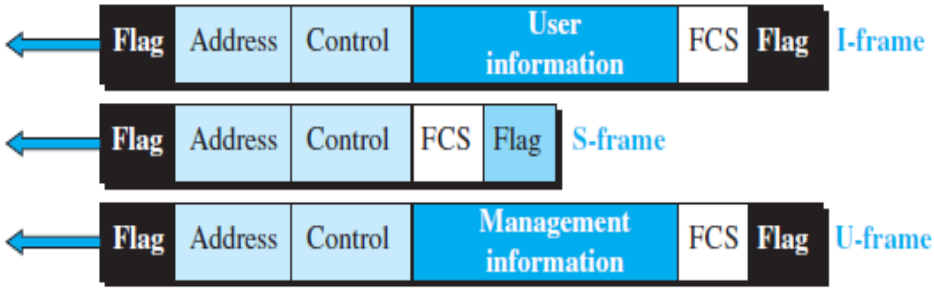
***Information frames (I-frames)*:** Iframes are used to data-link user data and control information relating to user data (piggybacking).

***Supervisory frames (S-frames)*:** S-frames are used only to transport control information.

***Unnumbered frames (U-frames)*:** U-frames are reserved for system management. Information carried by U-frames is intended for managing the link itself.

Each frame in HDLC may contain up to six fields: A beginning flag field, an address field, a control field, an information field, a frame check sequence (FCS) field, and an ending flag field. In multiple-frame transmissions, the ending flag of one frame can serve as the beginning flag of the next frame.

Figure 11.16 *HDLC frames*



Let us now discuss the fields and their use in different frame types.

Flag field: This field contains synchronization pattern 01111110, which identifies both the beginning and the end of a frame.

Address field: This field contains the address of the secondary station. If a primary station created the frame, it contains a *to* address. If a secondary station creates the frame, it contains a *from* address. The address field can be one byte or several bytes long, depending on the needs of the network.

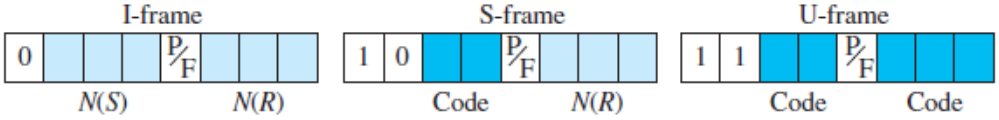
Control field: The control field is one or two bytes used for flow and error control. The interpretation of bits are discussed later.

Information field: The information field contains the user's data from the network layer or management information. Its length can vary from one network to another.

FCS field. The frame check sequence (FCS) is the HDLC error detection field. It can contain either a 2- or 4-byte CRC.

The control field determines the type of frame and defines its functionality. So let us discuss the format of this field in detail. The format is specific for the type of frame, as shown in Figure 11.17.

Figure 11.17 *Control field format for the different frame types*



Control Field for I-Frames

I-frames are designed to carry user data from the network layer. In addition, they can include flow- and error-control information (piggybacking). The subfields in the control field are used to define these functions. The first bit defines the type. If the first bit of the control field is 0, this means the frame is an I-frame. The next 3 bits, called $N(S)$, define the sequence number of the frame. Note that with 3 bits, we can define a sequence number between 0 and 7. The last 3 bits, called $N(R)$, correspond to the acknowledgment number when piggybacking is used. The single bit between $N(S)$ and $N(R)$ is called the P/F bit. The P/F field is a single bit with a dual purpose. It has meaning only when it is set (bit=1) and can mean poll or final. It means *poll* when the frame is sent by a primary station to a secondary (when the address field contains the address of the receiver). It means *final* when the frame is sent by a secondary to a primary (when the address field contains the address of the sender).

Control Field for S-Frames

Supervisory frames are used for flow and error control whenever piggybacking is either impossible or inappropriate. S-frames do not have information fields. If the first 2 bits of the control field are 10, this means the frame is an S-frame. The last 3 bits, called $N(R)$, correspond to the acknowledgment

number (ACK) or negative acknowledgment number (NAK), depending on the type of S-frame. The 2 bits called *code* are used to define the type of S-frame itself. With 2 bits, we can have four types of S-frames, as described below:

Receive ready (RR). If the value of the code subfield is 00, it is an RR S-frame. This kind of frame acknowledges the receipt of a safe and sound frame or group of frames. In this case, the value of the *N*(*R*) field defines the acknowledgment number.

Receive not ready (RNR). If the value of the code subfield is 10, it is an RNR Sframe. This kind of frame is an RR frame with additional functions. It acknowledges the receipt of a frame or group of frames, and it announces that the receiver is busy and cannot receive more frames. It acts as a kind of congestion-control mechanism by asking the sender to slow down. The value of *N*(*R*) is the acknowledgment number.

Reject (REJ). If the value of the code subfield is 01, it is an REJ S-frame. This is a NAK frame, but not like the one used for Selective Repeat ARQ. It is a NAK that can be used in Go-Back-*N* ARQ to improve the efficiency of the process by informing the sender, before the sender timer expires, that the last frame is lost or damaged. The value of *N*(*R*) is the negative acknowledgment number.

Selective reject (SREJ). If the value of the code subfield is 11, it is an SREJ Sframe. This is a NAK frame used in Selective Repeat ARQ. Note that the HDLC Protocol uses the term *selective reject* instead of *selective repeat*. The value of *N*(*R*) is the negative acknowledgment number.

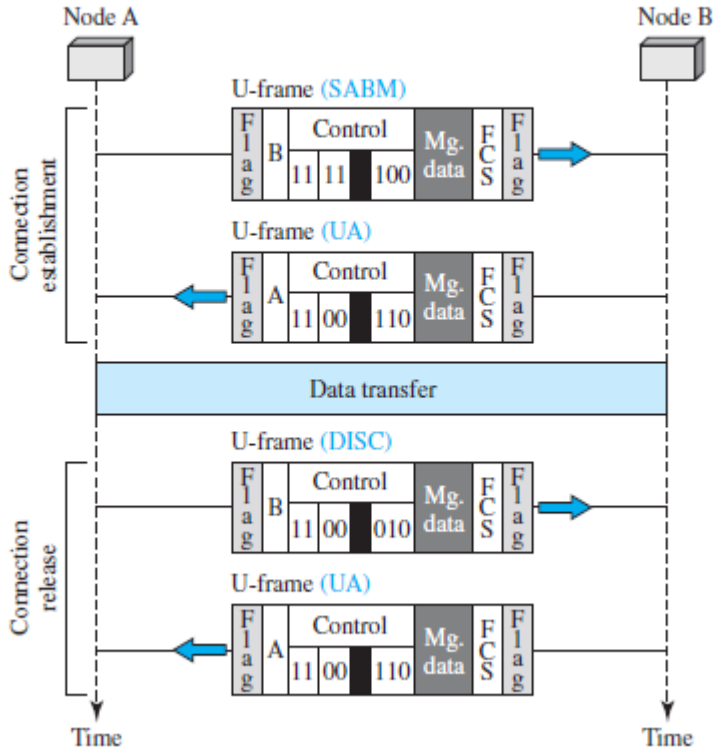
Control Field for U-Frames

Unnumbered frames are used to exchange session management and control information between connected devices. Unlike S-frames, U-frames contain an information field, but one used for system management information, not user data. As with S-frames, however, much of the information carried by U-frames is contained in codes included in the control field. U-frame codes are divided into two sections: a 2-bit prefix before the P/ F bit and a 3-bit suffix after the P/F bit. Together, these two segments (5 bits) can be used to create up to 32 different types of U-frames.

Example 11.5

Figure 11.18 shows how U-frames can be used for connection establishment and connection release. Node A asks for a connection with a set asynchronous balanced mode (SABM) frame; node B gives a positive response with an unnumbered acknowledgment (UA) frame. After these two exchanges, data can be transferred between the two nodes (not shown in the figure). After data transfer, node A sends a DISC (disconnect) frame to release the connection; it is confirmed by node B responding with a UA (unnumbered acknowledgment).

Figure 11.18 *Example of connection and disconnection*



1.4 POINT-TO-POINT PROTOCOL (PPP)

One of the most common protocols for point-to-point access is the **Point-to-Point Protocol (PPP).** Today, millions of Internet users who need to connect their home computers to the server of an Internet service provider use PPP. The majority of these users have a traditional modem; they are connected to the Internet through a telephone line, which provides the services of the physical layer. But to control and manage the transfer of data, there is a need for a point-to-point protocol at the data-link layer.

1.4.1 Services

The designers of PPP have included several services to make it suitable for a point-to-point protocol, but have ignored some traditional services to make it simple.

Services Provided by PPP

- PPP defines the format of the frame to be exchanged between devices. It also defines how two devices can negotiate the establishment of the link and the exchange of data.
- PPP is designed to accept payloads from several network layers (not only IP). Authentication is also provided in the protocol, but it is optional.
- The new version of PPP, called *Multilink PPP*, provides connections over multiple links. One interesting feature of PPP is that it provides network address configuration. This is particularly useful when a home user needs a temporary network address to connect to the Internet.

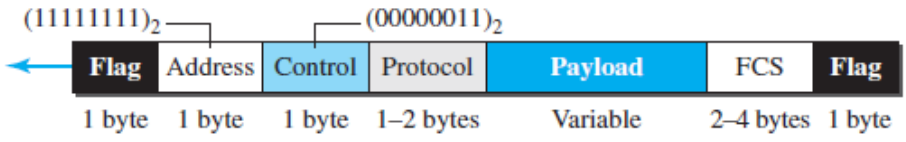
Services Not Provided by PPP

- PPP does not provide flow control. A sender can send several frames one after another with no concern about overwhelming the receiver.
- PPP has a very simple mechanism for error control. A CRC field is used to detect errors. If the frame is corrupted, it is silently discarded; the upper-layer protocol needs to take care of the problem. Lack of error control and sequence numbering may cause a packet to be received out of order.
- PPP does not provide a sophisticated addressing mechanism to handle frames in a multipoint configuration.

1.4.2 Framing

PPP uses a character-oriented (or byte-oriented) frame. Figure 11.20 shows the format of a PPP frame. The description of each field follows:

Figure 11.20 *PPP frame format*



Flag. A PPP frame starts and ends with a 1-byte flag with the bit pattern 01111110.

Address. The address field in this protocol is a constant value and set to 11111111 (broadcast address).

Control. This field is set to the constant value 00000011 (imitating unnumbered frames in HDLC). As we will discuss later, PPP does not provide any flow control. Error control is also limited to error detection.

Protocol. The protocol field defines what is being carried in the data field: either user data or other information. This field is by default 2 bytes long, but the two parties can agree to use only 1 byte.

Payload field. This field carries either the user data or other information that we will discuss shortly. The data field is a sequence of bytes with the default of a maximum of 1500 bytes; but this can be changed during negotiation. The data field is byte-stuffed if the flag byte pattern appears in this field. Because there is no field defining the size of the data field, padding is needed if the size is less than the maximum default value or the maximum negotiated value.

FCS. The frame check sequence (FCS) is simply a 2-byte or 4-byte standard CRC.

Byte Stuffing

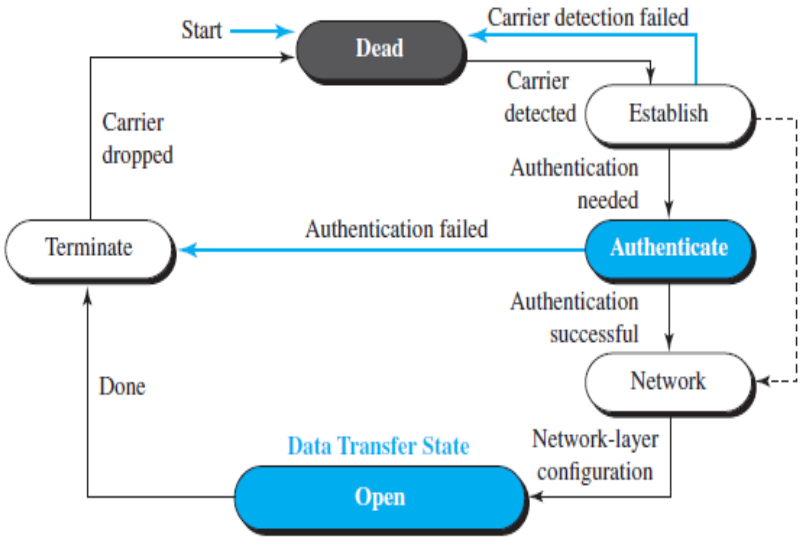
Since PPP is a byte-oriented protocol, the flag in PPP is a byte that needs to be escaped whenever it appears in the data section of the frame. The escape byte is 01111101, which means that

very time the flag like pattern appears in the data, this extra byte is stuffed to tell the receiver that the next byte is not a flag. Obviously, the escape byte itself should be stuffed with another escape byte.

1.4.3 Transition Phases

- A PPP connection goes through phases which can be shown in a ***transition phase*** diagram.
- The transition diagram, which is an FSM, starts with the ***dead*** **state**. In this state, there is no active carrier (at the physical layer) and the line is quiet. When one of the two nodes starts the communication, the connection goes into the ***establish*** state.
- In this state, options are negotiated between the two parties. If the two parties agree that they need **authentication** (for example, if they do not know each other), then the system needs to do authentication (an extra step); otherwise, the parties can simply start communication.
- The link-control protocol packets, discussed shortly, are used for this purpose. Several packets may be exchanged here. Data transfer takes place in the ***open*** state. When a connection reaches this state, the exchange of data packets can be started. The connection remains in this state until one of the endpoints wants to terminate the connection.
- In this case, the system goes to the ***terminate*** state. The system remains in this state until the carrier (physical-layer signal) is dropped, which moves the system to the ***dead*** state again.

Figure 11.21 *Transition phases*



1.4.4 Multiplexing

Although PPP is a link-layer protocol, it uses another set of protocols to establish the link, authenticate the parties involved, and carry the network-layer data. Three sets of protocols are defined to make PPP powerful:

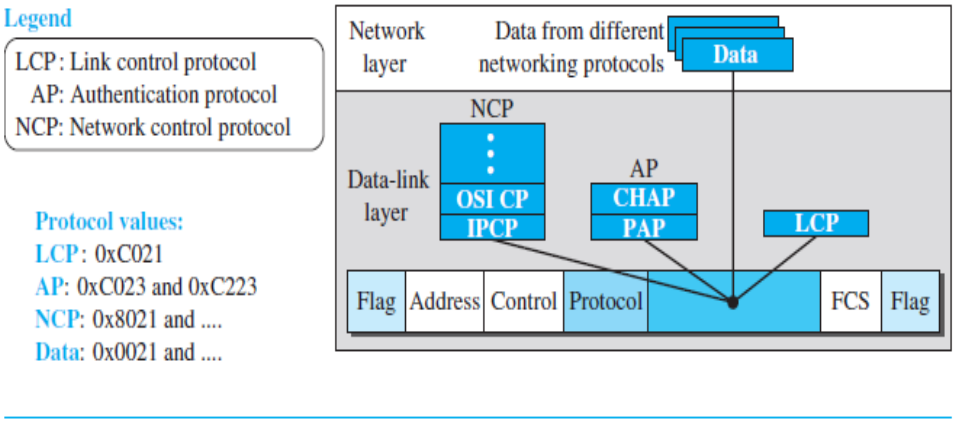
- ***The Link Control Protocol (LCP),***
- ***Two Authentication Protocols (APs), and***
- ***Several Network Control Protocols (NCPs).***

At any moment, a PPP packet can carry data from one of these protocols in its data field, as shown in Figure 11.22. Note that there are one LCP, two APs, and several NCPs. Data may also come from several different network layers.

Link Control Protocol

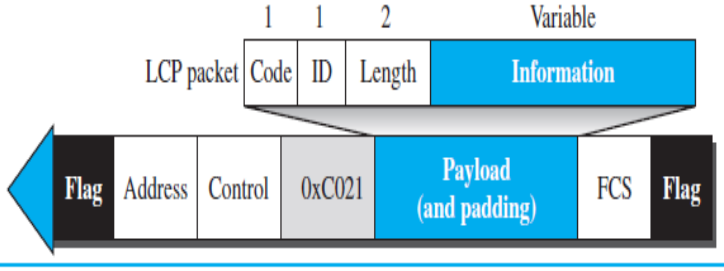
The **Link Control Protocol (LCP)** is responsible for establishing, maintaining, configuring, and terminating links. It also provides negotiation mechanisms to set options between the two endpoints. Both endpoints of the link must reach an agreement about the options before the link can be established. See Figure 11.21.

Figure 11.22 *Multiplexing in PPP*



All LCP packets are carried in the payload field of the PPP frame with the protocol field set to C021 in hexadecimal (see Figure 11.23).

Figure 11.23 *LCP packet encapsulated in a frame*



The code field defines the type of LCP packet. There are 11 types of packets, as shown in Table 11.1.

Table 11.1 *LCP packets*

Code	Packet Type	Description
0x01	Configure-request	Contains the list of proposed options and their values
0x02	Configure-ack	Accepts all options proposed
0x03	Configure-nak	Announces that some options are not acceptable
0x04	Configure-reject	Announces that some options are not recognized
0x05	Terminate-request	Request to shut down the line
0x06	Terminate-ack	Accept the shutdown request
0x07	Code-reject	Announces an unknown code
0x08	Protocol-reject	Announces an unknown protocol
0x09	Echo-request	A type of hello message to check if the other end is alive
0x0A	Echo-reply	The response to the echo-request message
0x0B	Discard-request	A request to discard the packet

There are three categories of packets.

The first category, comprising the first four packet types, is used for link configuration during the establish phase.

The second category, comprising packet types 5 and 6, is used for link termination during the termination phase.

The last five packets are used for link monitoring and debugging.

The ID field holds a value that matches a request with a reply. One endpoint inserts a value in this field, which will be copied into the reply packet. The length field defines the length of the entire LCP packet. The information field contains information, such as options, needed for some LCP packets.

There are many options that can be negotiated between the two endpoints. Options are inserted in the information field of the configuration packets. In this case, the information field is divided into three fields: ***option type, option length, and option data***. We list some of the most common options in Table 11.2.

Table 11.2 *Common options*

Option	*Default*
Maximum receive unit (payload field size)	1500
Authentication protocol	None
Protocol field compression	Off
Address and control field compression	Off

Authentication Protocols

Authentication plays a very important role in PPP because PPP is designed for use over dial-up links where verification of user identity is necessary. *Authentication* means validating the identity of a user who needs to access a set of resources. PPP has created two protocols for authentication: Password Authentication Protocol and Challenge Handshake Authentication Protocol. Note that these protocols are used during the authentication phase.

PAP

The **Password Authentication Protocol (PAP)** is a simple authentication procedure with a two-step process:

a. The user who wants to access a system sends an authentication identification (usually the user name) and a password.

b. The system checks the validity of the identification and password and either accepts or denies connection.

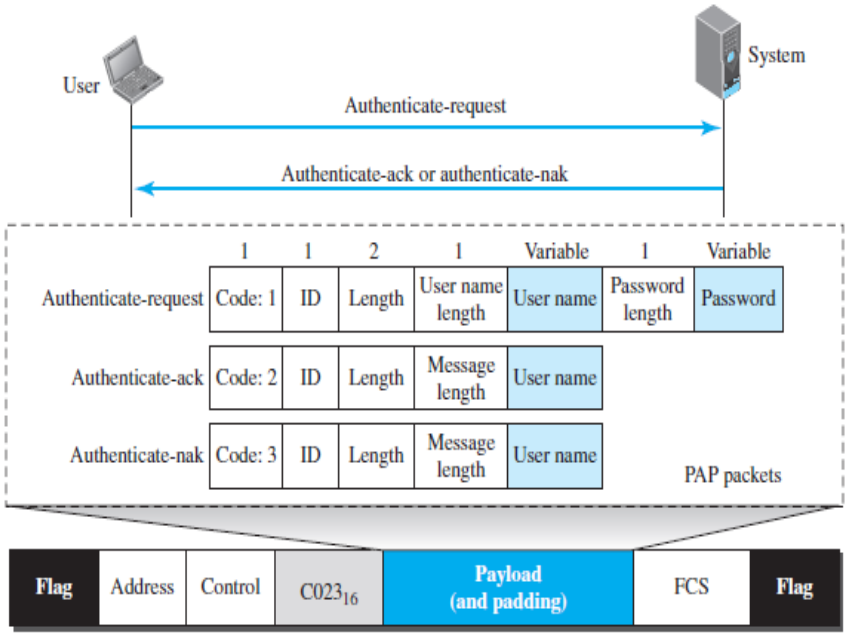
Figure 11.24 shows the three types of packets used by PAP and how they are actually exchanged. When a PPP frame is carrying any PAP packets, the value of the protocol field is 0xC023. The three PAP packets are

Authenticate-request: The first packet is used by the user to send the user name and password.

Authenticate-ack: The second is used by the system to allow access.

Authenticate-nak. The third is used by the system to deny access.

Figure 11.24 *PAP packets encapsulated in a PPP frame*



CHAP

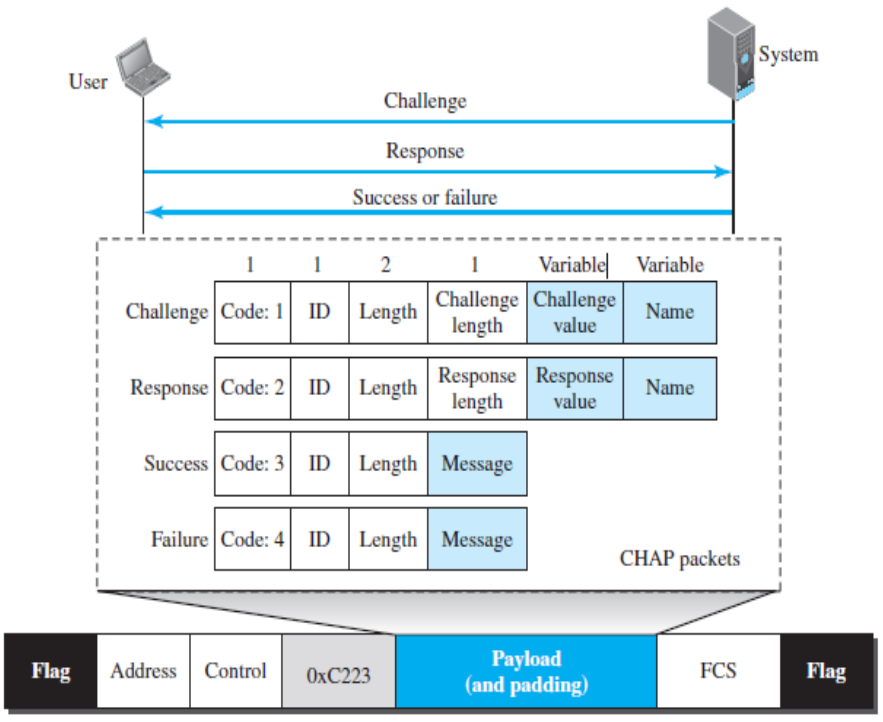
The **Challenge Handshake Authentication Protocol (CHAP)** is a three-way handshaking authentication protocol that provides greater security than PAP. In this method, the password is kept secret; it is never sent online.

a. The system sends the user a challenge packet containing a challenge value, usually a few bytes.

b. The user applies a predefined function that takes the challenge value and the user's own password and creates a result. The user sends the result in the response packet to the system.

c. The system does the same. It applies the same function to the password of the user (known to the system) and the challenge value to create a result. If the result created is the same as the result sent in the response packet, access is granted; otherwise, it is denied. CHAP is more secure than PAP, especially if the system continuously changes the challenge value. Even if the intruder learns the challenge value and the result, the password is still secret. Figure 11.25 shows the packets and how they are used.

Figure 11.25 *CHAP packets encapsulated in a PPP frame*



CHAP packets are encapsulated in the PPP frame with the protocol value C223 in hexadecimal. There are four CHAP packets:

Challenge: The first packet is used by the system to send the challenge value.

Response: The second is used by the user to return the result of the calculation.

Success: The third is used by the system to allow access to the system.

Failure: The fourth is used by the system to deny access to the system.

Network Control Protocols

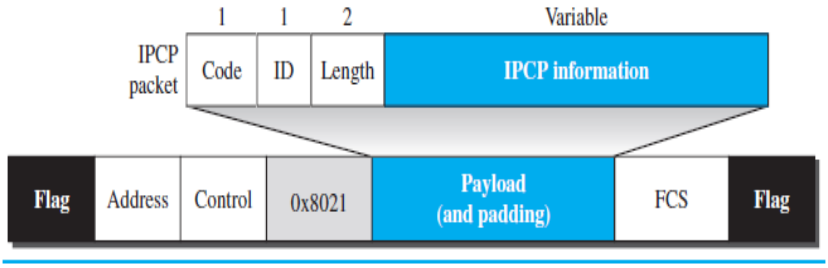
PPP is a multiple-network-layer protocol. It can carry a network-layer data packet from protocols defined by the Internet, OSI, Xerox, DECnet, AppleTalk, Novel, and so on. To do this, PPP has defined a specific Network Control Protocol for each network protocol.

For example, IPCP (Internet Protocol Control Protocol) configures the link for carrying IP data packets. Xerox CP does the same for the Xerox protocol data packets, and so on. Note that none of the NCP packets carry network-layer data; they just configure the link at the network layer for the incoming data.

IPCP

One NCP protocol is the **Internet Protocol Control Protocol (IPCP).** This protocol configures the link used to carry IP packets in the Internet. IPCP is especially of interest to us. The format of an IPCP packet is shown in Figure 11.26. Note that the value of the protocol field in hexadecimal is 8021.

Figure 11.26 *IPCP packet encapsulated in PPP frame*



IPCP defines seven packets, distinguished by their code values, as shown in Table 11.3.

Table 11.3 *Code value for IPCP packets*

Code	*IPCP Packet*
0x01	Configure-request
0x02	Configure-ack
0x03	Configure-nak
0x04	Configure-reject
0x05	Terminate-request
0x06	Terminate-ack
0x07	Code-reject

Other Protocols

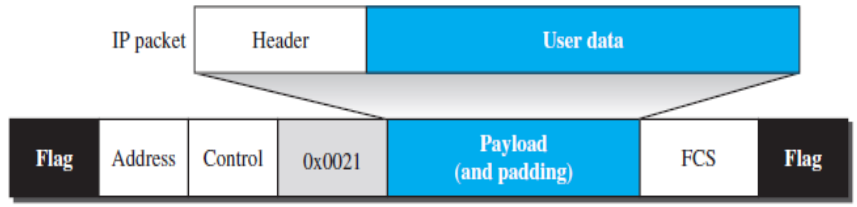
There are other NCP protocols for other network-layer protocols. The OSI Network Layer Control Protocol has a protocol field value of 8023; the Xerox NS IDP Control Protocol has a protocol field value of 8025; and so on.

Data from the Network Layer

After the network-layer configuration is completed by one of the NCP protocols, the users can exchange data packets from the network layer. Here again, there are different protocol fields for different network layers.

For example, if PPP is carrying data from the IP network layer, the field value is 0021 (note that the three rightmost digits are the same as for IPCP). If PPP is carrying data from the OSI network layer, the value of the protocol field is 0023, and so on. Figure 11.27 shows the frame for IP.

Figure 11.27 *IP datagram encapsulated in a PPP frame*

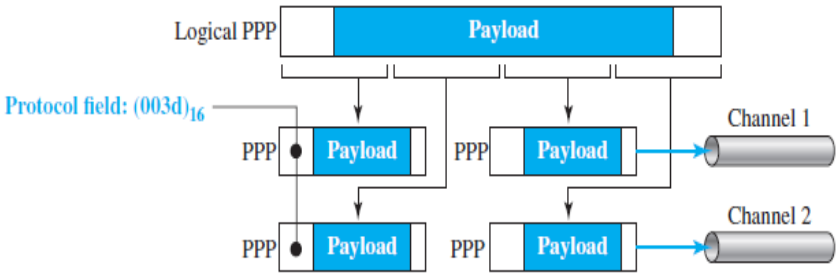


Multilink PPP

PPP was originally designed for a single-channel point-to-point physical link. The availability of multiple channels in a single point-to-point link motivated the development of Multilink PPP. In this case, a logical PPP frame is divided into several actual PPP frames. A segment of the logical frame is carried in the payload of an actual PPP frame, as shown in Figure 11.28. To show that the actual PPP frame is carrying a fragment of a logical PPP frame, the protocol field is set to (003d)16. This new development adds complexity.

For example, a sequence number needs to be added to the actual PPP frame to show a fragment's position in the logical frame.

Figure 11.28 *Multilink PPP*



Media Access Control: *Random Access, Controlled Access, Channelization.*

2.4 Random Access:

Networks can be divided into two categories:

1) point-to-point connections (WAN)
2) Broadcast channels. OR random access channels (LAN)

Any broadcast network, the key issue is how to determine who gets to use the channel. Consider a conference call in which six people, on six different telephones, are all connected so that each one can hear and talk to all the others. It is very likely that when one of them stops speaking, two or more will start talking at once. When only a single channel is available, determining who should go next is much harder. Many protocols for solving the problem are known and form the contents of this chapter. In the literature, broadcast channels are sometimes referred to as **multi-access channels** or **random access channels**.

The protocols used to determine who goes next on a multi-access channel belong to a sub layer of the data link layer called the **MAC** (**Medium Access Control**) sub layer. The MAC sub layer is especially important in LANs, many of which use a multi-access channel as the basis for communication.

In random-access or contention methods, no station is superior to another station and none is assigned control over another. At each instance, a station that has data to send uses a procedure defined by the protocol to make a decision on whether or not to send. This decision depends on the state of the medium (idle or busy).

Two important features give this method its name.

First, there is no scheduled time for a station to transmit. Transmission is random among the stations. That is why these methods are called *random access*.

Second, no rules specify which station should send next. Stations compete with one another to access the medium. These methods are also called ***contention*** methods.

In a random-access method, each station has the right to the medium without being controlled by any other station. If more than one station tries to send, there is an access conflict—***collision***—and the frames will be either destroyed or modified. To avoid access conflict or to resolve it when it happens, each station follows a procedure that answers the following questions:

- When can the station access the medium?
- What can the station do if the medium is busy?
- How can the station determine the success or failure of the transmission?
- What can the station do if there is an access conflict?

The **FOUR** important random access methods are

Aloha

In a random access methods a important protocol known as ALOHA, which used a very simple procedure called **multiple access (MA).** Many algorithms for allocating a multiple access channel. In 1970 Norman Abramson devised a new and elegant method to solve the channel allocation problem. ALOHA system, used ground-based radio broadcasting, the basic idea is applicable to any system in which uncoordinated users are competing for the use of a single shared channel.

Carrier sense multiple access (CSMA)

The method was improved with the addition of a procedure that forces the station to sense the medium before transmitting. This was called *carrier sense multiple access* (*CSMA*).

carrier sense multiple access with collision detection (CSMA/CD)

This method which tells the station what to do when a collision is detected

carrier sense multiple access with collision avoidance (CSMA/CA)

This method which tries to avoid the collision.

ALOHA

The earliest random access method, was developed at the University of Hawaii in early 1970. It was designed for a radio (wireless) LAN, but it can be used on any shared medium.

It is obvious that there are potential collisions in this arrangement. The medium is shared between the stations. When a station sends data, another station may attempt to do so at the same time. The data from the two stations collide and become garbled.

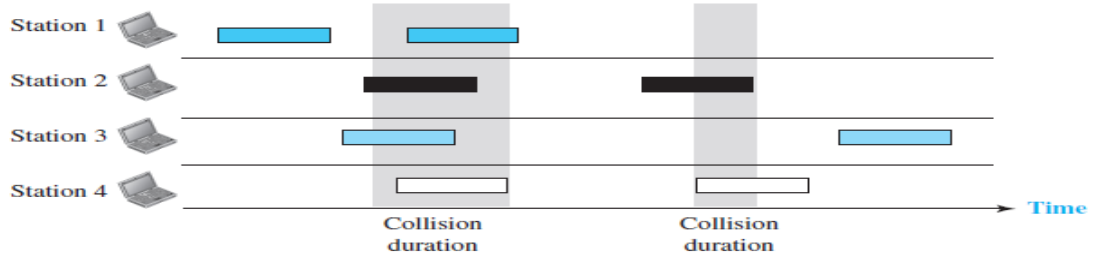
There are two versions of ALOHA:

- **Pure ALOHA** ,it does not require global time synchronization
- **Slotted ALOHA**, it require global time synchronization.

Pure ALOHA

The original ALOHA protocol is called ***pure ALOHA.*** This is a simple but elegant protocol. The idea is that each station sends a frame whenever it has a frame to send (multiple access). Since there is only one channel to share, there is the possibility of collision between frames from different stations.

Figure 12.2 *Frames in a pure ALOHA network*



There are four stations (unrealistic assumption) that contend with one another for access to the shared channel. The figure shows that each station sends two frames; there are a total of eight frames on the shared medium. Some of these frames collide because multiple frames are in contention for the shared channel.

Figure 12.2 shows that only two frames survive: one frame from station 1 and one frame from station 3. We need to mention that even if one bit of a frame coexists on the channel with one bit from another frame, there is a collision and both will be destroyed. It is obvious that we need to resend the frames that have been destroyed during transmission.

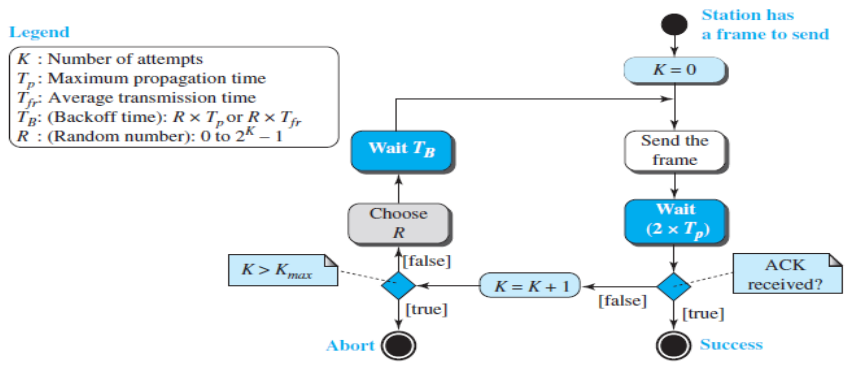
The pure ALOHA protocol relies on acknowledgments from the receiver. When a station sends a frame, it expects the receiver to send an acknowledgment. If the acknowledgment does not arrive after a **time-out period**, the station assumes that the frame (or the acknowledgment) has been destroyed and resends the frame.

A collision involves two or more stations. If all these stations try to resend their frames after the time-out, the frames will collide again. Pure ALOHA dictates that when the time-out period passes, each station waits a random amount of time before resending its frame. The randomness will help avoid more collisions. We call this time the ***backoff time*** T_B.

Pure ALOHA has a second method to prevent congesting the channel with retransmitted frames. After a maximum number of retransmission attempts K_{max}, a station must give up and try later. Figure 12.3 shows the procedure for pure ALOHA based on the above strategy.

The time-out period is equal to the maximum possible round-trip propagation delay, which is twice the amount of time required to send a frame between the two most widely separated stations (2 × Tp). The backoff time T_B is a random value that normally depends on K (the number of attempted unsuccessful transmissions). The formula for T_B depends on the implementation. One common formula is the ***binary exponential backoff***. In this method, for each retransmission, a multiplier R = 0 to $2^K - 1$ is randomly chosen and multiplied by T_p (maximum propagation time) or T_{fr} (the average time required to send out a frame) to find T_B. Note that in this procedure, the range of the random numbers increases after each collision. The value of K_{max} is usually chosen as 15.

Figure 12.3 *Procedure for pure ALOHA protocol*



Vulnerable time

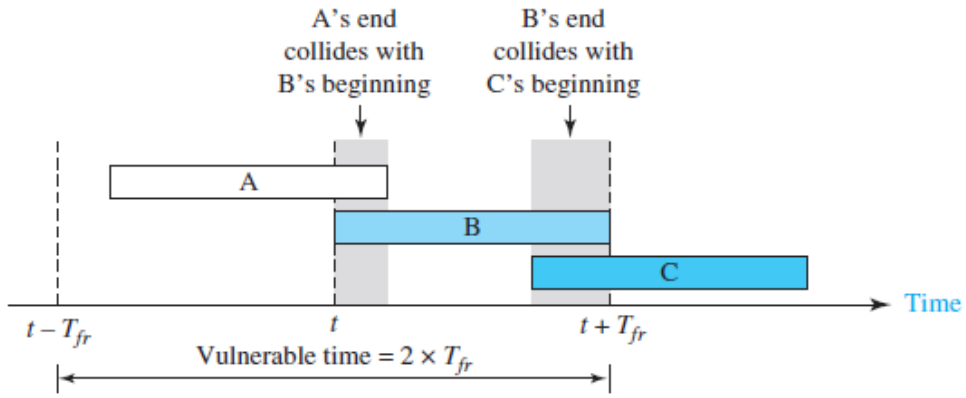
Let us find the **vulnerable time**, the length of time in which there is a possibility of collision. We assume that the stations send fixed-length frames with each frame taking T_{fr} seconds to send. Figure 12.4 shows the vulnerable time for station B.

Station B starts to send a frame at time *t*. Now imagine station A has started to send its frame after $t - T_{fr}$. This leads to a collision between the frames from station B and station A. On the other hand, suppose that station C starts to send a frame before time t + Tfr. Here, there is also a collision between frames from station B and station C.

Looking at Figure 12.4, we see that the vulnerable time during which a collision may occur in pure ALOHA is 2 times the frame transmission time.

$$\textit{Pure ALOHA vulnerable time} = 2 * T_{fr}$$

Figure 12.4 *Vulnerable time for pure ALOHA protocol*



Throughput

Let us call **G** the average number of frames generated by the system during one frame transmission time. Then it can be proven that the average number of successfully transmitted frames for pure ALOHA is $S = G \times e^{-2G}$. The maximum throughput $\mathbf{S_{max}}$ is 0.184, for G = 1/2. (We can find it by setting the derivative of S with respect to G to 0; see Exercises.)

If one-half a frame is generated during one frame transmission time (one frame during two frame transmission times), then 18.4 percent of these frames reach their destination successfully.

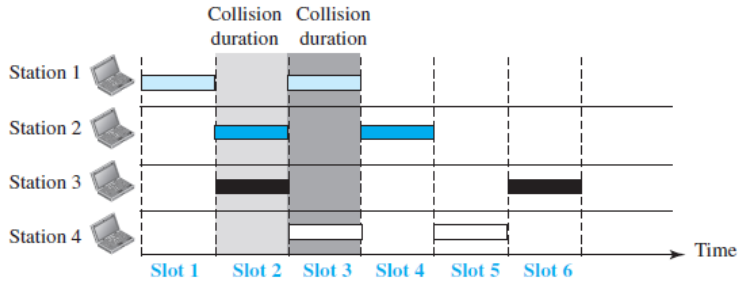
We expect G = 1/2 to produce the maximum throughput because the vulnerable time is 2 times the frame transmission time. Therefore, if a station generates only one frame in this vulnerable time (and no other stations generate a frame during this time), the frame will reach its destination successfully.

Slotted ALOHA:

Pure ALOHA has a vulnerable time of $2 \times T_{fr}$. This is so because there is no rule that defines when the station can send. A station may send soon after another station has started or just before another station has finished. **Slotted ALOHA** was invented to improve the efficiency of **pure ALOHA.**

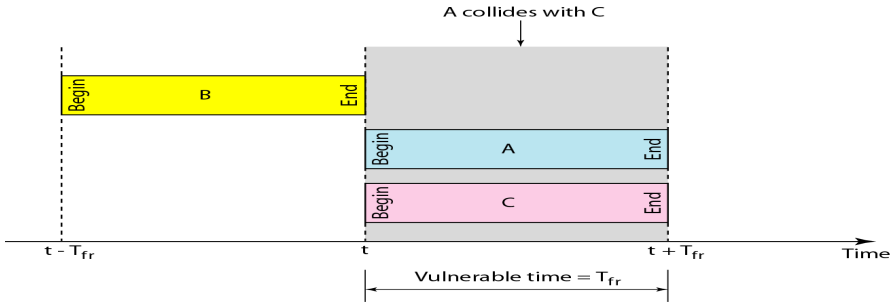
In slotted ALOHA we divide the time into slots of T_{fr} seconds and force the station to send only at the beginning of the time slot. Figure 12.5 shows an example of frame collisions in slotted ALOHA.

Figure 12.5 *Frames in a slotted ALOHA network*



Because a station is allowed to send only at the beginning of the synchronized time slot, if a station misses this moment, it must wait until the beginning of the next time slot. This means that the station which started at the beginning of this slot has already finished sending its frame. Of course, there is still the possibility of collision if two stations try to send at the beginning of the same time slot. However, the vulnerable time is now reduced to one-half, equal to *Tfr*.

Slotted ALOHA vulnerable time = T_{fr}



Throughput

It can be proven that the average number of successful transmissions for slotted ALOHA is $\mathbf{S = G \times e^{-G}}$. The maximum throughput Smax is 0.368, when G = 1. In other words, if one frame is generated during one frame transmission time, then 36.8 percent of these frames reach their destination successfully. We expect G = 1 to produce maximum throughput because the vulnerable time is equal to the frame transmission time. Therefore, if a station generates only one frame in this vulnerable time (and no other station generates a frame during this time), the frame will reach its destination successfully.

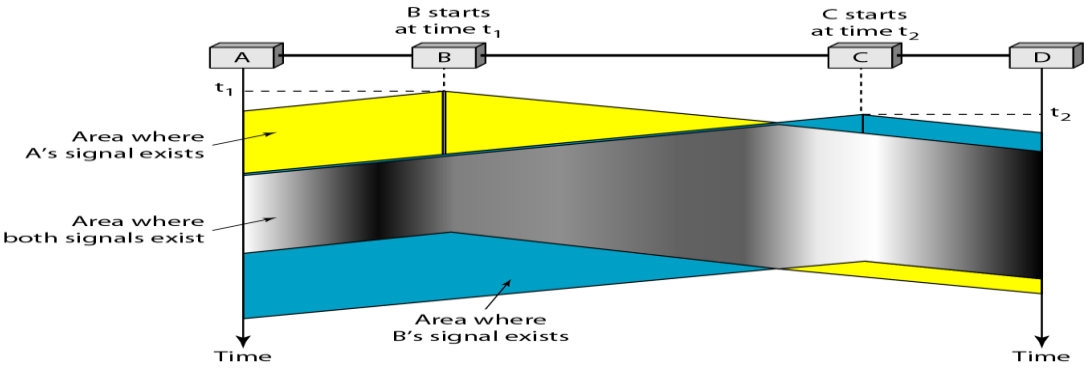
Carrier sense multiple access (CSMA)

The maximum channel utilization that can be achieved is 1/e. This is due to wastage of bandwidth because of collisions. This wastage can be reduces by avoiding transmissions that are certain to cause collisions. If any station is transmitting data that is known by the remaining all other stations, no other station will transmit. It is necessary to detect by a station what other stations are doing, that is by sensing medium for the presence of carrier signals, a station can determine whether there is an ongoing transmission. These protocols are called **CSMA**.

In local area networks, however, it is possible for stations to detect what other stations are doing, and adapt their behavior accordingly. These networks can achieve a much better utilization than 1/e. In this section we will discuss some protocols for improving performance.

CSMA can reduce the possibility of collision, but it cannot eliminate it. The reason for this is shown in Figure 12.7, a space and time model of a CSMA network. Stations are connected to a shared channel (usually a dedicated medium). The possibility of collision still exists because of propagation delay; when a station sends a frame, it still takes time (although very short) for the first bit to reach every station and for every station to sense it. In other words, a station may sense the medium and find it idle, only because the first bit sent by another station has not yet been received.

Protocols in which stations listen for a carrier (i.e., a transmission) and act accordingly are called carrier sense protocols.

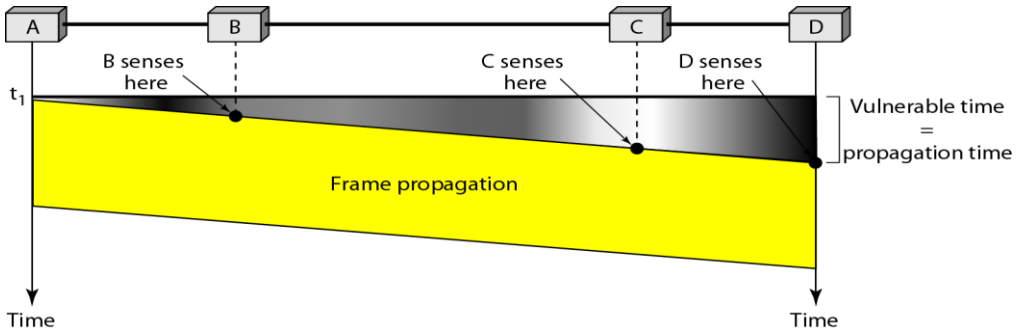


At time t1, station B senses the medium and finds it idle, so it sends a frame. At time t2 (t2 > t1), station C senses the medium and finds it idle because, at this time, the first bits from station B have not reached station C. Station C also sends a frame. The two signals collide and both frames are destroyed.

Vulnerable Time

The vulnerable time for CSMA is the propagation time Tp. This is the time needed for a signal to propagate from one end of the medium to the other. When a station sends a frame and any other station tries to send a frame during this time, a collision will result. But if the first bit of the frame reaches the end of the medium, every station will already have heard the bit and will refrain from sending. Figure 12.8 shows the worst case. The leftmost station, A, sends a frame at time t1,

which reaches the rightmost station, D, at time t1 + Tp. The gray area shows the vulnerable area in time and space.

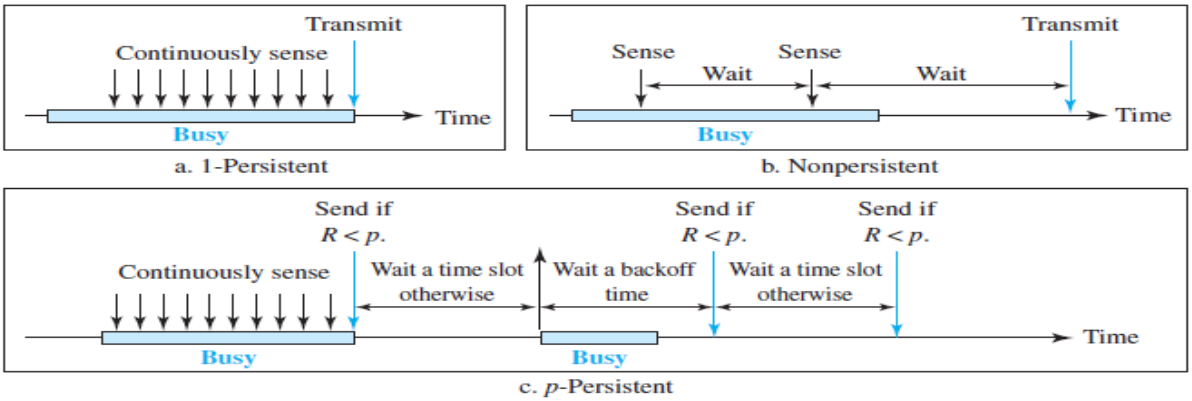


Persistence methods:

There are three different types of CSMA protocols

- **1-persistent CSMA**
- **Non-persistent CSMA**
- **P-persistent CSMA**

Figure 12.9 *Behavior of three persistence methods*



1-persistent CSMA

The first carrier sense protocol that we will study here is called **1-persistent CSMA-** Carrier Sense Multiple Access). When a station has data to send, it first listens to the channel to see if anyone else is transmitting at that moment. There are two possibilities that may occur, channel is busy or idle

- If the channel is busy, the station waits until it becomes idle, then they transmit their frames
- If the station detects an idle channel, it transmits a frame.

If a collision occurs, the station waits a random amount of time and starts all over again. The protocol is called 1-persistent because the station transmits with a probability of 1 when it finds the channel idle.

The propagation delay has an important effect on the performance of the protocol.

1) If the propagation delay is less, there will still be collisions. If two stations become ready in the middle of a third station's transmission, both will wait politely until the transmission ends and then both will begin transmitting exactly simultaneously, resulting in a collision.
2) If the propagation is longer, suppose a station begins sending, another station will become ready to send, it senses the idle channel. If the first station's signal has not yet reached the second one, , resulting in a collision.

   The longer the propagation delay, the more important this effect becomes, and the worse the performance of the protocol.

Non-persistent CSMA:

A second carrier sense protocol is **non-persistent CSMA**. In this protocol, a conscious attempt is made to be less greedy than in the previous one. Before sending, a station senses the channel. If no one else is sending, the station begins doing so itself. However, if the channel is already in use, the station does not continually sense it for the purpose of seizing it immediately upon detecting the end of the previous transmission. Instead, it waits a random period of time and then repeats the algorithm. Consequently, this algorithm leads to better channel utilization but longer delays than 1-persistent CSMA.

P-persistent CSMA:

The last protocol is **p-persistent CSMA**. It combines the elements of the above two schemes. It applies to slotted channels and works as follows. When a station becomes ready to send, it senses the channel.

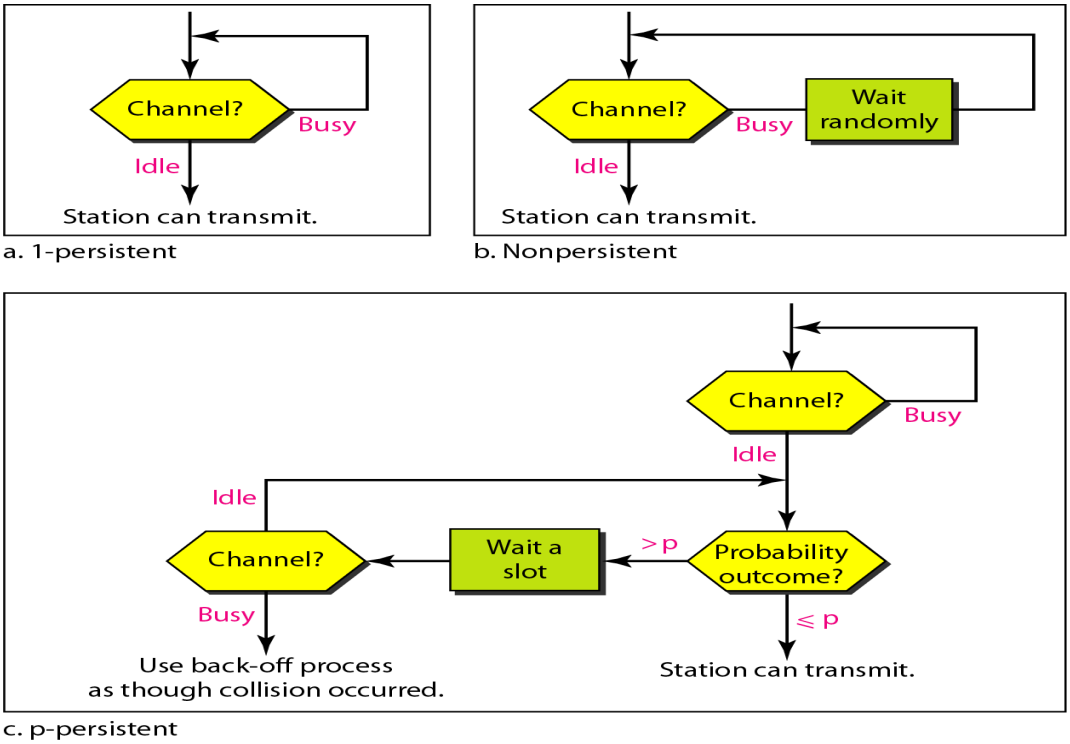
If the channel is idle, the following occurs

1) With a probability p, the station transmits its packet
2) With a probability q = 1 - p, it defers until the next slot.

a. If the line is idle, it goes to step 1.

b. If the line is busy, it acts as though a collision has occurred and uses the ***back off*** procedure.

If that slot is also idle, it either transmits or defers again, with probabilities p and q. This process is repeated until either the frame has been transmitted or another station has begun transmitting. In the latter case, the unlucky station acts as if there had been a collision (i.e., it waits a random time and starts again). If the station initially senses the channel busy, it waits until the next slot and applies the above algorithm.

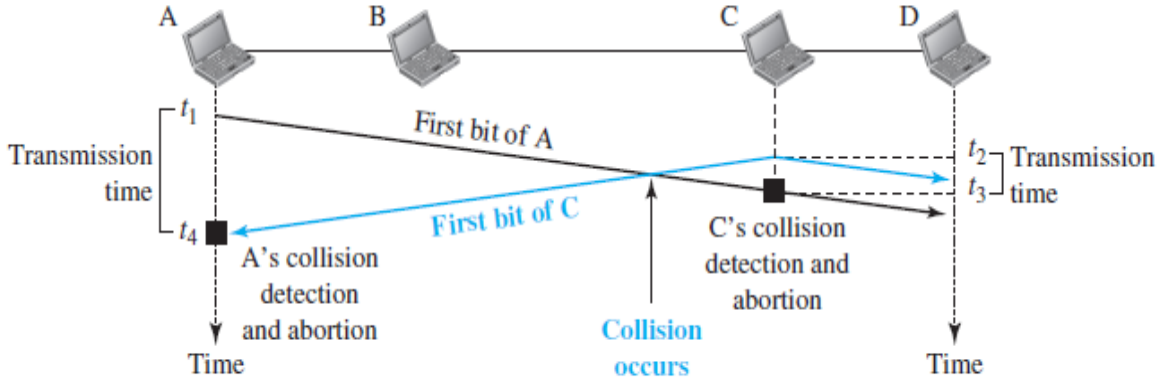


CSMA/CD

Carrier sense multiple access with collision detection (CSMA/CD) augments the algorithm to handle the collision. In this method, a station monitors the medium after it sends a frame to see if the transmission was successful. If so, the station is finished. If, however, there is a collision, the frame is sent again.

In Carrier sense multiple accesses with collision detection (CSMA/CD) the first bits transmitted by the two stations involved in the collision. Although each station continues to send bits in the frame until it detects the collision, we show what happens as the first bits collide. In Figure 12.11, stations A and C are involved in the collision.

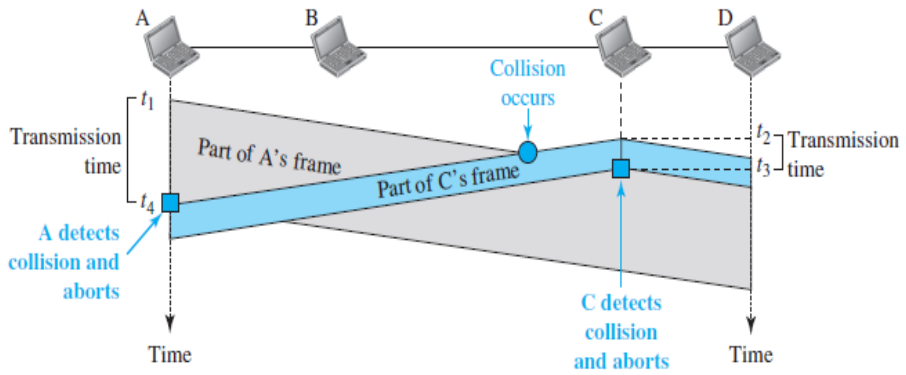
Figure 12.11 *Collision of the first bits in CSMA/CD*



At time *t*1, station A has executed its persistence procedure and starts sending the bits of its frame. At time *t*2, station C has not yet sensed the first bit sent by A. Station C executes its persistence procedure and starts sending the bits in its frame, which propagate both to the left and to the right. The collision occurs sometime after time *t*2. Station C detects a collision at time *t*3 when it receives the first bit of A's frame. Station C immediately (or after a short time, but we assume immediately) aborts transmission. Station A detects collision at time *t*4 when it receives the first bit of C's frame; it also immediately aborts transmission. Looking at the figure, we see that A transmits for the duration *t*4-*t*1; C transmits for the duration *t*3 - *t*2.

Now that we know the time **durations for the two transmissions**, we can show a more complete graph in Figure 12.12.

Figure 12.12 *Collision and abortion in CSMA/CD*



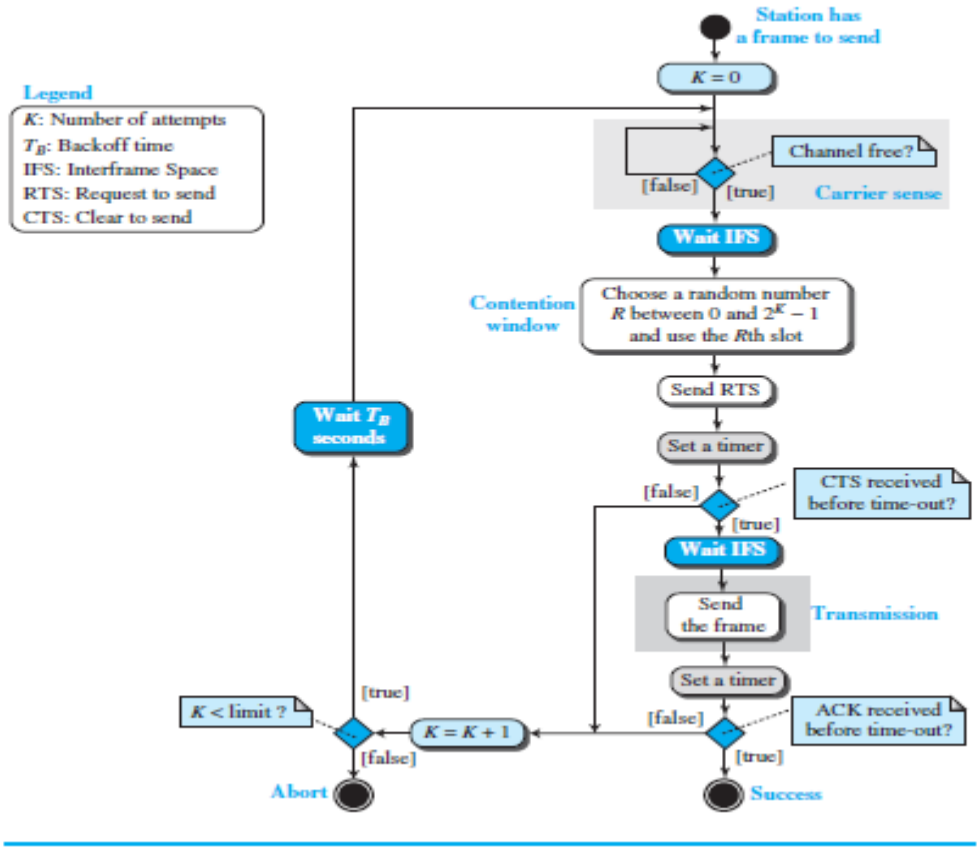
CSMA/CA

Carrier sense multiple access with collision avoidance (CSMA/CA) was invented for wireless networks. Collisions are avoided through the use of CSMA/CA's three strategies:

- The interframe space,
- The contention window, and
- Acknowledgments, as

shown in Figure 12.15. We discuss RTS and CTS frames later.

Figure 12.15 *Flow diagram of CSMA/CA*



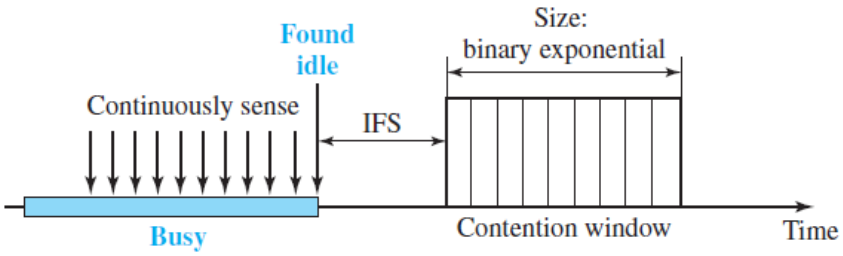
Interframe Space (IFS). First, collisions are avoided by deferring transmission even if the channel is found idle. When an idle channel is found, the station does not send immediately. It waits for a period of time called the ***interframe space*** or ***IFS.*** Even though the channel may appear idle when it is sensed, a distant station may have already started transmitting. The distant station's signal has not yet reached this station. The IFS time allows the front of the transmitted signal by the distant station to reach this station. After waiting an IFS time, if the channel is still idle, the station can send, but it still needs to wait a time equal to the contention window (described next).

The IFS variable can also be used to prioritize stations or frame types. For example, a station that is assigned shorter IFS has a higher priority.

Contention Window. The **contention window** is an amount of time divided into slots. A station that is ready to send chooses a random number of slots as its wait time. The number of slots in the window changes according to the binary exponential backoff strategy. This means that it is set to one slot the first time and then doubles each time the station cannot detect an idle channel after the IFS time. This is very similar to the *p*-persistent method except that a random outcome defines the number of slots taken by the waiting station. One interesting point about the contention window is that the station needs to sense the channel after each time slot.

However, if the station finds the channel busy, it does not restart the process; it just stops the timer and restarts it when the channel is sensed as idle. This gives priority to the station with the longest waiting time. See Figure 12.16.

Figure 12.16 *Contention window*



Acknowledgment. With all these precautions, there still may be a collision resulting in destroyed data. In addition, the data may be corrupted during the transmission. The positive acknowledgment and the time-out timer can help guarantee that the receiver has received the frame.

Frame Exchange Time Line

Figure 12.17 shows the exchange of data and control frames in time.

1. Before sending a frame, the source station senses the medium by checking the energy level at the carrier frequency.

a. The channel uses a persistence strategy with backoff until the channel is idle.

b. After the station is found to be idle, the station waits for a period of time called the ***DCF interframe space (DIFS);*** then the station sends a control frame called the *request to send (RTS)*.

2. After receiving the RTS and waiting a period of time called the ***short interframe space (SIFS),*** the destination station sends a control frame, called the *clear to send (CTS)*, to the source station. This control frame indicates that the destination station is ready to receive data.

3. The source station sends data after waiting an amount of time equal to SIFS.

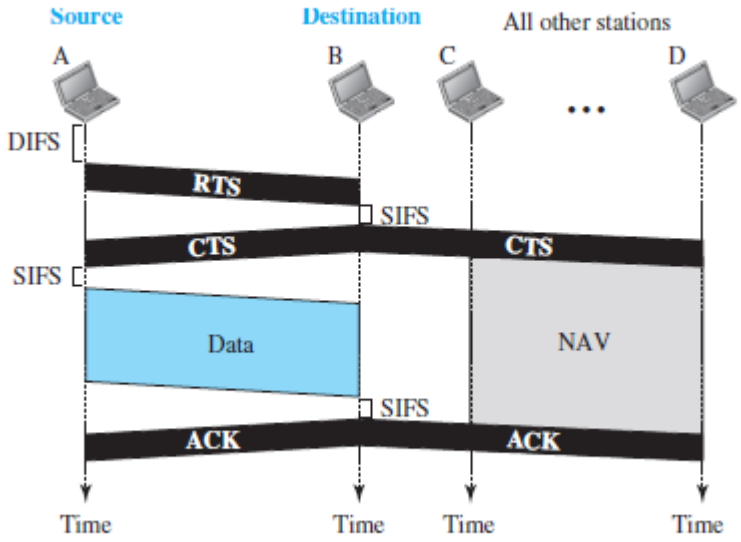
4. The destination station, after waiting an amount of time equal to SIFS, sends an acknowledgment to show that the frame has been received. Acknowledgment is needed in this protocol because the station does not have any means to check for the successful arrival of its data at the destination. On the other hand, the lack of collision in CSMA/CD is a kind of indication to the source that data have arrived.

Network Allocation Vector

When a station sends an RTS frame, it includes the duration of time that it needs to occupy the channel. The stations that are affected by this transmission create a timer called a **network allocation vector (NAV)** that shows how much time must pass before these stations are allowed to check the channel for idleness. Each time a station accesses the system and sends an RTS frame, other

stations start their NAV. Each station, before sensing the physical medium to see if it is idle, first checks its NAV to see if it has expired. Figure 12.17 shows the idea of NAV.

Figure 12.17 *CSMA/CA and NAV*



Collision During Handshaking

What happens if there is a collision during the time when RTS or CTS control frames are in transition, often called the *handshaking period?* Two or more stations may try to send RTS frames at the same time. These control frames may collide. However, because there is no mechanism for collision detection, the sender assumes there has been a collision if it has not received a CTS frame from the receiver. The backoff strategy is employed, and the sender tries again.

Hidden-Station Problem

The solution to the hidden station problem is the use of the handshake frames (RTS and CTS). Figure 12.17 also shows that the RTS message from B reaches A, but not C. However, because both B and C are within the range of A, the CTS message, which contains the duration of data transmission from B to A, reaches C. Station C knows that some hidden station is using the channel and refrains from transmitting until that duration is over.

CSMA/CA and Wireless Networks

CSMA/CA was mostly intended for use in wireless networks. The procedure described above, however, is not sophisticated enough to handle some particular issues related to wireless networks, such as hidden terminals or exposed terminals. We will see how these issues are solved by augmenting the above protocol with handshaking features.

2.5 Controlled Access

In **controlled access,** the stations consult one another to find which station has the right to send. A station cannot send unless it has been authorized by other stations.

The important three controlled-access methods.

- ***Reservation***
- ***Polling***
- ***Token Passing***

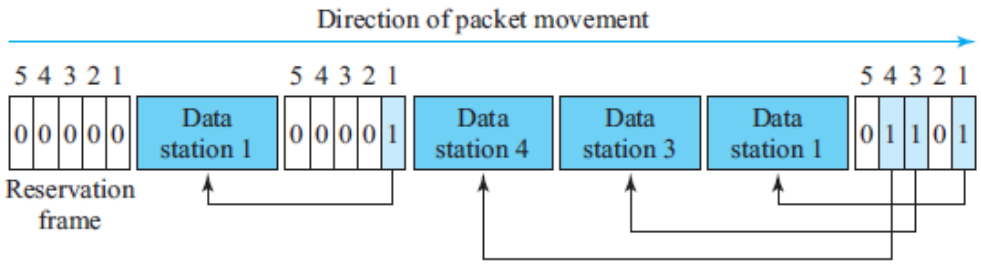
Reservation:

In the **reservation** method,

- A station needs to make a reservation before sending data.
- Time is divided into intervals. In each interval, a reservation frame precedes the data frames sent in that interval.
- If there are *N* stations in the system, there are exactly *N* reservation minislots in the reservation frame.
- Each minislot belongs to a station. When a station needs to send a data frame, it makes a reservation in its own minislot.

The stations that have made reservations can send their data frames after the reservation frame. Figure 12.18 shows a situation with five stations and a five-minislot reservation frame. In the first interval, only stations 1, 3, and 4 have made reservations. In the second interval, only station 1 has made a reservation.

Figure 12.18 *Reservation access method*

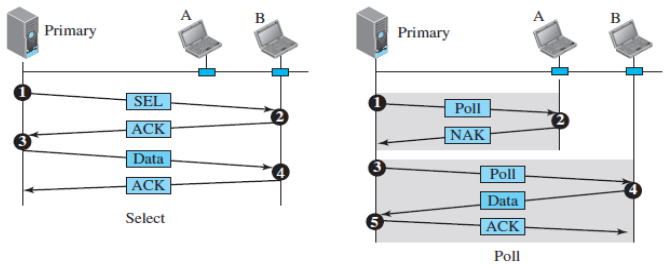


Polling

- **Polling** works with topologies in which one device is designated as a ***primary station*** and the other devices are ***secondary stations*.**
- All data exchanges must be made through the primary device even when the ultimate destination is a secondary device.
- The primary device controls the link;

- The secondary devices follow its instructions. It is up to the primary device to determine which device is allowed to use the channel at a given time.
- The primary device, therefore, is always the initiator of a session (see Figure 12.19). This method uses poll and select functions to prevent collisions. However, the drawback is if the primary station fails, the system goes down.

Figure 12.19 *Select and poll functions in polling-access method*



Select

The *select* function is used whenever the primary device has something to send. Remember that the primary controls the link. If the primary is neither sending nor receiving data, it knows the link is available. If it has something to send, the primary device sends it. What it does not know, however, is whether the target device is prepared to receive. So the primary must alert the secondary to the upcoming transmission and wait for an acknowledgment of the secondary's ready status. Before sending data, the primary creates and transmits a select (SEL) frame, one field of which includes the address of the intended secondary.

Poll

The *poll* function is used by the primary device to solicit transmissions from the secondary devices. When the primary is ready to receive data, it must ask (poll) each device in turn if it has anything to send. When the first secondary is approached, it responds either with a NAK frame if it has nothing to send or with data (in the form of a data frame) if it does. If the response is negative (a NAK frame), then the primary polls the next secondary in the same manner until it finds one with data to send. When the response is positive (a data frame), the primary reads the frame and returns an acknowledgment (ACK frame), verifying its receipt.

Token Passing

- In the **token-passing** method, the stations in a network are organized in a logical ring.
- For each station, there is a *predecessor* and a *successor*.
- The predecessor is the station which is logically before the station in the ring.
- The successor is the station which is after the station in the ring.
- The current station is the one that is accessing the channel now.
- The right to this access has been passed from the predecessor to the current station.

- The right will be passed to the successor when the current station has no more data to send.

In this method, a special packet called a ***token*** circulates through the ring. The possession of the token gives the station the right to access the channel and send its data. When a station has some data to send, it waits until it receives the token from its predecessor. It then holds the token and sends its data. When the station has no more data to send, it releases the token, passing it to the next logical station in the ring. The station cannot send data until it receives the token again in the next round. In this process, when a station receives the token and has no data to send, it just passes the data to the next station. Token management is needed for this access method. Stations must be limited in the time they can have possession of the token. The token must be monitored to ensure it has not been lost or destroyed.

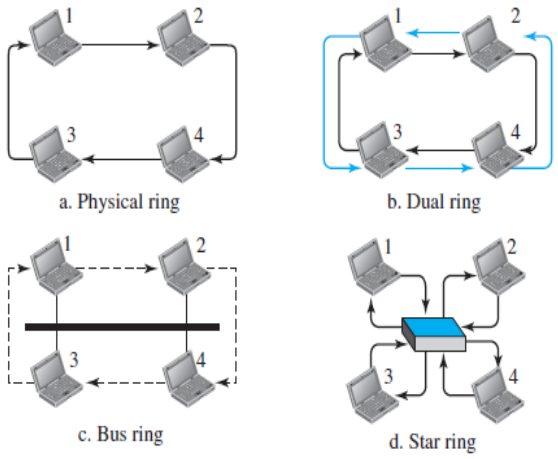
For example, if a station that is holding the token fails, the token will disappear from the network. Another function of token management is to assign priorities to the stations and to the types of data being transmitted. And finally, token management is needed to make low-priority stations release the token to high-priority stations.

Logical Ring

In a token-passing network, stations do not have to be physically connected in a ring. The ring can be a logical one. Figure 12.20 shows four different physical topologies that can create a logical ring.

- ***Physical ring***
- ***Dual ring***
- ***Bus ring***
- ***Star ring***

Figure 12.20 *Logical ring and physical topology in token-passing access method*



In the **physical ring topology**, when a station sends the token to its successor, the token cannot be seen by other stations; the successor is the next one in line. This means that the token does

not have to have the address of the next successor. The problem with this topology is that if one of the links the medium between two adjacent stations fails, the whole system fails.

The **dual ring topology** uses a second (auxiliary) ring which operates in the reverse direction compared with the main ring. The second ring is for emergencies only (such as a spare tire for a car). If one of the links in the main ring fails, the system automatically combines the two rings to form a temporary ring. After the failed link is restored, the auxiliary ring becomes idle again. Note that for this topology to work, each station needs to have two transmitter ports and two receiver ports. The high-speed Token Ring networks called *FDDI (Fiber Distributed Data Interface)* and *CDDI (Copper Distributed Data Interface)* use this topology.

In the **bus ring topology**, also called a token bus, the stations are connected to a single cable called a *bus.* They, however, make a logical ring, because each station knows the address of its successor (and also predecessor for token management purposes). When a station has finished sending its data, it releases the token and inserts the address of its successor in the token. Only the station with the address matching the destination address of the token gets the token to access the shared media. The Token Bus LAN, standardized by IEEE, uses this topology.

In a **starring topology**, the physical topology is a star. There is a hub, however, that acts as the connector. The wiring inside the hub makes the ring; the stations are connected to this ring through the two wire connections. This topology makes the network less prone to failure because if a link goes down, it will be bypassed by the hub and the rest of the stations can operate. Also adding and removing stations from the ring is easier. This topology is still used in the Token Ring LAN designed by IBM.

2.6 CHANNELIZATION

Channelization is a multiple-access method in which the available bandwidth of a link is shared in time, frequency, or through code, among different stations. There are three channelization protocols:

- ***FDMA***
- ***TDMA***
- ***CDMA.***

FDMA

In telecommunications, **frequency-division multiplexing (FDM)** is a technique by which the total bandwidth available in a communication medium is divided into a series of non-overlapping frequency bands, each of which is used to carry a separate signal. This allows a single transmission medium such as a cable or optical fiber to be shared by multiple independent signals. Another use is to carry separate serial bits or segments of a higher rate signal in parallel.

The most natural example of **frequency-division multiplexing** is radio and television broadcasting, in which multiple radio signals at different frequencies pass through the air at the same time. Another example is cable television, in which many television channels are carried simultaneously on a single cable. FDM is also used by telephone systems to transmit multiple telephone calls through high capacity trunklines, communications satellites to transmit multiple channels of data on uplink and downlink radio beams, and broadband DSL modems to transmit large amounts of computer data through twisted pair telephone lines, among many other uses.

An analogous technique called wavelength division multiplexing is used in fiber-optic communication, in which multiple channels of data are transmitted over a single optical fiber using different wavelengths (frequencies) of light.

The multiple separate information **(modulation)** signals that are sent over an FDM system, such as the video signals of the television channels that are sent over a cable TV system, are called baseband signals. At the source end, for each frequency channel, an electronic oscillator generates a carrier signal, a steady oscillating waveform at a single frequency that serves to "carry" information. The carrier is much higher in frequency than the baseband signal. The carrier signal and the baseband signal are combined in a modulator circuit. The modulator alters some aspect of the carrier signal, such as its amplitude, frequency, or phase, with the baseband signal, "piggybacking" the data onto the carrier.

The result of **modulating** (mixing) the carrier with the baseband signal is to generate sub-frequencies near the carrier frequency, at the sum (fC + fB) and difference (fC − fB) of the frequencies. The information from the modulated signal is carried in sidebands on each side of the carrier frequency. Therefore, all the information carried by the channel is in a narrow band of frequencies clustered around the carrier frequency, this is called the passband of the channel.

Similarly, additional baseband signals are used to modulate carriers at other frequencies, creating other channels of information. The carriers are spaced far enough apart in frequency that the

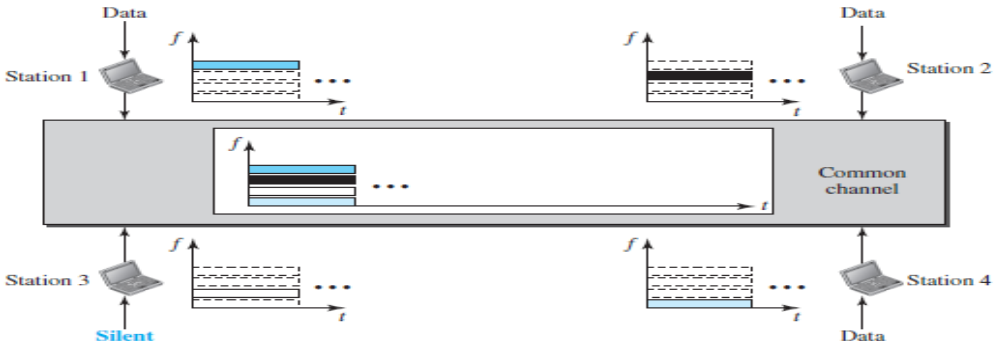
bands of frequencies occupied by each channel, the passbands of the separate channels, do not overlap. All the channels are sent through the transmission medium, such as a coaxial cable, optical fiber, or through the air using a radio transmitter. As long as the channel frequencies are spaced far enough apart that none of the passbands overlap, the separate channels will not interfere with each other. Thus the available bandwidth is divided into "slots" or channels, each of which can carry a separate modulated signal.

For example, the **coaxial cable** used by cable television systems has a bandwidth of about 1000 MHz, but the passband of each television channel is only 6 MHz wide, so there is room for many channels on the cable (in modern digital cable systems each channel in turn is subdivided into sub-channels and can carry up to 10 digital television channels).

At the destination end of the cable or fiber, or the radio receiver, for each channel a local oscillator produces a signal at the carrier frequency of that channel, that is mixed with the incoming modulated signal. The frequencies subtract, producing the baseband signal for that channel again. This is called **demodulation.** The resulting baseband signal is filtered out of the other frequencies and output to the user.

- In **frequency-division multiple access (FDMA),** the available bandwidth is divided into frequency bands.
- Each station is allocated a band to send its data.
- Each band is reserved for a specific station, and it belongs to the station all the time.
- Each station also uses a band pass filter to confine the transmitter frequencies.
- To prevent station interferences, the allocated bands are separated from one another by small ***guard bands***
- FDMA specifies a predetermined frequency band for the entire period of communication. This means that stream data can easily be used with FDMA.
- FDMA is a physical layer technique that combines the loads from low-bandwidth channels and transmits them by using a high-bandwidth channel.
- The channels that are combined are low-pass. The multiplexer modulates the signals, combines them, and creates a band-pass signal. The bandwidth of each channel is shifted by the multiplexer
- FDMA, on the other hand, is an access method in the **data-link layer**. The **data-link layer** in each station tells its physical layer to make a **band-pass signal** from the data passed to it.
- The signal must be created in the allocated band. There is no physical multiplexer at the physical layer. The signals created at each station are automatically band-pass-filtered. They are mixed when they are sent to the common channel.
- FDMA is used in cellular networks.

Figure 12.21 *Frequency-division multiple access (FDMA)*

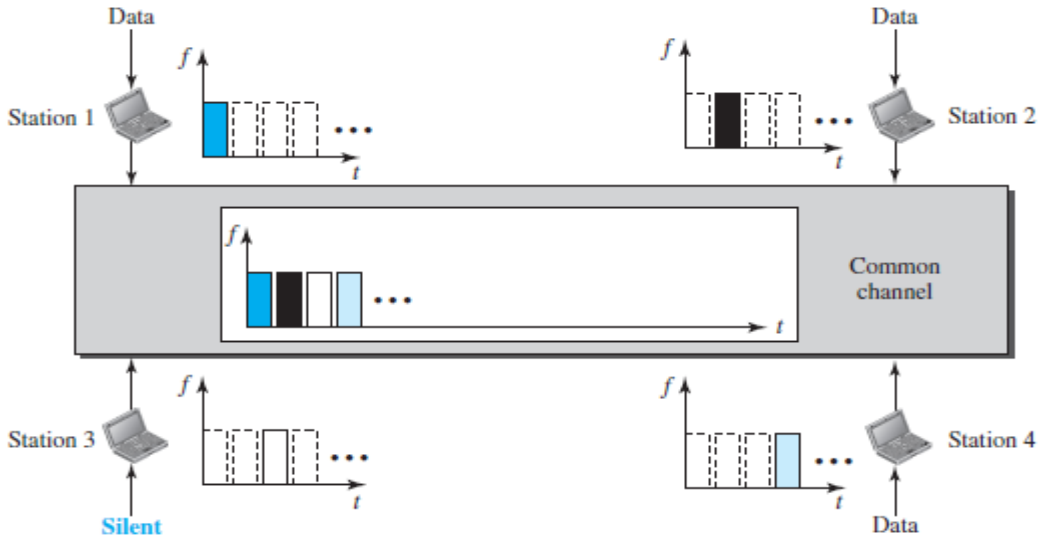


TDMA

Time-division multiplexing (**TDM**) is a method of transmitting and receiving independent signals over a common signal path by means of synchronized switches at each end of the transmission line so that each signal appears on the line only a fraction of time in an alternating pattern. It is used when the bit rate of the transmission medium exceeds that of the signal to be transmitted.

In **time-division multiple access (TDMA),** the stations share the bandwidth of the channel in time. Each station is allocated a time slot during which it can send data. Each station transmits its data in its assigned time slot.

Figure 12.22 *Time-division multiple access (TDMA)*



- The main problem with TDMA lies in achieving synchronization between the different stations.
- Each station needs to know the beginning of its slot and the location of its slot. This may be difficult because of propagation delays introduced in the system if the stations are spread over a large area.

- To compensate for the delays, we can insert *guard times*. Synchronization is normally accomplished by having some synchronization bits (normally referred to as *preamble bits*) at the beginning of each slot.
- TDMA is a physical layer technique that combines the data from slower channels and transmits them by using a faster channel.
- The process uses a physical multiplexer that interleaves data units from each channel.
- TDMA, on the other hand, is an access method in the data-link layer.
- The data-link layer in each station tells its physical layer to use the allocated time slot. There is no physical multiplexer at the physical layer.
- TDMA used in wired communications.
- The mobile station or the user need not tune into a certain frequency ie.., the same frequency can be used all the times. Such a flexible access scheme provided by TDMA.
- Due to ability to use the same frequency, media access can be controlled through simple and efficient algorithms.
- TDMA makes access to timely divided channels with the same frequency.
- Token ring, ATM, Ethernet are the MAC schemes for wired networks, works on the principle of TDMA.

CDMA

Code-division multiple access (CDMA) was conceived several decades ago. Recent advances in electronic technology have finally made its implementation possible. **CDMA** differs from **FDMA** in that only one channel occupies the **entire bandwidth** of the link. It differs from TDMA in that all stations can send data simultaneously; there is no timesharing.

Code division multiplexing (CDM) is a networking technique in which multiple data signals are combined for simultaneous transmission over a common frequency band.

When CDM is used to allow multiple users to share a single communications channel, the technology is called code division multiple access (CDMA).

CDMA provides a certain amount of built-in security, as the transmissions of multiple users are mixed together within the frequency spectrum. The spreading code is required to decode a specific transmission.

CDMA is used as the access method in many mobile phone standards. IS-95, also called "cdmaOne", and its 3G evolution CDMA2000, are often simply referred to as "CDMA", but UMTS, the 3G standard used by GSM carriers, also uses "wideband CDMA", or W-CDMA, as well as TD-CDMA and TD-SCDMA, as its radio technologies.

Analogy

Let us first give an analogy. CDMA simply means communication with different codes.

Idea

Let us assume we have four stations, 1, 2, 3, and 4, connected to the same channel. The data from station 1 are *d*1, from station 2 are *d*2, and so on. The code assigned to the first station is *c*1, to the second is *c*2, and so on. We assume that the assigned codes have two properties.

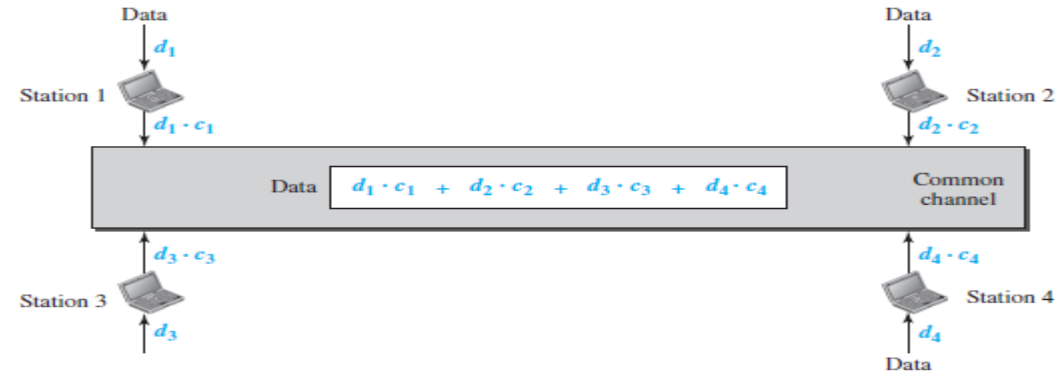
1. If we multiply each code by another, we get 0.

2. If we multiply each code by itself, we get 4 (the number of stations).

With these two properties in mind, let us see how the above four stations can send data using the same common channel, as shown in Figure 12.23. Station 1 multiplies (a special kind of multiplication, as we will see) its data by its code to get *d*1. *c*1. Station 2 multiplies its data by its code to get *d*2.*c*2, and so on. The data that go on the channel are the sum of all these terms, as shown in the box. Any station that wants to receive data from one of the other three multiplies the data on the channel by the code of the sender. For example, suppose stations 1 and 2 are talking to each other. Station 2 wants to hear what station 1 is saying. It multiplies the data on the channel by *c*1, the code of station 1.

Because (*c*1. *c*1) is 4, but (*c*2. *c*1), (*c*3. *c*1), and (*c*4. *c*1) are all 0s, station 2 divides the result by 4 to get the data from station 1.

Figure 12.23 *Simple idea of communication with code*



$$\text{data} = (d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4) \cdot c_1$$
$$= d_1 \cdot c_1 \cdot c_1 + d_2 \cdot c_2 \cdot c_1 + d_3 \cdot c_3 \cdot c_1 + d_4 \cdot c_4 \cdot c_1 = 4 \times d_1$$

Chips

CDMA is based on coding theory. Each station is assigned a code, which is a sequence of numbers called *chips*, as shown in Figure 12.24. The codes are for the previous example. Later in this chapter we show how we chose these sequences. For now, we need to know that we did not choose the sequences randomly; they were carefully selected. They are called ***orthogonal sequences*** and have the following properties:

Figure 12.24 *Chip sequences*

1. Each sequence is made of N elements, where N is the number of stations.
2. If we multiply a sequence by a number, every element in the sequence is multiplied by that element. This is called multiplication of a sequence by a scalar. For example,

$$2 \cdot [+1 +1 -1 -1] = [+2 +2 -2 -2]$$

3. If we multiply two equal sequences, element by element, and add the results, we get N, where N is the number of elements in each sequence. This is called the ***inner product*** of two equal sequences. For example,

$$[+1 +1 -1 -1] \cdot [+1 +1 -1 -1] = 1 + 1 + 1 + 1 = 4$$

4. If we multiply two different sequences, element by element, and add the results, we get 0. This is called the *inner product* of two different sequences. For example,

$$[+1 +1 -1 -1] \cdot [+1 +1 +1 +1] = 1 + 1 - 1 - 1 = 0$$

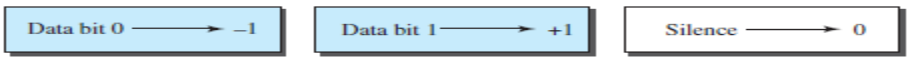
5. Adding two sequences means adding the corresponding elements. The result is another sequence. For example,

$$[+1 +1 -1 -1] + [+1 +1 +1 +1] = [+2 +2 \ 0 \ 0]$$

Data Representation

We follow these rules for encoding: If a station needs to send a 0 bit, it encodes it as −1. If it needs to send a 1 bit, it encodes it as +1. When a station is idle, it sends no signal, which is interpreted as a 0. These are shown in Figure 12.25.

Figure 12.25 *Data representation in CDMA*



Encoding and Decoding

As a simple example, we show how four stations share the link during a 1-bit interval. The procedure can easily be repeated for additional intervals. We assume that stations 1 and 2 are sending a 0 bit and channel 4 is sending a 1 bit. Station 3 is silent. The data at the sender site are translated to −1, −1, 0, and +1. Each station multiplies the corresponding number by its chip (its orthogonal sequence), which is unique for each station. The result is a new sequence which is sent to the channel. For simplicity, we assume that all stations send the resulting sequences at the same time. The sequence on the channel is the sum of all four sequences as defined before. Figure 12.26 shows the situation.

Now imagine that station 3, which we said is silent, is listening to station 2. Station 3 multiplies the total data on the channel by the code for station 2, which is [+1 -1 +1 -1], to get

$$[-1 -1 -3 +1] \cdot [+1 -1 +1 -1] = -4/4 = -1 = \textbf{bit 1}$$

Connecting Devices and Virtual LAN's: *Connecting Devices.*

CONNECTING DEVICES

Hosts and networks do not normally operate in isolation. C**onnecting devices** is used to connect hosts together to make a network or to connect networks together to make an internet. Connecting devices can operate in different layers of the Internet model.

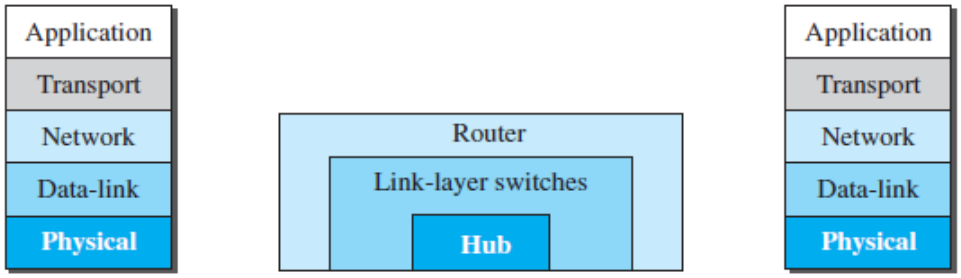
There are three kinds of ***connecting devices:***

Hubs: Hubs today operate in the first layer of the Internet model.

Link-layer switches*:* Link-layer switches operate in the first two layers.

Routers*:* Routers operate in the first three layers.

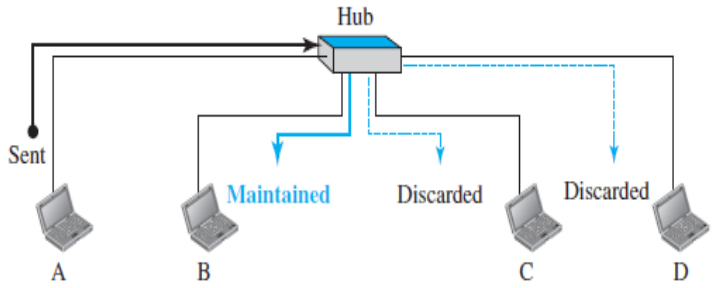
Figure 17.1 *Three categories of connecting devices*



Hubs

- A **hub** is a device that operates only in the physical layer. Signals that carry information within a network can travel a fixed distance before attenuation endangers the integrity of the data.
- A **repeater** receives a signal and, before it becomes too weak or corrupted, *regenerates* and *retimes* the original bit pattern. The repeater then sends the refreshed signal.
- A repeater was used to connect two segments of a LAN to overcome the length restriction of the coaxial cable. Today, however, Ethernet LANs use star topology.
- In a star topology, a repeater is a multiport device, often called a *hub that* can be used to serve as the connecting point and at the same time function as a repeater.
- when a packet from station A to station B arrives at the hub, the signal representing the frame is regenerated to remove any possible corrupting noise, but the hub forwards the packet from all outgoing ports except the one from which the signal was received.
- The frame is broadcast. All stations in the LAN receive the frame, but only station B keeps it. The rest of the stations discard it.
- A hub or a repeater is a physical-layer device. They do not have a link-layer address and they do not check the link-layer address of the received frame. They just regenerate the corrupted bits and send them out from every port.

Figure 17.2 *A hub*



Link-Layer Switches

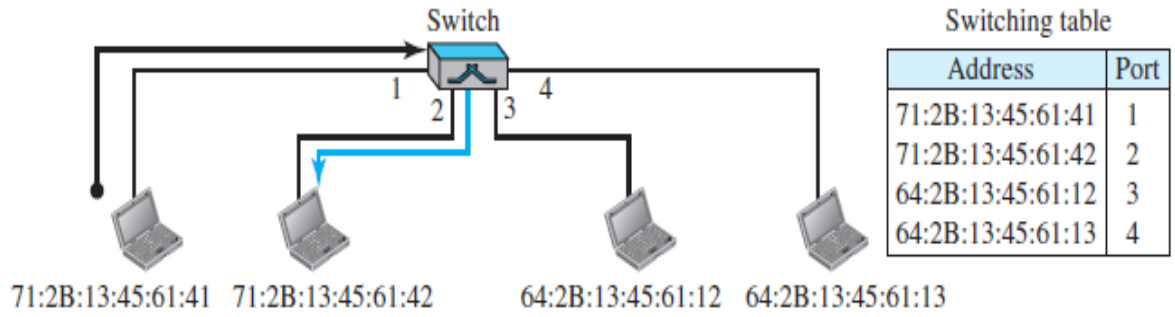
A **link-layer switch** (or ***switch***) operates in both the **physical** and the **data-link layers**. As a physical-layer device, it regenerates the signal it receives. As a link-layer device, the link-layer switch can check the MAC addresses (source and destination) contained in the frame.

Filtering

A link-layer switch has **filtering** capability. It can check the destination address of a frame and can decide from which outgoing port the frame should be sent.

Let us give an example. In Figure 17.3, we have a LAN with four stations that are connected to a link-layer switch. If a frame destined for station 71:2B:13:45:61:42 arrives at port 1, the link-layer switch consults its table to find the departing port. According to its table, frames for 71:2B:13:45:61:42 should be sent out only through port 2; therefore, there is no need for forwarding the frame through other ports.

Figure 17.3 *Link-layer switch*



Switching table

Address	Port
71:2B:13:45:61:41	1
71:2B:13:45:61:42	2
64:2B:13:45:61:12	3
64:2B:13:45:61:13	4

Transparent Switches

A **transparent switch** is a switch in which the stations are completely unaware of the switch's existence. If a switch is added or deleted from the system, reconfiguration of the stations is unnecessary. According to the IEEE 802.1d specification, a system equipped with transparent switches must meet **three** criteria:

- Frames must be **forwarded** from one station to another.
- The forwarding table is automatically made **by learning frame** movements in the network.
- **Loops** in the system must be prevented.

Forwarding

A transparent switch must correctly forward the frames,.

Learning

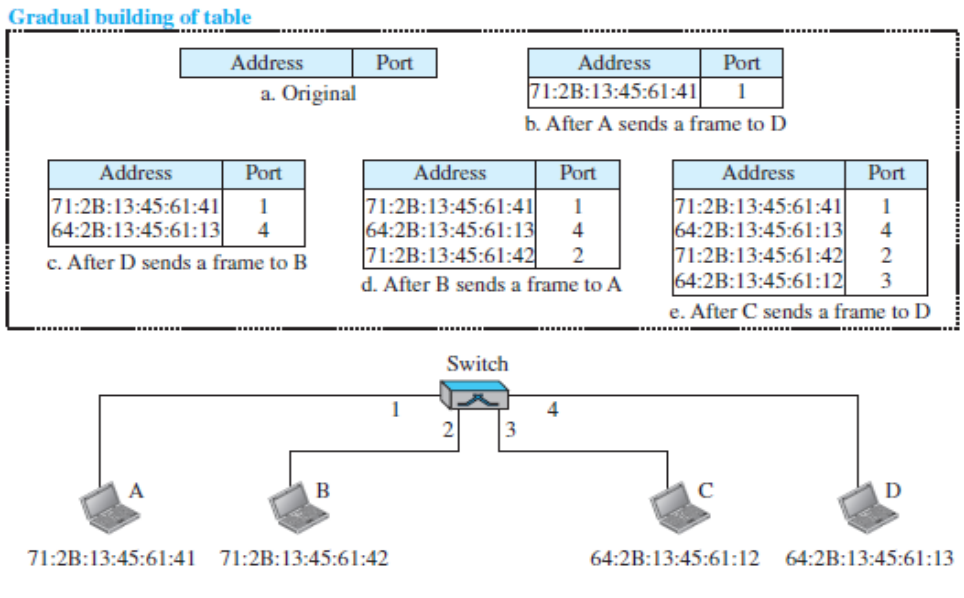
The earliest switches had switching tables that were static. The system administrator would manually enter each table entry during switch setup.

If a station was added or deleted, the table had to be modified manually. The same was true if a station's MAC address changed, which is not a rare event. For example, putting in a new network card means a new MAC address.

A better solution to the static table is a dynamic table that maps addresses to ports (interfaces) automatically. To make a table dynamic, we need a switch that gradually learns from the frames' movements. To do this, the switch inspects both the destination and the source addresses in each frame that passes through the switch.

The destination address is used for the forwarding decision (table lookup); the source address is used for adding entries to the table and for updating purposes.

Figure 17.4 *Learning switch*



1. When station A sends a frame to station D, the switch does not have an entry for either D or A. The frame goes out from all three ports; the frame floods the network. However, by looking at the source address, the switch learns that station A must be connected to port 1. This means that frames destined for A, in the future, must be sent out through port 1. The switch adds this entry to its table. The table has its first entry now.

2. When station D sends a frame to station B, the switch has no entry for B, so it floods the network again. However, it adds one more entry to the table related to station D.

3. The learning process continues until the table has information about every port. However, note that the learning process may take a long time. For example, if a station does not send out a frame (a rare situation), the station will never have an entry in the table.

Loop Problem

Transparent switches work fine as long as there are no redundant switches in the system. Systems administrators, however, like to have redundant switches (more than one switch between a pair of LANs) to make the system more reliable. If a switch fails, another switch takes over until the failed one is repaired or replaced. Redundancy can create loops in the system, which is very undesirable. Loops can be created only when two or more broadcasting LANs (those using hubs, for example) are connected by more than one switch.

Figure 17.5 shows a very simple example of a loop created in a system with two LANs connected by two switches.

1. Station A sends a frame to station D. The tables of both switches are empty. Both forward the frame and update their tables based on the source address A.

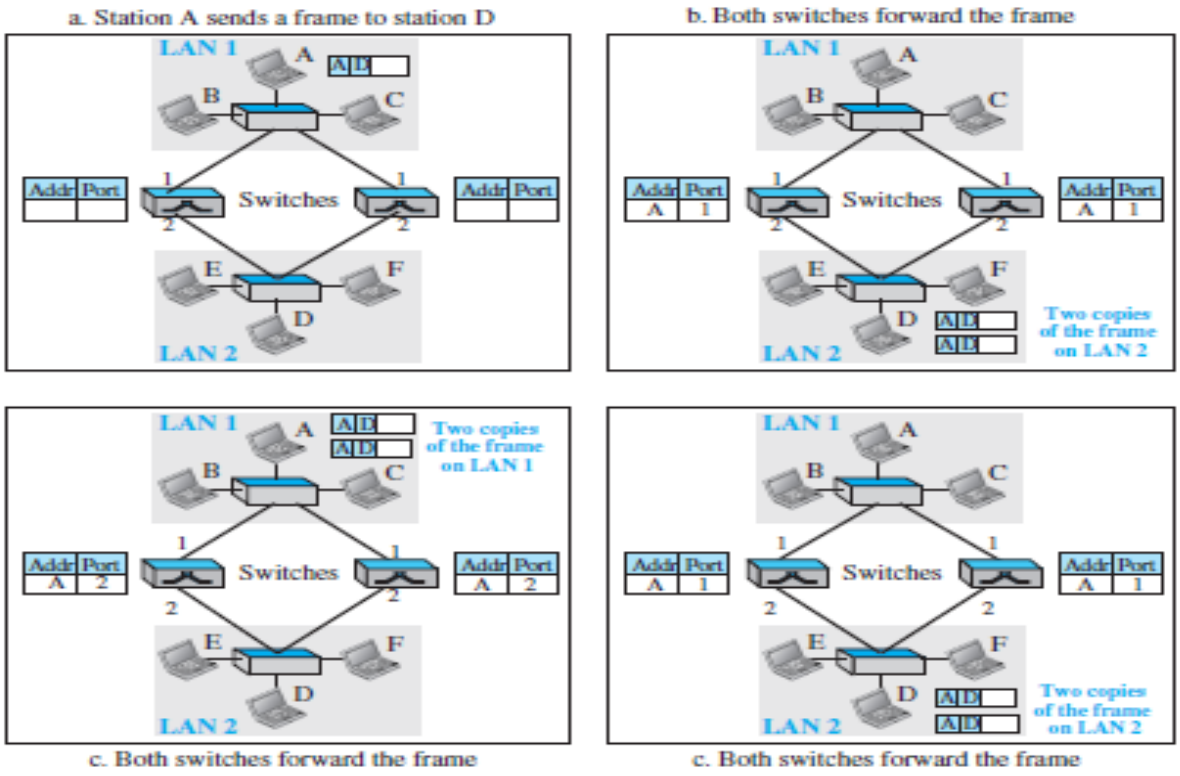
2. Now there are two copies of the frame on LAN 2. The copy sent out by the left switch is received by the right switch, which does not have any information about the destination address D; it forwards the frame. The copy sent out by the right switch is received by the left switch and is sent out for lack of information about D.

Note that each frame is handled separately because switches, as two nodes on a broadcast network sharing the medium, use an access method such as CSMA/CD. The tables of both switches are updated, but still there is no information for destination D.

3. Now there are two copies of the frame on LAN 1. Step 2 is repeated, and both copies are sent to LAN2.

4. The process continues on and on. Note that switches are also repeaters and regenerate frames. So in each iteration, there are newly generated fresh copies of the frames.

Figure 17.5 *Loop problem in a learning switch*



Spanning Tree Algorithm

To solve the looping problem, the IEEE specification requires that switches use the spanning tree algorithm to create a loop-less topology. In graph theory, a **spanning tree** is a graph in which there is no loop. In a switched LAN, this means creating a topology in which each LAN can be reached from any other LAN through one path only (no loop). We cannot change the physical topology of the system because of physical connections between cables and switches, but we can create a logical topology that overlays the physical one. Figure 17.6 shows a system with four LANs and five switches. We have shown the physical system and its representation in graph theory.

The connecting arcs show the connection of a LAN to a switch and vice versa. To find the spanning tree, we need to assign a cost (metric) to each arc. The interpretation of the cost is left up to the systems administrator. We have chosen the minimum hops. The hop count is normally 1 from a switch to the LAN and 0 in the reverse direction.

The process for finding the spanning tree involves three steps:

1. Every switch has a built-in ID (normally the serial number, which is unique). Each switch broadcasts this ID so that all switches know which one has the smallest ID. The switch with the smallest ID is selected as the *root* switch (root of the tree). We assume that switch S1 has the smallest ID. It is, therefore, selected as the root switch.

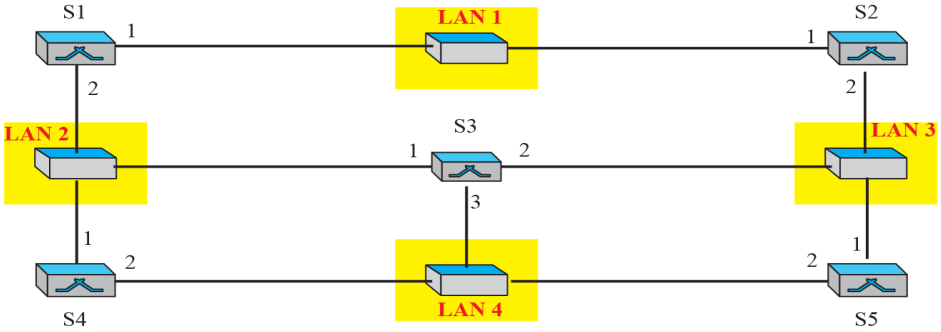
2. The algorithm tries to find the shortest path (a path with the shortest cost) from the root switch to every other switch or LAN. The shortest path can be found by examining the total cost from the root switch to the destination. Figure 17.7 shows the shortest paths.

3. The combination of the shortest paths creates the shortest tree, which is also shown in Figure 17.7.

4. Based on the spanning tree, we mark the ports that are part of it, the **forwarding ports,** which forward a frame that the switch receives. We also mark those ports that are not part of the spanning tree, the **blocking ports,** which block the frames received by the switch. Figure 17.8 shows the logical systems of LANs with forwarding ports (solid lines) and blocking ports (broken lines).

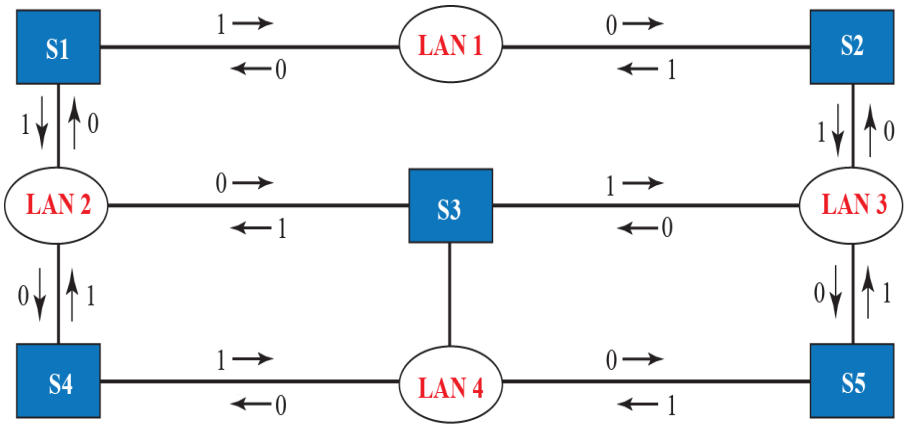
Note that there is only one path from any LAN to any other LAN in the spanning tree system. This means there is only one path from one LAN to any other LAN. No loops are created. You can prove to yourself that there is only one path from LAN 1 to LAN 2, LAN 3, or LAN 4. Similarly, there is only one path from LAN 2 to LAN 1, LAN 3, and LAN 4. The same is true for LAN 3 and LAN 4.

We have described the spanning tree algorithm as though it required manual entries. This is not true. Each switch is equipped with a software package that carries out this process dynamically.



a. Actual system

Figure 17.6: A system of connected LANs and its graph (Part a)



b. Graph representation with cost assigned to each arc

Figure 17.6: A system of connected LANs and its graph (Part b)

Figure 17.7 *Finding the shortest paths and the spanning tree in a system of switches*

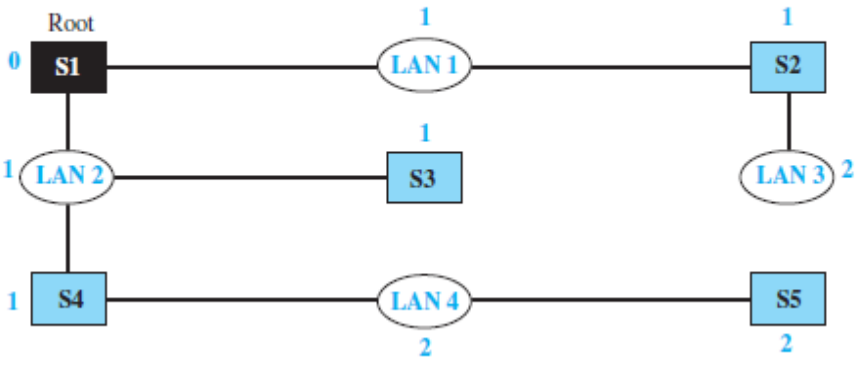
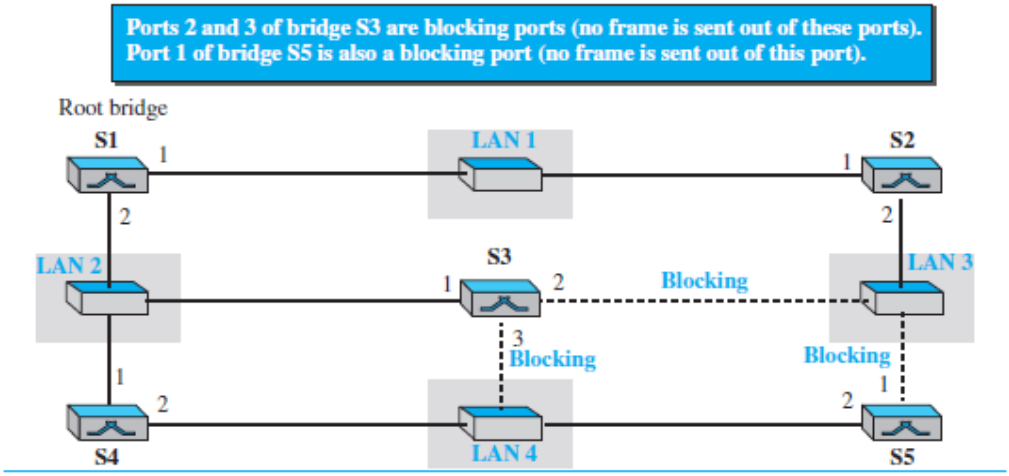


Figure 17.8 *Forwarding and blocking ports after using spanning tree algorithm*



Advantages of Switches

A link-layer switch has several advantages over a hub. We discuss only two of them here.

Collision Elimination

A link-layer switch eliminates the collision. This means increasing the average bandwidth available to a host in the network. In a switched LAN, there is no need for carrier sensing and collision detection; each host can transmit at any time.

Connecting Heterogeneous Devices

A link-layer switch can connect devices that use different protocols at the physical layer (data rates) and different transmission media. As long as the format of the frame at the data-link layer does not change, a switch can receive a frame from a device that uses twisted-pair cable and sends data at 10 Mbps and deliver the frame to another device that uses fiber-optic cable and can receive data at 100 Mbps.

Routers

- A **router** is a three-layer device; it operates in the physical, data-link, and network layers.
- As a physical-layer device, it regenerates the signal it receives. As a link-layer device, the router checks the physical addresses (source and destination) contained in the packet. As a network-layer device, a router checks the network-layer addresses.
- A router can connect networks. In other words, a router is an internetworking device; it connects independent networks to form an internetwork. According to this definition, two networks connected by a router become an internetwork or an internet.

There are three major differences between a router and a repeater or a switch.

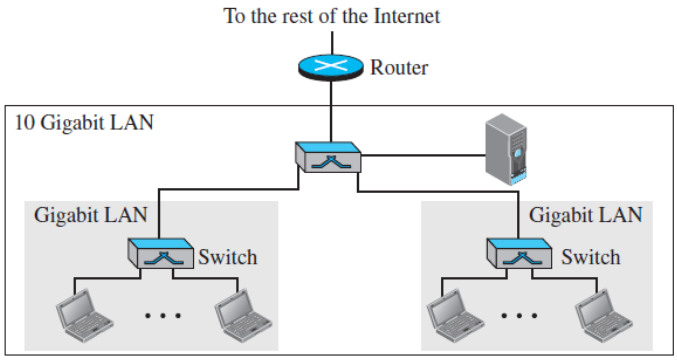
1. A router has a physical and logical (IP) address for each of its interfaces.

2. A router acts only on those packets in which the link-layer destination address matches the address of the interface at which the packet arrives.

3. A router changes the link-layer address of the packet (both source and destination) when it forwards the packet.

Let us give an example. In Figure 17.9, assume an organization has two separate buildings with a Gigabit Ethernet LAN installed in each building. The organization uses switches in each LAN. The two LANs can be connected to form a larger LAN using 10 Gigabit Ethernet technologies that speeds up the connection to the Ethernet and the connection to the organization server. A router then can connect the whole system to the Internet. A router, will change the MAC addresses it receives because the MAC addresses have only local jurisdictions.

Figure 17.9 *Routing example*



VIRTUAL LANS

A station is considered part of a LAN if it physically belongs to that LAN. The criterion of membership is geographic. What happens if we need a virtual connection between two stations belonging to two different physical LANs? We can roughly define a **virtual local area network (VLAN)** as a local area network configured by software, not by physical wiring.

Let us use an example to elaborate on this definition. Figure 17.10 shows a switched LAN in an engineering firm in which nine stations are grouped into three LANs that are connected by a switch.

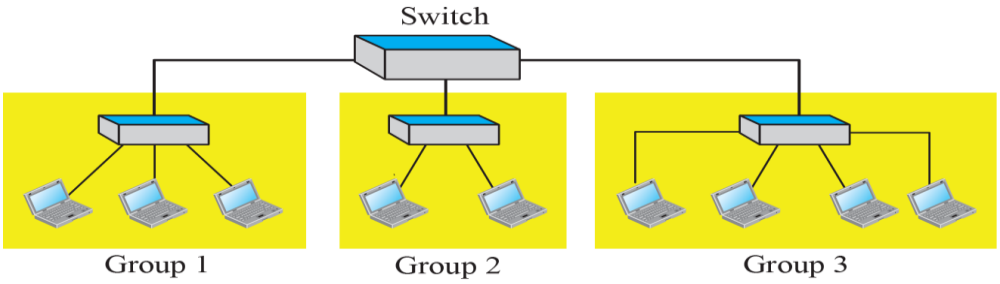


Figure 17.10: A switch connecting three LANs by wire

The first three engineers work together as the first group, the next two engineers work together as the second group, and the last four engineers work together as the third group. The LAN is configured to allow this arrangement.

But what would happen if the administrators needed to move two engineers from the first group to the third group, to speed up the project being done by the third group?. The LAN configuration would need to be changed. The network technician must rewire. The problem is repeated if, in another week, the two engineers move back to their previous group. In a switched LAN, changes in the work group mean physical changes in the network configuration.

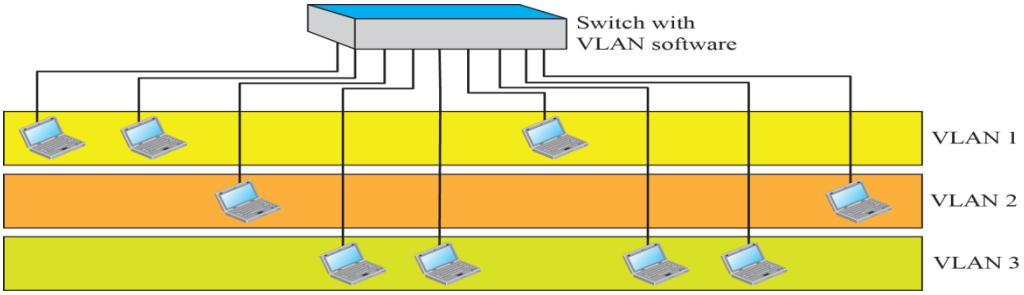


Figure 17.11: A switch using VLAN software

Figure 17.11 shows the same switched LAN divided into VLANs. The whole idea of VLAN technology is to divide a LAN into logical, instead of physical, segments. A LAN can be divided into several logical LANs, called *VLANs*. Each VLAN is a work group in the organization. If a person moves from one group to another, there is no need to change the physical configuration. The group membership in VLANs is defined by software, not hardware. Any station can be logically moved to another VLAN. All members belonging to a VLAN can receive broadcast messages sent to that

particular VLAN. This means that if a station moves from VLAN 1 to VLAN 2, it receives broadcast messages sent to VLAN 2, but no longer receives broadcast messages sent to VLAN 1.

It is obvious that the problem in our previous example can easily be solved by using VLANs. Moving engineers from one group to another through software is easier than changing the configuration of the physical network.

VLAN technology even allows the grouping of stations connected to different switches in a VLAN. Figure 17.12 shows a backbone local area network with two switches and three VLANs. Stations from switches A and B belong to each VLAN.

This is a good configuration for a company with two separate buildings. Each building can have its own switched LAN connected by a backbone. People in the first building and people in the second building can be in the same work group even though they are connected to different physical LANs.

From these three examples, we can see that a VLAN defines broadcast domains. VLANs group stations belonging to one or more physical LANs into broadcast domains. The stations in a VLAN communicate with one another as though they belonged to a physical segment.

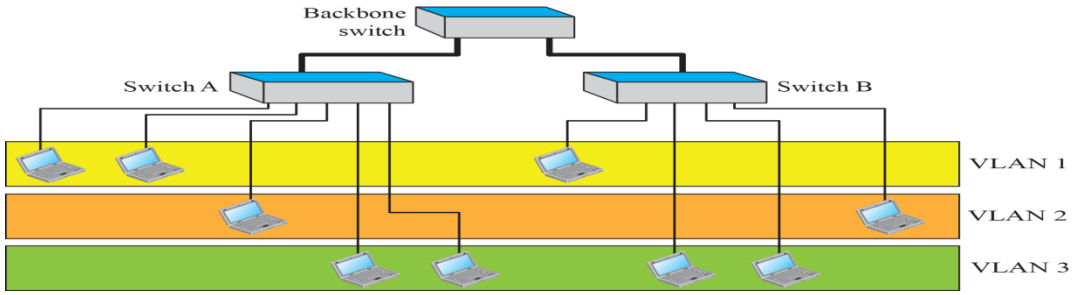


Figure 17.12: Two switches in a backbone using VLAN software

1.Membership

What characteristic can be used to group stations in a VLAN? Vendors use different characteristics such as interface numbers, port numbers, MAC addresses, IP addresses, IP multicast addresses, or a combination of two or more of these.

Interface Numbers

Some VLAN vendors use switch interface numbers as a membership characteristic. For example, the administrator can define that stations connecting to ports 1, 2, 3, and 7 belong to VLAN 1, stations connecting to ports 4, 10, and 12 belong to VLAN 2, and so on.

MAC Addresses

Some VLAN vendors use the 48-bit MAC address as a membership characteristic. For example, the administrator can stipulate that stations having MAC addresses E2:13:42:A1:23:34 and F2:A1:23:BC:D3:41 belong to VLAN 1.

IP Addresses

Some VLAN vendors use the 32-bit IP address (see Chapter 18) as a membership characteristic.

For example, the administrator can stipulate that stations having IP addresses 181.34.23.67, 181.34.23.72, 181.34.23.98, and 181.34.23.112 belong to VLAN 1.

Multicast IP Addresses

Some VLAN vendors use the multicast IP address (see Chapter 21) as a membership characteristic. Multicasting at the IP layer is now translated to multicasting at the datalink layer.

Combination

Recently, the software available from some vendors allows all these characteristics to be combined. The administrator can choose one or more characteristics when installing the software. In addition, the software can be reconfigured to change the settings.

2.Configuration

How are the stations grouped into different VLANs? Stations are configured in one of three ways: manually, semiautomatically, and automatically.

Manual Configuration

In a manual configuration, the network administrator uses the VLAN software to manually assign the stations into different VLANs at setup. Later migration from one VLAN to another is also done manually. Note that this is not a physical configuration; it is a logical configuration. The term *manually* here means that the administrator types the port numbers, the IP addresses, or other characteristics, using the VLAN software.

Automatic Configuration

In an automatic configuration, the stations are automatically connected or disconnected from a VLAN using criteria defined by the administrator.

For example, the administrator can define the project number as the criterion for being a member of a group. When a user changes projects, he or she automatically migrates to a new VLAN.

Semiautomatic Configuration

A semiautomatic configuration is somewhere between a manual configuration and an automatic configuration. Usually, the initializing is done manually, with migrations done automatically.

3.Communication between Switches

In a multi-switched backbone, each switch must know not only which station belongs to which VLAN, but also the membership of stations connected to other switches.

For example, in Figure 17.12, switch A must know the membership status of stations connected to switch B, and switch B must know the same about switch A. Three methods have been devised for this purpose: table maintenance, frame tagging, and time-division multiplexing.

Table Maintenance

In this method, when a station sends a broadcast frame to its group members, the switch creates an entry in a table and records station membership. The switches send their tables to one another periodically for updating.

Frame Tagging

In this method, when a frame is traveling between switches, an extra header is added to the MAC frame to define the destination VLAN. The frame tag is used by the receiving switches to determine the VLANs to be receiving the broadcast message.

Time-Division Multiplexing (TDM)

In this method, the connection (trunk) between switches is divided into time-shared channels (see TDM in Chapter 6). For example, if the total number of VLANs in a backbone is five, each trunk is divided into five channels. The traffic destined for VLAN 1 travels in channel 1, the traffic destined for VLAN 2 travels in channel 2, and so on. The receiving switch determines the destination VLAN by checking the channel from which the frame arrived.

IEEE Standard

In 1996, the IEEE 802.1 subcommittee passed a standard called 802.1Q that defines the format for frame tagging. The standard also defines the format to be used in multi-switched backbones and enables the use of multivendor equipment in VLANs. IEEE 802.1Q has opened the way for further standardization in other issues related to VLANs. Most vendors have already accepted the standard.

4.Advantages

There are several advantages to using VLANs.

Cost and Time Reduction

VLANs can reduce the migration cost of stations going from one group to another. Physical reconfiguration takes time and is costly. Instead of physically moving one station to another segment or even to another switch, it is much easier and quicker to move it by using software.

Creating Virtual Work Groups

VLANs can be used to create virtual work groups. For example, in a campus environment, professors working on the same project can send broadcast messages to one another without the necessity of belonging to the same department. This can reduce traffic if the multicasting capability of IP was previously used.

Security

VLANs provide an extra measure of security. People belonging to the same group can send broadcast messages with the guaranteed assurance that users in other groups will not receive these messages.

UNIT III

The network layer: Network layer design issues,Routing algorithms,Congestion control algorithms,Quality of service,Internetworking.

The network layer in the internet: IPV4 Addresses, IPV6, Internet control protocol,OSPF, BGP,IP, ICMPV4,IGMP

INTRODUCTION

Functions of network layer:

- The basic function of network layer is to provide end to end communications capability to the transport layer which lies above it. Network layer is the lowest layer that deals with end to end communication.
- To achieve the goal, the network layer must know about the topology of the communication subnet ie.., set all routers and choose appropriate path through it. Network layer also take care of loading of the chosen root.
- The network layer protocols are concerned with the exchange of packets of information between transport layer entities
- A packet is a group of bits that includes data plus source and destination address
- The service provided by the network layer to the transport layer is called network service
- The functions carried out by a layer are different from its services. Functions are those activities which are carried out by a layer in order to provide services.
- The network layer functions are carried out by adding a header to every network service data unit (NSDU) forming network protocol data unit(NPDU). The header contains all the information necessary for carrying out functions
    1. It keeps track which MAC address to send.
    2. It makes routing of data through network from source to destination.
    3. Virtual circuits are established in this layer.
    4. It translates logical network address in to physical machine address.
    5. It breaks large packets into smaller so that it will be accepted by the frame of data link layer.
    6. Flow control of packet information of congestion avoidance is concern of protocol.
    7. It determines the quality of service (QOS) parameter.

Network Layer Design Issues:

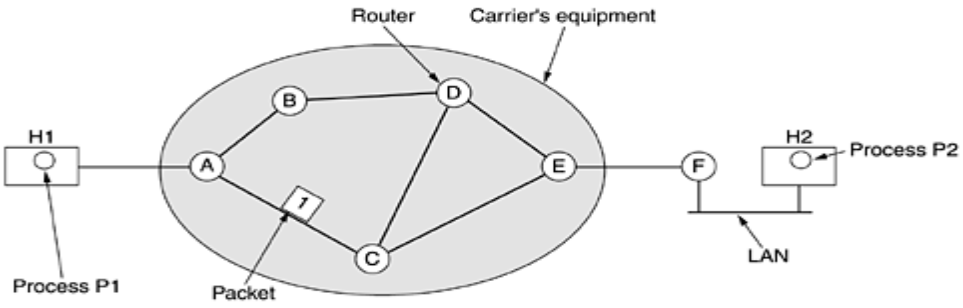
Some of the issues that the designers of the network layer must develop. These issues include the service provided to the transport layer.

- ❖ Store-and-Forward Packet Switching
- ❖ Services Provided to the Transport Layer
- ❖ Implementation of Connectionless Service
- ❖ Implementation of Connection-Oriented Service

Store-and-Forward Packet Switching

The major components of the network are the ISP'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.
- *Host 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.
- 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
- The above process or mechanism is called **store and forward packet switching**.



The environment of the network layer protocols.

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.

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

The network layer should provide connection oriented service or connectionless service.

Connection oriented service:

The subnet is unreliable and host should do the error control and flow control themselves.

Connectionless service:

Network service should be connectionless;these can be accomplished with primitives **SEND PACKET and RECEIVE PACKET.** The service is connectionless, and hence there is no packet ordering and flow control. The overhead of this type of service is that the packet must contain full destination address. Full destination address is required because each packet sent is carried independently of its predecessor packets.

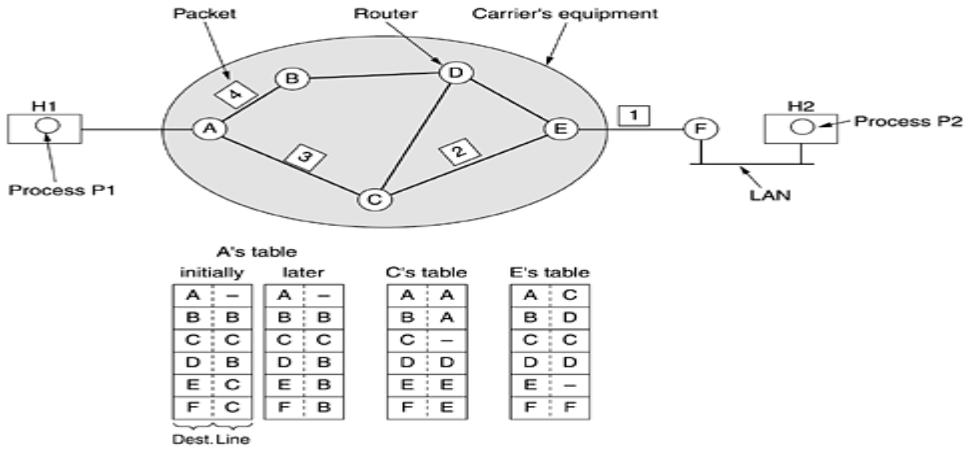
IfSubnet should provide **connection oriented service**, which is reliable, supports real time and provides high quality of service. The packets sent at transmitter will reach the destination in the same order.

Implementation of Connectionless Service:

A connectionless network service is also known as **datagram's**. The implementation of **connectionless service** is done by injecting the packets into the subnet individually and routed independently. In this case there is no advance set up is required. In this case, the packets are frequently called **datagram's** (in analogy with telegrams) and the subnet is called a **datagram subnet**. The packets sent from one end will travel in different order and finally reach destination in different order. Hence the packets grouping and ordering is required at destination end.

In case of **connection oriented service** the path is established between source nodes to the destination node before any data transfer takes place. The connection setup in advance is known virtual circuit and the subnet is called a **virtual circuit subnet**

A message M has to be transmitted from host 1 to host 2 on datagram subnet.



Routing within a datagram subnet.

Step 1: process running on host 1 gives message to transport layer and its adds header to the front of the message and passes into the network layer.

Step 2: 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.**

Step 3: Now the packets are injected on to the subnet independently. Each router has internal routing table telling where to send packets for each possible destination. Each table entry has a pair of consisting of a destination and outgoing line to use for the destination.

Step 4: At each router, all packets are stored for some time and checksum is verified and forwarded to the next router according to the information available in the router table.

Step 5: At A, packets 1, 2, and 3 are stored briefly, having arrived on the incoming link and had their checksums verified. Then each packet is forwarded according to A's table, onto the outgoing link to C within a new frame. Packet 1 is then forwarded to E and then to F. When it gets to F, it is sent within a frame over the LAN to H2. Packets 2 and 3 follow the same route.

Step 6:something different happens to **packet 4**. When it gets to **A** it is sent to **router B**, even though it is also destined for **F**. For some reason, A decided to send **packet 4** via a different route than that of the first three packets. Perhaps it has learned of a traffic jam somewhere along the **ACE** path and updated its routing table, as shown under the label ''later.''

Step 5: All these packets 1,2,3,4, arrive at the destination host 2 at different order. So, these packets are grouped and ordered. All these packets are grouped into message M and given to the transport layer.

The algorithm that manages the tables and makes the routing decisions is called the **routing algorithm**.

Implementation of Connection-Oriented Service

Connection oriented network is also known as **virtual circuit**. Virtual circuit is similar to telephone system. A route which consists of a logical connection is first established two users. The connection that is established is not a decided path between stations. The path is generally shared by many other virtual connections.

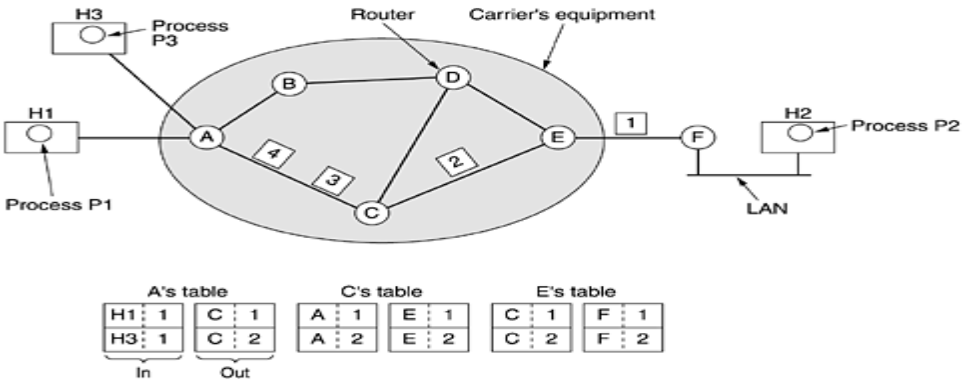
The process is completed in three main phases

- ***Establishment phase***
- ***Data transfer phase***
- ***Connection release phase***

Establishment phase: During setup of logical connections the two users not only agree to setup a connection between them but also decide upon the quality of service associated with the connection.

Data transfer phase: during this phase it performs flow control and error control services. The error control service ensures correct sequencing of packets and correct arrival of packets. The flow control services ensure a slow receiver from being overwhelming with data from faster transmitter.

Connection release phase: when a station wish to close down the virtual circuit, one station can terminate the connection with a clear request packet.



Routing within a virtual-circuit subnet.

Host H1has 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.

Host H3also 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.

Here is the conflict here because although Acan easily distinguish connection 1 packets from H1 from connection 1 packets from H3 but router C cannot do this. For this reason, router 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.

Compare Virtual-Circuit and Datagram Subnets:

Several trade-offs exist between virtual circuits and datagrams.One trade-off is setup time versus address parsing time.

1. Using **virtual circuits**requires a setup phase, which takes time and consumes resources. Once this price is paid, figuring out what to do with a data packet in a virtual-circuitnetwork is easy: the router just uses the circuit number to index into a table tofind out where the packet goes.
2. In a **datagram network**, no setup is needed but amore complicated lookup procedure is required to locate the entry for the destination
3. A related issue is that the destination addresses used in **datagram networks** arelonger than circuit numbers used in **virtual-circuit networks** because they have aglobal meaning.
4. Yet another issue is the amount of table space required in router memory. A**datagram network** needs to have an entry for every possible destination, whereas avirtual-circuit network just needs an entry for each virtual circuit.
5. **Virtual circuits** have some advantages in guaranteeing **quality of service** and avoiding congestion within the network because resources (e.g., buffers, bandwidth, and CPU cycles) can be reserved in advance, when the connection is established.
6. Once the packets start arriving, the necessary bandwidth and router capacity will be there. With a **datagram network, congestion** avoidance is more difficult.
7. Virtual circuits also have a **vulnerability problem**. If a router crashes and loses its memory, even if it comes back up a second later, all the virtual circuits passing through it will have to be aborted.
8. In contrast, if a **datagram router goes** down, only those users whose packets were queued in the router at the time need suffer.The loss of a communication line is fatal to virtual circuits using it, but can easily be compensated for if datagrams are used. Datagrams also allow the routers to balance the traffic throughout the network, since routes can be changed partway through a long sequence of packet transmissions.

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

Routing Algorithms:

- A router is an **internetworking device** that connects two or more networks and forwards packets from one network to another. It is used to manage the traffic on the network and find the best route for packets to be sent. Routers use routing algorithms to find the best route to destination.
- The main function of the network layer is routing packets from the source machine to thedestination machine during routing; there may be intermediate hops present in between source and destination machine. Suitable routing algorithms are required to transfer the packets on the network.
- The **routing algorithm** is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on. If the subnet uses datagram's internally, this design must be made anew for every arriving data packet.
- 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.This is sometimes called **session routing** because a route remains in force for an entire user session.

The router has two functions. They are

- ***Routing***
- ***Forwarding***

In the **routing** function, an algorithm finds the optimal path, and in **forwarding** function, each router forwards the packet. The process is responsible for filling and updating the tables. A packet in the subnet takes several paths.

Properties of routing algorithm:

Certain properties are desirable in a routing algorithm are correctness, simplicity, robustness, stability, fairnessand optimality, efficiency

Correctness: It could able to deliver packets from source to destination without failure or without error for other nodes.

Simplicity: The function should be simple in operation.

Robustness: means the routing algorithm 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 to be rebooted every time some router crashes.

Stability: The routing function should react to contingencies slowly that are neither fast nor too slow. Routing algorithms that never converge to equilibrium, no matter how long they run. A stable algorithm reaches equilibrium and stays there

Fairness and optimality: some performance criteria may give higher priority to the exchange of packets between neighbor stations compared to an exchange distant station. This policy may maximize average throughput but will appear unfair to the station that primarily needs to communicate with distant stations.

Efficiency: The efficiency routing function involves the processing overhead at each node and often a transmission overhead.

Classification of routing algorithms:

Two major classes which come under the routing algorithms are Non-adaptive and Adaptive algorithms

Non-adaptive algorithms: Non-adaptive algorithms are **static and offline**. Their decision does not depend upon the prevailing conditions of traffic and topology of the network. Route for each possible network is computed in advance and are loaded into the router tables when a route is up. The disadvantage of static routing is its inability to respond quickly to network failure.

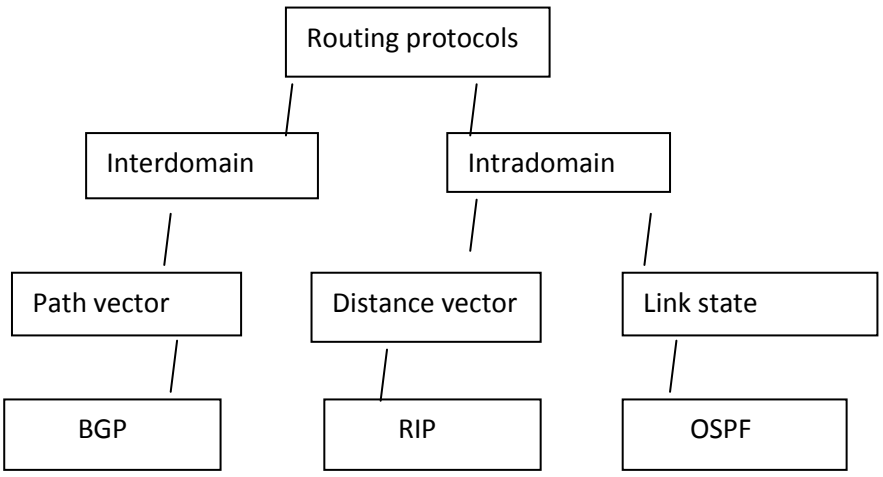
Example: *shortest path and flooding*.

Adaptive algorithms: adaptive algorithms on the contrary are dynamic and online. They collect the information about the state of the network and make routing decisions based on the latest information. The disadvantage iof dynamic routing is its complexity in the router.

Example:*distant vector routing and link state routing*.

Intra and interdomain routing:

- An internet is divided into autonomous systems. An autonomous system is a group of group of networks and routers under authority of a single administration.
- Routing inside an autonomous system is referred to as intradomain routing.
- Routing between autonomous systems is referred to as interdomain routing.
- *Distance vector and link state routing* is the example of **intradomain routing protocols**.
- *Path vector* is an example of **interdomain protocol.**



Border Gateway Protocol (BGP) is a standardized exterior gateway protocol designed to exchange routing and reachability information between autonomous systems (AS) on the Internet.[1] The protocol is often classified as a path vector protocol, but is sometimes also classed as a distance-vector routing protocol. The Border Gateway Protocol does not use Interior Gateway Protocol(**IGP**) metrics, but makes routing decisions based on paths, network policies and/or rule-sets configured by a network administrator. The Border Gateway Protocol plays a key role in the overall operation of the Internet and is involved in making core routing decisions.

BGP may be used for routing within an AS. In this application it is referred to as Interior Border Gateway Protocol, Internal BGP or iBGP. In contrast, the **Internet** application of the protocol may be referred to as Exterior Border Gateway Protocol, External BGP or EBGP.

BGP is the successor to the Exterior Gateway Protocol. BGP is currently the most widely used exterior gateway protocol by Internet service providers. BGP was originally designed to help transition from the core Arpanet model to a decentralized system that included the NSFNET backbone and its associated regional networks.

The **Routing Information Protocol (RIP)** is one of the oldest distance-vector routing protocols, which employs the hop count as arouting metric. RIP prevents routing loops by implementing a limit on the number of hops allowed in a path from the source to a destination. The maximum number of hops allowed for RIP is 15. This hop limit, however, also limits the size of networks that RIP can support. A **hop count** of 16 is considered an infinite distance, in other words the route is considered unreachable. RIP implements the split horizon, route poisoning and holddown mechanisms to prevent incorrect routing information from being propagated.

Originally, each RIP router transmitted full updates every 30 seconds. In the early deployments, routing tables were small enough that the traffic was not significant. As networks grew in size, however, it became evident there could be a massive traffic burst every 30 seconds, even if the routers had been initialized at random times. It was thought, as a result of random initialization, the routing updates would spread out in time, but this was not true in practice. Sally Floyd and Van Jacobson showed in 1994 that, without slight randomization of the update timer, the timers synchronized over time. In most current networking environments, RIP is not the preferred choice for routing as its time to converge and scalability are poor compared to EIGRP, OSPF, or IS-IS (the latter two beinglink-state routing protocols), and a hop limit severely limits the size of network it can be used in. However, it is easy to **configure**, because RIP does not require any parameters on a router unlike other protocols.

RIP uses the User Datagram Protocol (UDP) as its transport protocol, and is assigned the reserved port number **520**

Open Shortest Path First (OSPF) is a routing protocol for Internet Protocol (IP) networks. It uses a link state routing algorithm and falls into the group of interior routing protocols, operating within a **single** autonomous system (AS). It is defined as OSPF Version 2 in RFC 2328 (1998) for IPv4.[1] The updates for IPv6 are specified as OSPF Version 3 in RFC 5340 (2008).[2]

OSPF is perhaps the most widely used interior gateway protocol (IGP) in large enterprise networks. IS-IS, another link-state dynamic routing protocol, is **more** common in large service provider networks. The most widely used exterior gateway protocol is the Border Gateway Protocol (BGP), the **principal** routing protocol between autonomous systems on the **Internet.**

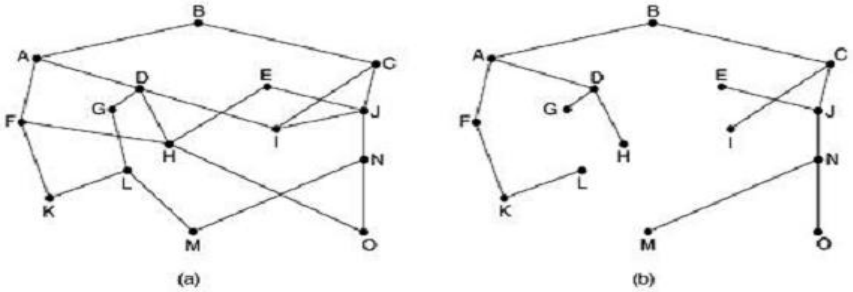
Comparison between intra-domain and inter-domain routing:

S.no	Intra domain routing	Inter domain routing
1	Routing with and AS	Routing between AS
2	Ignore the internet outside the AS	Assumes that the internet consists of a collection of interconnected AS's
3	Protocols for intra domain routing are also called interior gateway protocols	Protocols for inter domain routing are also called Exterior gateway protocols
4	Popular protocols are RIP and OSPF	Routing protocols BGP

1. TheOptimality Principle:

- In the network, if we consider any two nodes there exist multiple routers. But optimal route is one, which has the shortest distance between any two nodes, without regard to network topology or traffic.
- Some Specific algorithms may be helpful to note that one can make a general statement about optimal routes without regard to network topology or traffic. This statement is known as the optimality principle.
- The optimality principle states that, "if router J is one path from router 1 to router k, then the optimal path from J to K also falls along the same route."
- If a route better than r2 existed from J to K, it could be concatenated with r1 to improve the route from I to K, contradicting our statement that r1r2 is optimal.
- The set of optimal routes from all sources to a given destination form a tree rooted at the destination. Such a tree is called a sink tree.The distance metric is the number of hops. A sink tree is not necessarily unique; other trees with the same path lengths may exist.

- Sink tree does not contain any loops, so each packet will be delivered within a finite and bounded number of hopsduring the operation, Links and routers can go down and come back. So, different routers will have different nodes about the current topology.



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

Shortest path routing algorithm:

- Shortest path routing is based on subnet of the network. It is used to build a graph of the subnet, with each node of graph representing a router and each arc of the graph representing a communication line.
- To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph. Different ways of measuring the path length is the number of Hops, Geographical distance in kmts, Mean Queuing delay, Transmission delay, Functions of distance, Bandwidth, Average traffic, communication cost etc.,
- By exchanging the weighing function, the algorithm then computes the shortest path measured according to any one of a number of criteria.

Two algorithms for computing the shortest path between two nodes of a graph are known

1) ***Dijkstra's algorithm***
2) ***Bellman – ford algorithm***

Dijkstra's algorithm

Several algorithms for computing the shortest path between two nodes of a graph are known. This one is due to Dijkstra's (1959). Each node is labeled (in parentheses) with its distance from the source node along the best known path. Initially, no paths are known, so all nodes are labeled with infinity. As the algorithm proceeds and paths are found, the labels may change,reflecting better paths.

Stepwise proceeding of algorithm as follows:

Step 1: Each node is labeled with its distance from the source node along the best known path.

Step 2: Initially assume that no paths are found and all nodes can be labeled with infinity.

Step 3: As the algorithm proceeds and paths are found, the labels may change, reflecting better paths.

Step 4: All labels are initially tentative. When it is known that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter.

Problem 1:

Find the shortest path between node A and node H for the following graph by applying Dijkstra's algorithm. Show at each step.

Problem 2:

Find the shortest path between node A and node D for the following graph by applying Dijkstra's algorithm. Show at each step.

Bellman-Ford algorithm:

Bellman-Ford algorithm is similar to Dijkstra's algorithm but here the shortest path from a given source node is computed subject to the constraint that the path contain at most one link, i.e.., at each step least cost path with maximum number of links are found. Finally the least cost path to each node and the cost of that path is computed.

Problem:

Find the shortest path between node A and node H for the following graph by applying Dijkstra's algorithm. Show at each step.

3.Flooding:

Flooding is a static routing algorithm, in which every incoming packet is sent out on every outgoing line except the one it arrived on. The drawback of flooding generates vast numbers of duplicate packets. These duplicate packets are infinite and have to be controlled.

Various measures are available to control the generation of duplicate packet

1) Using hop counter in the header of each packet

To have a hop counter contained in the header of each packet, which is decremented ateach hop, with the packet being discarded when the counter reaches 'zero'. Initially, thehop counter should be initialized to the length of the path from source to destination.

2) Avoid sending packets for second time by using sequence numbers

To keep track of which packets have been flooded so that they can be avoided sendingsecond time. This is achieved by having the source router put a sequence number in eachpacket it arrives from its hosts. Each router then needs a list per source router tellingwhich sequence numbers originating at that source have already been seen. If anincoming packet is on the list, it is not flooded.

3) Selective Flooding:

The algorithm in which the routers do not send every incoming packet out on every line but only on those lines that are going approximately in the right direction.

The main advantage of flooding is the total traffic load that it generates, is directly proportional to the connectivity of the network.

Flooding requires much bandwidth. The advantage of flooding is that it is highly robust.

Uses of Selective Flooding:

1. In military, where large number of routers receive packets at any instant.

2. In distributed database applications, it is sometimes necessary to update all the databases concurrently.

3. Flooding is used in wireless networks, all messages transmitted by a station can be receive by all other stations with in its radio range.

4. Distance vector routing:

- Distance vector routing algorithm is the dynamic routing algorithm. It was designed mainly for small network topologies.This algorithm is also called as **BELLMAN- FORD** or **FORD-FULKERSON algorithm.**
- Each router maintains a table (i.e, a vector) giving the best known distance to each destination and which line to reach destination. These tables are updated by exchanging information with the neighbors periodically for updation. The three key principles to understand how this algorithm works are as follows:

  Knowledge about the whole network:

Each router shares the table with the entire network. It sends all of its collected knowledge about the network to its neighbors.

Routing only to neighbors:

Each router in the network periodically sends its knowledge about the network only to which it has direct link. This information is received and kept by each neighboring router and used routers own information about the network.

Information is shared periodically:

Sharing of its table information with neighbors is done periodically. For example for every 20 seconds each router sends its information about the whole network to its neighbors.

In this algorithm, each router maintains a routing table indexed by and containing one entry for each router in subnet. This entry contains 2 parts:

- The preferred outgoing line to use for that destination
- An estimate of time or distance (no of hops, or time delay or queue length) for that destination

- This algorithm operates by having each router maintain a table giving the best known distance to each destination and which line to use to get there. These tables are updated by exchanging information with the neighbors.
- Itwas the original ARPANET routing algorithm and was also used in the Internet under the nameRIP.
- The metric used might be **number of hops, time delay in milliseconds, total number of packets queued along the path, or something similar.**
- Assume that delay is used as a metric and that the router knows the delay to each of its neighbors. All nodes exchange information only with their neighboring nodes. Nodes participating in the same local network are considered neighboring nodes.
- If the metric is hops, the distance is just one hop. If the metric is queue length, the router simply examines each queue. If the metric is delay, the router can measure it directly with special ECHO packets that the receiver and measure the distance using timestamps.
- 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.
- 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.

A new routing table is computed as explained below:

1) Let us consider router H to be the node,which receives delay vectors from neighbors A, G, I, F. the delay vector and new table is shown in below fig.
2) The delay obtained from the neighbors are HA is 6, HI is 11, and HG is 8. HA is 6 obtained from neighbor router A. HI is obtained from neighbor router I. HF is obtained from neighbor router F. HG is obtained from neighbor router G.
3) From H to A, I, G and F delays are entered into new estimated table for H.

4) Now from H to B, H has to use out going line A and the delay from H to B can be calculated as HA + AB. HA is 6 and AB is 12. The estimated delay is 18. Similarly, from H to C.

5) Router H calculates delay via neighbors A,I, G and F as follows,

H – A – C via A	HA + AC =	(6+21) = 27
H – I – C via I	HI + IC =	(11+18) = 29
H – G – C viaG	HG + GC =	(8+21) = 29
H – F – C viaF	HF + FC =	(9+30) =39

The route H – A – C is the best route as it has minimum delay of 27. The new entry is made into the router table along with outgoing line F. similarly delay D,E can be calculated.

6) For D via A,I,G and F is

HA + AD = 6 + 25 = 31

HI + ID = 11 + 21 = 32

HG + GD = 8 + 24 = 32

HF + FD = 9 + 19 = 28

THE BEST ROUTE IS H – F- D AND DELAY IS 28

7) For E via A,I,G and F is

HA + AE = 6 + 41 = 47

HI + IE = 11 + 7 = 18

HG + GE = 8 + 23 = 31

HF + FE = 9 + 6 = 15

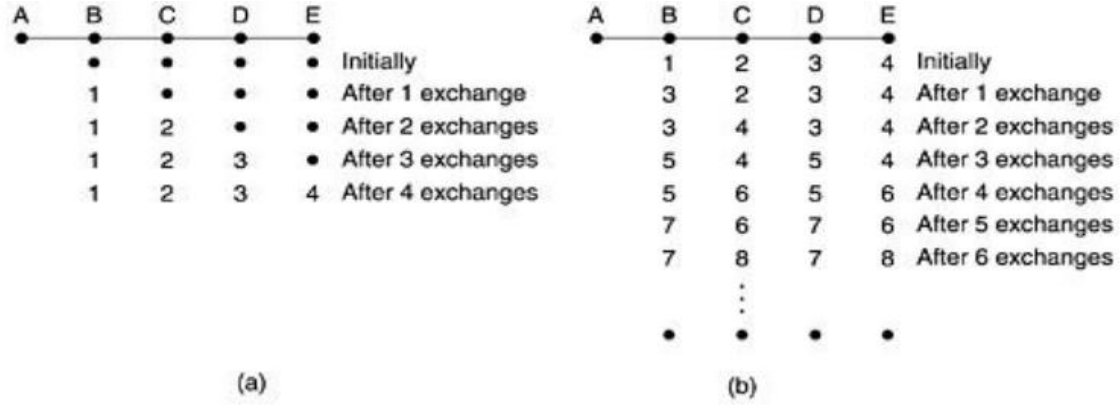
THE BEST ROUTE IS H – F – E AND DELAY IS 15

The Count-to-Infinity Problem

Distance vector routing has a serious drawback of converging slowly in large network. The important information failure is intimated very slowly. Thus it reacts rapidly to good news and slow to bad news.

The good news is processed for one vector exchange. Consider a simple subnet to study the progress of distance vector algorithm whose delay metric I the number of hops.Suppose A is down

initially and all the other routers know this and recorded the delay to A as infinity.



The count-to-infinity problem.

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 except B think that A is down, because the information is not exchanged with them. On the next exchange, it is clear that B has path of 1 length to A and updates its routing table to indicate a path of length 2. But D and C are still not aware of A.

So, 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.

Routers B, C, D, and E have distances to A of 1, 2, 3, and 4, respectively. Suddenly A goesdown, or alternatively, the line between A and B is cut. At the first packet exchange, B does not hear anything from A.but B can reach via C with a path length of 2. D does not update it entries for A on the first exchange.

On the second exchange, C notices that each of its neighbors claims to have a path to Aof length 3.now D makes a new distance to A as 4. This exchange produces different path lengths. This problem is known as counter to infinity problem.

5. Link State Routing:

Link state routing is the second major class of intra domain routing protocol. It is dynamic type routing algorithm. The link state routing algorithm was developed to overcome the drawback in distance vector routing. Two primary problems in dynamic type routing algorithm is

First, if the delay metric was queue length, the line bandwidth into account when choosing the routes.

Second, The problem of count to infinity, which takes long time to converge.The idea behind link state routing is simple and can be stated as five parts. In link state routing, Each router must do the following:

a) ***Learning about the neighbors***
b) ***Measuring the line cost***
c) ***Building link state packets***
d) ***Distributing the link state packets***
e) ***Computing the new routes***

a) Learning about the neighbors:

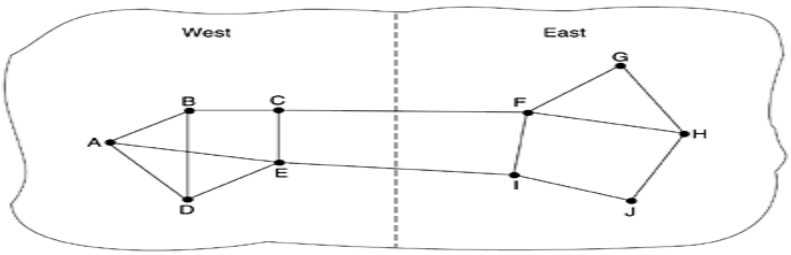
Learning about neighbors takes place when a router is booted. First, router tries to learn or know about neighbors. Learning of neighbors can be achieved by the router by sending HELLO packet on each point-to-point line. Neighbors will receive HELLO packets and reply by giving their network address.

These network addresses must be globally unique. When two or more routers are connected by a LAN, the situation is more complicated.

The router A, D, and G are directly connected to LAN. Further each of these routers are connected to one or more additional routers. The graph model may be constructed by introducing new artificial node T to which A,D and G are connected. Here, the LAN is considered as node itself. The possibility of going from A to D on the LAN is trough the path A-T-D

b) Measuring the line cost

To determine the cost for a line, a router sends a special ECHO packet over the line 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.



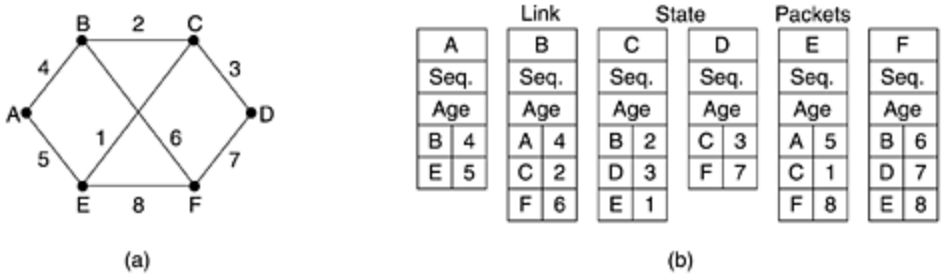
A subnet in which the East and West parts are connected by two lines.

c) Building Link State Packets

Building up of packets will be started as the information is collected. Information may be network address and delay. Each link state packet will have the following format as shown in the below

Source address	Sequence number	Packet life or age	Neighbor routers and delay

An example of building link state packets is shown in below



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

The link state packets are easy to build. These packets are built periodically at regular intervals. Even in case when a line or neighbor goes down or comes up, link state packets are built.

d) Distributing the Link State Packets

The link state packets must be distributed to all the neighbors' reliably. This distribution of packets is done, when router changes their routes. The different routers may be using different versions of the topology, which can lead to inconsistencies, loops, unreachable machines, and other related problems.

The distribution of link state is done by flooding. To avoid duplication or redundancy packets, sequence numbers are used. Thus, link state packet format has a filed for holding sequence number. Each packet contains a sequence number that is incremented for each new packet sent. All Routers will keep track of source router, sequence number.

When a new link state packet comes in, it is checked against the list of packets available in the entry of routing table. If the link state packet is new then it is forwarded to all lines except the one It arrived on (flooding).

If the link state packet is already available in entry of routing table, then it is discarded (duplicate). There is other way of finding duplicate packets ie.., with the sequence number. If the sequence number is lower than the highest one already stored, then packet is discarded as duplicate packet.

This algorithm has a few problems, but they are manageable.

First, The limitation in the length of sequence number. If the number of bits used for representing sequence numbers are less, it leads confusion. The solution to this problem is to use a 32 bit sequence number.

Second, 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.

Third, if a sequence number is ever corrupted and 65,540 is received instead of 4 (a 1-bit error), then remaining packets from 5 through 65,540 will be rejected as duplicate packets, since the current sequence number is thought to be 65,540.

e) Computing the New Routes

All the link state packets are accumulated at the router. Now, router can construct the entire subnet graph because every link is represented.

Apply dijkstra's algorithm to compute shortest path to all possible destinations. This algorithm can be run locally. For the subnet with n routers, each of which has K neighbors, the memory required to store input data in Kn. For large subnet, memory occupied will be more and also computation time can be an issue. Even in this situation, link state routing works well.

6. Hierarchical routing

Maintaining information about all the neighbor routers leads to many problems whennetwork size grows in size. As the network grows in size, the router routing table grows proportionally. Router consumes more memory space and also CPU time is required to search or scan the routing tale. More network bandwidth is wasted to distribute routing messages among neighbors. To overcome all these problems in a very large network, routing will have to be done hierarchically

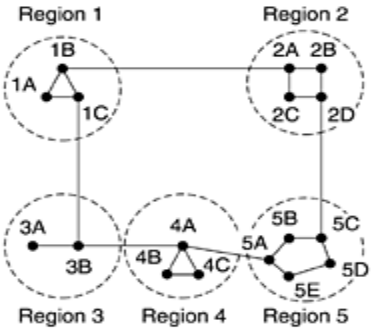
- In **hierarchical region**, the routers are divided into regions, with each router knowing all the details about how to route packets to destinations.
- Router in a region will have the details of its own region, but knowing nothing about the internal structure of other regions.
- In large networks where different networks are interconnected, each of the regions are considered separately in order to free the routers in one region from having to know the topological structure of the other ones.

- For huge networks, a two-level hierarchy may be insufficient; it may be necessary to group the regions into clusters, the clusters into zones.

  Different levels are used to compute the routes:

  - ***Level 1 - Routers***
  - ***Level 2 - Regions***
  - ***Level 3 - Clusters***
  - ***Level 4 - Zones***
  - ***Level 5 - Zonal regions***

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. 5-15(b). When routing is done hierarchically, as in Fig. 5-15(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. The advantage is that as 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

(b)

Hierarchical table for 1A

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

(c)

Figure 5-15 .Hierarchical routing.

The disadvantage is, increased path length. For example, the best route from 1*A* to 5*C* 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 increases load on line 1A – 5C.

7. Broadcast Routing

There are different types of routing mechanisms for different types of application. In some applications, hosts need to send messages to many or all other hosts. For example, a service

distributing weather reports, stock market updates etc. in broadcasting a packet is sent to all destinations in a network.

There are various methods proposed for broadcasting technique,

- ***Direct method***
- ***Flooding***
- ***Multidestination routing***
- ***Spanning tree***
- ***Reverse path forwarding***

Direct method:

Broadcasting method that requires no special features. Here source node simply sends a distinct packet to each destination. In this method, bandwidth is wasted and source is also required to maintain the list of all destinations.

Flooding:

The main problem in flooding is generation of too many duplicate packets and it also consumes more network bandwidth.

Multi destination routing:

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.

The router generates a new copy of the packet for each outgoing line. After a sufficient number of hops, each packet will carry only one destination and can be treated as a normal packet.

Spanning tree:

In this method either sink tree or spanning tree is used for broadcast. A **spanning tree** is a subset of the subnet that includes all the routers but contains no loops. Each router is provided with line belonging to the spanning tree.

When a packet is to be broadcasted, a copy of this packet is sent to all the spanning tree lines except the one it arrived on. The advantage of using spanning tree or sink tree is usage of bandwidth efficiently. The disadvantage of this method is to have knowledge of some spanning tree for the method to be applicable.

Reverse path forwarding:

In reverse path forwarding, even when the routers do not know anything at all about spanning trees, the packets are broadcasted.

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.

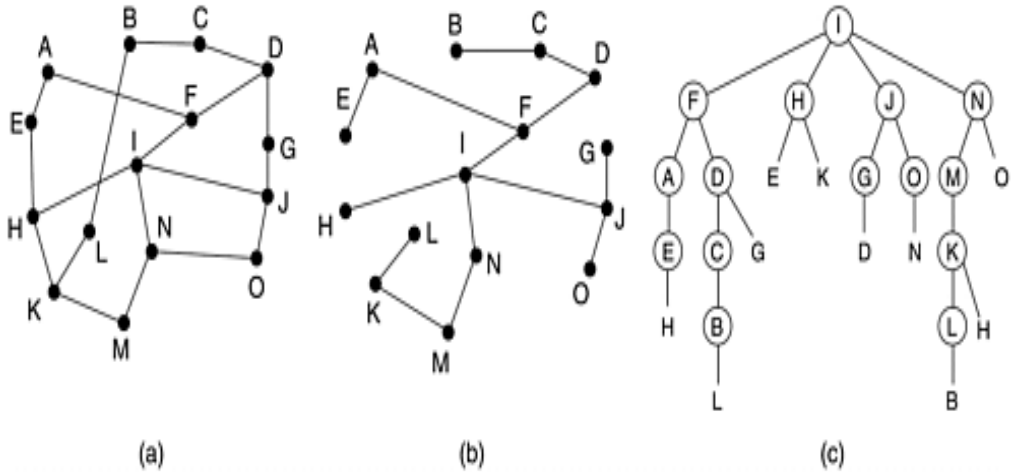


Fig 5-16 Reverse path forwarding. (a) A subnet. (b) A sink tree. (c) The tree built by reverse path forwarding

Example of reverse path forwarding is shown in Fig. 5-16. Part (a) shows a subnet, part (b) shows a sink tree for router I of that subnet, and part (c) shows how the reverse path algorithm works. On the first hop, I send packets to F, H, J, and N, as indicated by the second row of the tree. Each of these packets arrives on the preferred path to I (assuming that the preferred path falls along the sink tree) and is so indicated by a circle around the letter.

On the second hop, eight packets are generated, two by each of the routers that received a packet on the first hop. As it turns out, all eight of these arrive at previously unvisited routers, and five of these arrive along the preferred line. Of the six packets generated on the third hop, only three arrive on the preferred path (at C, E, and K); the others are duplicates. After five hops and 24 packets, the broadcasting terminates, compared with four hops and 14 packets had the sink tree been followed exactly.

The advantages of reverse path forwarding are:

- ✓ Efficient and easy to implement
- ✓ No need to know about spanning tree or overhead of destination list or bit map
- ✓ No mechanism is required to stop the process of flooding.

8. Multicast Routing

- Some applications, such as a **multiplayer game** or **live video** of a sports eventstreamed to many viewing locations, send packets to multiple receivers. Unlessthe group is very small, sending a distinct packet to each receiver is expensive.
- On the other hand, broadcasting a packet is wasteful if the group consists of, say,1000 machines on a million-node network, so that most receivers are not interestedin the message (or worse yet, they are definitely interested but are not supposedto see it). Thus, we need a way to send messages to well-defined groupsthat are numerically large in size but small compared to the network as a whole.
- Sending a message to such a group is called **multicasting**, and the routing algorithmused is called **multicast routing**. All **multicasting** schemes require someway to create and destroy groups and to identify which routers are members of agroup. How these tasks are accomplished is not of concern to the routing algorithm.
- For now, we will assume that each group is identified by a **multicast address**and that routers know the groups to which they belong.
- **Multicast routing schemes** build on the broadcast routing schemes, sending packets along spanning trees to deliver the packets to themembers of the group while making efficient use of bandwidth. However, thebest spanning tree to use depends on **whether the group isdense, with receiversscattered over most of the network, or sparse, with much of the network not belongingto the group.**
- If the group is dense, broadcast is a good start because it efficiently gets thepacket to allparts of the network. But broadcast will reach some routers that arenot members of the group, which is wasteful. The solution explored by Deeringand Cheriton (1990) is to **prune** the broadcast spanning tree by removing links thatdo not lead to members. **The result is an efficient multicast spanning tree.**
- As an example, consider the two groups, 1 and 2, in the network shown inFig. 5-16(a). Some routers are attached to hosts that belong to one or both ofthese groups, as indicated in the figure.
- A spanning tree for the leftmost router isshown in Fig. 5-16(b). This tree can be used for broadcast but is overkill for multicast,as can be seen from the two pruned versions that are shown next.

- InFig. 5-16(c), all the links that do not lead to hosts that are members of group 1have been removed. The result is the multicast spanning tree for the leftmostrouter to send to group 1. Packets are forwarded only along this spanning tree,which is more efficient than the broadcast tree because there are 7 links instead of10.

- Fig. 5-16(d) shows the multicast spanning tree after pruning for group 2. It isefficient too, with only five links this time. It also shows that different multicastgroups have different spanning trees.

- Various ways of **pruning** the spanning tree are possible. The simplest one canbe used if link state routing is used and each router is aware of the complete topology,including which hosts belong to which groups. Each router can then constructits own pruned spanning tree for each sender to the group in question byconstructing a sink tree for the sender as usual and then removing all links that donot connect group members to the **sink node**. **MOSPF (Multicast OSPF) is anexample of a link state protocol that works in this way (Moy, 1994).**

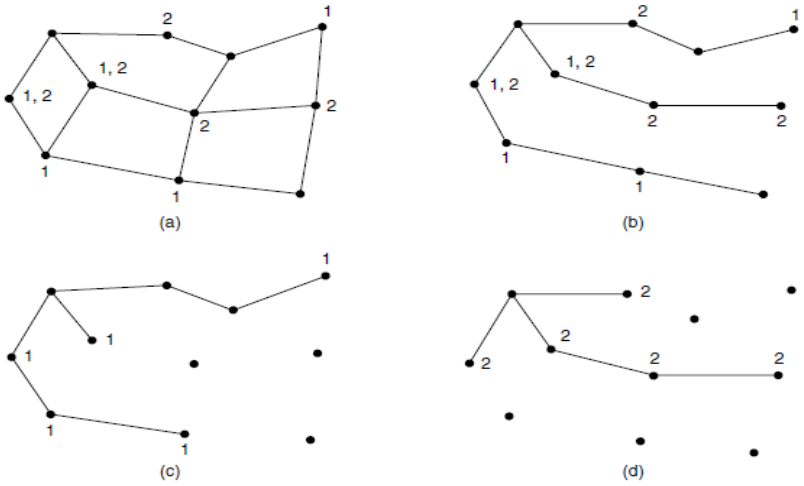


Figure 5-16. (a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.

With **distance vector routing**, a different **pruning** strategy can be followed.The basic algorithm is reverse path forwarding. However, whenever a router withno hosts interested in a particular group and no connections to other routers receivesa multicast message for that group, it responds with a PRUNE message, tellingthe neighbor that sent the message not to send it any more multicasts from thesender for that group. When a router with no group members among its own hostshas received such messages on all the lines to which it sends the multicast, it, too,can respond with a PRUNE message. In this way, the spanning tree is recursivelypruned. **DVMRP (Distance Vector Multicast Routing Protocol) is an exampleof a multicast routing protocol that works this way** (Waitzman et al., 1988).

9. Anycast Routing

There are so many types of delivery models in which a source sends to a singledestination (called unicast), to all destinations (called broadcast), and to a groupof destinations (called multicast). Another delivery model, called anycast issometimes also useful. In anycast, a packet is delivered to the nearest member of a group. Schemes that find these paths are called anycastrouting.

Why would we want anycast? Sometimes nodes provide a service, such as**time of day** or **content distribution** for which it is getting the right information allthat matters, not the node that is contacted; any node will do.

For example, anycastis used in the Internet as part of DNS.

We will not have to devise new routing schemes for anycast becauseregular **distance vector and link state routing** can produce anycast routes. Supposewe want to anycast to the members of group 1. They will all be given the address‘‘1,’’ instead of different addresses. Distance vector routing will distribute vectorsas usual, and nodes will choose the shortest path to destination 1. This will resultin nodes sending to the nearest instance of destination 1. The routes are shown inFig. 5-18(a). This procedure works because the routing protocol does not realizethat there are multiple instances of destination 1. That is, it believes that all theinstances of node 1 are the same node, as in the topology shown in Fig. 5-18(b).

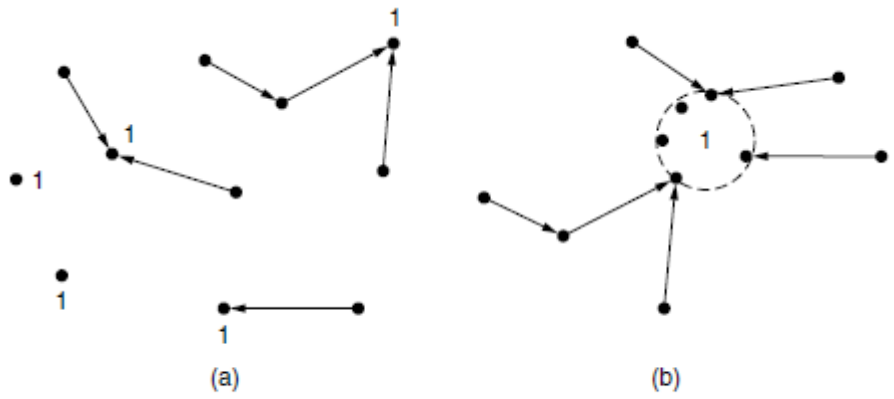


Figure 5-18. (a) Anycast routes to group 1. (b) Topology seen by the routing protocol.

This procedure works for link state routing as well, although there is theadded consideration that the routing protocol must not find seemingly short pathsthat pass through node 1. This would result in jumps through hyperspace, sincethe instances of node 1 are really nodes located in different parts of the network.However, link state protocols already make this distinction between routers andhosts. We glossed over this fact earlier because it was not needed for our discussion.

10. Routing for Mobile Hosts

Millions of people use computers while on the go, from truly mobile situationswith wireless devices in moving cars, to nomadic situations in which laptopcomputers are used in a series of different locations. We will use the term mobilehosts to mean either category, as distinct from stationary hosts that never move.Increasingly, people want to stay connected wherever in the world they may be, aseasily as if they were at home. These mobile hosts introduce a new complication**:to route a packet to a mobile host, the network first has to find it.**

Some discussion of this model is in order. A different model would be tore-compute routes as the mobile host moves and the topology changes. However,with a growing number of mobile hosts, this model would soon lead to the entirenetwork endlessly computing **new routes**. Using the home addresses greatly reducesthis burden.

Another alternative would be to provide mobility above the network layer,which is what typically happens with laptops today. When they are moved to newInternet locations, laptops acquire new network addresses. There is no associationbetween the old and new addresses; the network does not know that they belongedto the same laptop.

In this model, a laptop can be used to browse the Web, butother hosts cannot send packets to it (for example, for an incoming call), withoutbuilding a higher layer location service, for example, signing into **Skype** againafter moving. Moreover, connections cannot be maintained while the host is moving;new connections must be started up instead. Network-layer mobility is usefulto fix these problems.

The basic idea used for mobile routing in the Internet and cellular networks isfor the mobile host to tell a host at the home location where it is now. This host,which acts on behalf of the mobile host, is called **the home agent**. Once it knowswhere the mobile host is currently located, it can forward packets so that they aredelivered.

Fig. 5-19 shows mobile routing in action. A sender in the **northwest city** ofSeattle wants to send a packet to a host normally located across the United Statesin New York. The case of interest to us is when the mobile host is not at home.Instead, it is temporarily in San Diego.

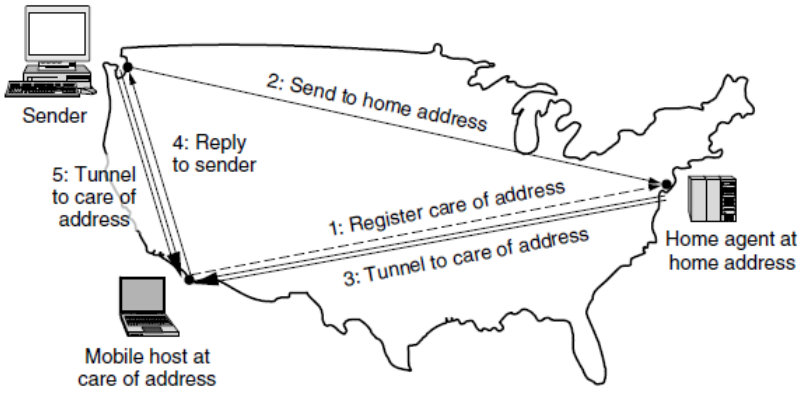


Figure 5-19. Packet routing for mobile hosts.

The mobile host in San Diego must acquire a local network address before itcan use the network. This happens in the normal way that hosts obtain networkaddresses; we will cover how this works for the Internet later in this chapter.

- Thelocal address is called a care of address. Once the mobile host has this address,it can tell its home agent where it is now. It does this by sending a registrationmessage to the home agent

(step 1) with the care of address. The message isshown with a dashed line in Fig. 5-19 to indicate that it is a control message, not adata message.

- Next, the sender sends a data packet to the mobile host using its permanentaddress (step 2). This packet is routed by the network to the host's home locationbecause that is where the home address belongs.
- In New York, the home agentintercepts this packet because the mobile host is away from home. It then wrapsor encapsulates the packet with a new header and sends this bundle to the care ofaddress (step 3). This mechanism is called **tunneling**. It is very important in theInternet so we will look at it in more detail later.
- When the encapsulated packet arrives at the care of address, the mobile hostunwraps it and retrieves the packet from the sender. The mobile host then sendsits reply packet directly to the sender (step 4). **The overall route is called trianglerouting because it may be circuitous if the remote location is far from the homelocation**.
- As part of step 4, the sender may learn the current care of address. Subsequentpackets can be routed directly to the mobile host by tunneling them to thecare of address (step 5), bypassing the home location entirely. If connectivity islost for any reason as the mobile moves, the home address can always be used toreach the mobile.

An important aspect that we have omitted from this description is security. Ingeneral, when a host or router gets a message of the form ''Starting right now,please send all of Stephany's mail to me,'' it might have a couple of questionsabout whom it is talking to and whether this is a good idea. Security informationis included in the messages so that their validity can be checked with cryptographicprotocols.

There are many variations on mobile routing. The scheme above is modeledon IPv6 mobility, the form of mobility used in the Internet and as part of IP-based **cellular networks such as UMTS.**

5.2.11 Routing in Ad Hoc Networks

We have now seen how to do routing when the hosts are mobile but the routersare fixed. An even more extreme case is one in which the routers themselvesare mobile. Among the possibilities are emergency workers at an earthquake site,military vehicles on a battlefield, a fleet of ships at sea, or a gathering of peoplewith laptop computers in an area lacking 802.11.

In all these cases, and others, each node communicates wirelessly and acts asboth a host and a router. Networks of nodes that just happen to be near each otherare called ad hoc networks or MANETs (Mobile Ad hoc NETworks). What makes ad hoc networks different from wired networks is that the topologyis suddenly tossed out the window. Nodes can come and go or appear in newplaces at the drop of a bit.

With a wired network, if a router has a valid path tosome destination, that path continues to be valid barring failures, which are hopefullyrare. With an ad hoc network, the topology may be changing all the time, sothe desirability and even the validity of paths can change spontaneously withoutwarning. Needless to say, these circumstances make routing in ad hoc networksmore challenging than routing in their fixed counterparts.

Many, many routing algorithms for ad hoc networks have been proposed.However, since ad hoc networks have been little used in practice compared tomobile networks, it is unclear which of these protocols are most useful. As an example,we will look at one of the most popular routing algorithms, **AODV (Adhoc On-demand Distance Vector)**. It is a relative ofthe distance vector algorithm that has been adapted to work in a mobile environment,in which nodes often have limited bandwidth and battery lifetimes.

3. Congestion Control Algorithms

When too many packets are present in (a part of) the subnet, performance degrades. **This situation is called congestion. Congestion** in a network occurs if users send data into the subnet at a rate greater than allowed by network resources. Network resources can be **communication links, routers, buffers in switches and soon.**

The networkand transport layers share the responsibility for **handling congestion**. Since congestionoccurs within the network, it is the network layer that directly experiencesit and must ultimately determine what to do with the excess packets. The most effective way to control **congestion is to reduce the load that the transport layer is placing on the network**. This requires the network and transport layersto work together.

Figure 5-21 depicts the onset of congestion. When the number of packetshosts send into the network is well within its **carrying capacity**, the number deliveredis proportional to the number sent. If twice as many are sent, twice as manyare delivered. However, as the offered load approaches the carrying capacity,bursts of traffic occasionally fill up the **buffers inside routers** and **some packets are lost**. These lost packets consume some of the capacity, so the number of deliveredpackets falls below the ideal curve. **The network is now congested**.

Unless the network is well designed, it may experience a congestion collapse,in which performance plummets as the offered load increases beyond the capacity.This can happen because packets can be sufficiently delayed inside the networkthat they are no longer useful when they leave the network.

For example, inthe early Internet, the time a packet spent waiting for a backlog of packets aheadof it to be sent over a slow 56-kbps link could reach the maximum time it was allowedto remain in the network. It then had to be thrown away.

A different failuremode occurs when senders retransmit packets that are greatly delayed, thinkingthat they have been lost. In this case, copies of the same packet will be deliveredby the network, again wasting its capacity. To capture these factors, the y-axis ofFig. 5-21 is given as goodput, which is the rate at which useful packets are deliveredby the network.

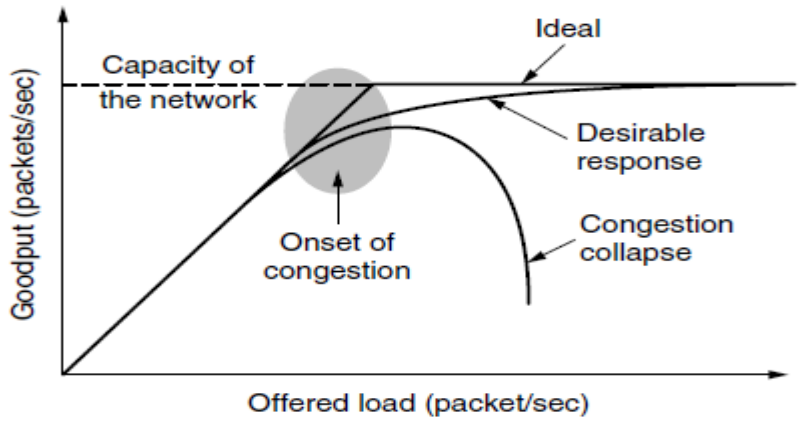


Figure 5-21. With too much traffic, performance drops sharply.

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

Congestion will occur due to many reasons. If the traffic increases, the routers are no longer able to cope and they begin losing packets. At very high traffic, performance degrades completely and delivery of packets will be less.

Reasons for Congestion:

1. If input packets coming from 3 or 4 lines, requires only one particular output line.

2. If routers are supplied with infinite amount of memory, packets take longtime to reachto the front of queue where duplicates are generated as they are timed out.

3. Slow processors cause congestion. When CPU performs tasks very slowly, then congestion may occur. Tasks can be queuing buffers, updating tables etc.., adding high speed processor will solve this problem.

4. Low bandwidth lines also cause congestion. . Upgrading the lines may help to avoid congestion to some extent.

5. Congestion feeds upon itself and cause congestion

1. General Principles of Congestion Control

The complex system such as computer network has various problems. All solutions to these problems are grouped into,

1. Open Loop congestion control (prevention)

2. Closed Loop congestion control (removal).

Open loop congestion controls are applied to prevent congestion before it happens. Open loop solutions attempt to solve the problem by good design. In this mechanism congestion control is handled by source or destination. Once the system is up and running, midcourse corrections are not made. Several tools for open-loop control include deciding when to accept new traffic, deciding when to discard packets.

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.

- Metrics to monitor the subnet for the congestion are: the average queue lengths, percentage of all packets discarded for lack of buffer space, the average packet delay, the standard deviation of packet delay and the number of packets that time out and are retransmitted.
- These closed loop algorithms are further divided into two categories:
  - ✓ **Implicit feedback** and
  - ✓ **Explicit feedback.**
- **Implicit feedback**: The source reduces the congestion existence by making local observations, such as the time needed for acknowledgments to come back.
- **Explicit feedback**: Packets are sent back from the point of congestion to warn the source

2.Congestion Prevention Policies

The congestion prevention policies are discussed at different levels in open loop systems. There are different **data link, network and transport policies** that 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

Policies that affect congestion.

Data link layer is concerned with four policies:

a) **Retransmission policy**: In retransmission policy, when sender waits for acknowledgment from the other end and sender time out has occurred and then once again the packets are to be retransmitted. The retransmission of packets depends on the ARQ protocol used. this leads congestion heavily

b) **Out-of –order Caching policy**: selective repeat is better than go back N in which receivers routinely discard all out-of-order packets, (packets arrived without order) they have to be retransmitted again later and this creates extra load.

c) **Acknowledgement policy**: If each packet is acknowledged immediately, acknowledged packets generate extra traffic. This policy deals with piggybacking.

d) **Flow Control policy**: In flow control mechanism, if the window size is big, then there may be a chance of losing the packets, A tight flow control scheme (ex: a small window) reduces the data rate and thus helps fights congestion.

Network layer adopts the following five policies:

a) **Virtual Circuits vs Data grams**: In network layer, many congestion control algorithms work only with virtual circuit subnet. This affects congestion as many algorithms work only with virtual circuits.

b) **Packets queuing and Service policy:** Relates to whether routers have one queue per input line, one queue per output line or both. It also relates to the order in which packets are processed.

c) **Packet Discard policy**: Packet discard policy, Tells which packet is dropped when there is no space. Good policy will avoid congestion and bad will make congestion worse.

d) **Routing Algorithm:**routing algorithm also play a very important role in avoiding congestion by spreading the traffic uniformly over all the lines.

e) **Packet lifetime management**: packet life time must be specified correctly. Deals with how long a packet may live before being discarded.If the packet life time is more, it may be block up lines for long time. If the packet life time is short, it may be die before reaching destination. Thus this packet needs retransmission, so, in the way packet life time management policies affect management of congestion.

Transport layer adopts the following five policies:

Timeout Determination: It is harder as transit time across the network is less predictable than transit time over a wire between two routers. If the time out interval is too short, extra packet will be unnecessarily sent due to retransmission. If the time out interval is too long, congestion will be reduced, but the response time will suffer whenever packets are lost.

3.Congestion Control in Virtual-Circuit Subnets

There are various approaches to control congestion in virtual circuit subnet. Some of these approaches are,

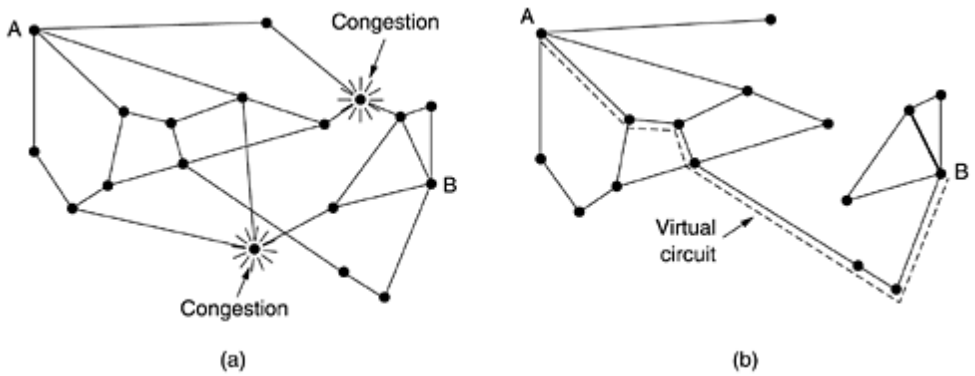
- ***Admission control***
- ***Route all new virtual circuits around new problem areas***
- ***Negotiate host and subnet during virtual circuit setup.***

Admission control:

Admission control is used to control during the new virtual circuit set up. If there is an indication of congestion (heavy traffic) on the subnet, no new virtual circuits are setup until congestion reduces. Thus, attempts to setup new connection at transport layer.

Route all new virtual circuits around new problem areas:

In this approach new connections are allowed but carefully route all new virtual circuit around problem areas .to illustrate this situation, consider the subnet in fig in which two routers are in congestion.



(a) A congested subnet. (b) A redrawn subnet that eliminates the congestion. A virtual circuit from A to B is also shown.

Suppose router A wants to setup a connection to another router B. A new route and subnet is considered by omitting the congested routers and all their lines. The new route is shown in dashed line, avoids the congested routers.

Negotiate host and subnet during virtual circuit setup:

An agreement is to be negotiated between the host and subnet when avirtual circuit is set up. This agreement normally specifies the volume and shape of the traffic,quality of service required and other parameters. To keep its part of agreement, the subnet willtypically reserve resources along the path when the circuit is set up, and thus avoids congestion.

4.Congestion Control in Datagram Subnets:

This is an approach that can be used in both virtual circuit and data gram subnets. Each router can easily monitor the utilization of its output lines and other resources using the formula….

$$U_{new} = au_{old}+(1-a)f$$

Where u = a variable that reflects the recent utilization of a line. Value lies between 0.0 and 1.0

a = constant that determines how fast the router forgets recent history.

F =a sample of instantaneous line utilization (either 0 or 1)

Various approaches are,

- ***The warning bit***
- ***Choke packets***
- ***Hop-by-hop choke packets***

The warning bit

This bit is used to signal the congestion on the subnet. The warning state is indicated by setting a special bit in the packet's header. When the packet arrived at its destination, the transport

entity copies the bit into the next acknowledgement. The acknowledgment is sent back to the source indicating the congestion. So the source then reduces the traffic. This warning is set as long as router is in warning state. As long as warning bits continue to flow in, the source continues to decrease its transmission rate. If warning bit is not set, then transmission rate can be increased.

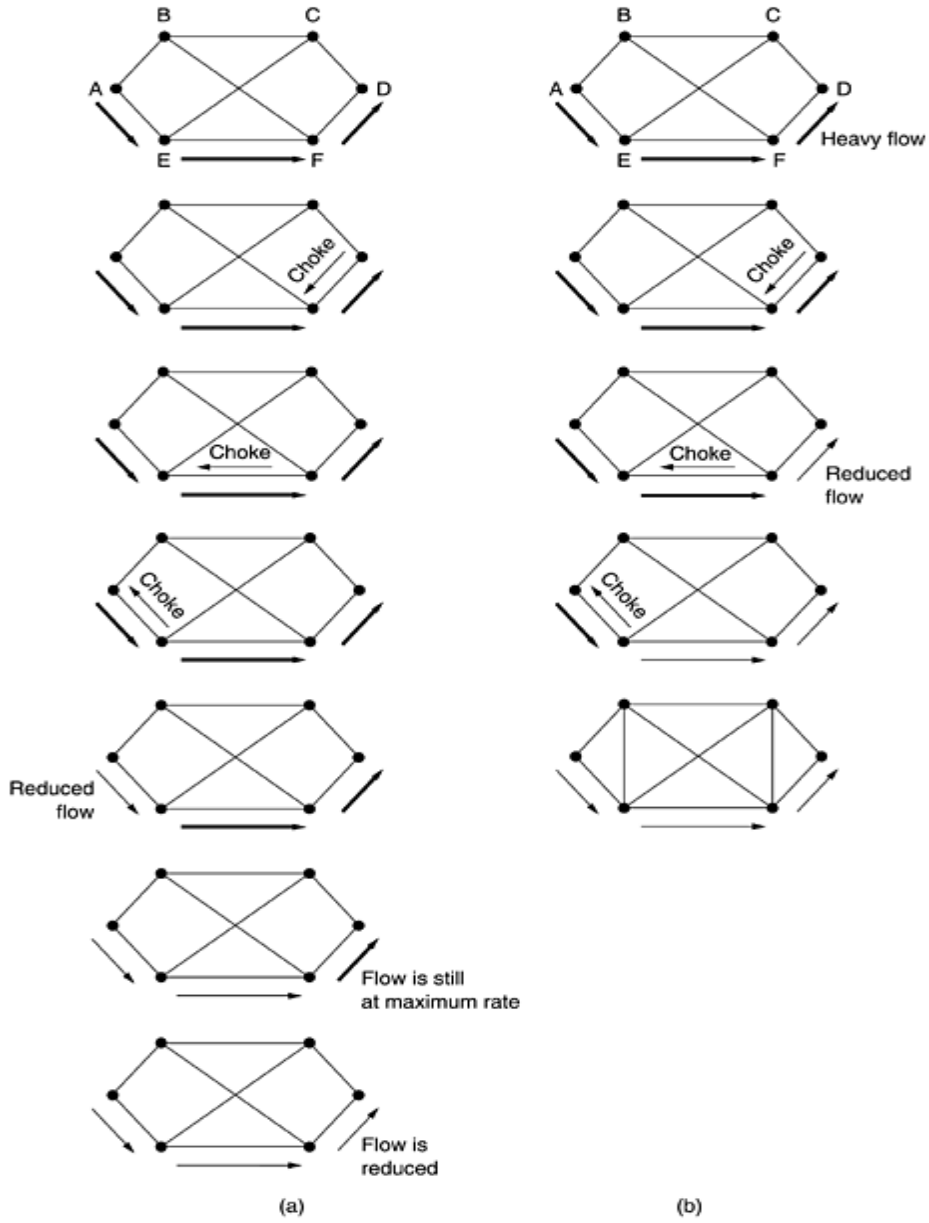
Choke packets

In this mechanism, special choke packets are used to inform about the congestion back to source machine. The router sends a choke packet back to the source, giving information about congestion on that line. If there is no congestion, then original packet is tagged so that it will not generate any more choke packets further along the path and is then forwarded in the usual way.

When the source host gets the choke packet, It is necessary to reduce the traffic sent to the specified destination by some percent. At every fixed interval of time, host will try to listen to choke packet. If choke packets arrives at the time of interval , this indicates line is still congested, so the source machine reduces the flow still more and ignore choke packets for fixed time of interval. If there is no choke packets arrived during the listening period, the host may increase the flow again. Host can reduce traffic by adjusting their policy parameters.

Hop-by-hop Choke packets

At high speeds and over long distance, sending a choke packet to the source host doesnot work well because the reaction is slow.



(a). A Choke packet that effects source

(b). A Choke packet that affectseachhop it passes through

Host ‘A’ sending traffic to host ‘B’ located at a very long distance from ‘A’. The choke packets are released at the time of congestion and as the 2 hosts are far situated from each other , It takes a maximum delay for choke packets to reach the host ‘A’ and reaction is similar.

An alternative approach to reduce the delay is to have the choke packets take effect at every hop it passes through, as shown in sequence of fig(ii) . Here, as soon as 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, 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.

5. Load shedding

Load shedding is a fancy way of saying that when routers are being inundated by 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, but usually it can do better than that. Which packet to discard 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. For multimedia, a new packet is more important than an old one. The former policy (old is better than new) is often called **wine** and the latter (new is better than old) is often called **milk**.

A step above this in intelligence requires cooperation from the senders. For many applications, some packets are more important than others. For example, certain algorithms for compressing video periodically transmit an entire frame and then send subsequent frames as differences from the last full frame. In this case, dropping a packet that is part of a difference is preferable to dropping one that is part of a full frame.

As another example, consider 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.
- The incentive might be in the form of money, with the **low-priority packets** being cheaper to send than the high-priority ones. Alternatively, senders might be allowed to send **high-priority packets** under conditions of light load, but as the load increased they would be discarded, thus encouraging the users to stop sending them.
- Another option is to allow hosts to **exceed the limits** specified in the agreement negotiated when the **virtual circuit was set up** (e.g., use a higher bandwidth than allowed), but subject to the condition that all excess traffic be marked as low priority. Such a strategy is actually not a bad idea, because it makes **more efficient use of idle resources,** allowing hosts to use

them as long as nobody else is interested, but without establishing a right to them when times get tough.

Random Early Detection

- A popular algorithm for doing this is called **RED** (**Random Early Detection**). In some transport protocols (including TCP), the response to lost packets is for the source to slow down.
- The reasoning behind this logic is that TCP was designed for wired networks and wired networks are very reliable, so lost packets are mostly due to buffer overruns rather than transmission errors. This fact can be exploited to help reduce congestion.
- By having routers drop packets before the situation has become hopeless (hence the ''early'' in the name), the idea is that there is time for action to be taken before it is too late. To determine when to start discarding, routers maintain a running average of their queue lengths.
- When the average queue length on some line exceeds a threshold, the line is said to be congested and action is taken.
- Since the router probably cannot tell which source is causing most of the trouble, picking a packet at random from the queue that triggered the action is probably as good as it can do.

One way is to send it a choke packet, as we have described. A problem with that approach is that it puts even more load on the already congested network. A different strategy is to just discard the selected packet and not report it. The source will eventually notice the lack of acknowledgement and take action. Since it knows that lost packets are generally caused by congestion and discards, it will respond by slowing down instead of trying harder. This implicit form of feedback only works when sources respond to lost packets by slowing down their transmission rate. In wireless networks, where most losses are due to noise on the air link, this approach cannot be used.

6.Jitter Control

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. The variation (i.e., standard deviation) in the packet arrival times is called **jitter**.

High jitter, for example, having some packets taking 20 msec and others taking 30 msec to arrive will give an uneven quality to the sound or movie. Jitter is illustrated in Fig. 5-29. In contrast, an agreement that 99 percent of the packets be delivered with a delay in the range of 24.5 msec to 25.5 msec might be acceptable.

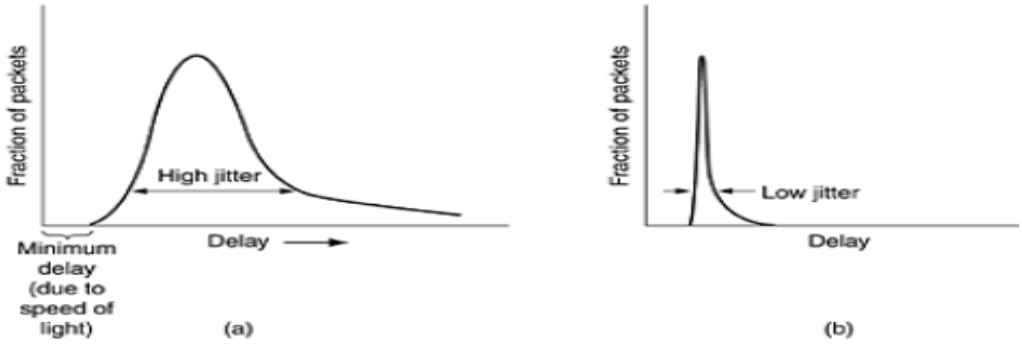


Figure 5-29. (a) High jitter. (b) Low jitter.

The range chosen must be feasible, of course. It must take into account the **speed-of-light transit time** and **the minimum delay** through the routers and perhaps leave a little slack for some inevitable delays.

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. This information is stored in the packet and updated at each hop. If the packet is ahead of schedule, it is held just long enough to get it back on schedule. If it is behind schedule, the router tries to get it out the door quickly.***

In fact, 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. In this way,packets that are ahead of schedule get slowed down and packets that are behind schedule get speeded up, in both cases reducing the amount of jitter.

In some applications, such as video on demand, jitter can be eliminated by buffering at the receiver and then fetching data for display from the buffer instead of from the network in real time. However, for other applications, especially those that require ***real-time interaction between people such as Internet telephony and videoconferencing, the delay inherent in buffering is not acceptable.***

5. Quality of service:

Quality of service (**QoS**) is the overall performance of a telephony or computer network, particularly the performance seen by the users of the network. To quantitatively measure quality of service, several related aspects of the network service are often considered, such as error rates, bit rate, throughput, transmission delay, availability, jitter, etc.

The attributes that can be used to describe the data flow are as follows:

1. **Reliability:**reliability is a necessary characteristic for data flow lack of reliability conveys loosing either a packet or acknowledgment, which results in retransmission of packets. But, reliability differs from one application another application

   For example: applications like e-mail, file transfer and internet access requires reliability of data transmission. Whereas applications like telephony, audio conferencing do not require reliability of data transmission.

2. **Jitter:** the variation in packet delay is known as jitter. **High jitter** conveys that difference between delays is large. **Low jitter** conveys that difference between delays is small.

   Example: For instance, assume that three packets has departure times 1,2,3 and arrival times 31,32,33 then the three packets will have the same delay ie.., 30 units of time. On the other hand, if three packets has arrival time 31,34,36then all packets will have different delay ie.., 30,32,33.

   In first case ie.., packets having same delay is acceptable for application like audio,video, whereas the second case (ie.., packets having different delays) is not acceptable.

3. **Delay:**delay is another characteristic of data flow. The degree of delay varies from one application to another application. Delay is much more important application like telephony, video conferencing, remote login, audio conferencing whereas it is less important for applications like e-mail, file transfer and internet access.

4. **Bandwidth:**the requirement of bandwidth varies from one application to another application. Application like video conferencing requires high bandwidth in order to refresh a color screen whereas application like e-mail needs less bandwidth.

In the field of computer networking and other packet-switched telecommunication networks, quality of service refers to traffic prioritization and resource reservation control mechanisms rather than the achieved service quality. Quality of service is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow.

Quality of service is particularly important for the transport of traffic with special requirements. In particular, developers have introduced technology to allow computer networks to

become as useful as telephone networks for audio conversations, as well as supporting new applications with even stricter service demands.

Multimedia applications in particular, often need a minimum throughput and maximum latency to work. An easy solution to provide good quality of service is to build a network with enough capacity for whatever traffic will be thrown at it. The name for this solution is **over provisioning**.

The most basic way to avoid congestion is to build a network that is well matched to the traffic that it carries. If there is a low-bandwidth link on the path along which most traffic is directed, congestion is likely. Sometimes resources can be added dynamically when there is serious congestion, for example, turning on spare routers or enabling lines that are normally used only as backups (to make the system fault tolerant) or purchasing bandwidth on the open market. More often, links and routers that are regularly heavily utilized are upgraded at the earliest opportunity. This is called **provisioning** and happens on a time scale of months, driven by long-term traffic trends.

The resulting network will carry application traffic without significant loss and, assuming a decent routing scheme, will deliver packets with low latency.

Quality of service mechanisms let a network with less capacity meet application requirements just as well at a lower cost. Moreover, over provisioning is based on expected traffic. All bets are off if the traffic pattern changes too much. With quality of service mechanisms, the network can honor the performance guarantees that it makes even when traffic spikes, at the cost of turning down some requests.

Four issues must be addressed to ensure quality of service:

1. What applications need from the network.
2. How to regulate the traffic that enters the network.
3. How to reserve resources at routers to guarantee performance.
4. Whether the network can safely accept more traffic.

No single technique deals efficiently with all these issues. Instead, a variety of techniques have been developed for use at the network (and transport) layer. Practical quality-of-service solutions combine multiple techniques. To this end, we will describe two versions of quality of service for the Internet called Integrated Services and Differentiated Services.

4.1 Application Requirements

A stream of packets from a source to a destination is called a **flow**. A flow might be all the packets of a **connection** in a **connection-orientednetwork**, or all the packets sent from **one process to another process** in a connectionlessnetwork.

The needs of each flow can be characterized by four primaryparameters:

- ***bandwidth,***
- ***delay,***
- ***jitter, and***
- ***Loss.***

The flow requires the**QoS**(**Quality of Service**).Several common applications and the stringency of their network requirementsare listed in Fig. 5-27. If the network requirements are less demandingthan application requirements in those cases that the application can improveon the **service provided by the network**.

In particular, networks do not needto be lossless for reliable file transfer, and they do not need to deliver packets withidentical delays for audio and video playout. Some amount of loss can be repairedwith **retransmissions**, and some amount of **jitte**r can be smoothed by bufferingpackets at the receiver. However, there is nothing applications can do to remedythe situation if the network provides too little bandwidth or too much delay.

Application	Bandwidth	Delay	Jitter	Loss
Email	Low	Low	Low	Medium
File sharing	High	Low	Low	Medium
Web access	Medium	Medium	Low	Medium
Remote login	Low	Medium	Medium	Medium
Audio on demand	Low	Low	High	Low
Video on demand	High	Low	High	Low
Telephony	Low	High	High	Low
Videoconferencing	High	High	High	Low

Figure 5-27. Stringency of applications' quality-of-service requirements.

The applications differ in their bandwidth needs, with **email, audio** in allforms, and **remote login** not needing much, but file sharing and video in all formsneeding a great deal.

File transfer applications, includingemail and video, are not delay sensitive. If all packets are **delayed** uniformlyby a few seconds, no harm is done. Interactive applications, such as **Websurfing and remote login, are more delay sensitive**. Real-time applications, suchas **telephony and videoconferencing, have strict delay requirements**. **Playing audio or video files from a serverdoes not require low delay.**

The variation (i.e., standard deviation) in the delay or packet arrival times iscalled **jitter**. The first **three** applications in Fig. 5-27 are not sensitive to the packetsarriving with irregular time

intervals between them. **Remote login** is somewhat sensitive to that, since updates 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 unless the application hides the jitter.

For **audio, a jitter** of even a few milliseconds is clearly audible. The **first four** applications have more stringent requirements on loss than audio and video because all bits must be delivered correctly. This goal is usually achieved with retransmissions of packets that are lost in the network by the transport layer.

Audio and video applications can tolerate some lost packets without retransmission because people do not notice short pauses or occasional skipped frames.

To accommodate a variety of applications, networks may support different categories of **QoS**. An influential example comes from ATM networks, which were once part of a grand vision for networking but have since become a niche technology. They support:

1. Constant bit rate (e.g., telephony): **Constant bit** rate is an attempt to simulate a wire by providing a uniform bandwidth and a uniform delay.

2. Real-time variable bit rate (e.g., compressed videoconferencing): **Variable bit rate** occurs when video is compressed, with some frames compressing more than others. Sending a frame with a lot of detail in it may require sending many bits, whereas a shot of a white wall may compress extremely well.

3. Non-real-time variable bit rate (e.g., watching a movie on demand): Movies on demand are not actually **real time** because a few seconds of video can easily be buffered at the receiver before playback starts, so jitter on the network merely causes the amount of stored-but-not-played video to vary.

4. Available bit rate (e.g., file transfer): **Available bit rate** is for applications such as email that are not sensitive to delay or jitter and will take what bandwidth they can get.

Example: file transfer

2. Traffic shaping:

Traffic shaping is a technique for regulating the average rate and burstiness of a flow of data that enters the network. The goal is to allow applications to transmit a wide variety of traffic that suits their needs, including some bursts, yet have a simple and useful way to describe the possible traffic patterns to the network.

When a flow is set up, the user and the network (i.e., the customer and the provider) agree on a certain traffic pattern (i.e., shape) for that flow. The customer says to the provider ‘‘My transmission pattern will look like this; can you handle it?’’

Sometimes this agreement is called an **SLA** (**Service Level Agreement**), especially when it is made over aggregate flows and long periods of time, such as all of the traffic for a given customer. As long as the customer fulfils her part of the bargain and only sends packets according to the agreed-on contract, the provider promises to deliver them all in a timely fashion.

- Traffic shaping reduces congestion and thus helps the network live up to its Promise.
- Monitoring a traffic flow is called **traffic policing**.
- Shaping and policing are not so important for peer-to-peer and other transfers that will consume any and all available bandwidth, but they are of great importance for real-time data, such as audio and video connections, which have stringent quality-of-service requirements.
- Traffic shaping is a technique of controlling the amount and rate of flow of the network traffic. There are two types of traffic shaping techniques they are
  - **Leaky bucket**
  - **Token bucket**

Leaky Buckets:

The leaky bucket implementation is used to control the rate at which traffic is sent to the network. A leaky bucket provides a mechanism by which bursty traffic can be shaped to present a smooth stream of traffic to the network, as opposed to bursts of low-volume and high-volume flows.

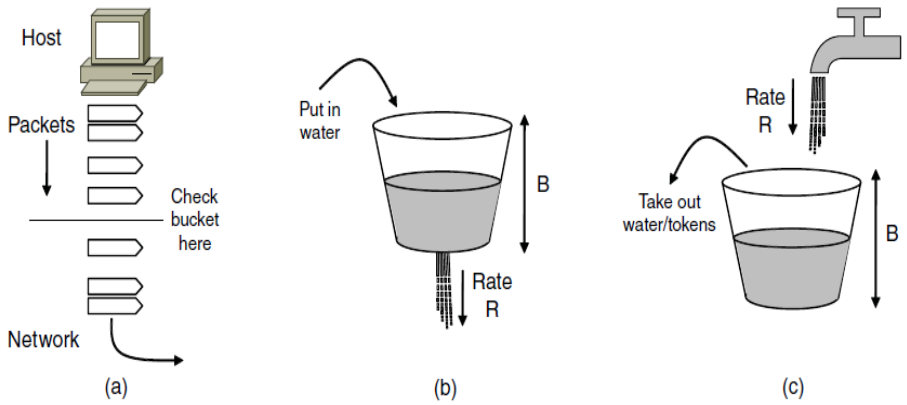
Leaky bucket implementation is same as a bucket a hole at the bottom. Whatever may be the rate at which water is being filled into the bucket, the water always flows out at a constant rate.

There are two different algorithms exist for two types of packets,

a) **Fixed size packet which uses originally leaky bucket algorithm**
b) **Variable size packet which uses byte counting leaky bucket algorithm**

Need of traffic shaping:

- Traffic shaping is a mechanism that alters the traffic characteristics of a stream of cells on a connection to achieve better efficiency, while meeting quality of service objective.
- Traffic shaping maintains cell sequence integrity on a connection. It modifies the traffic characteristics of cell flow with the consequence of increasing the mean cell transfer delay.
- It is possible to reduce congestion with the aid of traffic shaping.
- It is not important for transferring a file than transferring real-time traffic such as video, audio.
- It is alteration of traffic stream to increase the network efficiency.
- It controls the amount and rate at which traffic is sent to the network. This control can be accomplished in many ways and for many reasons but traffic shaping always results in delaying packet.
- Traffic shaping is the way of examining the flow of traffic and it is related practise of packet dropping.



(a) Shaping packets. (b) A leaky bucket. (c) A token bucket

Original leaky bucket algorithm:

a) Arriving packets in a bucket with a hole in the bottom
b) The bucket can queue at most '*b*' bytes. If a packet arrives when the bucket is full, the packet is discarded
c) Packet drain through the hole in the bucket in the network at a constant rate of '*r*' bytes/sec, thus smoothing traffic bursts.
d) The size 'b' of a packet is limited by the available memory of the system.

Byte-counting leaky bucket

a) Initialize a counter to k at the tick of the clock
b) If k is greater than the size of the packet send the packet and decrement the value of counter by packet size. Repeat this step until k is smaller than packet size.
c) Reset the counter and continue step-1.

Token bucket:

It is a technique for controlling the amount of traffic within a network. This algorithm allows the host in an idle state to gather credit in the form of tokens for the future purpose. At each clock tick, 'n' tokens are sent by the system to the bucket and a token is removed on each byte sent.

Consider for an example if the number of tokens 'n' is 50 and the host is idle for 100 ticks, then the number of tokens collected in a bucket is 5000. These collected tokens can be consumed with in a single tick with 5000 cells by host or it may take 1000 ticks with 5 cells/tick ie.., the bursts of data can be sent to the bucket till it becomes full.

This mechanism cab be implemented using a counter. Initially the token is set to 0. The counter is incremented or decremented by 1 on each addition of token an each transfer of a data unit, respectively. The host stops sending the data as soon as the counter becomes '0'.

3. Packet scheduling:

To improve the quality of service packet scheduling is used. As the packets are arriving from different flows at a switch/router for the purpose of processing. Some of scheduling is needed. A good scheduling is one that considers all the packets from different flows, equally and in appropriate manner for improving the quality of service.

Algorithms that allocate router resources among the packets of a flow and between competing flows are called **packet scheduling algorithms**. Three different kinds of resources can potentially be reserved for different flows:

a. Bandwidth: The first one, bandwidth, is the most obvious. If a flow requires 1 Mbps and the outgoing line has a capacity of 2 Mbps, trying to direct three flows through that line is not going to work. Thus, reserving bandwidth means not oversubscribing any output line.

b. Buffer space: A second resource is buffer space. When a packet arrives, it is buffered inside the **router** until it can be transmitted on the chosen outgoing line. The purpose of the buffer is to absorb **small bursts** of traffic as the flows contend with each other. If no buffer is available, the packet **has to be discarded** since there is no place to put it. For

good quality of service, some buffers might be reserved for a specific flow so that flow does not have to compete for buffers with other flows.

c. CPU cycles: CPU cycles may also be a scarce resource. It takes router CPU time to process a packet, so a router can process only a certain number of packets per second. While modern routers are able to process most packets quickly, some kinds of packets require greater CPU processing, such as the ICMP packets. Making sure that the CPU is not overloaded is needed to ensure timely processing of these packets.

Some of the scheduling polices are given below:

a) ***FIFO SCHEDULING:*** In this type of queuing all the incoming packets wait in a queue till the router becomes ready for processing them. if the packets arrival rate on an average is greater than the average processing rate, then the queue becomes full and all the new packets are discarded.

FIFO scheduling is simple to implement, but it is not suited to providing good quality of service because when there are multiple flows, one flow can easily affect the performance of the other flows. If the first flow is aggressive and sends large bursts of packets, they will lodge in the queue. Processing packets in the order of their arrival means that the aggressive sender can hog most of the capacity of the routers its packets traverse, starving the other flows and reducing their quality of service.

b) ***Priority queuing:*** In priority queuing, all the incoming packets are initially assigned to a priority class containing its own queue. The packets in the highest priority queue are processed first followed by the packets in the lowest-priority queue once the system starts processing a queue it will not stop till the packets in that queue are served.

Advantage*:* it offers better QoS than the FIFO scheduling as the highest priority traffic such as multimedia can be handled and destined first with much lesser delay.

Disadvantage: it leads to a condition called starvation where in the packets in the lower packets queues will never get a chance to be served as the processer busy in serving highest priority packets.

c) **Weighted fair queuing:** in this type of scheduling technique, all the incoming packets are assigned to the priority classes containing their own queues. The queues are then given weights based their priorities ie.., a queue with higher priority is assigned larger weight where as a queue with lower priority gets lesser weight. The system then starts processing the packets in a round robin fashion with random selection of packets from each queue depending upon the associated weights.

Example: if the weights are 5,4, and 3 then five packets are selected for processing from Q1, four from Q2 and three from Q3.

4. Admission control:

It is a technique of accepting or rejecting flow by a router or a switch based on the flow specifications. Prior to accepting the new flow, the flow specifications must be checked to determine its capacity in terms of CPU speed , bandwidth etc..,

Admission control used to control congestion, which is a performance guarantee. The reservations must be made at all of the routers along the route that the packets take through the network. Any routers on the path without reservations might become congested, and a single congested router can break the QoS guarantee.

Many routing algorithms find the single best path between each source and each destination and send all traffic over the best path. This may cause some flows to be rejected if there is not enough spare capacity along the best path. QoS guarantees for new flows may still be accommodated by choosing a different route for the flow that has excess capacity. This is called **QoS routing**.

It is also possible to split the traffic for each destination over multiple paths to more easily find excess capacity. A simple method is for routers to choose equal-cost paths and to divide the traffic equally or in proportion to the capacity of the outgoing links.

Given a path, the decision to accept or reject a flow is not a simple matter of comparing the resources (bandwidth, buffers, cycles) requested by the flow with the router's excess capacity in those three dimensions. It is a little more complicated than that. To start

with, although some applications may know about their bandwidth requirements, few know about **buffers** or **CPU cycles**, so at the minimum, a different way is needed to describe flows and translate this description to router resources.

Some applications may be willing to haggle about the **flow parameters** and others may not. For example, a movie viewer that normally runs at 30 frames/sec may be willing to drop back to 25 frames/sec if there is not enough free bandwidth to support 30 frames/sec. Similarly, the number of pixels per frame, audio bandwidth, and other properties may be adjustable.

Many parties may be involved in the flow negotiation (the sender, the receiver, and all the routers along the path between them), flows must be described accurately in terms of specific parameters that can be negotiated. A set of such parameters is called a **flow specification**. Typically, the sender (e.g., the video server) produces a flow specification proposing the parameters it would like to use. As the specification propagates along the route, each router examines it and modifies the parameters as need be. The modifications can only reduce the flow, not increase it (e.g., a lower data rate, not a higher one). When it gets to the other end, the parameters can be established.

As an example of what can be in a flow specification, consider the example of Fig. 5-32. It has five parameters. The first two parameters, the ***token bucket rate*** and ***token bucket size***, use a token bucket to give the maximum sustained rate the sender may transmit, averaged over a long time interval, and the largest burst it can send over a short time interval.

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

Figure 5-32. An example flow specification.

The third parameter, the ***peak data rate***, is the maximum transmission rate tolerated, even for brief time intervals. The sender must never exceed this rate even for short bursts.

The last two parameters specify the **minimum and maximum packet sizes**, including the transport and network layer headers (e.g., TCP and IP). The **minimum size** is useful because processing each packet takes some fixed time, no matter how short. A router may be prepared to handle 10,000 packets/sec of 1 KB each, but not be prepared to handle 1,00,000 packets/sec of 50 bytes each, even though this represents a lower data rate. The **maximum packet size** is important due to internal network limitations that may not be exceeded.

5. Integrated services:

The integrated services are designed to provide **quality of services(QoS).** In the internet and focus on the use of quality of service at the IP(network layer). In IP best effort delivery in IP do not guarantee required bandwidth for applications like real time applications audio, video. If such application acquires additional bandwidth then it may be harmful to other applications requiring less bandwidth. This situation results in congestion.

Integrate services are introduced to overcome the problems of IP. The integrated services are even called **IntServ.** It is a flow based quality of service model. Where the user is required to create a kind of virtual circuit flow from the source to the destination and inform all the routers of the required resources.

Signaling: flow based model can be implemented over IP(connectionless protocol) using a signalling protocol called **resource reservation protocol(RSVP).** This protocol provides the signalling mechanism that is used for reserving the internet resources.

Flow specification: a flow specification is defined when the source reserves the internet resources. It consists of two parts. They are:

1. **Respec (Resource Specification):** it defines the resource that is to be reserved by the flow.
2. **Tspec (Traffic Specification):** it defines the traffic characterisation of the flow.

Resource ReserVation protocol (RSVP):

- Resource reservation protocol is a network control protocol that enables internet application to obtain different quality of services for their data flow.
- RSVP is a set of communication rules that allows channels or paths on the internet to be reserved for the multicast or unicast transmission of video and other high bandwidth messages.
- The RSVP is intended to provide IP network with the capability to support the different performance requirements of different application types.
- RSVP provides receiver initiated setup of resource reservation for multicast or unicast data flows with scaling and robustness.

Unicast: In unicast both sender and receiver agree on a specific quality of service and the network between them should also support the same quality of service. If the network is overloaded then the expected quality of service is not provided and packets are delivered with low quality of service.

Multicast: it provides a greater amount of internetwork traffic if the application changes their destination dynamically. RSVP is used to control for congestion for multicasting. It allows

many senders to transmit data to different groups of receivers where each individual receiver can be allowed to dynamically change its path to utilize bandwidth better and to reduce congestion.

RSVP is different from normal multicast routing algorithms. RSVP works in congestion with routing protocol and stores the list along the routes that routing protocol calculates.

Working of RSVP: RSVP allows any receiver in a group to send a reservation message to the desired sender if it wishes to eliminate congestion. Using the reserved path forwarding algorithm this message is transmitted. Router at each hop reserves the bandwidth required. When the message reaches the sender, the bandwidth has already been reserved all the way from sender to receiver.

Example:

Host 1,2,3 are multicast senders and host 4,5,6 are multicast receivers. The points A,B,L specify the router. When host 4 wishes to establish a connection with host 1it sends a reservation message, during each hop the routers A,E,H,J reserves the channel and records the information about the reservation. Now the packets can be transmitted from host 1 to host 4 without congestion.

If host 6 wishes to reserve a channel with host 1 it also sends a message for a reservation. At router H, it sees that the bandwidth has already been reserved and therefore, it doesn't have to reserve anymore bandwidth.

The receiver must specify the sources from which it wants to receive data. It should specify if it wishes to change the source dynamically or not. This information is used by router to optimize bandwidth.

Characteristics of RSVP:

1. RSVP requests resources for simplex flows. ie.., a traffic stream in only one direction from sender to one or more receivers.
2. RSVP is not a routing protocol but works with future routing protocols.
3. RSVP is a receiver oriented in that the receiver of a data flow initiates and maintains the resource reservation for that flow.

Differentiate services:

Flow-based algorithms have the potential to offer good quality of service to one or more flows because they reserve whatever resources are needed along the route. They require an advance setup to establish each flow, something that does not scale well when there are thousands or millions of flows.

Also, they maintain internal per-flow state in the routers, making them vulnerable to router crashes. Finally, the changes required to the router code are substantial and involve complex router-to-router exchanges for setting up the flows.

IETF has also devised a simpler approach to quality of service, one that can be largely implemented locally in each router without advance setup and without having the whole path involved. This approach is known as **class-based** (as opposed to flow-based) quality of service.

Differentiated services can be offered by a set of routers forming an administrative domain. The administration defines a set of service classes with corresponding forwarding rules. If a customer subscribes to differentiated services, customer packets entering the domain are marked with the **class** to which they belong. This information is carried in the *Differentiated services* field of IPv4 and IPv6 packets. The classes are defined as **per hop behaviours** because they correspond to the **treatment the packet will receive at each router, not a guarantee across the network**.

Better service is provided to packets with some per-hop behaviors (e.g., premium service) than to others (e.g., regular service). Traffic within a class may be required to conform to some specific shape, such as a leaky bucket with some specified drain rate.

The difference between **flow-based quality of service** and **class-based quality of service**.

Consider an example: Internet telephony. With a flow-based scheme, each telephone call gets its own resources and guarantees. With a class-based scheme, all the telephone calls together get the resources reserved for the class telephony. These resources cannot be taken

away by packets from the Web browsing class or other classes, but no telephone call gets any private resources reserved for it alone.

Expedited Forwarding

The choice of service classes is up to each operator, but since packets are often forwarded between networks run by different operators, IETF has defined some network-independent service classes. The simplest class is **expedited forwarding**.

The idea behind expedited forwarding is very simple. Two classes of service are available:

a) regular: The vast majority of the traffic is expected to be regular.

b) expedited: A limited fraction of the packets are expedited. The expedited packets should be able to transit the network as though no other packets were present.

In this way they will get low loss, low delay and low jitter service just what is needed for VoIP. A symbolic representation of this ‘‘two-tube’’ system is given in Fig. 5-36. Note that there is still just one physical line. The two logical pipes shown in the figure represent a way to reserve bandwidth for different classes of service, not a second physical line.

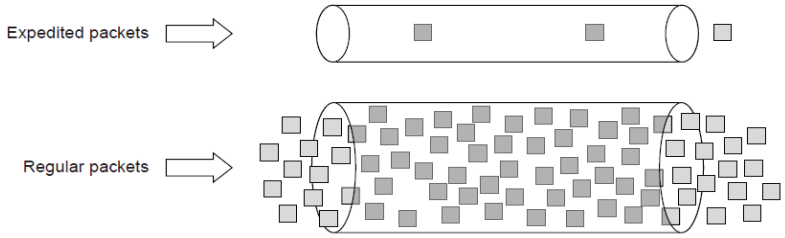


Figure 5-36. Expedited packets experience a traffic-free network.

One way to implement this strategy is as follows. Packets are classified as expedited or regular and marked accordingly. This step might be done on the sending host or in the ingress (first) router. The advantage of doing classification on the sending host is that more information is available about which packets belong to which flows. This task may be performed by networking software or even the operating system, to avoid having to change existing applications.

For example, it is becoming common for VoIP packets to be marked for expedited service by hosts. If the packets pass through a corporate network or ISP that supports expedited service, they will receive preferential treatment. If the network does not support expedited service, no harm is done.

If the marking is done by the host, the ingress router is likely to police the traffic to make sure that customers are not sending more expedited traffic than they have paid for. Within the network, the routers may have two output queues for each outgoing line, one for

expedited packets and one for regular packets. When a packet arrives, it is queued accordingly. The expedited queue is given priority over the regular one, for example, by using a priority scheduler. In this way, expedited packets see an unloaded network, even when there is, in fact, a heavy load of regular traffic.

Assured forwarding:

Assured forwarding specifies that there shall be **four priority classes**, each class having its own resources. The top three classes might be called **gold, silver, and bronze**. In addition, it defines three discard classes for packets that are experiencing congestion: **low, medium, and high.** Taken together, these two factors define 12 service classes.

Figure 5-37 shows one way packets might be processed under assured forwarding. The *first step* is to classify the packets into one of the four priority classes. As before, this step might be done on the sending host (as shown in the figure) or in the ingress router, and the rate of higher-priority packets may be limited by the operator as part of the service offering.

The *next step* is to determine the discard class for each packet. This is done by passing the packets of each priority class through a traffic policer such as a token bucket. The policer lets all of the traffic through, but it identifies packets that fit within small bursts as low discard, packets that exceed small bursts as medium discard, and packets that exceed large bursts as high discard. The combination of priority and discard class is then encoded in each packet.

Finally, the packets are processed by routers in the network with a packet scheduler that distinguishes the different classes. A common choice is to use weighted fair queueing for the four priority classes, **with higher classes given higher weights**. In this way, **the higher classes will get most of the bandwidth, but the lower classes will not be starved of bandwidth entirely.**

For example, if the weights double from one class to the next higher class, the higher class will get twice the bandwidth. Within a priority class, packets with a higher discard class can be preferentially dropped by running an algorithm such as RED (Random Early Detection), which we saw in Sec. 5.3.5. RED will start to drop packets as congestion builds but before the router has run out of buffer space. At this stage, there is still buffer space with which to accept low discard packets while dropping high discard packets.

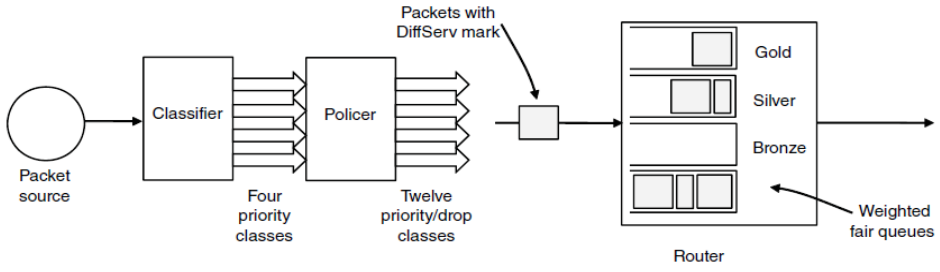


Figure 5-37. *A possible implementation of assured forwarding.*

Internetworking, The Network Layer in the Internet.

Internetworking:

Internetworking is connection of two or more networks. There are different types of networks such as local area networks (LAN), wide area networks (WAN) a variety of protocols are used at every layer and interconnection of these networks is called internet. Through internet it is possible to exchange information between hosts which are connected across various networks.The probable reasons for which a variety of networks and their protocols will exist are,

Firstly, the installed base of different networks is large and still growing.

Secondly, with the rate of computers and networks getting cheaper, it becomes more affordable.

When different networks have different technologies, then new hardware developments will ultimately result in respective software development.In **internetworking** there are various types of networks and their scenarios. There are basically four types of scenarios.

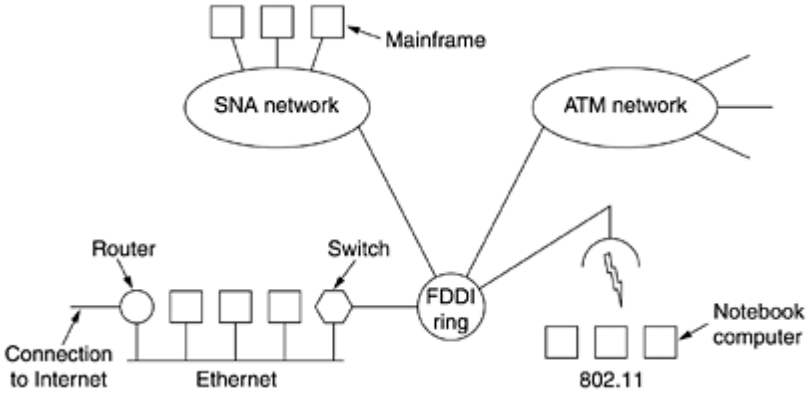
- *LAN-LAN*
- *LAN-WAN*
- *WAN-WAN*
- *LAN-WAN-LAN*

LAN-LAN: To exchange information between two departments of the same organization which are present on different host's different networks? By using bridges we can interconnect different LANs.

LAN-WAN: To send information to a remote host which is connected through wide area network, transfer information from LAN to WAN.

WAN-WAN: For transferring information between two different organizations.

LAN-WAN-LAN: For international organization which host on different branches communicate with each other. The information generated by first LAN will be transmitted to another LAN through wide area network.



A collection of interconnected networks.

For example, how different networks might be connected is shown in figure. A corporate network with multiple locations tied together by a wide area ATM network is shown in figure. At one of the locations, an FDDI optical backbone is used to connect Ethernet, an 802.11 wireless LAN and the corporate data center's SNA mainframe network. Purpose of interconnecting all these networks is to allow users on any of them to communicate with users on all the other ones and access data on any of them.

How networks differ?

Networks can differ at each layer. The list of some of the differences that can occur in the network layer

Item	Some Possibilities
Service offered	Connection oriented versus connectionless
Protocols	IP, IPX, SNA, ATM, MPLS, AppleTalk, etc.
Addressing	Flat (802) versus hierarchical (IP)
Multicasting	Present or absent (also broadcasting)
Packet size	Every network has its own maximum
Quality of service	Present or absent; many different kinds
Error handling	Reliable, ordered, and unordered delivery
Flow control	Sliding window, rate control, other, or none
Congestion control	Leaky bucket, token bucket, RED, choke packets, etc.
Security	Privacy rules, encryption, etc.
Parameters	Different timeouts, flow specifications, etc.
Accounting	By connect time, by packet, by byte, or not at all

Some of the many ways networks can differ.

When a packet starts at the source host and before reaching to the destination host, the following problems may occur,

1) A packet may have to travel through connection oriented and connection less network or vice versa.
2) Different maximum size packets are used by different networks.
3) The packet has to pass through different networks which are having different flow control, congestion control, Quality of Service and securities etc..,

How networks can be connected:

Network layer can be interconnected by using various devices. The various types of devices that are needed for internetworking are as follows.

Repeaters: it works at physical layer. They are used for amplifying or strengthening weak signals, so that the signals can travel over long distances.

Bridges: These are intelligent devices that are used to interconnect local area networks. The main function of a bridge is when a frame is received, it checks the destination MAC address of the frame, if it finds the particular destination on the other side of the LAN, then only, the frame is forwarded. Thus the bridge takes decisions either to forward the frames or not. Bridges are used at data link layer.

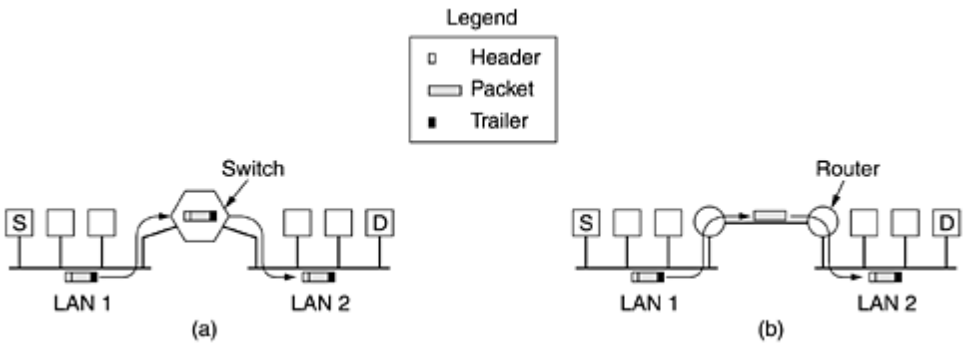
Routers or multiprotocol routers: The routers work at network layer so that the data can be transmitted across multiple networks. Multiple protocol router forwards the incoming packet over the line that belongs to a different network and make use of different protocols.

Transport gateways: This is used the transport layer for connecting two networks in the layer.

Application gateways: These gateways are used, in application layer for connecting two different parts of an application in the layer.

Let us consider internetworking in the network layer and see how that differs from switching in the data link layer.

The source machine S, wants to send a packet to the destination machine D. These machines are on different Ethernets connected by switch. Source S machine encapsulates the packet in the frame with MAC header and trailer. Frame arrives at the switch, which then determines that frame is intended to go to LAN 2 by examining at its MAC address. Switch moves the frame from LAN1 and transmits it to LAN2.



(a) Two Ethernets connected by a switch. (b) Two Ethernets connected by routers.

Let us consider the same situation but the two Ethernets connected by a pair of routers instead of switch. The routers are connected by a point-to-point line, possibly with a leased line, thousands of kilometers away. The frame is picked up by the router and the packet is removed from the frame. Then the router examines the IP address in the packet and looks up this address in its routing table.

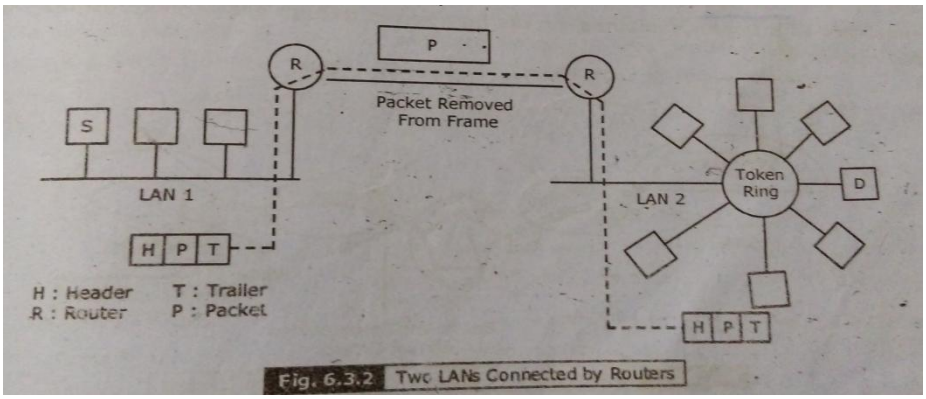


Fig. 6.3.2 Two LANs Connected by Routers

Based on the address available in the packet, it decides to send the packet to the remote router. At remote router the packet is put into data field of an Ethernet frame and transmitted on to Lan2.

The difference between switched case and routed case is shown below

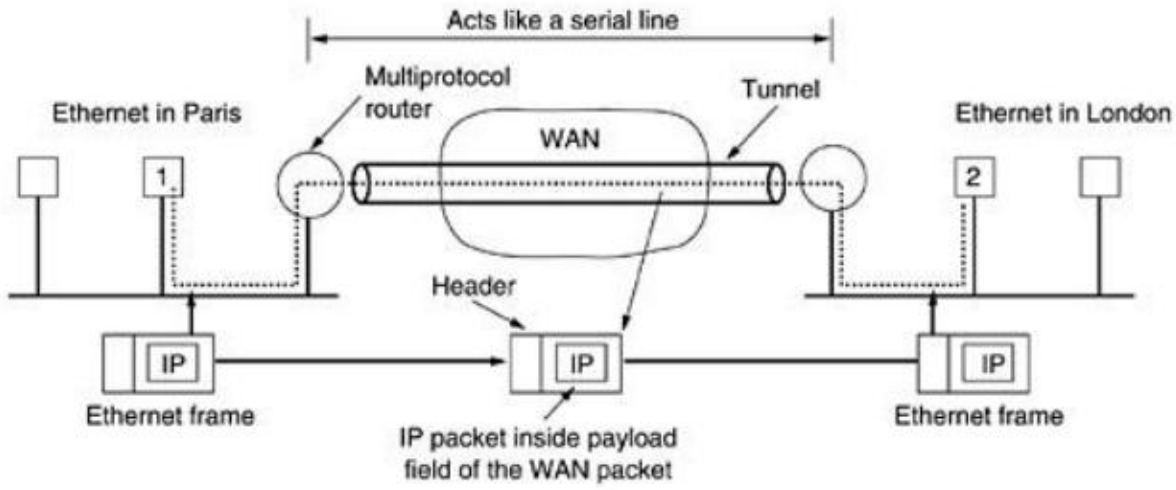
Table 6.3.1 Switched Case Vs Routed Case

S.No.	Switched Case	Routed Case
1)	Entire frame is transported. On the basis of MAC address.	The packet is extracted from the frame and the address in the packet is used for deciding where to send it.
2)	No need to understand network layer protocol.	Routers have to understand network layer protocol.

Tunneling:

- Tunneling is a special case of internetworking where source and destination are on the same type of network (but not with in single network) and there is a different network in between.
- Tunneling is a method of transmitting data that is intended for use only within a private network through a public network.
- Tunneling is generally done by encapsulating the private network data and protocol information within the public network transmission units, so that the private network protocol information appears to the public network as data.
- Tunneling allows the use of internet, which is public network to convey data on behalf of a private network.

Example: source and destination are in two TCP/IP Ethernet LANs and both LANs are connected by an ATM-based wide-area network.



Tunneling a packet .

From the above figure WAN is seen as a big tunnel where IP packets enters at one end. This IP packet is rolled in as ATM (WAN Protocol here) payload at one end and at other end, IP packet is taken out and inserted into Ethernet frame and destination receives it. All the routers in the WAN treat is an ATM cell only. They do not even know that IP packets exist inside it.

The tunneling concept is also used in virtual private LANs using public transport network. Consider a company having two branches located in two different places. The data can be transmitted over a public network between branches by using tunneling, where the router in one office rolls up its IP packet holding the address of the router at the other end. All the nodes in the public network view the packet destined to the router at other end, thus protecting the packets and hence private LAN is realized.

The advantage of tunneling is security that is provided for VPN and it allows interconnection of dissimilar networks.

Internetwork routing:

Internetwork routing is analogues to single subnet routing. In this routing a graph is constructed for a network where each router is represented by a node which is able to access information from every other gateway connected to the network. After the graph is constructed as in figure (b) for a network as in figure (a), different routing algorithms such as hierarchical routing and flow based algorithms are applied which result in two different routing algorithms.

Routing in internetworks adds additional complexity to the algorithm design compared to the case of intranet routing.

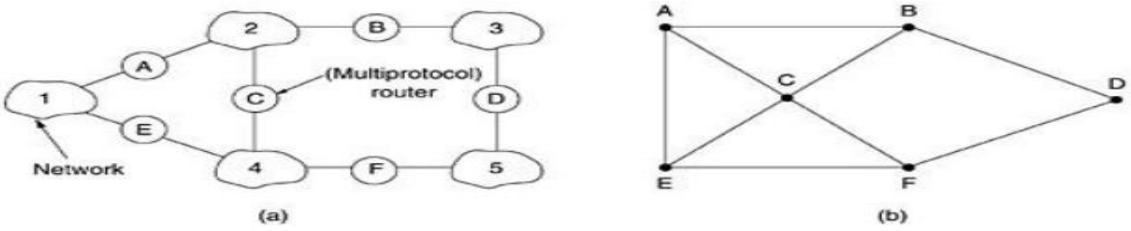
1) Since route may span through several networks across several countries, the individual policies of the countries about data and information exchange needs are to be considered.

2) Many a time, the interior routing model and algorithms of component networks add a lot of burden to the routing algorithm design for internetworks.

3) Variation in Quality-Of-Service in individual network complicates the achievement of global acceptable Quality-Of-Service.

It is already mentioned that, several component networks are connected through multiprotocol routers, which can understand frames of different protocols. So, a graph having these multiprotocol routers as nodes or vertices, connected by edges is constructed. An edge between two routers A and B shows the reachability of A from B and vice-versa ie.., A can directly send packets to B.

For example: below figure shows an internetwork of five networks connected by six multiprotocol routers. We observe here the gateway B in fig(a) can directly access A and C through network 2 and also D through network 3, which result graph is fig (b).



(a) An internetwork. (b) A graph of the internetwork.

Once the graph has been constructed, routing algorithms (link state routing or distance vector) are applied on it. So, there exist two level routing algorithms.

1) Interior gateway routing protocol, to route with in a network.
2) Exterior gateway routing protocol, to route across network.

Each network in an internetwork is called "autonomous system" (AS).

Packet Fragmentation

Each network or link imposes some maximum size on its packets. These limits have various causes, among them

1. Hardware (e.g., the size of an Ethernet frame).
2. Operating system (e.g., all buffers are 512 bytes).
3. Protocols (e.g., the number of bits in the packet length field).
4. Compliance with some (inter)national standard.
5. Desire to reduce error-induced retransmissions to some level.
6. Desire to prevent one packet from occupying the channel too long.

The result of all these factors is that the network designers are not free to choose any old maximum packet size they wish. Maximum payloads for some common technologies are 1500 bytes for Ethernet and 2272 bytes for 802.11. IP is more generous, allows for packets as big as 65,515 bytes.

Packets can either fixed size or packet size. The maximum packet size of a packet varies from network-to-network. Several factors will dominate the size of the packets like hardware, the operating systems, protocol design, and attempts to reduce the retransmission overhead in case of errors and to present the packet occupying the channel too long.

In internetworking, it is possible that a larger size packet can appear on a smaller size network. This is done by dividing the original packet in to smaller fragments which are treated as independent packets and are routed over a smaller size network. Dividing the larger packet in to smaller independent packets is called fragmentation

The obvious solution for forwarding packets when they enter the network having similar maximum packet size is to fragment the incoming packet and transmit only these fragments.

There are two strategies used for recombining the fragment into original packets. They are

a) Transparent fragmentation

b) Non-transparent fragmentation

Transparent fragmentation*:* In this type of fragmentation all the intermediate gateways fragment the larger packet in to smaller independent fragments which are recombined at the same exit gateway whose address is specified by the source gateway with each fragmented packet. A packet is fragmented when it enters a network and is reassembled when it leaves the same network, so that other networks do not know that fragmented occurred.

Disadvantages of Transparent fragmentation:

1) Each exit gateway must have knowledge about all the packets in order to mark end of packet.
2) Reassembling is done only at each exit gateway. Restricting all the fragments to the same node may cause some loss in performance.
3) Large overhead occurs due to repeated fragmenting and reassembling.

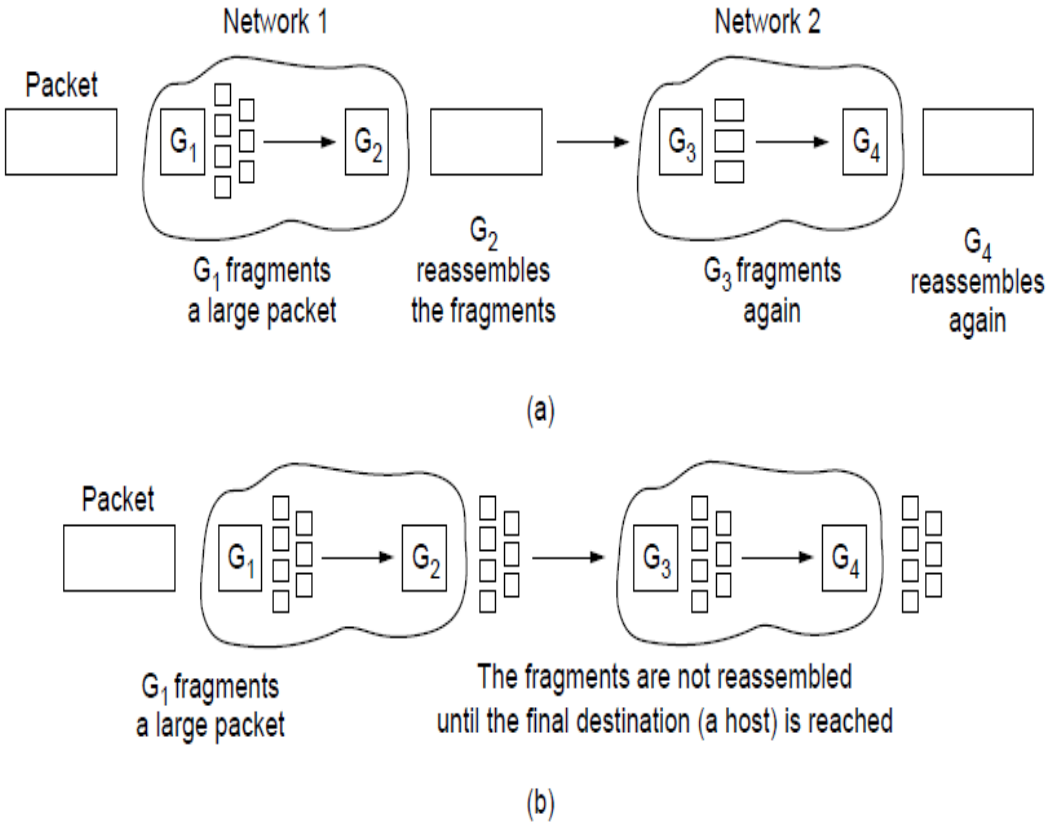
Non-transparent fragmentation: when a packet is fragmented, it is never recombined are intermediate nodes. The packets are recovered by combining only at the destination node.

Advantages of Non-Transparent fragmentation:

- Fragments need not exit through the same gateway.
- Reassembling is done only at the destination.

Disadvantages of Non-Transparent fragmentation:

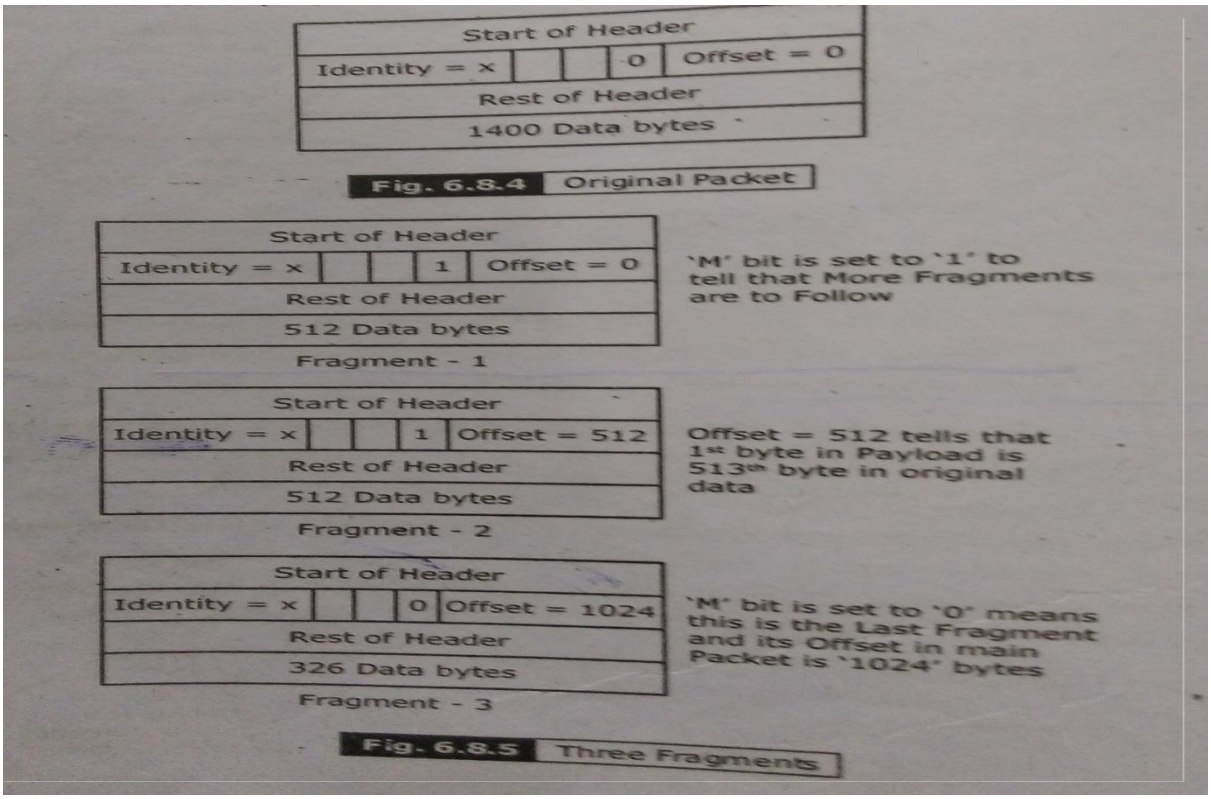
- Every destination host should have knowledge of reassembling the packets.
- Overhead increases as each fragment will reach destination where as in transparent fragmentation, overhead decreases as soon as the small packet exists the network.



(a) Transparent fragmentation. (b) Nontransparent fragmentation.

Although several mechanisms exist to a number of individual fragments and using this information original packet can be reconstructed at the end. But, the best mechanism is to store the offset of that fragment in original packet in terms of number of bytes.

Suppose two networks A and B with maximum payload sizes 1400, 512 bytes respectively are connected by router ‘R’. Assume IP exists at network layer in both the networks, A packet with 1400 payload is fragmented into 512, 512, 326 payload bytes. The ‘M’ set to ‘1’ in IP-header says whether more fragments are to follow or not. For last fragment ‘M’ bit is set to ‘0’ .



THE NETWORK LAYER IN THE INTERNET

The network layer protocol is IP (Internet Protocol). Internet can be viewed as a collection of subnets or autonomous systems (AS) that are interconnected. This interconnection is made up of high bandwidth lines and fast routers.

Internet is developed and designed by keeping the following 10 principles in mind.

1**. Make sure it works**. Do not finalize the design or standard until multiple prototypes have successfully communicated with each other.

2. Keep it simple. The design must be as simple as possible by removing the unnecessary components from the layer

3. Make clear choices. If there are several ways of doing the same thing, choose one. Having two or more ways to do the same thing is looking for trouble. Standards often have multiple options or modes or parameters because several powerful parties insist that their way is best.

4. Exploit modularity. This principle leads directly to the idea of having protocol stacks, each of whose layers is independent of all the other ones. In this way, if circumstances that require one module or layer to be changed, the other ones will not be affected.

5. Expect heterogeneity. Different types of hardware, transmission facilities, and applications will occur on any large network. To handle them, the network design must be simple, general, and flexible.

6. Avoid static options and parameters. If parameters are unavoidable (e.g., maximum packet size), it is best to have the sender and receiver negotiate a value than defining fixed choices.

7. Look for a good design; it need not be perfect. Often the designers have a good design but it cannot handle some weird special case. Rather than messing up the design, the designers should go with the good design and put the burden of working around it on the people with the strange requirements.

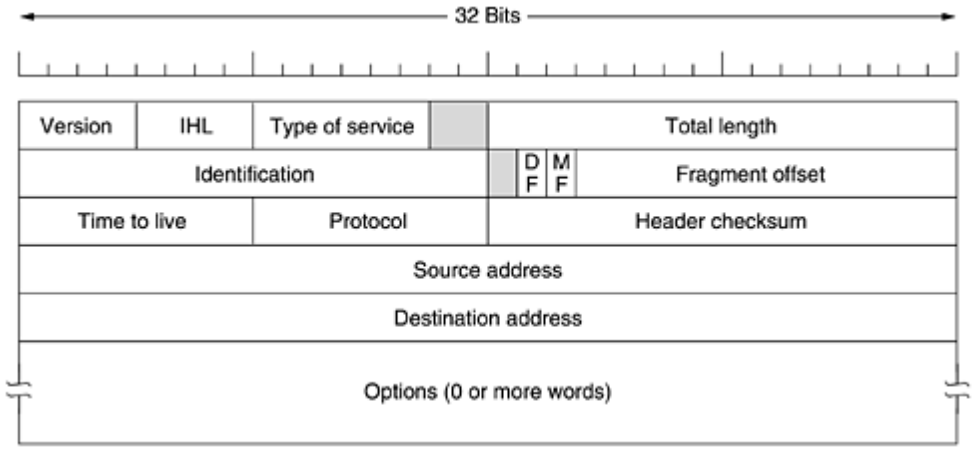
8. Be strict when sending and tolerant when receiving. In other words, only send packets that rigorously comply with the standards, but expect incoming packets that may not be fully conformant and try to deal with them.

9. Think about scalability. If the system is to handle millions of hosts and billions of users effectively, no centralized databases of any kind are tolerable and load must be spread as evenly as possible over the available resources.

10. Consider performance and cost. If a network has poor performance or outrageous costs, nobody will use it.

IP PROTOCOL

IP protocol is a protocol of network layer whose main objective is to support internetworking. An IP datagram consists of a header part and a text part. The header has a 20-byte fixed part and a variable length optional part. The header format is shown in Fig. 13.It is transmitted in big-endian order: from left to right, with the high-order bit of the Version field going first.



The IPv4 (Internet Protocol) header.

Version: Keeps track of which version of the protocol the datagram belongs to.

IHL: Tells how long the header is, in 32-bit words. The min and max values are 5 and 15 respectively, which are a multiple of 4.

Type of service: Allows the host to tell the subnet what kind of service it wants.

The field itself contains:

- 8-bit precedence field, where precedence field is a priority from 0-7.
- 8-flags D, T, R (Delay, Throughput, and Reliability)which is most cared parameter set.
- 2-bits, unused.

Total Length: The total length of both header and data maximum length is 65,535 bytes.

Identification: Allows the destination host to determine to which datagram a newly arrived fragment belongs.

DF(Don't Fragment): It is an order to routine not to fragment the datagram because the destination is incapable of putting the pieces back together again.

MF(More Fragments): All fragments except the last one have this bit set.

Fragment Offset: Tells where in the current datagram this fragment belongs.

Time to live: It is a counter used to limit packet lifetimes. It counts time in seconds, allowing a maximum lifetime of 255sec. It is decremented at each hop fill it reaches zero and then discarded.

Header Checksum: Verifies the header only.

Protocol: Tells the network layer to which transport process, the datagram is to be given. Eg :- TCP, UDP .... etc

SA: Indicates the network number and host number from which datagram has came from.

DA: Destination address and it indicates both the n/w number and host number to which destination, the datagram is to be delivered.

Options: This was designed to provide an escape to allow subsequent versions of protocol to include information not present in the original design, to permit experimenters to try out new ideas and to avoid allocating header bits to information that is rarely needed.

Option	Description
Security	Specifies how secret the datagram is
Strict source routing	Gives the complete path to be followed
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

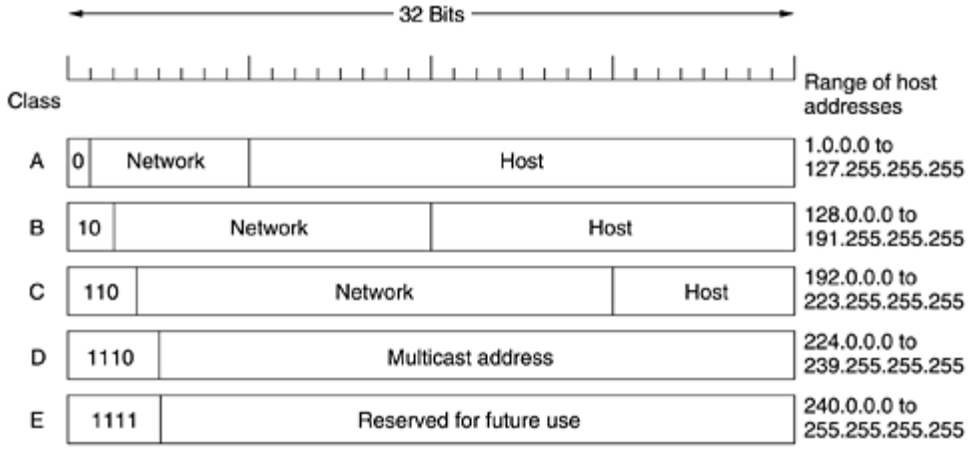
Some of the IP options.

IP ADDRESS:

IP address is required to identify any computer on the internet. Every host and router on the internet has an IP address, which includes its network number and host number. No two machines on the internet can have the same IP address. IP address is unique and all are 32- bit long and are used in the source address and destination address fields of IP packet.

To identify the network access the globe the network address is used. The host id is a unique address given to each host on the network.

IP protocol used 32-bit address for the network addressing depending on the applications the network address are classified in to 5 types class A, class B, class C, class D, and class E.



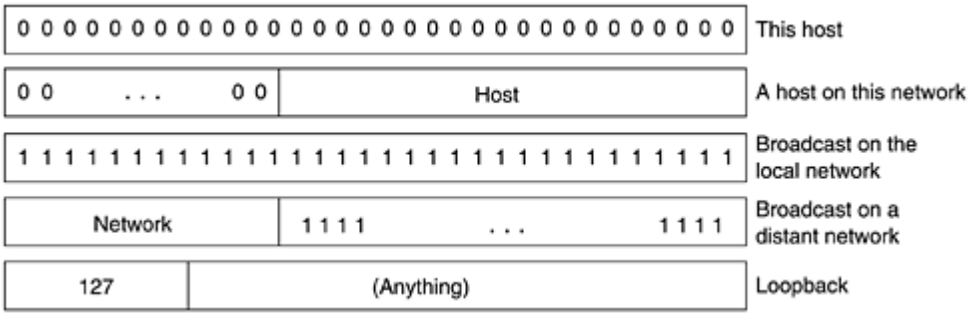
IP address formats.

1) Class A format allows up to 128 networks with 16 million hosts each
2) Class B format allows up to 16,384 networks with 64 k hosts
3) Class C format allows up to 2 million networks with 256 hosts
4) Class D format supports multicast, in which each datagram is directed to multiple hosts.
5) Class E format, address beginning with 1111 is reserved for future purpose.

Special IP addresses.

Multicast is a type of communication involving single source and multiple destination. Address starting with 1111 is reserved for future use. Network numbers are managed by NIC (network information center) and ICANN (internet corporation for assigned names and numbers) to avoid conflicts.

Each IP address is written in dotted decimal notation like 0.0.0.0 to 255.255.255.255 (lowest IP address to highest IP address)



Special IP addresses.

The IP address 0.0.0.0 is used by the hosts when they are being booted and IP address with '0' as the network number refers to the current network.

The IP address consisting all 1's access broadcasting on the local area network. The address with proper network number and all 1's in the host field allow machines to send broadcast packets to distant LAN's anywhere in the internet. And all address of the form 127.xx.yy.zz are reserved for loop back testing.

Subnets

- If a organization is large or its computers geographically distributed, it makes good sense to divide network into smaller ones, connected together by routers.
    - Reduced network traffic
    - Optimized network performance
    - Simplified network management
    - Facilities spanning large geographical distances.
- If **NIC** assign only one network address to an organization which have multiple network, that organization has a problem. A single network address can be used to refer to multiple physical networks. An organization can request individual network address for each one of its physical networks.
- If each router on the internet needed to know about each physical network, routing tables would be impossible huge. This is physical overhead on the router. To solve this type of problem, the subnet addressing method is used.
- To allow single network address to span multiple physical networks is called subnet addressing or sub netting or subnet routing.

- IP addresses are 32 bit long. 2 bytes of the address indicates a network ID and other 2 bytes indicates the host ID on the network.
- To reach host on the internet first network is reached by using first portion (n/w ID) of the address. After receiving the network then the host can be reached by using second portion (host ID) of the address.
- In classes A, B, C, IP addressing is designed with two levels of hierarchy. But, two level hierarchies are not enough; it is because we cannot have more than one physical network. The host cannot be organized into groups and all the hosts are at same level.
- If an organization needs to assemble the hosts into groups, the networks need to be divided into several sub networks (subnets).

Example:

A university wants to group its hosts according to department. Here the university has one network address, but need several sub network address for each department. In sub netting, a network is divided into several smaller groups with each sub network having its own sub network address.

CIDR—Classless InterDomain Routing

- Dividing the IP address space in to A, B and C classes turned out to be inflexible. IP is rapidly becoming a victim of its own popularity; it is running out of address. In 1993, the class full address space restriction was lifted. An arbitrary prefix length to indicate the network number, known as classless inter domain routing (CIDR), was adopted in place of the class full scheme.
- In CIDR network address, the network part of an IP address can be any number of bits long, rather than being constrained to 8, 16 or 24 bits.
- The CIDR, each routing address has the dotted-decimal from a.b.c.d/x, where x indicates the number of leading bits in 32-bit quality that constitutes the network portion of the address.
- In CIDR, each routing table entry is extended by giving it a 32 bit mask. The routing table for all networks constitutes of an array of IP address, subnet mask and outgoing line as triples.

When a packet comes in, its destination IP address is first extracted. Then the routing table is scanned entry-by-entry, masking the destination address and comparing it with the table entry for a match. It is possible that multiple entries match, in which case the longest mask is used.

Example:

NAT—Network Address Translation

The problem of running out of IP addresses is not a theoretical problem that might occur at some point in the distant future. It is happening right here and right now. The long-term solution is for the whole Internet to migrate to IPv6, which has 128-bit addresses. This transition is slowly occurring, but it will be years before the process is complete. As a consequence, some people felt that a quick fix was needed for the short term. This quick fix came in the form of **NAT** (**Network Address Translation**).

The basic idea behind NAT is to assign each company a single IP address (or at most, a small number of them) for Internet traffic. Every computer inside the company or organization gets a unique IP address, which is used for routing traffic. Companies internally uses three ranges of IP addresses have been declared as private. The three reserved ranges are:

10.0.0.0 – 10.255.255.255/8 (16,777,216 hosts)

172.16.0.0 – 172.31.255.255/12 (1,048,576 hosts)

192.168.0.0 – 192.168.255.255/16 (65,536 hosts)

The rule is that no packets contain three addresses must appear on the internet.

Operation of NAT:

The operation of the NAT is discussed step by step

Step 1: Within the company premises, every machine has a unique address of the form 10.x.y.z. However, when a packet leaves the company premises, it passes through a NAT box that converts the internal IP source address to true IP address.

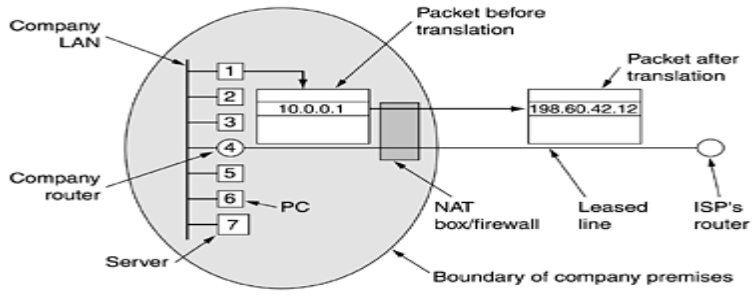
Internal IP source address 10.0.0.2 is converted to the company's true IP addresses 192.50.12.1. NAT is combined and placed with the firewall, which provides security by controlling the information going out and coming in.

Step 2: when the reply comes back from outside the company premise, it is naturally addressed to 192.50.12.1. so NAT has to know which address to replace it with. Nat designers observed that most IP packets carry either TCP or UDP used port for holding the connection. These ports are used to make NAT work.

Step 3: Each outgoing TCP/UDP message contains both a source port and destination port. Ports are used to identify the process during the connection on both ends.

Mapping between the internal IP source address and true IP address are done by using source port field. Whenever an outgoing packet enters the NAT box, the internal IP address 10.x.y.z source address is replace by the company true address. TCP source port field is replaced by an index into NAT box's translation table. This translation table can hold 65,536 entries and contains the original IP address and original source port. Finally, both the TCP and IP headers checksums are recomputed and inserted into the packet.

Step 4: when the packet arrive at the NAT box from the ISP, the source port in the TCP header is extracted. This port is used as index into NAT box mapping table. Entry is located in the internal IP address and TCP source port is extracted and inserted into the packet. Then the both TCP and IP checksums are recomputed and inserted into packet. The packet is then passed to the company router for normal delivery using 10.x.y.z address.



Placement and operation of a NAT box.

Limitations of NAT:

1) NAT violates the architectural model of IP, which states that every IP address uniquely identifies a single machine worldwide.
2) Nat changes the internet from a connection less network to a kind of connection oriented network. If Nat box crashes, it loses mapping table and all its top connections are destroyed.
3) NAT violates the most fundamental rule of protocol layering ie.., layer to be kept independent. But if TCP ports are upgraded to 32 bit ports, then NAT will fail.
4) Some applications insert IP address in the body of the text. The receiver then extracts this address and uses them. NAT does not know about these addresses, it cannot replace them, so Nat will fail.
5) On the internet if new protocol is used other than TCP or UDP, NAT will fail. Because NAT box will not able to locate TCP source port correctly.
6) TCP source port field is 16 bits; at most 65,536 machines can be mapped into an IP address. But first 4096 ports are reserved for special purpose. Only 61,440 machines can be mapped.

IPv6

IPv4 provides the host to host communication between systems in the Internet. IPv4 has played a central role in the internetworking environment for many years. It has provided flexible enough to work on many different technologies.

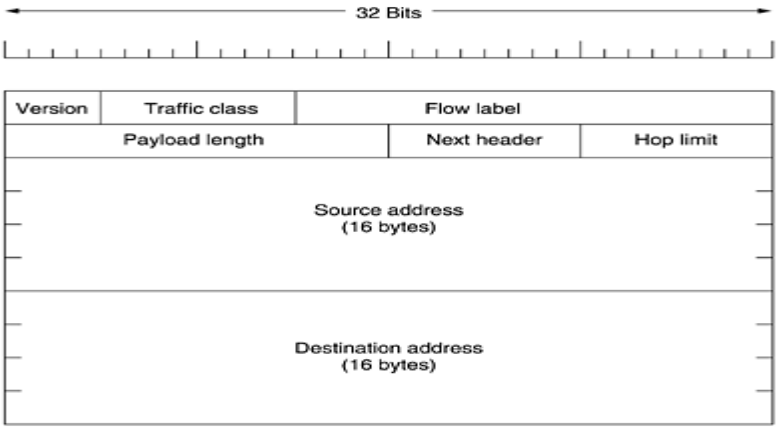
In the early 1990 the IETF (internet engineering task force) began to work on the successor of IPv4 that would solve the address exhaustion problem and other scalability problems. this was because the 32-bit IP address was being used up, with the new networks and IP nodes being added to the internet at very high rate. To catch this need a large IP address of 128- bit address IPv6 was developed.

IETF started work on a new version of IP, Its major goals were:

1. Support billions of hosts, even with inefficient address space allocation.
2. Reduce the size of the routing tables.
3. Simplify the protocol, to allow routers to process packets faster.
4. Provide better security (authentication and privacy) than current IP.
5. Pay more attention to type of service, particularly for real-time data.
6. Aid multicasting by allowing scopes to be specified.
7. Make it possible for a host to roam without changing its address.
8. Allow the protocol to evolve in the future.
9. Permit the old and new protocols to coexist for years.

The Main IPv6 Header

The format of IPv6 header is



Version (4 bits): internet protocol version number is 6. IP v4 is always 4

Priority (4 bits): it is used to distinguish between packets whose sources can be flow controlled and those that cannot. (0…7 = for transmission that are capable of slowing down in the event of congestion while 8…15 = real time traffic whose sending rate is constant)

Flow label (20 bits): may be used by a host to label those packets for which it is requesting special handling by routers with in a network.

Payload length (16-bits): length of remainder of IPV6 packet following the header in octets.

Next header: it tells which transport protocol handler (Eg: TCP, UDP) to pass the packet to.

Hop limit: it is used to keep packets from living for ever.

Source address: tells the address of the source from which the data is coming

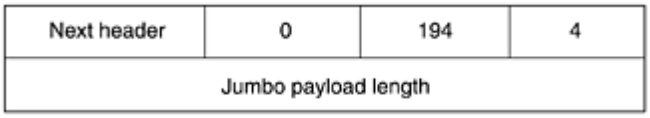
Destination address: tells the address of the destination to where it should go.

Extension Headers:

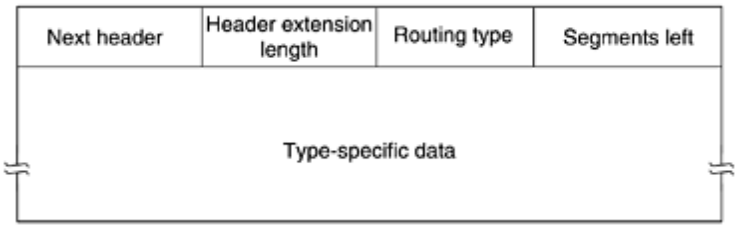
IPv6 has introduced the concept of extension headers. These headers can be supplied to provide extra information, but encoded in an efficient way. There are six types of extension headers. They are as follows:

Extension header	Description
Hop-by-hop options	Miscellaneous information for routers
Destination options	Additional information for the destination
Routing	Loose list of routers to visit
Fragmentation	Management of datagram fragments
Authentication	Verification of the sender's identity
Encrypted security payload	Information about the encrypted contents

Hop-by –hop options header: It is used for information that all routers along the path must examine. Data grams using this header extension are called **JUMBOGRAMS**. So far, only one option has been defined i.e., support of data grams exceeding 64k. The format is represented as follows:



Routing Header: It lists one or more routers that must be visited on the way to the destination. Both strict routing and loose routing are available, but they are combined. The format can be represented as:



Fragmentation header: it deals with fragmentation. The header holds the datagram identifier, fragment number, and a bit telling whether more fragments will follow.

Authentication header: it provides a mechanism by which the receiver of a packet can be sure of who sent it.

Encrypted security payload header: it is used for packets that must be sent secretly.

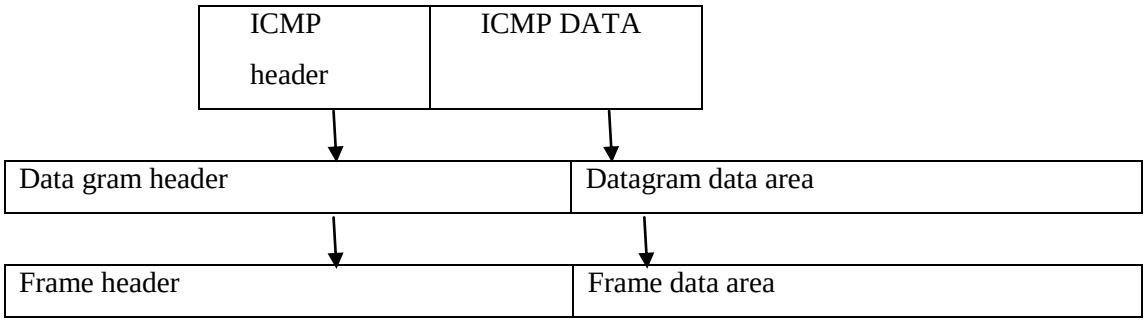
Destination options header: it is intended for fields that need only be interpreted at the destination host.

INTERNET CONTROL PROTOCOLS:

Internet has several protocols used in the network layer, such as ICMp, ARP, RARP, DHCP and BOOTP.

ICMP protocol:

- ICMP stands for Internet Control Message Protocol. The ICMP allows routers to send error or control messages to other routers or hosts. ICMP provides communication between the internet protocol software on one machine and the internet protocol software on another machine.
- Technically ICMP is an **error reporting mechanism**. It provides a way for routers that encounter an error report the error to the original source. ICMP messages require two level of encapsulation. Each ICMP message travels across the internet in the data portion of an IP datagram, which itself travels across each physical network in the data portion of a frame.
- Each ICMP has own format



Two levels of ICMP encapsulation

- ICMP sends messages by encapsulating them in IP packets and setting the header's protocol field to 1. **The sole function of the ICMP is to report problems, not to correct them**. Responsibility for error correction lies with the sender. ICMP can send messages to on to source, not to an intermediate router. Because the datagram carries only the address of the original sender and the final destination. It does not know the address of the previous routers that passed it along.

ICMP message format

Type	Code	checksum
unused		
IP header and original datagram(64 bits)		

Type: size of the field is 8-bit. Type field is used to identify the ICMP message type

Code: size of the code is 8-bit. It provides the information or parameter of the message type.

Check sum: size is 16-bit. Check sum is entire ICMP message

ICMP type filed defines of the message as well as its format. The type field includes.

Type field	ICMP message type
0	Echo reply
3	Destination unreachable
4	Source quench
5	Redirect
8	Echo request
11	Time exceeded
12	Parameter problem
13	Timestamp request
14	Time stamp replay
15	Information request
16	Information replay
17	Address mask request
18	Address mask replay

- The DESTINATION UNREACHABLE message is used when the subnet or a router cannot locate the destination or when a packet with the DF bit cannot be delivered because a "small packet" network stands in the way.
- The TIME EXCEEDED message is sent when a packet is dropped because its counter has reached zero. This event is a symptom that packets are looping, that there is enormous congestion, or that the timer values are being set too low.
- The PARAMETER PROBLEM message indicates that an illegal value has been detected in a header field. This problem indicates a bug in the sending host's IP software or possibly in the software of a router transited.
- The SOURCE QUENCH message was formerly used to throttle hosts that were sending too many packets. When a host received this message, it was expected to slow down.
- The REDIRECT message is used when a router notices that a packet seems to be routed wrong. It is used by the router to tell the sending host about the probable error.
- The ECHO and ECHO REPLY messages are used to see if a given destination is reachable and alive. Upon receiving the ECHO message, the destination is expected to send an ECHO REPLY message back.
- The TIMESTAMP REQUEST and TIMESTAMP REPLY messages are similar, except that the arrival time of the message and the departure time of the reply are recorded in the reply. This facility is used to measure network performance.

ARP (Address Resolution Protocol)

The hosts or routers are recognized at network layer by IP address. An IP address is an internetwork address. However to identify a host irrespective of network layer protocol, we need another address called MAC address.

There is no dependence between local address and IP address, the only method is establishing the mapping using tables. As a result of network configuration, each interface knows its local address and IP address. This mapping can be considered as a table distributed over individual network interfaces.

Any time a router or hosts has an IP datagram to send another host or a router. It has the logical IP address of the receiver. But the IP datagram has encapsulated in a frame to be able to pass through physical network. This means the sender needs the physical address of the receiver. A mapping corresponds a logical address to physical address.

The mapping can be done dynamically, which means that the sender asks the receiver to announce its physical address when needed. ARP is used for this purpose.

ARP associates an IP address with its physical address. Anytime a host or a router needs to find physical address of another host or a router on its network, it sends an ARP query packet. The packet includes the physical address and IP address of the sender and the IP address of the sender.

Consider the following example computer A and computer B shares a physical network. Each computer has an assigned IP address IA and IB physical address PA and PB. The goal is to device low level software that hides physical address and allows higher level programs to work only with internet address.

RARP, BOOTP, and DHCP

Reverse address resolution protocol, bootstrap protocols are no where used now a days. They are replaced by DHCP (dynamic host configuration protocol). RARP assigns IP address to a known MAC address. Suppose a diskless workstation is booted. It gets its binary version of OS from remote system. In such cases, the work station broadcasts its MAC address. RARP server sees this MAC address and find its IP address from local configuration files and gives reply. However IP address can be coded into boot image itself.

RARP demands the presence of RARP server on each network. Such RARP requests cannot be broadcasted onto other LAN's. to avoid this problem, BOOTP was designed. RARP uses Ethernet broadcast messages, where as BOOTP and UDP messages which can be forwarded over routers. A serious problem with BOOTP is, it used manual configuration of tasks mapping IP address to MAC address.

Whenever new host is added, it cannot use BOOTP until administrator assigns IP address to it and enters entry into BOOTP configuration tasks. To avoid it, BOOTP was extended to DHCP. DHCP supports both manual and automatic assignment. DHCP will have DHCP server, one for LANs and a DHCP relay agent one per LAN.

OSPF

The Interior Gateway Routing protocol: OSPF (Open Shortest Path First)

The Internet is made up of a large number of Autonomous systems, in which each as is operated by a different organization and can use its own routing algorithm inside. A Routing algorithm with in an Autonomous system is called an **Interior Gateway Protocol.**

The distance vector routing is also an interior gateway protocol, which is based on the bellman-ford algorithm, but works on the smaller autonomous systems for large autonomous systems we go for OSPF.

The OSPF is a link state routing protocol. OSPF is based on the distributed map concept all nodes have a copy of the network map, which is regularly updated. The OSPF computes the shortest path to the other routers. OSPF protocol is now widely used as the interior router protocol in TCP/IP networks.

OSPF is classified as an internal gateway protocol because it supports routing with one autonomous system only. The exchange between the autonomous systems is the responsibility of another protocol an external gateway protocol. OSPF supports one or many networks.

Design goals and requirements of OSPF as follows:

The algorithm is published in OPEN literature, and hence "**O**" in '**OSPF**'.

1. This protocol supports a variety of distance metrics, including physical distance, delay….etc...
2. It is a dynamic algorithm, which adapts to changes in the topology automatically and quickly.
3. It supports routing based on type of service.
4. It does load balancing by splitting the load over multiple lines.
5. It provides support for hierarchical systems.
6. It provides medium of security and authentication of routing messages are necessary because if false messages reach network may breakdown.
7. It provides a way to deal with routers that were connected to Internet via a tunnel.

OSPF supports 3 kinds of connections and networks:

1) Point-to-Point lines between exactly 2 routers

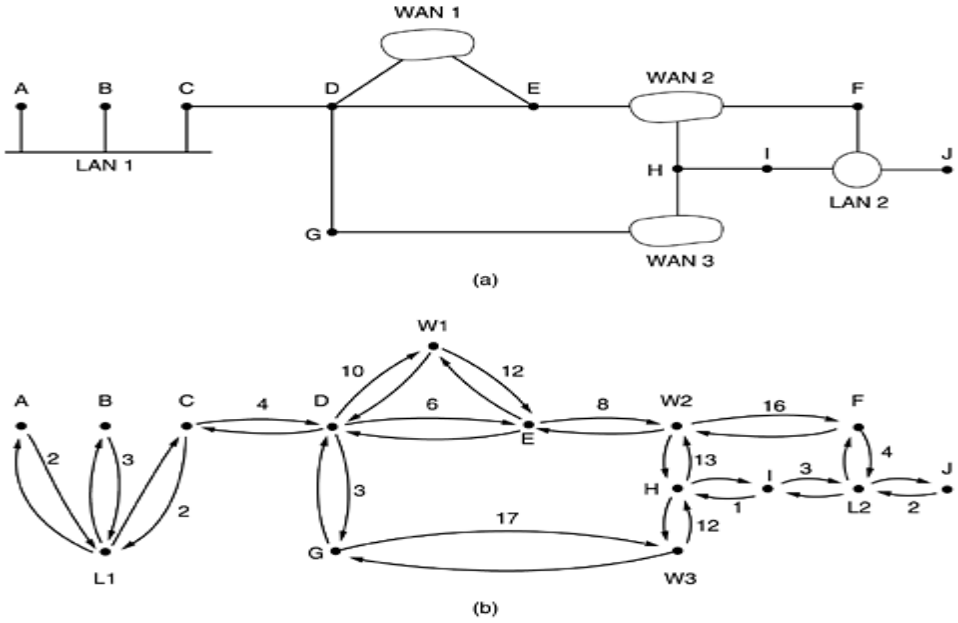
2) Multi-access networks with broadcasting (**Eg:** most LANS)

3) Multi-access networks without broadcasting (**Eg:** most Packet Switched WANS)

Multi-access network: It is a n/w that can have multiple routers on it, each of which can directly communicate with all the others

OSPF works by abstracting the collection of actual networks, routers and lines into a directed graph in which each arc is assigned a cost (distance, delay etc). It then computes the shortest path based on the weights on the arcs. A serial connection between two routers is represented by a pair of arcs, one in each direction. Their weights may be different. A multi-access network is represented by a node for the n/w itself plus a node for each router. The arcs from n/w node to the routers have

weight '0' and are omitted from the graph. What OSPF fundamentally does is represent the actual n/w like a graph and to compute the shortest path from every router to every other router.



(a) An autonomous system. (b) A graph representation of (a).

An Autonomous System (AS) in the internet is large and so is divided up into numbered areas, where an area is an n/w or a set of contiguous n/w & and is a generalization of a subnet. Every 'AS' has a backbone area, called area-'0'.All areas are connected to the backbone, possibly by Tunnels (represented as an arc in the graph).The topology of areas and backbone are not visible outside the backbone.

With in an area, each router has the same link state database and runs the same shortest path algorithm. Its main job is to calculate the shortest path from itself to every other router in the area, including the router that is connected to the backbone, of which there must be at least one. The OSPF handles the service routing type by having 3 multiple graphs labeled using different metrics accordingly. So that separate routes are allowed for optimizing delay, throughput and reliability.

During Normal Operation, 3 kinds of routes may be needed:-

• **Intra-area:** These are easiest since the source router already knows the shortest path to the destination router.

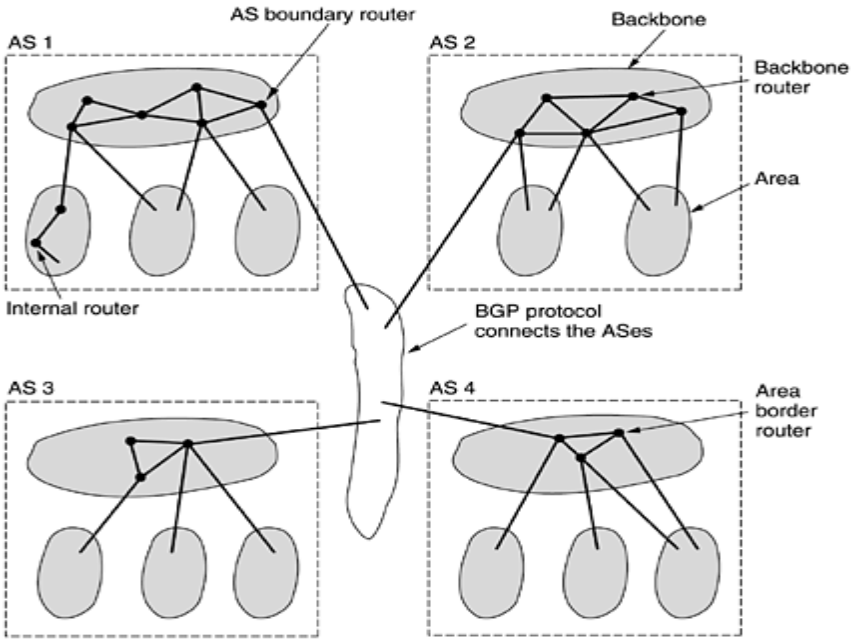
• **Inter-area:** Always proceeds in 3 steps:-

(a) Go from source to the backbone.

(b) Go across the backbone to destination area.

(c) Go to the destination

• **Inter-AS:** Packets are routed from source to destination being encapsulated or tunneled when going to an area whose only destination to the backbone is a tunnel.



The relation between ASes, backbones, and areas in OSPF.

OSPF distinguishes 4 classes of routers:

1) **Internal Routers:** These are wholly with in one area.

2) **Area Border Routers:** These connect 2 or more areas

3) **Backbone Routers:** These are on the backbone

4) **AS Boundary Routers:** These talks to routers in other autonomous systems.

When a router boots, it sends **HELLO** messages on all of its point-to-point lines and multicasts then on LAN's to the group consisting of all other routers. OSPF works by exchanging information between **adjacent routers**. One router is elected as the **Designated Router** and it is said to be adjacent to all the other routers and exchanges information with them.

During normal operation, each router periodically floods **LINK STATE UPDATE** messages along with sequence numbers to each of its adjacent routers by which it can differentiate new from the old ones. This message gives its state and provides the costs used in topological database. Used when a line goes up/down. Each partner can request link state information from the other one using **LINK STATE REQUEST** messages, by which each pair of adjacent routers check to see who has the most recent data. All these messages are sent as row IP packets. **DATA BASE DESCRIPTION** messages give the sequence numbers of all the link state entries currently held by the sender. Used when a line is brought up.

Message type	Description
Hello	Used to discover who the neighbors are
Link state update	Provides the sender's costs to its neighbors
Link state ack	Acknowledges link state update
Database description	Announces which updates the sender has
Link state request	Requests information from the partner

The five types of OSPF messages.

Using Flooding, each router informs all the other routers in its area of its neighbors and costs. This information allows each router to construct the graph for its areas and compute the shortest path. In addition, the backbone routers accept information from area border routers in order to compute the best route from each backbone router other router. This information is propagated back to the area border routers, which advertise it within these areas. Using this information, a router about to send an inter area packet, can select the best exit router to the backbone.

The Exterior Gateway Routing Protocol: (BGP-Border Gateway protocol)

An autonomous system used for routing across autonomous systems is called "exterior gateway protocol". An algorithm for routing between ASes is called an **Exterior Gateway Routing Protocol**. It moves packets as efficiently as possible from source to destination, using different policies, examples are:-

1) No transit traffic through certain ASes.

2) Never put Iraq on a route starting at pentagon.

3) Traffic starting or ending at IBM should not transit MICROSOFT.

4) Policies are manually configured into each BGP router 2 BGP routers are said to be connected if they share a common n/w.

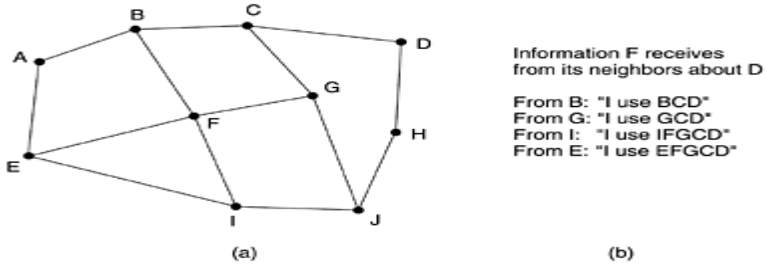
Networks are grouped into 3 categories depending on BGP traffic transit they are:-

Stub autonomous system: An autonomous system that has only a single connection to one another autonomous system is called stub autonomous system. Such autonomous system will carry only local traffic. "Local traffic" is defined as traffic that originates at or terminates on nodes with in an autonomous system; transit traffic is defined as traffic that passes through and autonomous system.

Multi-homed autonomous system: An autonomous system that has connections to more than one another autonomous system but refuses to carry transit traffic.

Transit Networks: autonomous system that has connections to more than one other autonomous system and that is designed to carry both transit and local traffic.

The main goal of inter domain routing is to find a path to the destination which is loop free ie.., we are concerned with only reach ability than optimality. There will be at least one node in each autonomous system, which is called as "BGP speaker", who is spokes person to the entire autonomous system. BGP speakers establish BGP sessions to other BGP speakers in other autonomous systems. These sessions are used to exchange reach ability among autonomous systems.



***(a) A set of BGP routers. (b) Information sent to* F.**

Pairs of BGP routers communicate with each other by establishing TCP connections so that reliable communication is provided and all n/w details are hided. BGP is fundamentally a distance vector protocol, which keeps track of the exact path used besides maintaining cost to each destination and it related to each of its neighbors.

From the above figure, consider F's routing table. Suppose that it uses the path FGCD to get to D. When the neighbors give it routing information, they provide their complete paths as from which the best path is chosen by 'F'. After seeing the neighbors information, 'F' discards the paths from I and E as they pass through 'F' itself. Any one of the paths from B or G is selected.

Neighbor	Route used
B	BCD
E	EFGCD
G	GCD
I	IFGCD

The selection is done in this way:

Every BGP router contains a module that examines routes to a given destination and scores them, returning a number for the "distance" to that distance for each route. Any route violating a policy constraint automatically gets a score of "$\propto$ (infinity)". The router then adopts the route with the shortest distance.

BGP easily solves even the count-to-infinity problem in this way:

Suppose, 'G' crashes or the line 'FG' goes down 'F' receives routes from 3 remaining neighbors, which are BCD, IFGCD, EFGCD respectively from B, I, E from which FBCD chosen as its new route (IFGCD, EFGCD pass through 'F' itself and so neglected.)

INTERNET MULTICASTING:

Communication between a single sender and a single receiver is called unicast. When it involves a single source and multiple destinations, it is called multicasting. IP supports Multicasting using class-D addresses. Each class-D address identifies a group of hosts. 28 bits are available for identifying groups, so over 250 million groups can exist at the same time. When a process sends a packet to a class-D address, best efforts of the group addressed, but no guarantees are given. Some members may not receive.

Example: video conferencing, online auction, online banking and soon….

Two kinds of group addresses are supported:

1. Permanent Addresses

2. Temporary Addresses

1. Permanent Addresses: They are always there and don't have to be set up.

Some examples: 224.0.0.1 -- All systems on a LAN.

222.0.0.2 -- All OSPF Routers on a LAN.

224.0.0.5 -- All OSPF Routers on a LAN.

224.0.0.6 -- All designed OSPF routers on a LAN.

2. Temporary Addresses: They must be created before they can be used. A process can ask its host to join a specific group or to leave the group. When the last process on a host leaves a group, that group is no longer present on the host.

About once a minute, each multicast router sends a hardware (**Eg:** DLL) multicast to the hosts on its LAN (284.0.0.1 address) asking them to report back on the groups their processes currently belong to. Each host sends back responses for all the class-D addresses it is interested in. These query and response packets use a protocol called **IGMP (Internet Group Management Protocol).**

Multicast Routing is done using **SPANNING TREES**. Each multicast router exchanges information with its neighbors using a modified distance vector protocol in order for each one to construct a spanning tree per group covering all group members. Various optimizations are used to prune tree to eliminate routers and networks not interested in particular groups. The protocol makes heavy use of tunneling to avoid bothering nodes not in a spanning tree.

UNIT IV

The Transport Layer: The Transport Service, Elements of Transport Protocols, Congestion control, **The Internet Transport Protocols**: UDP, **The Internet Transport Protocols**: TCP, Performance problems in computer networks, Network performance measurement

INTRODUCTION

The transport layer is above the network layer and provides services to the upper layers. The task of the transport layer is to provide reliable, cost-effective data transport from the source machine to the destination machine and is responsible for establishing and releasing connections and error control functions.

4.1 THE TRANSPORT SERVICE:

These services are provides to the application layer. There are two set of primitives provided by transport layer. These services are provided in the form of interface commonly used in internet.

The following services are the transport services,

1) Services provided to the upper layers.
2) Transport service primitives
3) Berkeley sockets.

Services provided to the upper layers

The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective service users, normally processes in the application layer. To achieve this goal, the transport layer make of the services provided by the network layer. The hardware and/or software within the transport that does the work is called the transport entity. The transport entity can be located in the open system kernel, in a separate user process, in a library package bound into network applications, conceivably on the network interface card. The relationship of the network, transport, a application layers is illustrated in Fig.1.

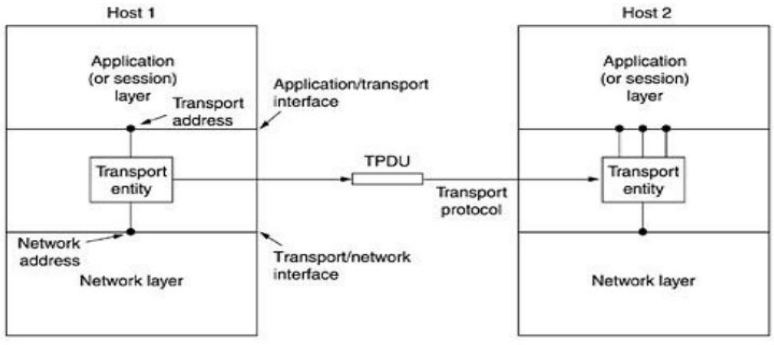


Fig. The network, transport, and application layers

TPDU (Transport Protocol Data Unit): Transmissions of message between 2 transport entities are carried out by TPDU. The transport entity carries out the transport service primitives by blocking the caller and sending a packet the service. Encapsulated in the payload of this packet is a transport layer message for the server's transport entity.

Transport Entity: The hardware and/or software which makes use of services provided by the N/W layer, (with in the transport layer) is called transport entity.

Transport Service Provider: Layers 1 to 4 are called Transport Service Provider.

Transport Service User: The upper layers i.e., layers 5to 7 are called Transport Service User.

Transport Service Primitives: Which allow transport users (application programs) to access the transport service.

There are two types of transport services similar to network services

1) *Connection oriented transport service*
2) *Connection less transport service*

Connections have three phases

1) *Connection establishment*
2) *Data transfer*
3) *Connection release*

Transport service primitives

Transport service primitives are used by the users to access the transport service. The transport layer must provide some operations to application programs, that is, a transport service interface. Each transport service has its own interface.

The main difference is that the network service is intended to model the service offered by real networks, warts and all. Real networks can lose packets, so the network service is generally unreliable. The (connection-oriented) transport service, in contrast, is reliable.

Let us consider some primitives for simple transport service. Transport service is connection oriented and allows application to establish, use and then release connections.

Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

The primitives for a simple transport service

Eg: Consider an application with a server and a number of remote clients.

1. The server executes a **"LISTEN"** primitive by calling a library procedure that makes a system call to block the server until a client turns up.

2. when a client wants to talk to the server, it executes a **"CONNECT"** primitive, with **"CONNECTION REQUEST" TPDU** sent to the server.

3. When it arrives, the TE unblocks the server and sends a **"CONNECTION ACCEPTED"** TPDU back to the client.

4. When it arrives, the client is unblocked and the connection is established. Data can now be exchanged using "SEND" and "RECEIVE" primitives.

5. When a connection is no longer needed, it must be released to free up table space within the 2 transport entries, which is done with "DISCONNECT" primitive by sending "DISCONNECTION REQUEST" TPDU. This disconnection can b done either by asymmetric variant (connection is released, depending on other one) or by symmetric variant (connection is released, independent of other one).

TPDU (**Transport Protocol Data Unit**) for messages sent from transport entity to transport entity. Each TPDU consists of general four fields,

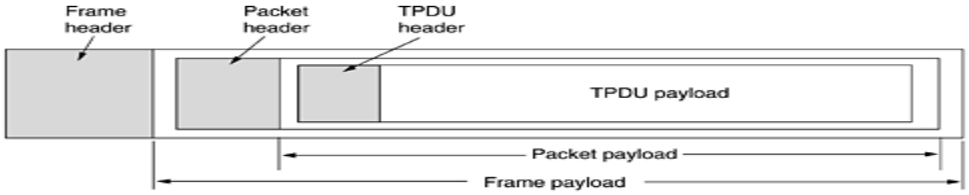
Length	Fixed parameter	Variable parameter	data

Length: occupies first byte and indicates total number of bytes in the TPDU

Fixed parameters: it contains parameter or control fields that are commonly present in all transport layer packets. It consists of code, source reference, sequence number and credit allocation.

Variable parameter: it contains parameters which occur frequently and are used for management of control codes.

Data: contain regular data coming from the upper layer each layer will append their headers. At transport layer, TPDUs messages are exchanged. TPDUs are embedded within the packet and packets are embedded within the frame at the data link layer.. this forms the nesting of TPDUs packets and frames.



Nesting of TPDUs, packets, and frames.

Berkeley Sockets: These primitives are socket primitives used in Berkely UNIX for TCP.

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

The first four primitives in the list are executed in that order by servers.

The **SOCKET** primitive creates a new end point and allocates table space for it within the transport entity. The parameters of the call specify the addressing format to be used, the type of service desired (e.g., reliable byte stream), and the protocol. A successful SOCKET call returns an ordinary file descriptor for use in succeeding calls, the same way an OPEN call does.

The **BIND** primitive assigns a name to unnamed socket. It is used for assigning address to the sockets that are newly created.

The **LISTEN** call, which allocates space to queue incoming calls for the case that several clients try to connect at the same time.

The **LISTEN** primitive is executed by the connection oriented server, an actual connection from some client process is waiting to have the server execute an **ACCEPT** primitive.

CONNECT primitive is used to establish a connection with the server

SEND and **RECEIVE** primitives are similar **READ** and **WRITE** primitives that they can be used to transmit and receive data using the full-duplex communication channel

CLOSE primitive is used to close the socket.

4.2 Elements of Transport Protocols:

The transport service is implemented by a transport protocol between the 2 transport entities. The elements are:

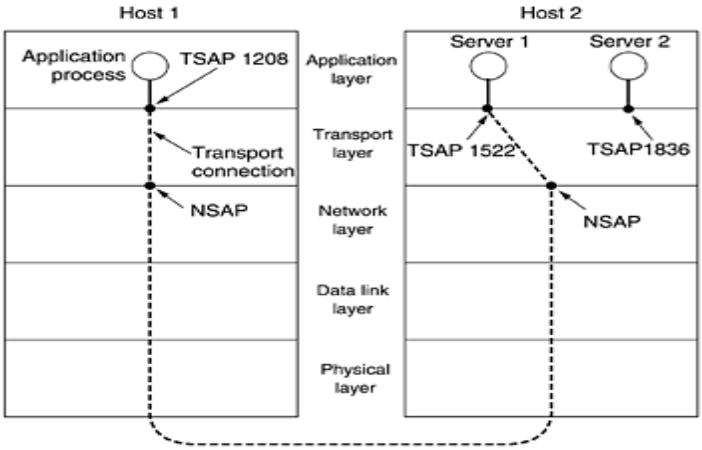
- ***Addressing***
- ***Connection establishment***
- ***Connection release***
- ***Flow control and buffering***
- ***multiplexing***
- ***Crash recovery***

A) ADDERSSING:

When an application process wishes to set up a connection to a remote application process, it must specify which one to connect it to, i.e., TSAPs and NSAPs must be specified by the transport layer and n/w layer respectively.

TSAP-----Transport Service Access Point

NSAP---- Network Service Access Point



TSAPs, NSAPs, and transport connections.

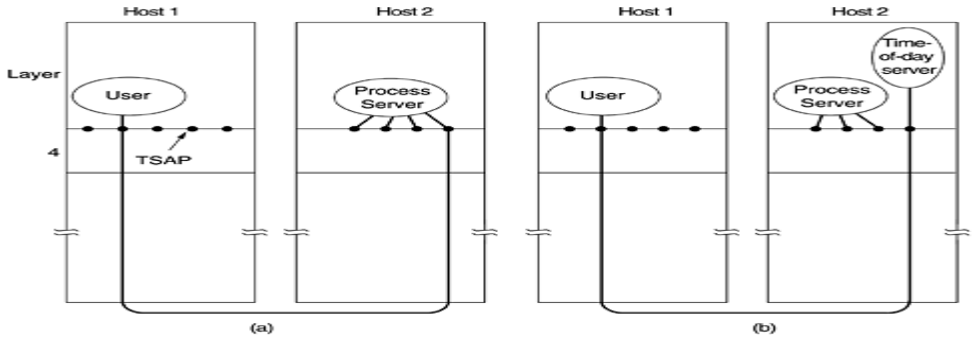
Application processes, both clients and servers, can attach themselves to a TSAP to establish a connection to a remote TSAP. These connections run through NSAPs on each host, as shown. The purpose of having TSAPs is that in some networks, each computer has a single NSAP, so some way is needed to distinguish multiple transport end points that share that NSAP.

A possible scenario for a transport connection is as follows.

1. A time of day server process on host 2 attaches itself to TSAP 1522 to wait for an incoming call. How a process attaches itself to a TSAP is outside the networking model and depends entirely on the local operating system. A call such as our LISTEN might be used, for example.
2. An application process on host 1 wants to find out the time-of-day, so it issues a CONNECT request specifying TSAP 1208 as the source and TSAP 1522 as the destination. This action ultimately results in a transport connection being established between the application process on host 1 and server 1 on host 2.
3. The application process then sends over a request for the time.
4. The time server process responds with the current time.
5. The transport connection is then released.

B) Connection establishment:

Establishing a connection sounds easy, but it is actually surprisingly tricky. The problem occurs when the network can lose, store, and duplicate packets. When there is heavy congestion on the subnet, the acknowledgment will not get back in time. Due to this delay, the packets are retransmitted two or three times. After some time the original packets may arrive at destination following different route. This mislead to generation of duplicate packets. These duplicate packets create lot of problem and confusion in real time applications.



(A) a user process in host 1 establishes a connection with a time-of-day server in host 2.

(B) Process server spawns requested server with user process

With packet lifetimes bounded, it is possible to devise a fool proof way to establish Connections safely. Packet lifetime can be bounded to a known maximum using one of the Following techniques:

- Restricted subnet design
- Putting a hop counter in each packet
- Time stamping in each packet

Restricted subnet design

The first method includes any method that prevents packets from looping, combined with some way of bounding congestion delay over the (now known) longest possible path.

Putting a hop counter in each packet

The second method consists of having the hop count initialized to some appropriate value and decremented each time the packet is forwarded. When packet hop count becomes zero, packets are discarded.

Time stamping in each packet

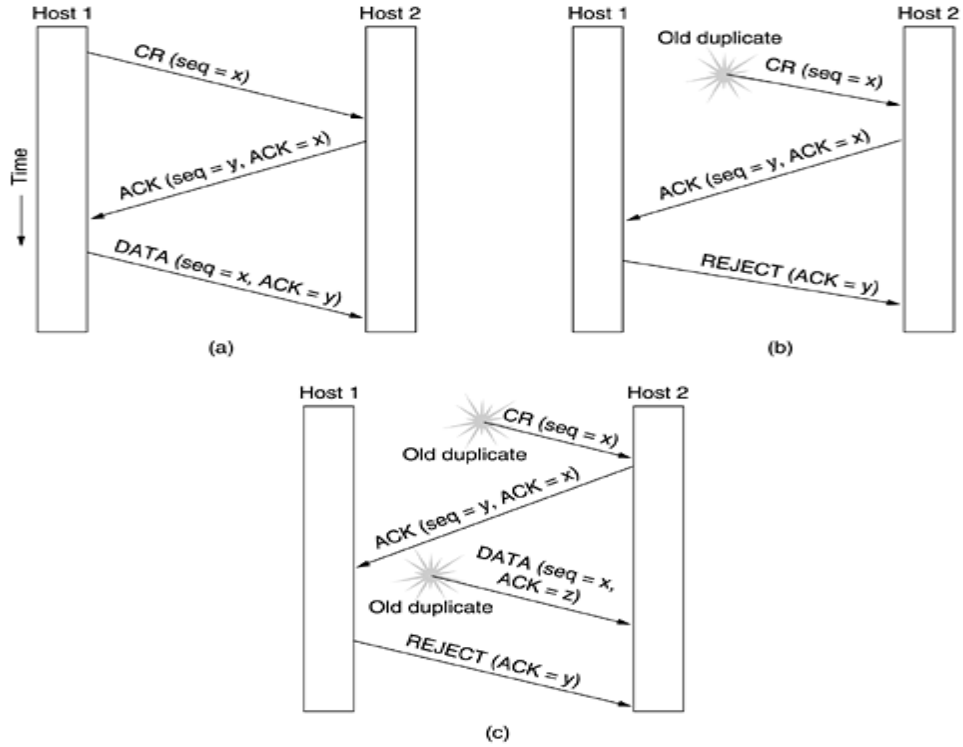
The third method requires each packet to bear the time it was created, with the routers agreeing to discard any packet older than some agreed-upon time. This method requires clocks to be synchronized because this method works on time factor

Three way handshake

There are many problems, before establishing a connection. After a connection is established, the clock based method solves the delayed duplicate problem, for data TPDUs are delayed, and then there is potential problem in getting both sides to agree on the initial sequence number. So, connection will not be established if control TPDUs are not exchanged properly. To overcome this problem Tomlinson introduced three way handshakes in 1975.

Using a 3-way hand shake, a connection can be established. This establishment protocol doesn't require both sides to begin sending with the same sequence number.

The three protocol scenarios for establishing a connection using three way handshake is explained three cases,



Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNECTION REQUEST. (a) Normal operation. (b) Old duplicate CONNECTION REQUEST appearing out of nowhere. (c) Duplicate CONNECTION REQUEST and duplicate ACK.

Case 1: Normal operation

Host-1 chooses a sequence number, 'x' and sends a CONNECTION REQUEST TPDU containing it to host-2. host-2 replies with a CONNECTION ACCEPTED TPDU acknowledging 'x' and announcing its own initial sequence number 'y'. finally, host-1 acknowledges host-2's choice of an initial sequence number in the first data TPDU that it sends.

Case 2: Delayed duplicate Connection_Request TPDUs

The first TPDU is a delayed duplicate CONNECTION REQUEST from an old connection. This TPDU arrives at host-2 without host-1's knowledge. Host-2 reacts to this TPDU by sending host-1 a CONNECTION ACCEPTED TPDU, in effect asking for verification that host-1 was indeed trying to set up a new connection. When host-1 rejects host-2's attempt to establish, host-2 realizes that it was tricked by a delayed duplicate and abandons the connection.

Case 2: duplicate Connection_Request and duplicate ACK

Host-2 gets a delayed CONNECTION REQUEST and replies to it. At this point, it is crucial to realize that host-2 has proposed using 'y' as the initial sequence number for host-2 to host-1 traffic, knowing that no TPDUs containing sequence number 'y' or acknowledgements to 'y' are still in existence. When the second delayed TPDU arrives at host-2, the fact that 'z' has been acknowledged rather than y tells host-2 that this ,too, is an old duplicate.

C) Connection release:

Connection release is easier than establishing connection. The connections are released in two ways. A connection is released using either asymmetric or symmetric variant.

Asymmetric release

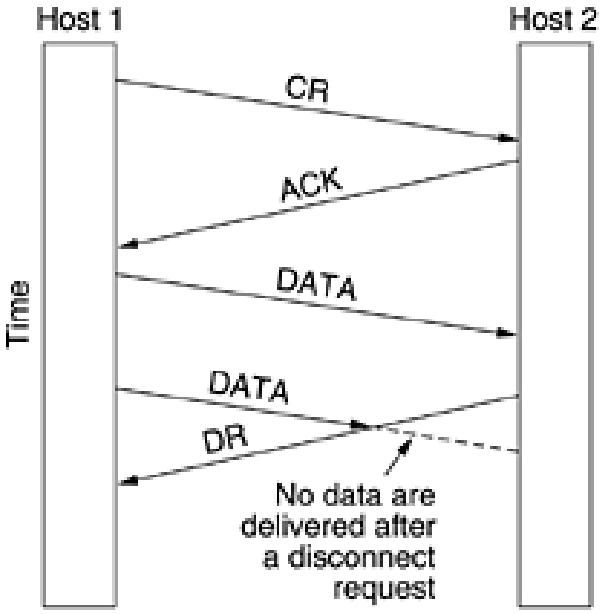
In asymmetric release, either side can release connection or there may be chances of loosing the data

Symmetric release

In symmetric release, the connection is treated as two separate unidirectional connections and requires either side to be released separately. If user on one side releases connection, but still it has to wait for other side to release connection. Here, there is no chance of losing the data.

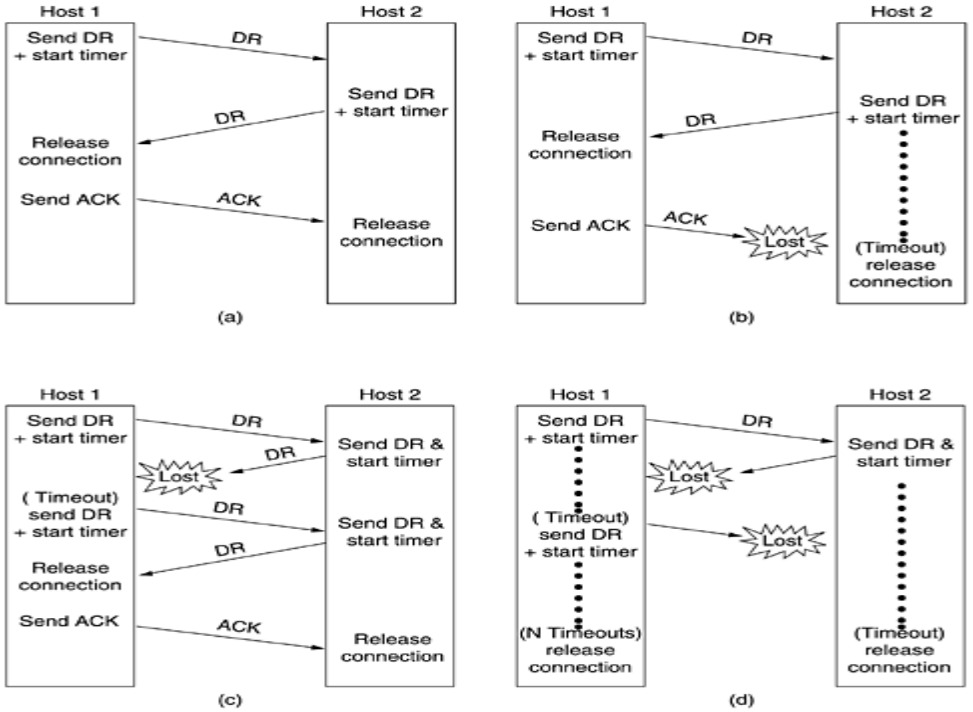
Consider a scenario of below fig where release abrupt and may result in data loss.

After the connection is established, host 1 sends a TPDU that arrives properly at host 2. Then host 1 sends another TPDU. Unfortunately, host 2 issues a DISCONNECT before the second TPDU arrives. The result is that the connection is released and data are lost. Symmetric release does the job when each process has a fixed amount of data to send and clearly knows when it has sent it. Both side will agree upon and thus connection release which does not result in loss of data.



Abrupt disconnection with loss of data.

Four protocol scenarios for releasing a connection are discussed with the following cases



Four protocol scenarios for releasing a connection. (a) Normal case of three-way handshake. (b) Final ACK lost. (c) Response lost. (d) Response lost and subsequent DRs lost.

Case 1: Normal connection release

One of the user sends a DISCONNECTION REQUEST TPDU in order to initiate connection release. When it arrives, the recipient sends back a DRTPDU, too, and starts a timer. When this DR arrives, the original sender sends back an ACK-TPDU and releases the connection. Finally, when the ACK-TPDU arrives, the receiver also releases the connection.

Case 2:Final ACK TPDU is lost.

Initial process is done in the same way as in fig-(a). if the final ACKTPDU is lost, the situation is saved by the timer. When the timer is expired, the connection is released.

Case 3 Response lost

If the second DR is lost, the user initiating the disconnection will not receive the expected response, and will timeout and starts all over again.

Case 4: Both response and subsequent DRs lost

Same as in fig-(c) except that all repeated attempts to retransmit the DR are assumed to be failed due to lost TPDUs. After 'N' entries, the sender just gives up and releases the connection.

D) Flow control and Buffering:

Flow control is done by having a sliding window on each connection to keep a fast transmitter from over running a slow receiver. Buffering must be done by the sender, if the network service is unreliable. The sender buffers all the TPDUs sent to the receiver. The buffer size varies for different TPDUs.

They are:

- ***Chained Fixed-size Buffers***
- ***Chained Variable-size Buffers***
- ***One large Circular Buffer per Connection***

(A). Chained Fixed-size Buffers:

If most TPDUs are nearly the same size, the buffers are organized as a pool of identical size buffers, with one TPDU per buffer.

(B). Chained Variable-size Buffers:

This is an approach to the buffer-size problem. i.e., if there is wide variation in TPDU size, from a few characters typed at a terminal to thousands of characters from file transfers, some problems may occur:

- If the buffer size is chosen equal to the largest possible TPDU, space will be wasted whenever a short TPDU arrives.
- If the buffer size is chosen less than the maximum TPDU size, multiple buffers will be needed for long TPDUs.

To overcome these problems, we employ variable-size buffers.

(c). One large Circular Buffer per Connection:

A single large circular buffer per connection is dedicated when all connections are heavily loaded.

E) Multiplexing:

In networks that use virtual circuits within the subnet, each open connection consumes some table space in the routers for the entire duration of the connection. If buffers are dedicated to the virtual circuit in each router as well, a user who left a terminal logged into a remote machine, there is need for multiplexing. There are 2 kinds of multiplexing:

1. Up-ward multiplexing:

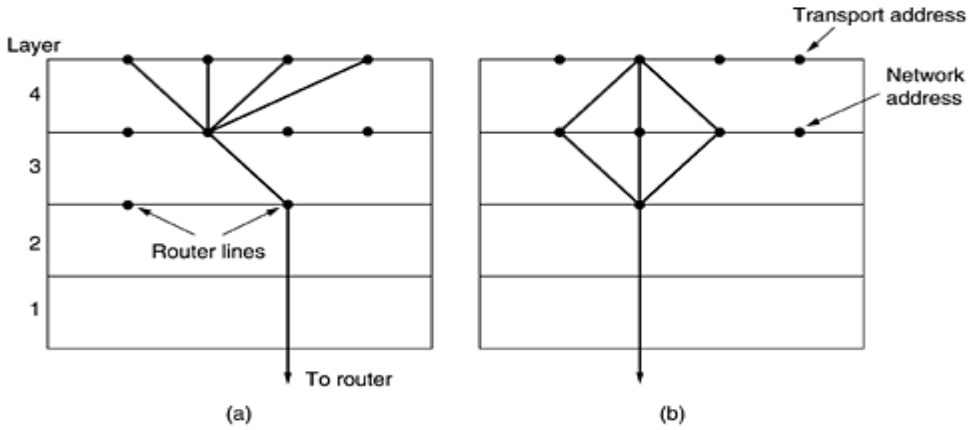
In the below figure, all the 4 distinct transport connections use the same network connection to the remote host. When connect time forms the major component of the carrier's bill, it is up to the transport layer to group port connections according to their destination and maps each group onto the minimum number of port connections.

2. Down-ward multiplexing:

• If too many transport connections are mapped onto the one network connection, the performance will be poor.

• If too few transport connections are mapped onto one network connection, the service will be expensive.

The possible solution is to have the transport layer open multiple connections and distribute the traffic among them on round-robin basis, as indicated in the below figure:



(a) Upward multiplexing. (b) Downward multiplexing.

F) Crash recovery:

To recover from server crashes……..

If there is a transmission between a client and a server and mean while if the server crashes, the server will send a broadcast TPDU to all other hosts, announcing that it just crashed. Each client can be in any of 2 states:

- One TPDU outstanding, SI
- No TPDUs outstanding, SO

THE INTERNET TRANPORT PROTOCOLS (TCP AND UDP)

The internet has 2 main protocols in the transport layer they are:

1. Connection-Oriented Protocol (TCP)

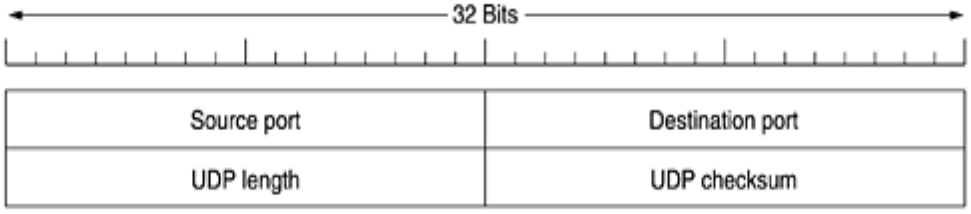
2. Connection-Less Protocol (UDP)

THE INTERNET TRANPORT PROTOCOLS:UDP

Introduction to UDP

The Internet protocol suite supports a connectionless transport protocol, **UDP** (**User Datagram Protocol**). UDP provides a way for applications to send encapsulated IP datagram's and send them without having to establish a connection.

UDP transmits **segments** consisting of an 8-byte header followed by the payload. The header is shown in Fig. 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. It provides a connection-less service for application level procedures. UDP is present on the top of IP. UDP adds a port addressing capability to IP. The header is as follows:



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

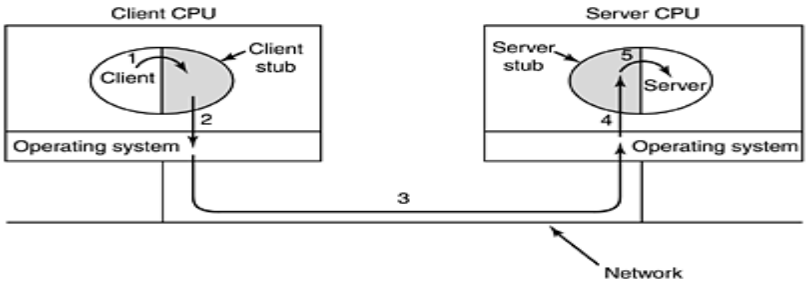
UDP checksum:

Includes the UDP header, the UDP data padded out to an even number o Of bytes if need be. It is an optional field

Remote Procedure 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 programmer. This technique is known as **RPC** (**Remote Procedure Call**). The calling procedure is known as the client and the called procedure is known as the server.

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.



Steps in making a remote procedure call. The stubs are shaded.

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 kernel sending the message from the client machine to the server machine.

Step 4 is the kernel passing the incoming packet to the server stub. Finally,

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 concept of remote procedure calls provides hiding of all network code into the stub procedures. This prevents the application programs, the client and server, from having to worry about details such as sockets, network byte order, and the like. One goal is to make the writing of distributed applications.

Problems with RPC:

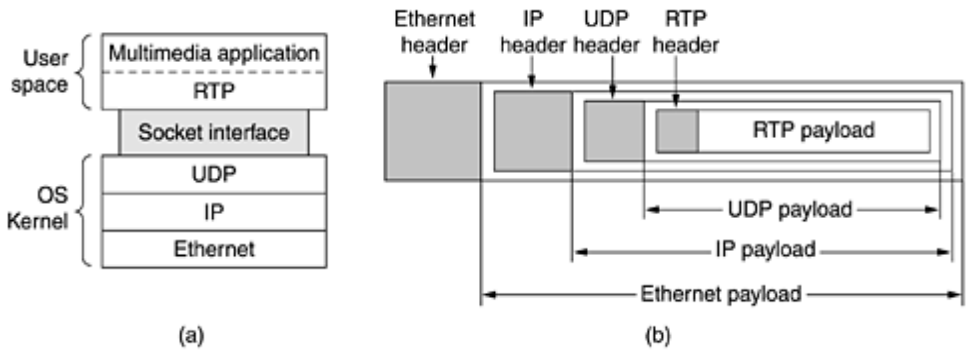
- With RPC, passing pointers is impossible because the client and server are in different address spaces.
- A second problem is that in weakly-typed languages, like C, it is perfectly legal to write a procedure that computes the inner product of two vectors (arrays), without specifying how large either one is. Each could be terminated by a special value known only to the calling and called procedure.
- A third problem is that it is not always possible to deduce the types of the parameters.
- A fourth problem relates to the use of global variables. Normally, the calling and called procedure can communicate by using global variables, in addition to communicating via parameters. If the called procedure is now moved to a remote machine, the code will fail because the global variables are no longer shared.

The Real-Time Transport Protocol

The Real-Time Transport Protocol was developed by the Audio/Video Transport working group of the IETF standards organization. RTP defines a standardized packet format for delivering audio and video over IP networks. RTP is used extensively in communication and entertainment systems that involve streaming media, internet radio,, internet telephony ,video conferencing and other multimedia applications

The position of RTP in the protocol stack is somewhat strange. It was decided to put RTP in user space and have it (normally) run over UDP. It operates as follows.

The multimedia application consists of multiple audio, video, text, and possibly other streams. These are fed into the RTP library, which is in user space along with the application. This library then multiplexes the streams and encodes them in RTP packets, which it then stuffs into a socket. At the other end of the socket (in the operating system kernel), UDP packets are generated and embedded in IP packets. If the computer is on an Ethernet, the IP packets are then put in Ethernet frames for transmission. The protocol stack for this situation is shown in Fig.(a). The packet nesting is shown in Fig. (b).



(a) The position of RTP in the protocol stack. (b) Packet nesting.

RTP Packet Header

The RTP Header has minimum size of 12 bytes. After a header, optional header extensions may be present. This is followed by RTP payload, the format of which is determined by the particular class of application. The fields in the header as follows:

1) **Version:**

The first word contains the *Version* field, which is already at 2 bits. Current version is 2.

2) P (Padding):

The *P* bit indicates that the packet has been padded to a multiple of 4 bytes. The last padding byte tells how many bytes were added. It occupies 1- bit used to indicate if there is extra padding bytes at the end of RTP packet. Padding might be used to fill up a block of certain size.

3) **X(Extension):**

The *X* bit indicates that an extension header is present. It occupies 1-bit.

4) CC(CSRC Count):

4-bits contains the number of CSRC identifiers that follow the fixed header

5) M(Market):

1-bit is used at application level and defined by a profile. If it is set it means that the current data has some special relevance for the application.

6) PT(payload):

7-bit indicates the format of the payload and determine its interpretation by the application. This is specified by an RTP profile.

7) Sequence number:

The sequence number is incremented by one for each TRP data packet sent and is to be used by the receiver to detect packet loss and to restore packet sequence. The RTP does not specify any action on packet loss, if is left to the application to make appropriate action.

8) Timestamp:

31-bit used to enable the receiver to play back the received samples at appropriate intervals.

9) SSRC:

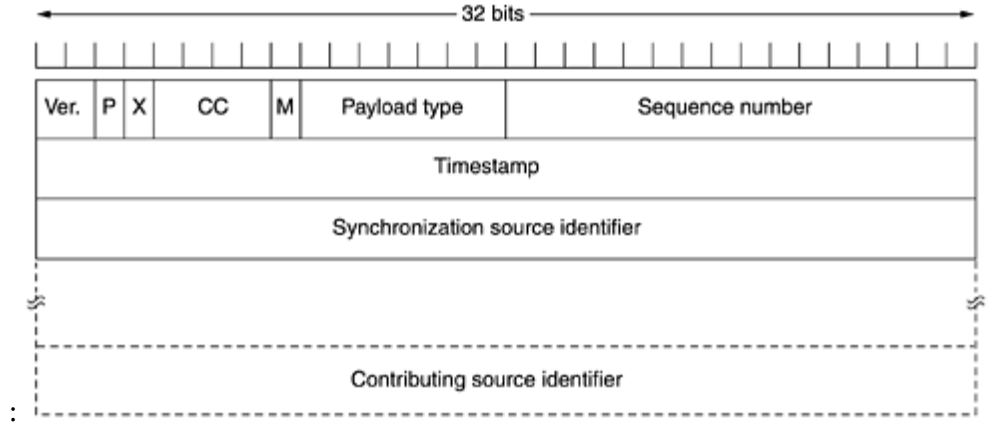
32-bit synchronization source identifier source identifier uniquely identifies the source of a stream. The synchronization sources within the same RTP session will be unique.

10) CSRC:

Contributing source IDs enumerate contributing sources to a stream which has been generated from multiple sources.

11) Extension

Optional the first 32-bit word contains a profile-specific identifier and a length specifier that indicates the length of the extension in 32-bit units, excluding the 32-bits of the extension header.



:

The RTP header.

THE INTERNET TRANPORT PROTOCOLS: TCP

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 datagram's 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 datagram's into messages in proper sequence.

The different issues to be considered are:

- ✓ **The TCP Service Model**
- ✓ **The TCP Protocol**
- ✓ **The TCP Segment Header**
- ✓ **The Connection Establishment**
- ✓ **The TCP Connection Release**
- ✓ **The TCP Connection Management Modeling**
- ✓ **TCP Transmission Policy**
- ✓ **TCP Congestion Control**
- ✓ **TCP Timer Management.**
- ✓ **Wireless TCP and UDP**
- ✓ **Transactional TCP**

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

TCP service is obtained by both the sender and receiver creating end points, called sockets. Each socket has a socket number (address) consisting of the IP address of the host and a 16-bit number local to that host, called a **port**. A port is the TCP name for a TSAP. For TCP service to be obtained, a connection must be explicitly established between a socket on the sending machine and a socket on the receiving machine.

A socket may be used for multiple connections at the same time. Two or more connections may terminate at the same socket. Connections are identified by the socket identifiers at both ends that is, (*socket1*, *socket2*). No virtual circuit numbers or other identifiers are used.

Port numbers below 1024 are called **well-known ports** and are reserved for standard services. For example, any process wishing to establish a connection to a host to transfer a file using FTP can connect to the destination host's port 21 to contact its FTP daemon.

Port	Protocol	Use
21	FTP	File transfer
23	Telnet	Remote login
25	SMTP	E-mail
69	TFTP	Trivial file transfer protocol
79	Finger	Lookup information about a user
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
119	NNTP	USENET news

Some assigned ports.

The TCP Protocol:

Every byte on a TCP connection has its own 32-bit sequence number, used both for acknowledgement and window mechanism. The sending and receiving TCP entities exchange data in the form of segments [20 byte header + optional part] followed by zero or more data bytes] segments can be united or splitted, by the TCP s/w accordingly, based on 2 limits:

1. Each segment must fit in 65,535 byte IP payload.
2. Each segment must fit in MTU (Max. Transfer Unit) of a network.

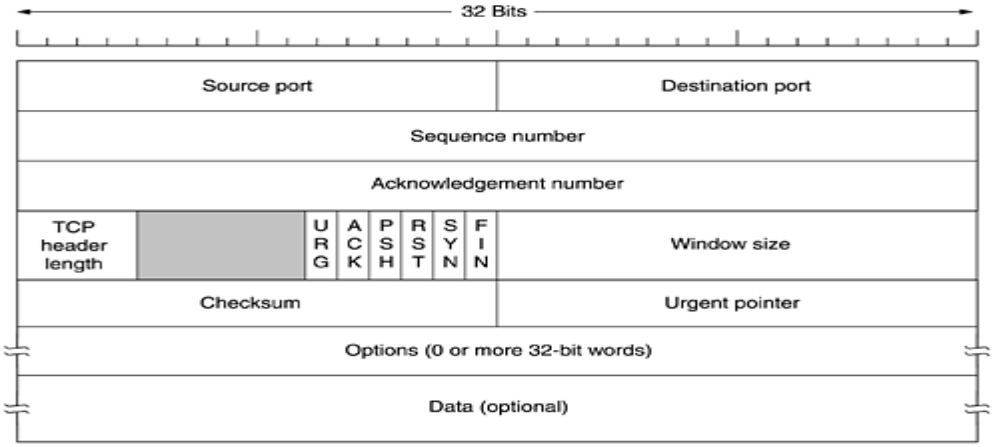
The basic protocol used by TCP entities is the **"SLIDING WINDOW PROTOCOL"** when a sender transmits a segment, it also starts a timer. When the segment arrives at destination, the receiving TCP entity sends back a segment having an acknowledgement number equal to the next sequence number it expects to receive. If the sender timer goes off before the acknowledgement is received, the sender transmits the segment again.

TCP at the design time must deal with different problems like:

- Since segments can be fragmented, it is possible that part of a transmitted segment arrives and is acknowledged, while the rest is lost.
- Segments can also arrive out of order
- Segments can also be delayed so long in transit that the sender times out and retransmits them
- If retransmitted segment takes other route than the original and reaches the receiver, duplicates must be differentiated from the original ones.
- Segments may occasionally hit congested networks along its path.

The TCP Segment Header:

Every TCP segment begins with a fixed format, 20 byte header. The fixed header may be followed by header options. Segments without data are commonly used for acknowledgments and control messages.



The TCP header.

Every segment begins with a fixed-format 20 byte header followed by header options. Then follows, up to 65,535-20-20=65,515 data bytes if any.

Different fields of TCP header are:

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.

SYN : This bit is used to establish connections. Connection request has SYN = 1 and ACK = 0. This indicates that the piggy back acknowledgement filed is not in use.

Connection reply has SYN = 1 and ACK = 1. This indicate the connection reply does bear an acknowledgment.

FIN : Used to release a connection.

Window size : This 16-bit window size field is used for flow control. It is used to indicate the number of bytes that a receiver is willing to accept. TCP uses variable-sized sliding window.

Checksum: This field is used for extra reliability in algorithm is simply used to add up all the 16 bit words in one's complement sum is taken. At the receiver end, receiver perform the calculation on the entire segment, including checksum filed, the result should be zero.

Urgent pointer:

"Urgent data" is another feature of TCP in which the sending application puts some control information in the data stream when an interactive user hits the DEL or CTRL-C key to break off a remote computation that has already begun, and gives it to TCP along with URGENT flag. This event causes TCP to stop accumulating data and transmit everything it has for that connection immediately.

When the urgent data are received at the destination, the receiving application is interrupted and reads the data stream to find the urgent data.

Options filed: This provides extra facilities that are not covered by the regular header. The most important option is the one that allows each host to specify the maximum, TCP payload it is willing to accept.

TCP Connection Establishment:

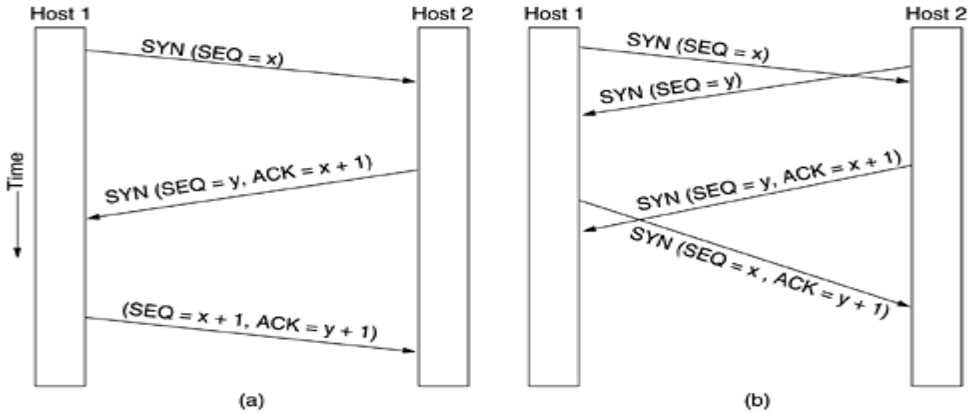
Connections are established in TCP using the 3-way handshake. To establish a connection between a server and a client following actions take place at both sides:

Server

Passively waits for an incoming connection by executing LISTEN and ACCEPT primitives, either specifying a specific source or nobody in particular.

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 additionally some user data.



TCP CONNECTION SETUP (a) TCP connection establishment in the normal case. (b) Call collision.

1). When a TCP segment arrive at destination, the TCP entity then ,checks to see if there is a process that has done a LISTEN on port given in destination port field.

2). If not, it sends a reply with RST bit-ON to reject the connection other wise, the process is given incoming TCP segment, by which the connection can be accepted or rejected.

3). Acknowledgement is sent back, if accepted.

TCP Connection Release:

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. When both directions have been shut down, the connection is released.

Four TCP segments are needed to release a connection, one *FIN* and one *ACK* for each direction. 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:

Connection management can be represented using the state diagram. The various primitives or states that are used are:

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

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

TCP Connection management from client's point of view:

1. **Client** issues a command to its **TCP** to request a connection. Client send a **CONNECT** primitive after listening to **LISTEN** primitive from server
2. Client send a **SYN** segment along with **CONNECT** primitive and moves to **SYN-SENT** state.
3. After receiving **SYN + ACK** from server, client goes to **ESTABLISHED** state which is the normal state for data transfer
4. When client does not have any data to send it executes **CLOSE** primitive.
5. CLOSE primitive is sent along with **FIN** bit which waits for the **ACK** and goes to **FIN-WAIT1** state.
6. When it receives the **ACK** from the sent **FIN**, it goes to it goes to **FIN-WAIT2** state and waits for **FIN** segment from server.
7. When it is received, client sends an **ACK** and goes to **TIME-WAIT** state

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

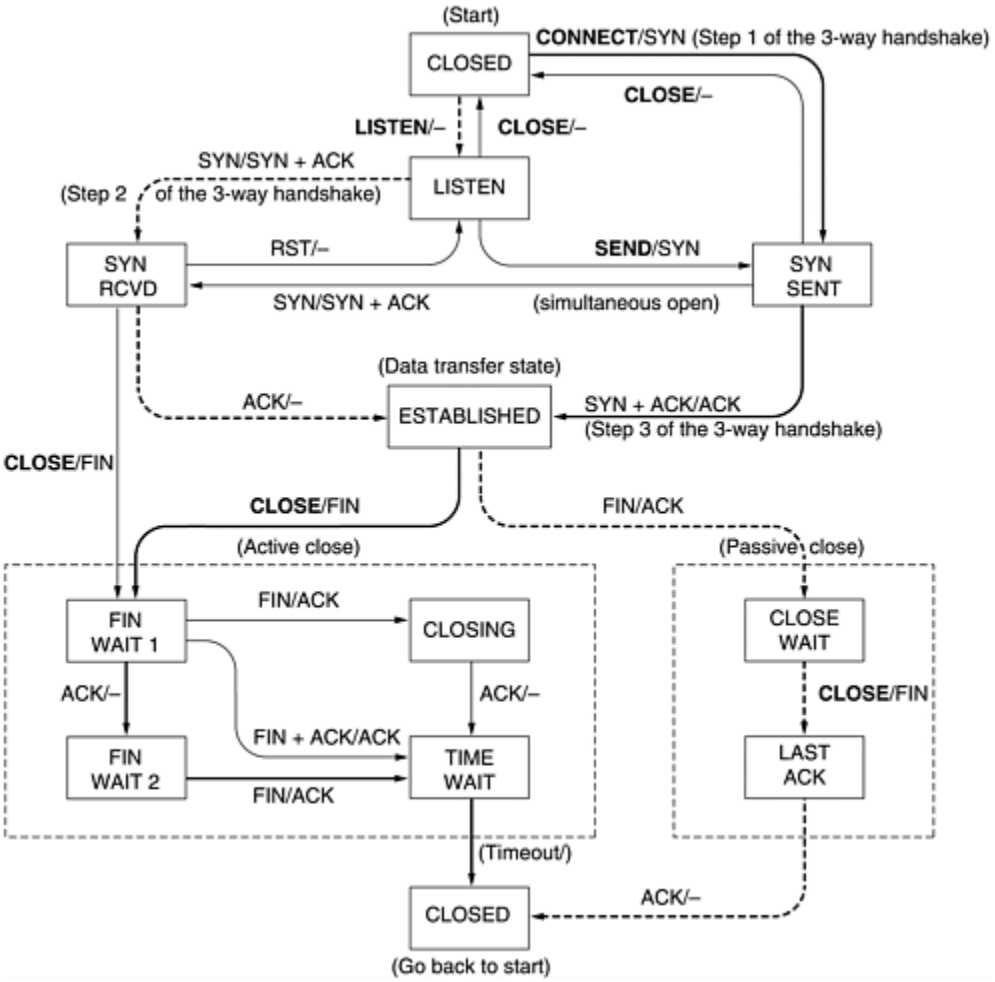
3. When the servers **SYN** is itself acknowledged the 3-way handshake is complete and server goes to the **ESTABLISHED** state. Data transfer can now occur.

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 connection management finite state machine. The heavy solid line is the normal path for a client. The heavy dashed line is the normal path for a server. The light lines are unusual events. Each transition is labeled by the event causing it and the action resulting from it, separated by a slash.

TCP Transmission policy:

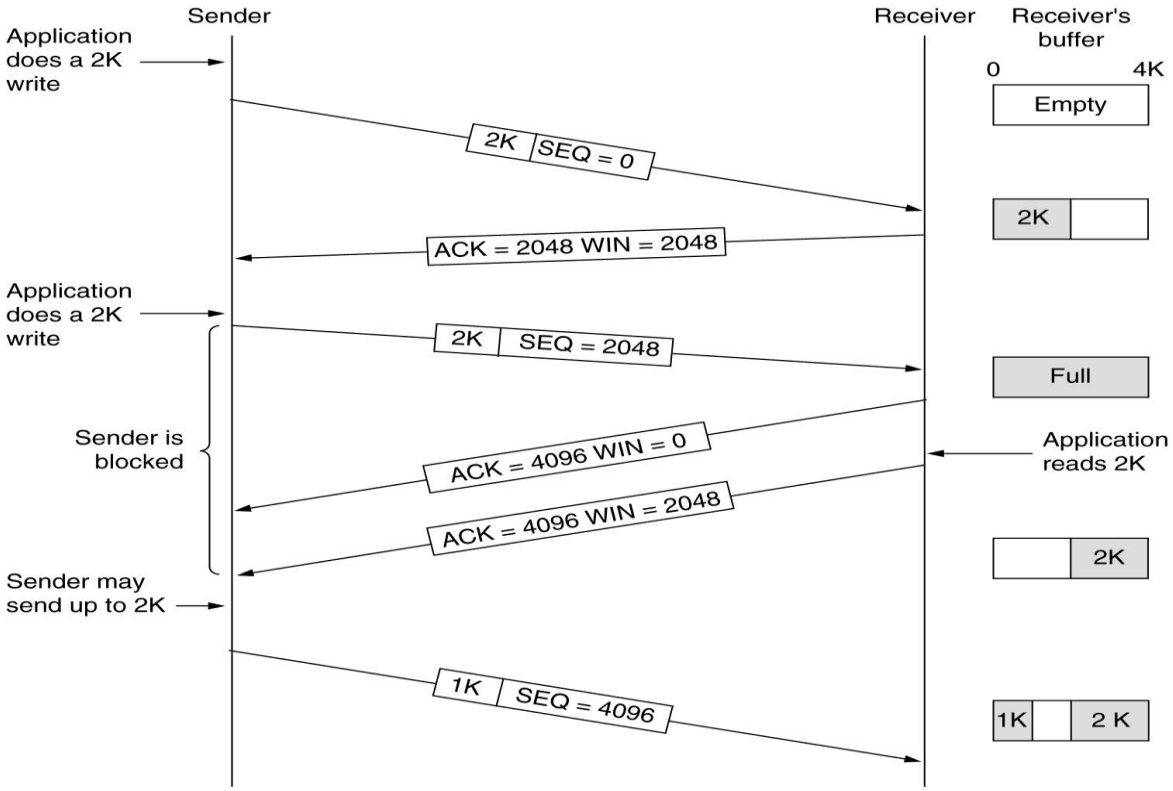
1. In the above example, the receiver has 4096-byte buffer.

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.

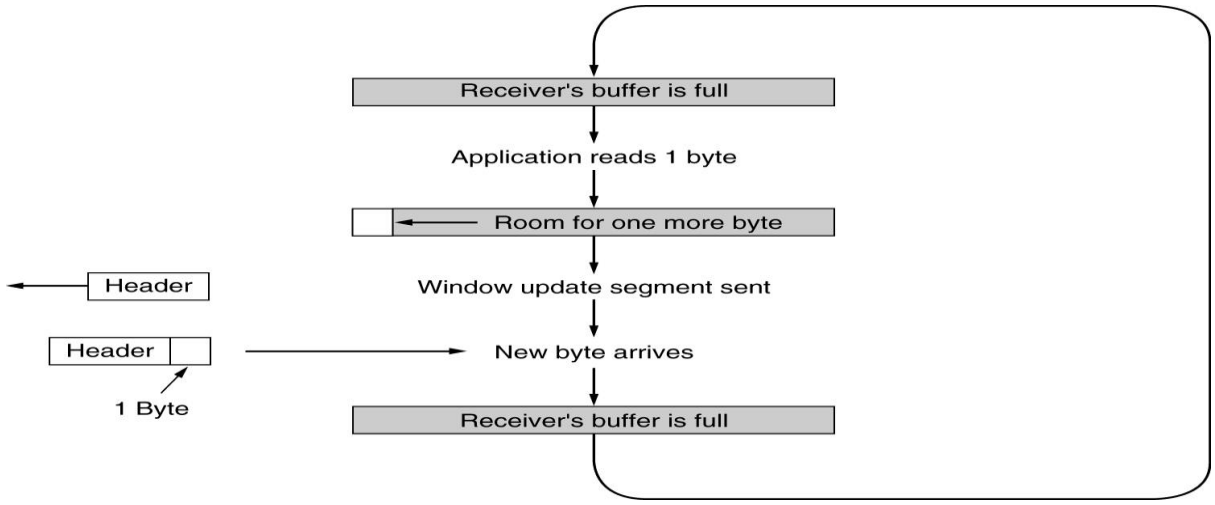
4. Now the sender transmits another 2048 bytes which are acknowledged, but the advertised window is'0'.

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.



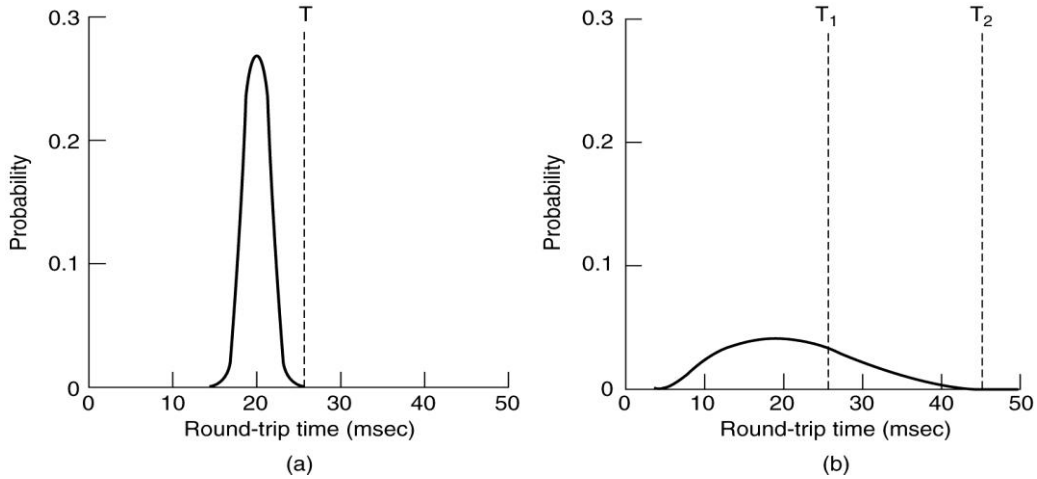
- Initially the TCP buffer on the receiving side is full and the sender knows this(win=0)
- Then the interactive application reads 1 character from tcp stream.
- Now, the receiving TCP sends a window update to the sender saying that it is all right to send 1 byte.
- The sender obligates and sends 1 byte.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

TCP Timer Management

TCP uses multiple timers (at least conceptually) to do its work. The most important of these is the **RTO** (**Retransmission TimeOut**). When a segment is sent, a retransmission 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 os started again).

This problem is much more difficult in the transport layer than in data link protocols such as 802.11. In the latter case, the expected delay is measured in microseconds and is highly predictable (i.e., has a low variance), so the timer can be set to go off just slightly after the acknowledgement is expected, as shown in Fig. 6-42(a). Since acknowledgements are rarely delayed in the data link layer (due to lack of congestion), the absence of an acknowledgement at the expected time generally means either the frame or the acknowledgement has been lost.

TCP is faced with a radically different environment. The probability density function for the time it takes for a TCP acknowledgement to come back looks more like Fig. 6-42(b) than Fig. 6-42(a). It is larger and more variable. Determining the round-trip time to the destination is tricky. Even when it is known, deciding on the timeout interval is also difficult. If the timeout is set too short, say, *T*1 in Fig. 6-42(b), unnecessary retransmissions will occur, clogging the Internet with useless packets.



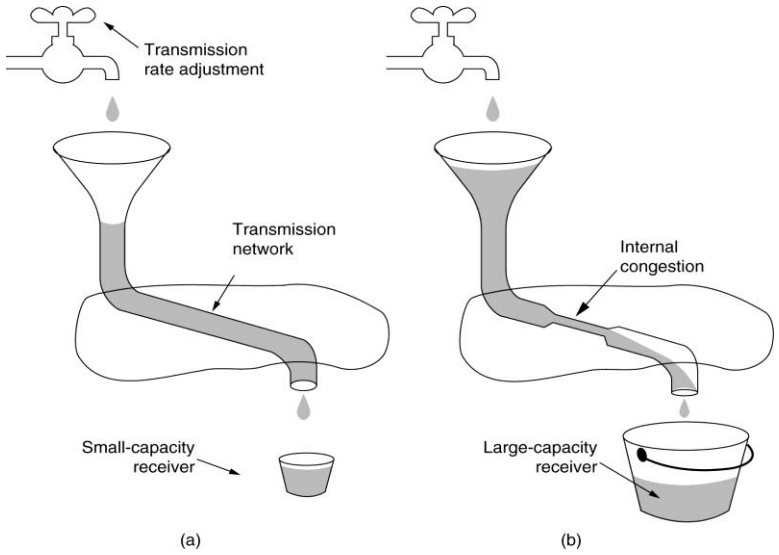
(a) Probability density of ACK arrival times in the data link layer.

(b) Probability density of ACK arrival times for TCP.

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 with in the n/w. lets see this problem occurs.

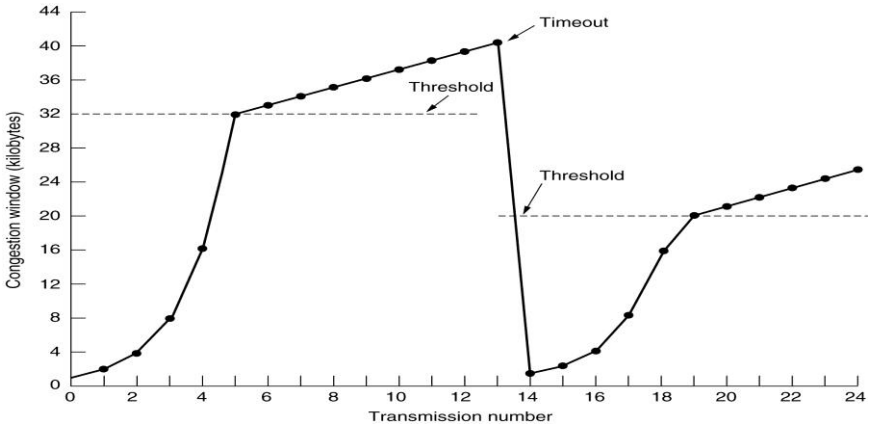


In fig (a): We see a thick pipe leading to a small- capacity receiver. As long as the sender does not send more water than the bucket can contain, no water will be lost.

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 segment 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.



An example of the Internet congestion algorithm.

4.3 CONGESTION CONTROL

- If the transport entities on many machines send too many packets into the network too quickly, the network will become congested, with performance degraded as packets are delayed and lost.
- Controlling congestion to avoid this problem is the combined responsibility of the network and transport layers.
- Congestion occurs at routers, so it is detected at the network layer. However, congestion is ultimately caused by traffic sent into the network by the transport layer. The only effective way to control congestion is for the transport protocols to send packets into the network more slowly.
- The Internet relies heavily on the transport layer for congestion control, and specific algorithms are built into TCP and other protocols.

4.3.1 Desirable Bandwidth Allocation

- It is to find a good allocation of bandwidth to the transport entities that are using the network.
- A good allocation will deliver good performance because it uses all the available bandwidth but avoids congestion, it will be fair across competing transport entities, and it will quickly track changes in traffic demands.
- We must specify the state in which a good congestion control algorithm will operate the network.
- The goal is more than to simply avoid congestion.

Efficiency and Power

An efficient allocation of bandwidth across transport entities will use all of the network capacity that is available. However, it is not quite right to think that if there is a 100-Mbps link, **five transport entities should get 20 Mbps each.** They should usually get less than 20 Mbps for good performance. The reason is that the traffic is often bursty. The **goodput** (or rate of useful packets arriving at the receiver) as a function of the offered load. This curve and a matching curve for the delay as a function of the offered load are given in Fig. 6-19.

As the load increases in Fig. 6-19(a**) goodput** initially increases at the same rate, but as the load approaches the capacity, **goodput** rises more gradually. This falloff is because bursts of traffic can occasionally mount up and cause some losses at buffers inside the network. If the transport protocol is poorly designed and retransmits packets that have been delayed but not lost, the network can enter congestion collapse. In this state, senders are furiously sending packets, but increasingly little useful work is being accomplished.

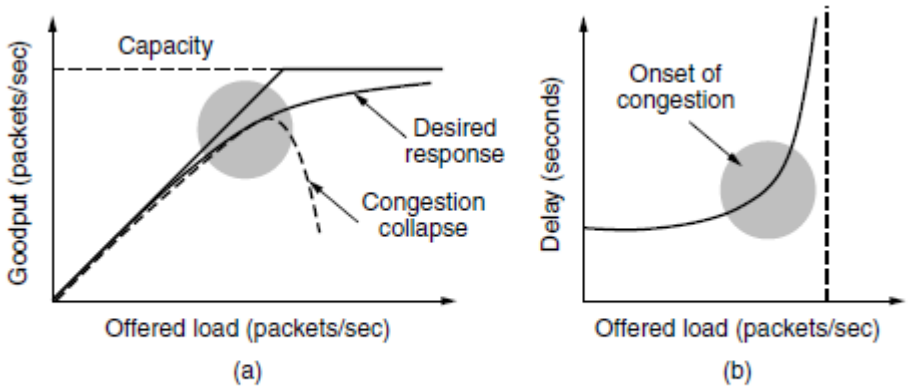


Figure 6-19. (a) Goodput and (b) delay as a function of offered load.

The corresponding **delay** is given in Fig. 6-19(b) Initially the delay is fixed, representing the propagation delay across the network. As the load approaches the capacity, the delay rises, slowly at first and then much more rapidly. This is again because of bursts of traffic that tend to mound up at high load. The delay cannot really go to infinity, except in a model in which the routers have infinite buffers. Instead, packets will be lost after experiencing the maximum buffering delay.

For both **goodput and delay**, performance begins to degrade at the onset of congestion. Intuitively, we will obtain the best performance from the network if we allocate bandwidth up until the delay starts to climb rapidly. This point is below the capacity. To identify it, Kleinrock (1979) proposed the metric of **power**, where

$$\text{power} = \text{load}/\text{delay}$$

Power will initially rise with offered load, as delay remains small and roughly constant, but will reach a maximum and fall as delay grows rapidly. The load with the highest power represents an efficient load for the transport entity to place on the network.

Max-Min Fairness

How to divide bandwidth between different transport senders. These sounds like a simple question to answer give all the senders an equal fraction of the bandwidth but it involves several considerations.

The first consideration is to ask what this problem has to do with congestion control. After all, if the network gives a sender some amount of bandwidth to use, the sender should just use that much bandwidth. However, it is often the case that networks do not have a strict bandwidth reservation for each flow or connection. They may for some flows if quality of service is supported, but many connections will seek to use whatever bandwidth is available or be lumped together by the network under a common allocation.

A second consideration is what a fair portion means for flows in a network. It is simple enough if N flows use a single link, in which case they can all have $1/N$ of the bandwidth (although efficiency will dictate that they use slightly less if the traffic is bursty). But what happens if the flows

have different, but overlapping, network paths? For example, one flow may cross three links, and the other flows may cross one link. The three-link flow consumes more network resources. It might be fairer in some sense to give it less bandwidth than the one-link flows. It should certainly be possible to support more one-link flows by reducing the bandwidth of the three-link flow. This point demonstrates an inherent tension between fairness and efficiency.

The form of fairness that is often desired for network usage is **max-min fairness**. An allocation is max-min fair if the bandwidth given to one flow cannot be increased without decreasing the bandwidth given to another flow with an allocation that is no larger. That is, increasing the bandwidth of a flow will only make the situation worse for flows that are less well off.

Let us see an example. A max-min fair allocation is shown for a network with four flows, *A*, *B*, *C*, and *D*, in Fig. 6-20. Each of the links between routers has the same capacity, taken to be 1 unit, though in the general case the links will have different capacities. Three flows compete for the bottom-left link between routers *R4* and *R5*. Each of these flows therefore gets 1/3 of the link. The remaining flow, *A*, competes with *B* on the link from *R2* to *R3*. Since *B* has an allocation of 1/3, *A* gets the remaining 2/3 of the link. Notice that all of the other links have spare capacity. However, this capacity cannot be given to any of the flows without decreasing the capacity of another, lower flow. For example, if more of the bandwidth on the link between *R2* and *R3* is given to flow *B*, there will be less for flow *A*. This is reasonable as flow *A* already has more bandwidth. However, the capacity of flow *C* or *D* (or both) must be decreased to give more bandwidth to *B*, and these flows will have less bandwidth than *B*. Thus, **the allocation is max-min fair.**

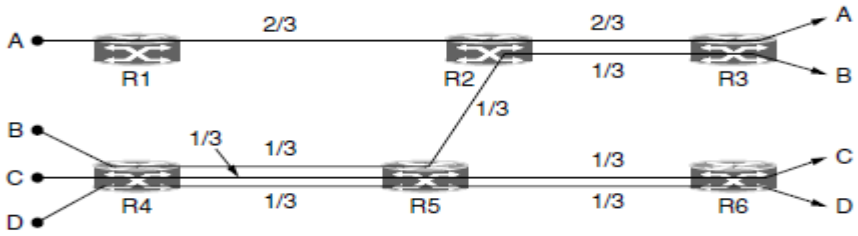


Figure 6-20. Max-min bandwidth allocation for four flows.

Max-min allocations can be computed given a global knowledge of the network. An intuitive way to think about them is to imagine that the rate for all of the flows starts at zero and is slowly increased. When the rate reaches a bottleneck for any flow, then that flow stops increasing. The other flows all continue to increase, sharing equally in the available capacity, until they too reach their respective bottlenecks.

A third consideration is the level over which to consider fairness. A network could be fair at the level of connections, connections between a pair of hosts, or all connections per host.

For example, defining fairness per host means that a busy server will fare no better than a mobile phone, while defining fairness per connection encourages hosts to open more connections. Given that there is no clear answer, fairness is often considered per connection, but precise fairness is

usually not a concern. It is more important in practice that no connection be starved of bandwidth than that all connections get precisely the same amount of bandwidth. In fact, with TCP it is possible to open multiple connections and compete for bandwidth more aggressively. This tactic is used by bandwidth-hungry applications such as **BitTorrent** for **peer-to-peer file sharing**.

4.3.2 Regulating the Sending Rate

The sending rate may be limited by two factors. The first is flow control, in the case that there is insufficient buffering at the receiver. The second is congestion, in the case that there is insufficient capacity in the network.

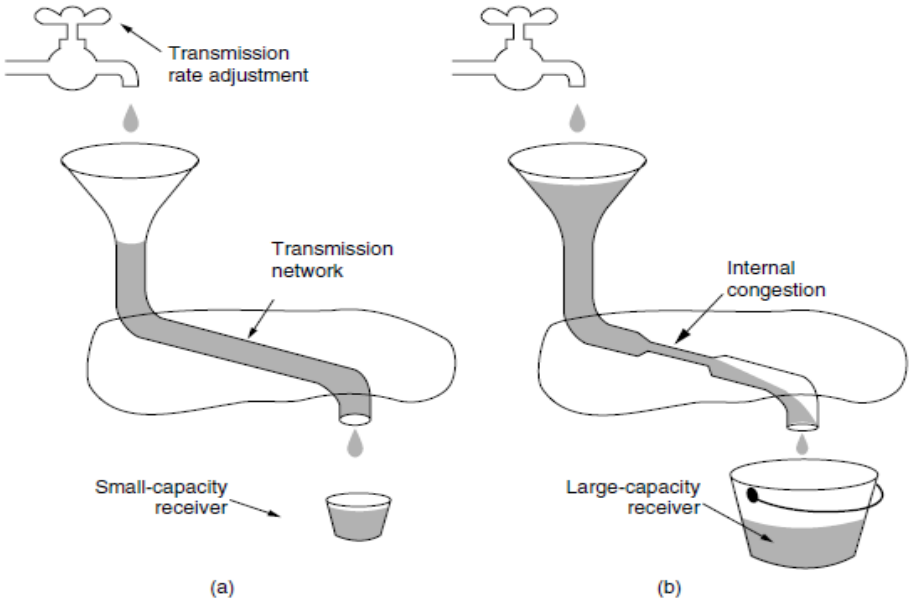


Figure 6-22. (a) A fast network feeding a low-capacity receiver. (b) A slow network feeding a high-capacity receiver.

In Fig. 6-22, we see this problem illustrated hydraulically. In Fig. 6-22(a), we see a thick pipe leading to a small-capacity receiver. This is a flow-control limited situation. As long as the sender does not send more water than the bucket can contain, no water will be lost. In Fig. 6-22(b), the limiting factor is not the bucket capacity, but the internal carrying capacity of the network.

If too much water comes in too fast, it will back up and some will be lost (in this case, by overflowing the funnel). These cases may appear similar to the sender, as transmitting too fast causes packets to be lost. However, they have different causes and call for different solutions.

We have already talked about a flow-control solution with a variable-sized window. Now we will consider a congestion control solution. Since either of these problems can occur, the transport protocol will in general need to run both solutions and slow down if either problem occurs.

The way that a transport protocol should regulate the sending rate depends on the form of the feedback returned by the network. Different network layers may return different kinds of feedback. The feedback may be explicit or implicit, and it may be precise or imprecise.

An example of an explicit, precise design is when routers tell the sources the rate at which they may send. Designs in the literature such as XCP (eXplicit Congestion Protocol) operate in this manner. An explicit, imprecise design is the use of ECN (Explicit Congestion Notification) with TCP. In this design, routers set bits on packets that experience congestion to warn the senders to slow down, but they do not tell them how much to slow down.

In other designs, there is no explicit signal. FAST TCP measures the roundtrip delay and uses that metric as a signal to avoid congestion. Finally, in the form of congestion control most prevalent in the Internet today,

TCP with drop-tail or RED routers, packet loss is inferred and used to signal that the network has become congested. There are many variants of this form of TCP, including CUBIC TCP, which is used in Linux. Combinations are also possible. For example, Windows includes Compound TCP that uses both packet loss and delay as feedback signals.

Protocol	Signal	Explicit?	Precise?
XCP	Rate to use	Yes	Yes
TCP with ECN	Congestion warning	Yes	No
FAST TCP	End-to-end delay	No	Yes
Compound TCP	Packet loss & end-to-end delay	No	Yes
CUBIC TCP	Packet loss	No	No
TCP	Packet loss	No	No

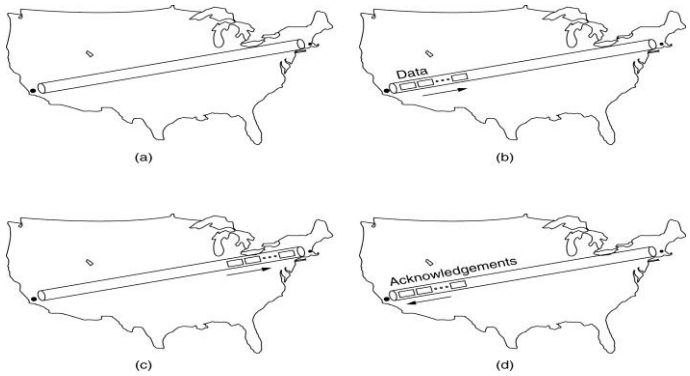
Figure 6-23. Signals of some congestion control protocols.

If an explicit and precise signal is given, the transport entity can use that signal to adjust its rate to the new operating point. For example, if XCP tells senders the rate to use, the senders may simply use that rate. In the other cases, however, some guesswork is involved. In the absence of a congestion signal, the senders should decrease their rates. When a congestion signal is given, the senders should decrease their rates. The way in which the rates are increased or decreased is given by a **control law**. These laws have a major effect on performance.

Performance problems in computer networks:

Some performance problems, such as congestion, are caused by temporary resource overloads. If more traffic suddenly arrives at a router than the router can handle, congestion will build up and performance will suffer.

Performance also degrades when there is a structural resource imbalance. For example, if a gigabit communication line is attached to a low-end PC, the poor CPU will not be able to process the incoming packets fast enough and some will be lost. These packets will eventually be retransmitted, adding delay, wasting bandwidth, and generally reducing performance.



The state of transmitting one megabit from San Diego to Boston

(a) At t = 0, (b) After 500 μsec, (c) After 20 msec, (d) after 40 msec.

Consider, for example, sending a 64-KB burst of data from San Diego to Boston in order to fill the receiver's 64-KB buffer. Suppose that the link is 1 Gbps and the one-way speed-of-light-in-fiber delay is 20 msec. Initially, at t = 0, the pipe is empty, as illustrated in Fig. 6-41(a). Only 500 μsec later, in Fig. 6-41(b), all the TPDUs are out on the fiber. The lead TPDU will now be somewhere in the vicinity of Brawley, still deep in Southern California. However, the transmitter must stop until it gets a window update.

After 20 msec, the lead TPDU hits Boston, as shown in Fig. 6-41(c) and is acknowledged. Finally, 40 msec after starting, the first acknowledgement gets back to the sender and the second burst can be transmitted. Since the transmission line was used for 0.5 msec out of 40, the efficiency is about 1.25 percent. This situation is typical of older protocols running over gigabit lines.

A useful quantity to keep in mind when analyzing network performance is the **bandwidth-delay product**. It is obtained by multiplying the bandwidth (in bits/sec) by the round-trip delay time (in sec). The product is the capacity of the pipe from the sender to the receiver and back (in bits).

For the example of Fig. 6-41 the bandwidth-delay product is 40 million bits. In other words, the sender would have to transmit a burst of 40 million bits to be able to keep going full speed until the first acknowledgement came back. It takes this many bits

to fill the pipe (in both directions). This is why a burst of half a million bits only achieves a 1.25 percent efficiency: it is only 1.25 percent of the pipe's capacity.

The conclusion that can be drawn here is that for good performance, the receiver's window must be at least as large as the bandwidth-delay product, preferably somewhat larger since the receiver may not respond instantly. For a transcontinental gigabit line, at least 5 megabytes are required.

If the efficiency is terrible for sending a megabit, imagine what it is like for a short request of a few hundred bytes. Unless some other use can be found for the line while the first client is waiting for its reply, a gigabit line is no better than a megabit line, just more expensive.

Another performance problem that occurs with time-critical applications like audio and video is jitter. Having a short mean transmission time is not enough. A small standard deviation is also required. Achieving a short mean transmission time along with a small standard deviation demands a serious engineering effort.

Network Performance Measurement

When a network performs poorly, its users often complain to the folks running it, demanding improvements. To improve the performance, the operators must first determine exactly what is going on. To find out what is really happening, the operators must make measurements. In this section we will look at network performance measurements.

The basic loop used to improve network performance contains the following steps:

1. Measure the relevant network parameters and performance.
2. Try to understand what is going on.
3. Change one parameter.

These steps are repeated until the performance is good enough or it is clear that the last drop of improvement has been squeezed out.

Measurements can be made in many ways and at many locations (both physically and in the protocol stack). The most basic kind of measurement is to start a timer when beginning some activity and see how long that activity takes. For example, knowing how long it takes for a TPDU to be acknowledged is a key measurement. Other measurements are made with counters that record how often some event has happened (e.g., number of lost TPDUs). Finally, one is often interested in knowing the amount of something, such as the number of bytes processed in a certain time interval.

Measuring network performance and parameters has many potential pitfalls. Below we list a few of them. Any systematic attempt to measure network performance should be careful to avoid these.

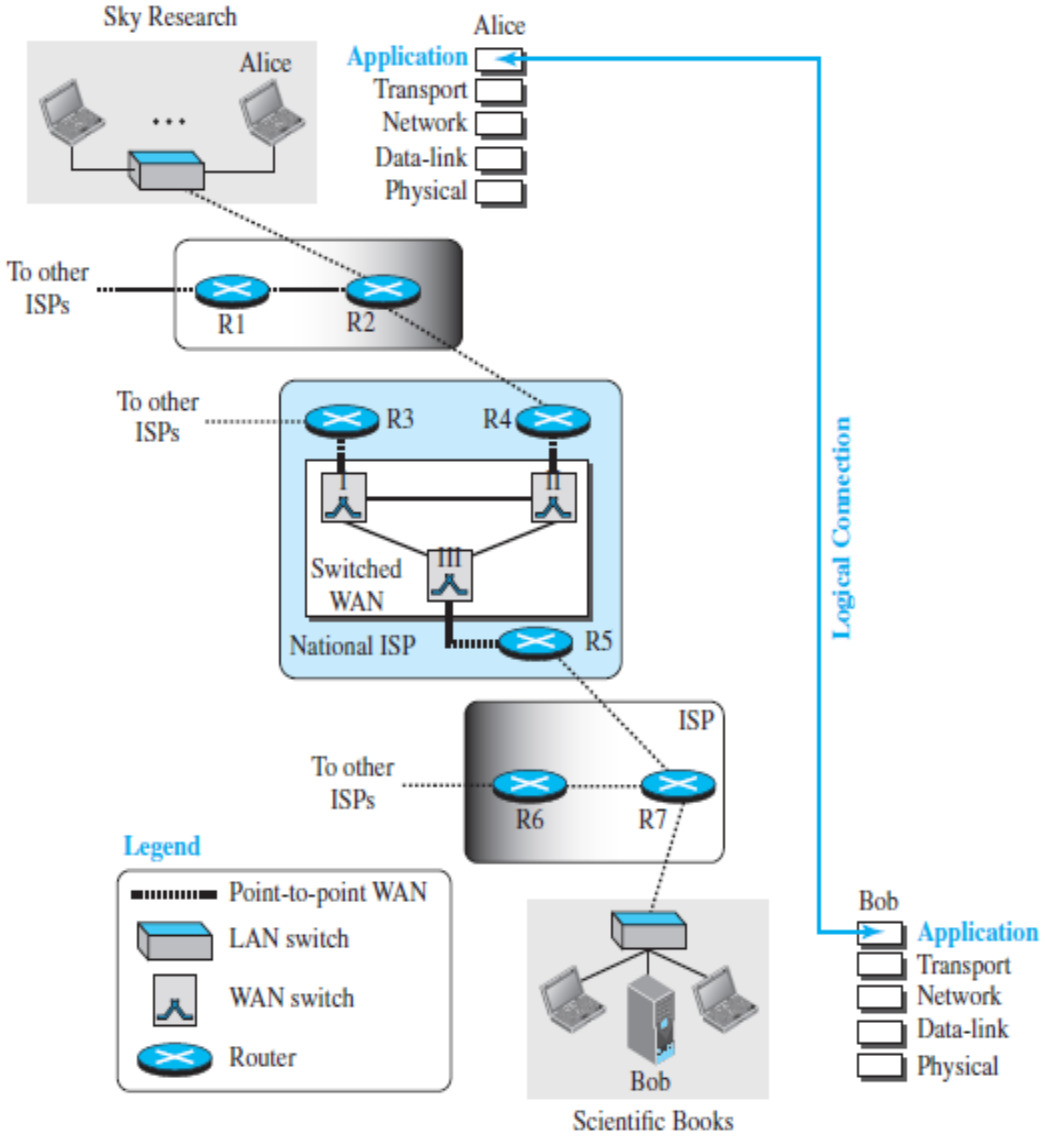
UNIT V

Introduction to Application Layer: Introduction, Client Server Programming, WWW and HTTP, FTP, E-mail, TELNET, Secure Shell, Domain Name System, SNMP.

Introduction:

The application layer provides services to the user. Communication is provided using a logical connection, which means that the two application layers assume that there is an imaginary direct connection through which they can send and receive messages.

Figure 25.1 *Logical connection at the application layer*



The figure shows the same scenario we have seen for other layers, but this time the logical connection is between two application layers.

5.1.1 Providing Services

- Communication networks that started before the Internet were designed to provide services to network users. All these networks are designed to provide one specific service.
- The Internet was originally designed for the same purpose: to provide service to users around the world. The layered architecture of the TCP/IP protocol suite, however, makes the Internet more flexible than other communication networks such as postal or telephone networks.
- Each layer in the suite was originally made up of one or more protocols, but new protocols can be added or some protocols can be removed or replaced by the Internet authorities.
- If a protocol is added to each layer, it should be designed in such a way that it uses the services provided by one of the protocols at the lower layer.
- If a protocol is removed from a layer, care should be taken to change the protocol at the next higher layer that supposedly uses the services of the removed protocol.
- The application layer, however, is somewhat different from other layers in that it is the highest layer in the suite. The protocols in this layer do not provide services to any other protocol in the suite; they only receive services from the protocols in the transport layer.
- The application layer is the only layer that provides services to the Internet user.

For example:

The telephone network was originally designed to provide voice service: to allow people all over the world to talk to each other.

Standard and Nonstandard Protocols

The protocols used in the first four layers of the TCP/IP suite need to be standardized and documented. The application-layer protocols can be both standard and nonstandard.

Standard Application-Layer Protocols

There are several application-layer protocols that have been standardized and documented by the Internet authority, and we are using them in our daily interaction with the Internet.

Each standard protocol is a pair of computer programs that interact with the user and the transport layer to provide a specific service to the user.

In the case of these application protocols, we should know what types of services they provide, how they work, and the options that we can use with these applications, and so on.

Nonstandard Application-Layer Protocols

A programmer can create a nonstandard application-layer program if he can write two programs that provide service to the user by interacting with the transport layer.

It is the creation of a nonstandard (proprietary) protocol, which does not even need the approval of the Internet authorities if privately used, that has made the Internet so popular worldwide.

A private company can create a new customized application protocol to communicate with all of its offices around the world using the services provided by the first four layers of the TCP/IP protocol suite without using any of the standard application programs. What is needed is to write programs, in one of the computer languages that use the available services provided by the transport-layer protocols.

5.1.2 Application-Layer Paradigms

Two application programs to interact with each other:

- One running on a computer somewhere in the world.
- The other running on another computer somewhere else in the world.

The two programs need to send messages to each other through the Internet infrastructure. Both application programs are able to request services and provide services.

Two paradigms have been developed during the lifetime of the Internet to answer this question: the ***client-server paradigm*** and the ***peer-to-peer paradigm***.

Traditional Paradigm: Client-Server

The traditional paradigm is called the **client-server paradigm**. It was the most popular paradigm until a few years ago. In this paradigm, the service provider is an application program, called the **server process**. It runs continuously, waiting for another application program, called the **client process**.

To make a connection through the Internet and ask for service. There are normally some server processes that can provide a specific type of service, but there are many clients that request service from any of these server processes.

The server process must be running all the time; the client process is started when the client needs to receive service. The client-server paradigm is similar to some available services out of the territory of the Internet.

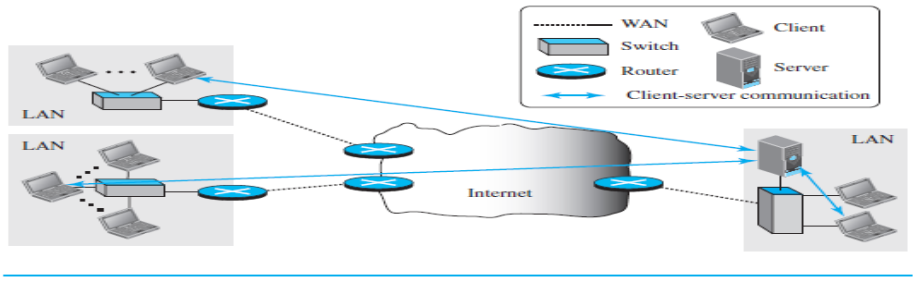
For example: A telephone directory center in any area can be thought of as a server; a subscriber that calls and asks for a specific telephone number can be thought of as a client. The directory center must be ready and available all the time; the subscriber can call the center for a short period when the service is needed.

Although the communication in the client-server paradigm is between two application programs, the role of each program is totally different.

Figure 25.2 shows an example of a client-server communication in which three clients communicate with one server to receive the services provided by this server.

One problem with this paradigm is that the concentration of the communication load is on the shoulder of the server, which means the server should be a powerful computer. Even a powerful computer may become overwhelmed if a large number of clients try to connect to the server at the same time. **Another problem** is that there should be a service provider willing to accept the cost and create a powerful server for a specific service, which means the service must always return some type of income for the server in order to encourage such an arrangement. Several traditional services are still using this paradigm, including the World Wide Web (WWW) and its vehicle Hypertext Transfer Protocol (HTTP), file transfer protocol (FTP), secure shell (SSH), e-mail, and so on.

Figure 25.2 *Example of a client-server paradigm*



New Paradigm: Peer-to-Peer

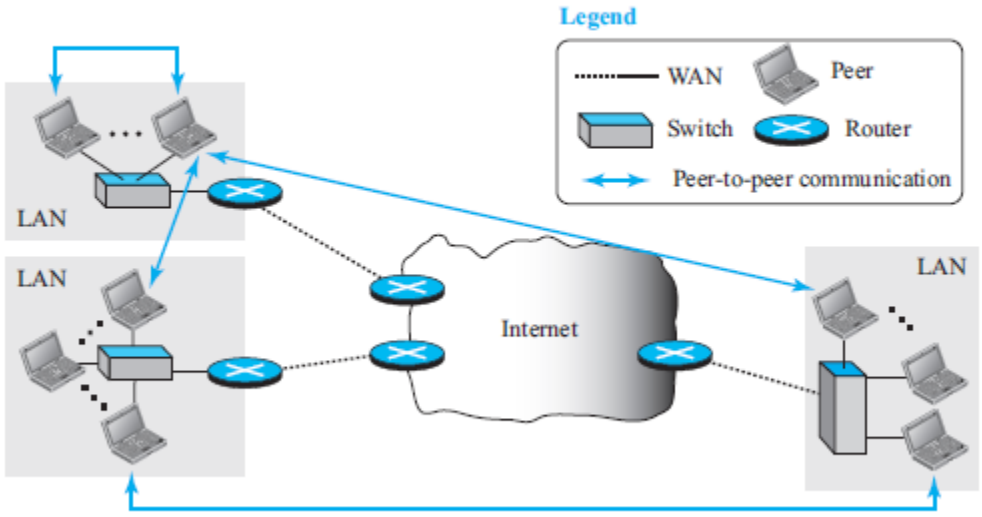
A new paradigm, called the **peer-to-peer paradigm** (often abbreviated *P2P paradigm*) has emerged to respond to the needs of some new applications. In this paradigm, there is no need for a server process to be running all the time and waiting for the client processes to connect. The responsibility is shared between peers. A computer connected to the Internet can provide service at one time and receive service at another time. A computer can even provide and receive services at the same time. Figure 25.3 shows an example of communication in this paradigm.

One of the areas that really fit in this paradigm is the Internet telephony. Communication by phone is indeed a peer-to-peer activity; no party needs to be running forever waiting for the other party to call. Another area in which the peer-to-peer paradigm can be used is when some computers connected to the Internet have something to share with each other.

For example: If an Internet user has a file available to share with other Internet users, there is no need for the file holder to become a server and run a server process all the time waiting for other users to connect and retrieve the file. Although the peer-to-peer paradigm has been proved to be easily scalable and cost-effective in eliminating the need for expensive servers to be running and maintained all the time, there are also some challenges. The main challenge has been security.

It is more difficult to create secure communication between distributed services than between those controlled by some dedicated servers. The other challenge is applicability; it appears that not all applications can use this new paradigm.

Figure 25.3 *Example of a peer-to-peer paradigm*



5.2 CLIENT-SERVER PROGRAMMING

In a client-server paradigm, communication at the application layer is between two running application programs called ***processes****:*

- A *client* : A client is a running program that initializes the communication by sending a request
- A *server*: A server is another application program that waits for a request from a client. The server handles the request received from a client, prepares a result, and sends the result back to the client.

1. Application Programming Interface

Computer program is normally written in a computer language with a predefined set of instructions that tells the computer what to do. A computer language has a set of instructions for mathematical operations, a set of instructions for string manipulation, a set of instructions for input/ output access. If we need a process to be able to communicate with another process, we need a new set of instructions to tell the lowest **four layers** of the TCP/IP suite to open the connection, send and receive data from the other end, and close the connection.

A set of instructions of this kind is normally referred to as an **application programming interface (API)**. An interface in programming is a set of instructions between **two** entities. **One** of the entities is the process at the application layer and the other is the ***operating system*** that encapsulates the first four layers of the

TCP/IP protocol suite. A computer manufacturer needs to build the first four layers of the suite in the operating system and include an API. In this way, the processes running at the application layer are able to communicate with the operating system when sending and receiving messages through the Internet. Several APIs have been designed for communication.

Three among them are common: **socket interface, Transport Layer Interface** (**TLI),** and **STREAM**.

Socket interface*:** the most common interface, to give a general idea of network communication at the application layer. ***Socket interface started in the early 1980s at UC Berkeley as part of a UNIX environment. The ***socket interface*** is a set of instructions that provide communication between the application layer and the operating system, as shown in Figure 25.4. It is a set of instructions that can be used by a process to communicate with another process. The idea of sockets allows us to use the set of all instructions already designed in a programming language for other ***sources and sinks***.

For example: In most computer languages, like C, C++, or Java, we have several instructions that can read and write data to other ***sources and sinks*** such as a keyboard (a source), a monitor (a sink), or a file (source and sink). We can use the same instructions to read from or write to sockets.

Figure 25.4 *Position of the socket interface*

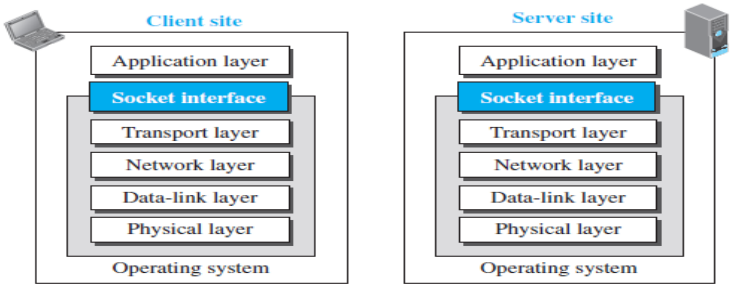
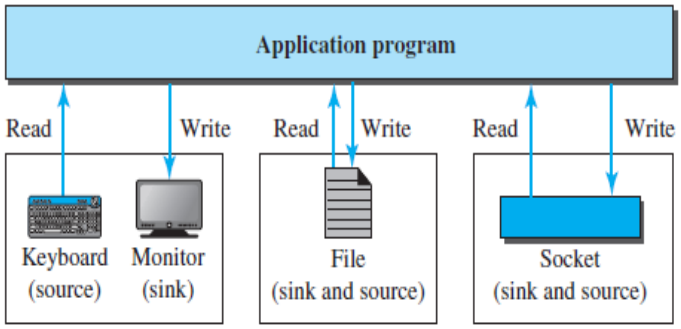


Figure 25.5 *Sockets used the same way as other sources and sinks*



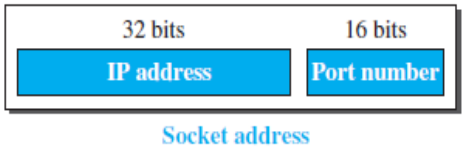
Socket Addresses

The interaction between a client and a server is two-way communication. In a two-way communication, we need a pair of addresses:

- ***local (sender) and***
- ***Remote (receiver).***

- The local address in one direction is the remote address in the other direction and vice versa. Since communication in the client-server paradigm is between two sockets, we need a pair of **socket addresses** for communication.
- A local socket address and a remote socket address.
- A socket address should first define the computer on which a client or a server is running.
- A computer in the Internet is uniquely defined by its IP address, a 32-bit integer in the current Internet version. Several client or server processes may be running at the same time on a computer, which means that we need another identifier to define the specific client or server involved in the communication.
- An application program can be defined by a port number, a 16-bit integer. This means that a socket address should be a combination of an IP address and a port number as shown in Figure 25.7.

Figure 25.7 *A socket address*



A socket defines the end-point of the communication. A socket is identified by a pair of socket addresses, a local and a remote.

Server Site

The server needs a local (server) and a remote (client) socket address for communication.

Local Socket Address:

- The local (server) socket address is provided by the operating system.
- The operating system knows the IP address of the computer on which the server process is running.
- The port number of a server process, however, needs to be assigned.
- If the server process is a standard one defined by the Internet authority, a port number is already assigned to it.

For example: The assigned port number for a **Hypertext Transfer Protocol (HTTP)** is the integer **80**, which cannot be used by any other process. If the server process is not standard, the designer of the server process can choose a port number, in the range defined by the Internet authority, and assign it to the process. When a server starts running, it knows the local socket address.

Remote Socket Address

- The remote socket address for a server is the socket address of the client that makes the connection. Since the server can serve many clients, it does not know beforehand the remote socket address for communication.

- The server can find this socket address when a client tries to connect to the server. The client socket address, which is contained in the request packet sent to the server, becomes the remote socket address that is used for responding to the client.
- Although the local socket address for a server is fixed and used during its lifetime, the remote socket address is changed in each interaction with a different client.

Client Site

The client also needs a local (client) and a remote (server) socket address for communication.

Local Socket Address:

- The local (client) socket address is also provided by the operating system. The operating system knows the IP address of the computer on which the client is running.
- The port number, however, is a 16-bit temporary integer that is assigned to a client process each time the process needs to start the communication.
- The port number, however, needs to be assigned from a set of integers defined by the Internet authority and called the ephemeral (temporary) port numbers.
- The operating system, however, needs to guarantee that the new port number is not used by any other running client process.
- The operating system needs to remember the port number to be able to redirect the response received from the server process to the client process that sent the request.

Remote Socket Address:

Finding the remote (server) socket address for a client, however, needs more work. When a client process starts, it should know the socket address of the server it wants to connect to. We will have two situations in this case.

1. The user who starts the client process knows both the server port number and IP address of the computer on which the server is running. This usually occurs in situations when we have written client and server applications and we want to test them.
2. Although each standard application has a well-known port number, most of the time, we do not know the IP address. This happens in situations such as when we need to contact a web page, send an e-mail to a friend, copy a file from a remote site, and so on. In these situations, the server has a name, an identifier that uniquely defines the server process.

Examples of these identifiers are URLs, such as www.xxx.yyy, or e-mail addresses, such as xxxx@yyyy.com.

- The client process should now change this identifier (name) to the corresponding server socket address.
- The client process normally knows the port number because it should be a well-known port number, but the IP address can be obtained using another client-server application called the *Domain Name System (DNS).*

2. Using Services of the Transport Layer

A pair of processes provides services to the users of the Internet, human or programs. Pair of processes, however, needs to use the services provided by the transport layer for communication because there is no physical communication at the application layer.

There are three common transport-layer protocols in the TCP/IP suite:

- ***UDP(USER DATAGRAM PROTOCOL)***
- ***TCP(TRANSMISSION CONTROL PROTOCOL)***
- ***SCTP(STREAM CONTROL TRANSMISSION PROTOCOL)***

UDP Protocol

UDP provides connectionless, unreliable, datagram service. Connectionless service means that there is no **logical connection** between the two ends exchanging messages. Each message is an independent entity encapsulated in a datagram. UDP does not see any relation (connection) between consequent datagram's coming from the same source and going to the same destination.

UDP is not a reliable protocol. Although it may check that the data is not corrupted during the transmission, it does not ask the sender to resend the corrupted or lost datagram. An advantage of UDP is, ***it is message-oriented***. It gives boundaries to the messages exchanged. An application program may be designed to use UDP if it is sending small messages and the simplicity and speed is more important for the application than reliability.

For example*: some management and multimedia applications fit in this category.*

TCP Protocol

TCP provides connection-oriented, reliable, byte-stream service. TCP requires that **two ends** first create a logical connection between themselves by exchanging some connection-establishment packets. This phase, which is sometimes called ***handshaking***, establishes some parameters between the two ends, including the size of the data packets to be exchanged, the size of buffers to be used for holding the **chunk**s of data until the whole message arrives.

After the handshaking process, the two ends can send chunks of data in segments in each direction. By numbering the bytes exchanged, the continuity of the bytes can be checked.

For example: If some bytes are lost or corrupted, the receiver can request the resending of those bytes, which makes TCP a reliable protocol. TCP also can provide ***flow control and congestion control*.**

One problem with the TCP protocol is that it is not message-oriented. It does not put boundaries on the messages exchanged. Most of the standard applications that need to send long messages and require reliability may benefit from the service of the TCP.

SCTP Protocol

- SCTP provides a service which is a combination of the two other protocols. Like TCP, SCTP provides a connection-oriented, reliable service, but it is not byte stream oriented. It is a message-oriented protocol like UDP.
- SCTP can provide multi-stream service by providing multiple network-layer connections.
- SCTP is normally suitable for any application that needs reliability and at the same time needs to remain connected, even if a failure occurs in one network-layer connection.

3. Iterative Communication Using UDP

Communication between a client program and a server program can occur ***iteratively or concurrently***. Several ***client programs*** can access the same server program at the sametime; the server program can be designed to respond ***iteratively or concurrently***. An iterative server can process one client request at a time; it receives ***a request, processes it,*** and sends the response to the requestor before handling another request.

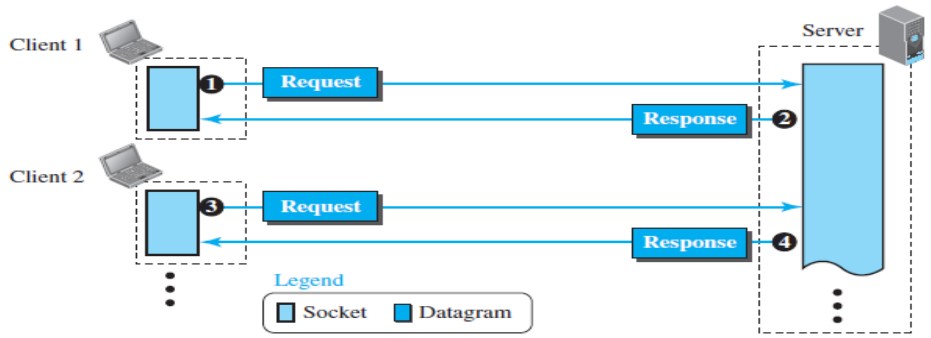
When the server is handling the request from a client, the requests from other clients, and even other requests from the same client, need to be queued at the server site and wait for the server to be freed. The received and queued requests are handled in the ***first-in, first-out fashion.***

Sockets Used for UDP

In UDP communication, the client and server use only one socket each. The socket created at the server site lasts forever; the socket created at the client site is closed (destroyed) when the client process terminates. Figure 25.8 shows the lifetime of the sockets in the server and client processes.

Different clients use different sockets, but the server creates only one socket and changes only the remote socket address each time a new client makes a connection. This is logical, because the server does know its own socket address, but does not know the socket addresses of the clients who need its services; it needs to wait for the client to connect before filling this part of the socket address.

Figure 25.8 *Sockets for UDP communication*



Flow Diagram

UDP provides a connectionless service, in which a client sends a request and the server sends back a response. Figure 25.9 shows a simplified flow diagram for iterative communication. There are **multiple clients**, but only one server. Each client is served in each iteration of the loop in the server.

Note: That there is no connection establishment or connection termination. Each client sends **a single datagram** and receives **a single datagram**. If a client wants to send two datagram's, it is considered as two clients for the server. The **second datagram** needs to wait for its turn. The diagram also shows the status of the socket after each action.

Server Process

The server makes a ***passive open***, in which it becomes ready for the communication, but it waits until a client process makes the connection. It creates an empty socket. It then binds the socket to the server and the well-know port, in which only part of the socket (the server socket address) is filled (binding can happen at the time of creation depending on the underlying language).

The server then issues a receive request command, which blocks until it receives a request from a client. The server then fills the rest of the socket (the client socket section) from the information obtained in the

request. The request is the process and the response is sent back to the client. The server now starts another iteration waiting for another request to arrive (an infinite loop). Note that in each iteration, the socket becomes only half-filled again; the client socket address is erased. It is totally filled only when a request arrives.

Client Process

The client process makes an ***active open***. It starts a connection. It creates an empty socket and then issues the send command, which fully fills the socket, and sends the request. The client then issues a receive command, which is blocked until a response arrives from the server. The response is then handled and the socket is destroyed.

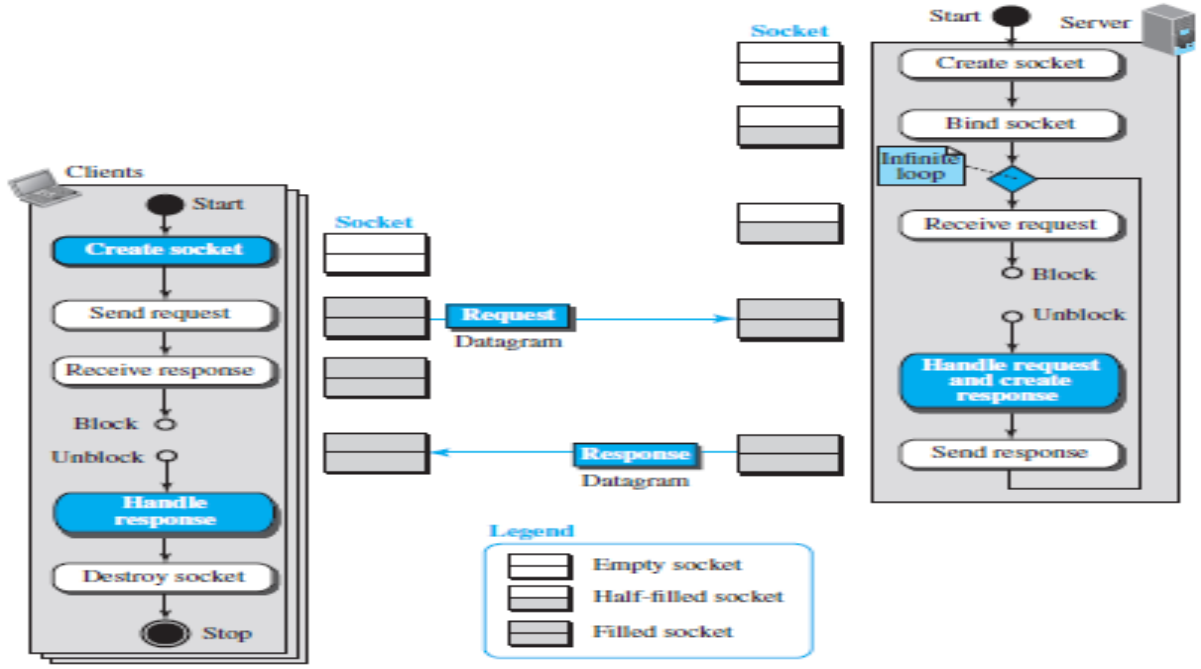


Figure 25.9 *Flow diagram for iterative UDP communication*

4. Iterative Communication Using TCP

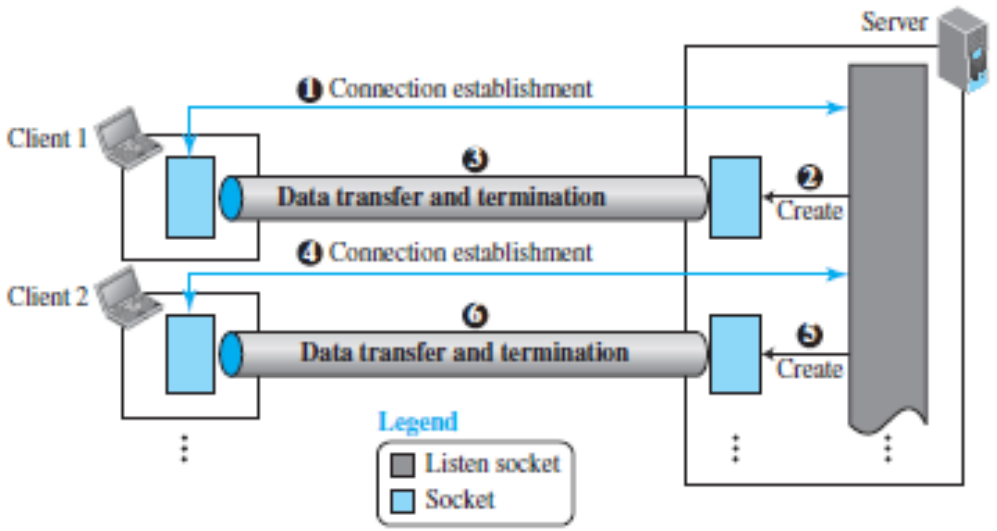
TCP is a connection-oriented protocol. Before sending or receiving data, a connection needs to be established between the client and the server. After the connection is established, the two parties can send and receive **chunks** of data as long as they have data to do so.

Sockets Used in TCP

The TCP server uses two different sockets, one for connection establishment and the other for data transfer. We call the first one the ***listen socket*** and the second the ***socket***. The reason for having two types of sockets is to separate the **connection phase** from the **data exchange phase**.

A server uses a listen socket to listen for a new client trying to establish connection. After the connection is established, the server creates a socket to exchange data with the client and finally to terminate the connection. The client uses only one socket for both connection establishment and data exchange (see Figure 25.10).

Figure 25.10 *Sockets used in TCP communication*



Flow Diagram

Figure 25.11 shows a simplified flow diagram for iterative communication using TCP. There are multiple clients, but only one server. Each client is served in each iteration of the loop. The flow diagram is almost similar to the one for UDP, but there are differences that we explain for each site.

Server Process

In Figure 25.11, the TCP server process, like the UDP server process, creates a socket and binds it, but these two commands create the listen socket to be used only for the connection establishment phase. The server process then calls the *listen* procedure, to allow the operating system to start accepting the clients, completing the connection phase, and putting them in the waiting list to be served.

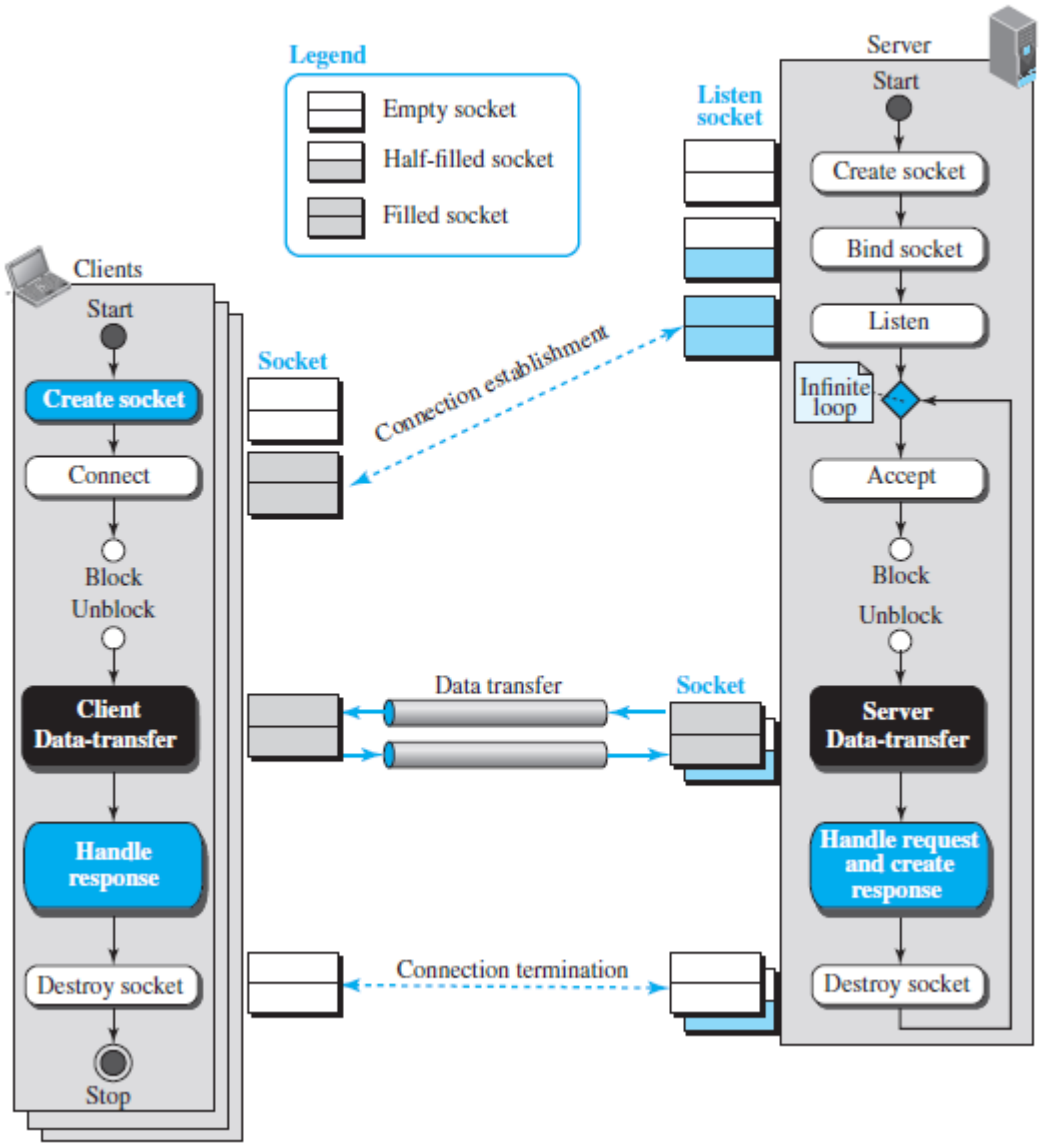
The server process now starts a loop and serves the clients one by one. In each iteration, the server process issues the *accept* procedure that removes one client from the waiting list of the connected clients for serving. If the list is empty, the *accept* procedure blocks until there is a client to be served. When the accept procedure returns, it creates a new socket for data transfer.

The server process now uses the client socket address obtained during the connection establishment to fill the remote socket address field in the newly created socket. At this time the client and server can exchange data.

Client Process

The client flow diagram is almost similar to the UDP version except that the *client data-transfer* box needs to be defined for each specific case. We do so when we write a specific program later.

Figure 25.11 *Flow diagram for iterative TCP communication*



5. Concurrent Communication

A concurrent server can process several client requests at the same time. This can be done using the available provisions in the underlying programming language.

- In C, a server can create several child processes, in which a child can handle a client.
- In Java, threading allows several clients to be handled by each thread.

5.3 WORLD WIDE WEB AND HTTP

5.3.1 World Wide Web

The Web today is a repository of information in which the documents, called ***web pages,*** are **distributed** all over the world and related documents are **linked** together.

The two terms in the above statement:

Distributed: Distribution allows the growth of the Web. Each web server in the world can add a new web page to the repository and announce it to all Internet users without overloading a few servers.

Linked: Linking allows one web page to refer to another web page stored in another server somewhere else in the world. The linking of web pages was achieved using a concept called ***hypertext,*** which was introduced many years before the advent of the Internet. The idea was to use a machine that automatically retrieved another document stored in the system when a link to it appeared in the document.

The Web implemented this idea electronically to allow the linked document to be retrieved when the link was clicked by the user.

USES:

- The term *hypertext,* coined to mean linked text documents, has been changed to ***hypermedia,*** to show that a web page can be a text document, an image, an audio file, or a video file.
- The purpose of the Web has gone beyond the simple retrieving of linked documents.
- The Web is used to provide electronic shopping and gaming. One can use the Web to listen to radio programs or view television programs whenever one desires without being forced to listen to or view these programs when they are broadcast.

Architecture

The WWW today is a distributed client-server service, in which a client using a browser can access a service using a server. The service provided is distributed over many locations called *sites.* Each site holds one or more web pages. Each web page can contain some links to other web pages in the same or other sites. A web page can be two types

Simple: A simple web page has no links to other web pages

Composite: Composite web page has one or more links to other web pages.

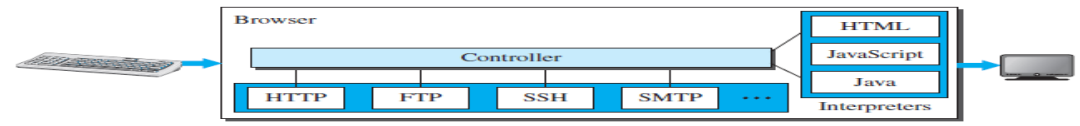
Each web page is a file with a name and address.

Web Client (Browser)

A variety of vendors offer commercial **browsers** that interpret and display a web page, and all of them use nearly the same architecture. Each browser usually consists of three parts:

- ***A controller,***
- ***Client protocols, and***
- ***Interpreters. (see Figure 26.2)***

Figure 26.2 *Browser*



- The controller receives input from the keyboard or the mouse and uses the client programs to access the document. After the document has been accessed, the controller uses one of the interpreters to display the document on the screen.
- The client protocol can be one of the protocols described later, such as HTTP or FTP.
- The interpreter can be HTML, Java, or JavaScript, depending on the type of document. Some commercial browsers include Internet Explorer, Netscape Navigator, and Firefox.

Web Server

The web page is stored at the server. Each time a request arrives, the corresponding document is sent to the client. To improve efficiency, servers normally store requested files in a cache in memory; memory is faster to access than a disk.

A server can also become more efficient through multithreading or multiprocessing. A server can answer more than one request at a time. Some popular web servers include Apache and Microsoft Internet Information Server.

Uniform Resource Locator (URL)

A web page, as a file, needs to have a unique identifier to distinguish it from other web pages. To define a web page, we need three identifiers: ***host, port,*** **and** ***path***. However, before defining the web page, we need to tell the browser what client-server application we want to use, which is called **the *protocol*.**

This means we need four identifiers to define the web page. The first is the type of vehicle to be used to fetch the web page; the last three make up the combination that defines the destination object (web page).

***Protocol*:** The first identifier is the abbreviation for the client-server program that we need in order to access the web page. Although most of the time the protocol is HTTP (Hyper-Text Transfer Protocol), we can also use other protocols such as FTP (File Transfer Protocol).

***Host*:** The host identifier can be the IP address of the server or the unique name given to the server. IP addresses can be defined in dotted decimal notation, (such as 64.23.56.17); the name is normally the domain name that uniquely defines the host, such as *forouzan.com*.

Port. The port, a 16-bit integer, is normally predefined for the client-server application.

For example, if the HTTP protocol is used for accessing the web page, the well-known port number is 80. However, if a different port is used, the number can be explicitly given.

***Path*:** The path identifies the location and the name of the file in the underlying operating system. The format of this identifier normally depends on the operating system. In UNIX, a path is a set of directory names followed by the file name, all separated by a slash. For example, */top/next/last/myfile* is a path that uniquely defines a file named *myfile*, stored in the directory *last*, which itself is part of the directory *next*, which itself is under the directory *top*. In other words, the path lists the directories from the top to the bottom, followed by the file name.

To combine these four pieces together, the **uniform resource locator (URL)** has been designed; it uses three different separators between the four pieces as shown below:

protocol://host/path	Used most of the time
protocol://host:port/path	Used when port number is needed

Web Documents

The documents in the WWW can be grouped into three broad categories:

- ***static,***
- ***dynamic,***
- ***Active.***

Static Documents

- **Static documents** are fixed-content documents that are created and stored in a server. The client can get a copy of the document only. The contents of the file are determined when the file is created, not when it is used. The contents in the server can be changed, but the user cannot change them.
- When a client accesses the document, a copy of the document is sent. The user can then use a browser to see the document. Static documents are prepared using one of several languages: *HyperText Markup Language* (HTML), *Extensible Markup Language* (XML), *Extensible Style Language* (XSL), and *Extensible Hypertext Markup Language* (XHTML).

Dynamic Documents

- A **dynamic document** is created by a web server whenever a browser requests the document. When a request arrives, the web server runs an application program or a script that creates the dynamic document.
- The server returns the result of the program or script as a response to the browser that requested the document. Because a fresh document is created for each request, the contents of a dynamic document may vary from one request to another.
- A very simple example of a dynamic document is the retrieval of the time and date from a server. Time and date are kinds of information that are dynamic in that they change from moment to moment.
- The client can ask the server to run a program such as the *date* program in UNIX and send the result of the program to the client. The *Common Gateway Interface* (*CGI*) was used to retrieve a dynamic document in the past, today's options include one of the scripting languages such as *Java Server Pages* (*JSP*), which uses the Java language for scripting, or *Active Server Pages* (*ASP*), a Microsoft product that uses Visual Basic language for scripting, or *ColdFusion*, which embeds queries in a Structured Query Language (SQL) database in the HTML document.

Active Documents

For many applications, we need a program or a script to be run at the client site. These are called ***active documents.*** For example, suppose we want to run a program that creates animated graphics on the screen or a program that interacts with the user. The program definitely needs to be run at the client site where the animation or interaction takes place. When a browser requests an active document, the server sends a copy of

the document or a script. The document is then run at the client (browser) site. One way to create an active document is to use *Java applets*, a program written in Java on the server. It is compiled and ready to be run. The document is in byte-code (binary) format. Another way is to use *Java Scripts* but download and run the script at the client site.

5.3.2 Hyper-Text Transfer Protocol (HTTP)

The **Hyper-Text Transfer Protocol (HTTP)** is used to define how the client-server programs can be written to retrieve web pages from the Web.

An HTTP client sends a request.

An HTTP server returns a response.

The server uses the port number 80.

The client uses a temporary port number. HTTP uses the services of TCP, which, as discussed before, is a **connection-oriented** and **reliable protocol**. This means that, before any transaction between the client and the server can take place, a connection needs to be established between them. After the transaction, the connection should be terminated. The client and server, does not need to worry about errors in messages exchanged or loss of any message, because the TCP is reliable and will take care.

Non-persistent versus Persistent Connections

The hypertext concept embedded in web page documents may require several requests and responses. If the web pages, objects to be retrieved, are located on different servers, we do not have any other choice than to create a new TCP connection for retrieving each object.

To find the objects located on the server, we have two choices: to retrieve each object using a new TCP connection or to make a TCP connection and retrieve them all.

- ***The first method is referred to as a non-persistent connection,***
- ***The second as a persistent connection.***

Non-persistent Connections

In a **non-persistent connection**, one TCP connection is made for each request/response. The following lists the steps in this strategy:

1. The client opens a TCP connection and sends a request.

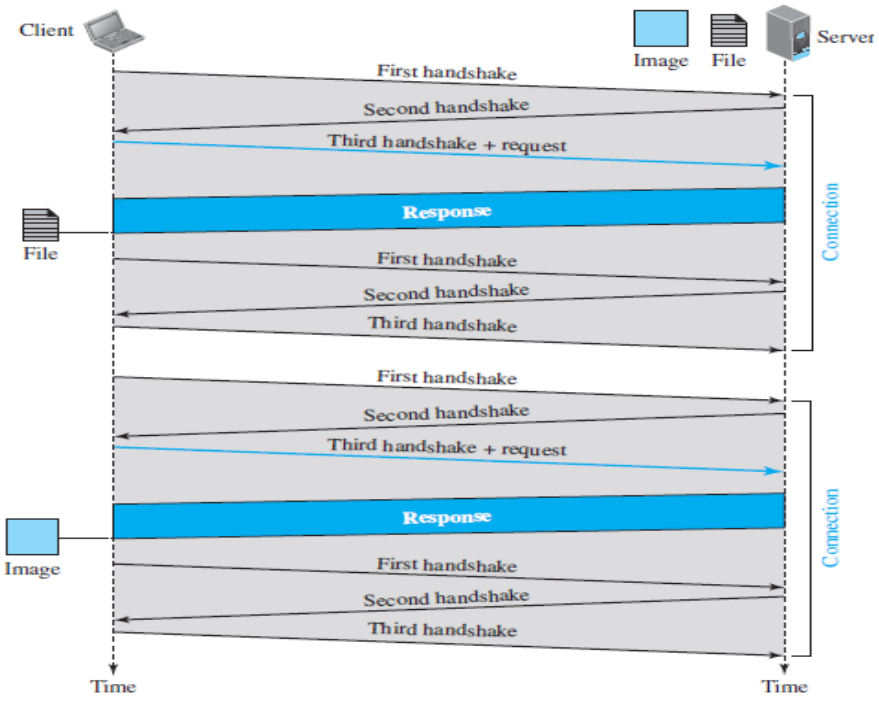
2. The server sends the response and closes the connection.

3. The client reads the data until it encounters an end-of-file marker; it then closes the Connection

EXAMPLE:

- The client needs to access a file that contains one link to an image. The text file and image are located on the same server. Here we need two connections.
- For each connection, TCP requires at least three handshake messages to establish the connection, but the request can be sent with the third one.
- After the connection is established, the object can be transferred.

- After receiving an object, another three handshake messages are needed to terminate the connection. This means that the client and server are involved in two connection establishments and two connection terminations.
- If the transaction involves retrieving 10 or 20 objects, the round trip times spent for these handshakes add up to a big overhead.



Persistent Connections

In a persistent connection, the server leaves the connection open for more requests after sending a response.

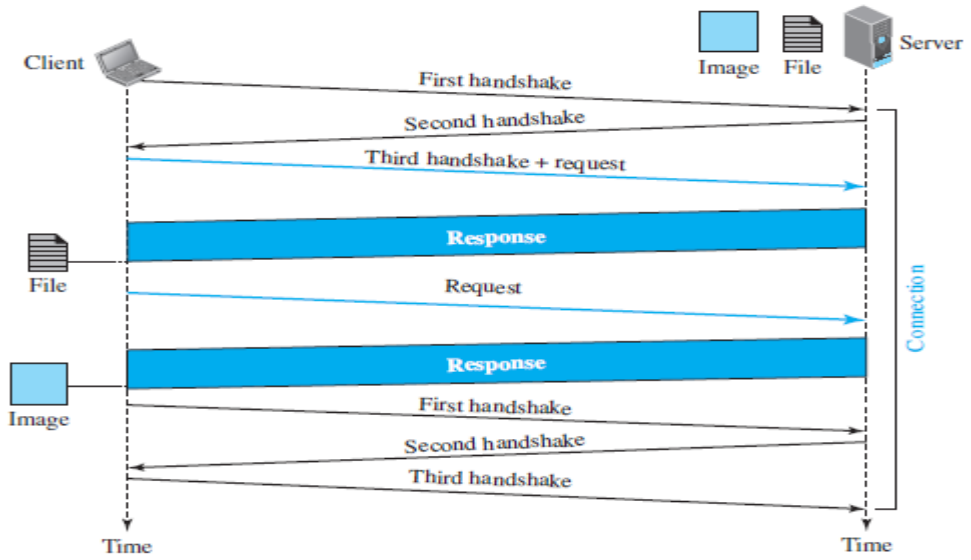
The server can close the connection at the request of a client or if a time-out has been reached.

The sender usually sends the length of the data with each response.

There are some occasions when the sender does not know the length of the data. This is the case when a document is created dynamically or actively. In these cases, the server informs the client that the length is not known and closes the connection after sending the data so the client knows that the end of the data has been reached.

Time and resources are saved using persistent connections. Only one set of buffers and variables needs to be set for the connection at each site. The round trip time for connection establishment and connection termination is saved.

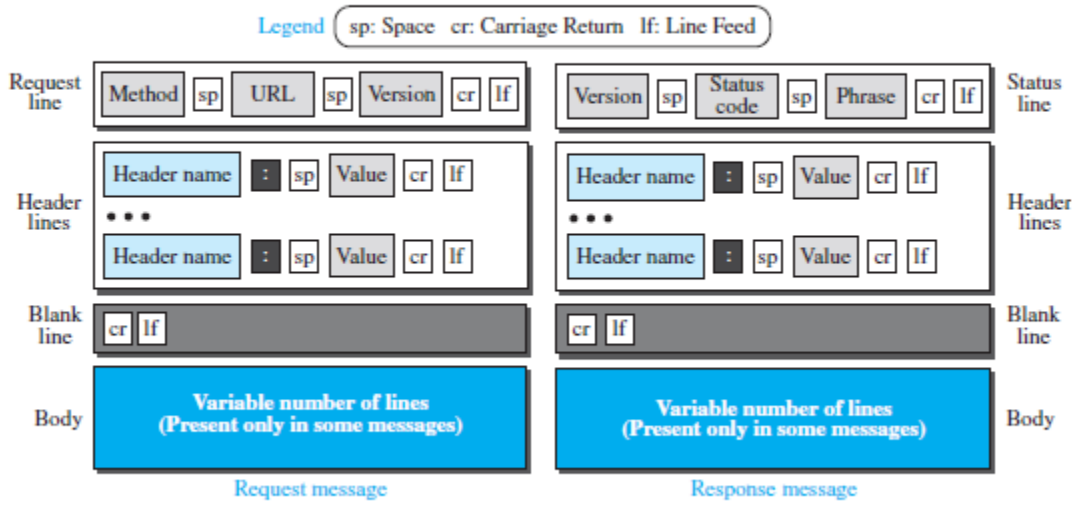
Example



Message Formats

The HTTP protocol defines the format of the request and response messages. We have put the two formats next to each other for comparison. Each message is made of four sections. The first section in the request message is called the *request line;* the first section in the response message is called the *status line*. The other three sections have the same names in the request and response messages. However, the similarities between these sections are only in the names; they may have different contents. We discuss each message type separately.

Figure 26.5 *Formats of the request and response messages*



Request Message

- The first line in a request message is called a request line. There are three fields in this line separated by one space and terminated by two characters (carriage return and line feed) as shown in Figure 26.5. The fields are called *method*, *URL*, and *version*.

- The method field defines the request types. In version 1.1 of HTTP, several methods are defined, as shown in Table 26.1. Most of the time, the client uses the GET method to send a request.
- The body of the message is empty.
- The HEAD method is used when the client needs only some information about the web page from the server, such as the last time it was modified. It can also be used to test the validity of a URL.
- The response message in this case has only the header section; the body section is empty. The PUT method is the inverse of the GET method; it allows the client to post a new web page on the server (if permitted).
- The POST method is similar to the PUT method, but it is used to send some information to the server to be added to the web page or to modify the web page.
- The TRACE method is used for debugging; the client asks the server to echo back the request to check whether the server is getting the requests.
- The DELETE method allows the client to delete a web page on the server if the client has permission to do so. The CONNECT method was originally made as a reserve method; it may be used by proxy servers, as discussed later. Finally, the OPTIONS method allows the client to ask about the properties of a web page.
- The second field, URL, was discussed earlier in the chapter. It defines the address and name of the corresponding web page. The third field, version, gives the version of the protocol; the most current version of HTTP is 1.1. After the request line, we can have zero or more *request header* lines.
- Each header line sends additional information from the client to the server. For example, the client can request that the document be sent in a special format.
- Each header line has a header name, a colon, a space, and a header value (see Figure 26.5). Table 26.2 shows some header names commonly used in a request.
- The value field defines the values associated with each header name. The list of values can be found in the corresponding RFCs.
- The body can be present in a request message. Usually, it contains the comment to be sent or the file to be published on the website when the method is PUT or POST.

Table 26.1 *Methods*

Method	*Action*
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
PUT	Sends a document from the client to the server
POST	Sends some information from the client to the server
TRACE	Echoes the incoming request
DELETE	Removes the web page
CONNECT	Reserved
OPTIONS	Inquires about available options

Table 26.2 *Request header names*

Header	*Description*
User-agent	Identifies the client program
Accept	Shows the media format the client can accept
Accept-charset	Shows the character set the client can handle
Accept-encoding	Shows the encoding scheme the client can handle
Accept-language	Shows the language the client can accept
Authorization	Shows what permissions the client has
Host	Shows the host and port number of the client
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Cookie	Returns the cookie to the server (explained later)
If-Modified-Since	If the file is modified since a specific date

Response Message

- The format of the response message is also shown in Figure 26.5. A response message consists of a status line, header lines, a blank line, and sometimes a body.
- The first line in a response message is called the *status line*. There are three fields in this line separated by spaces and terminated by a carriage return and line feed.
- The first field defines the version of HTTP protocol, currently 1.1. The status code field defines the status of the request. It consists of three digits. Whereas the codes in the 100 range are only informational, the codes in the 200 range indicate a successful request. The codes in the 300 range redirect the client to another URL, and the codes in the 400 range indicate an error at the client site. Finally, the codes in the 500 range indicate an error at the server site. The status phrase explains the status code in text form.
- After the status line, we can have zero or more *response header* lines. Each header line sends additional information from the server to the client.
- ***For example***, the sender can send extra information about the document. Each header line has a header name, a colon, a space, and a header value. We will show some header lines in the examples at the end of this section. Table 26.3 shows some header names commonly used in a response message.

Table 26.3 *Response header names*

Header	*Description*
Date	Shows the current date
Upgrade	Specifies the preferred communication protocol
Server	Gives information about the server
Set-Cookie	The server asks the client to save a cookie
Content-Encoding	Specifies the encoding scheme
Content-Language	Specifies the language
Content-Length	Shows the length of the document
Content-Type	Specifies the media type
Location	To ask the client to send the request to another site
Accept-Ranges	The server will accept the requested byte-ranges
Last-modified	Gives the date and time of the last change

The body contains the document to be sent from the server to the client. The body is present unless the response is an error message.

5.4FTP

File Transfer Protocol (FTP) is the standard protocol provided by TCP/IP for copying a file from one host to another.

For example: Two systems may use different file name conventions. Two systems may have different ways to represent data. Two systems may have different directory structures. All of these problems have been

solved by FTP in a very simple and elegant approach. Although we can transfer files using HTTP, FTP is a better choice to transfer large files or to transfer files using different formats.

The client has three components:

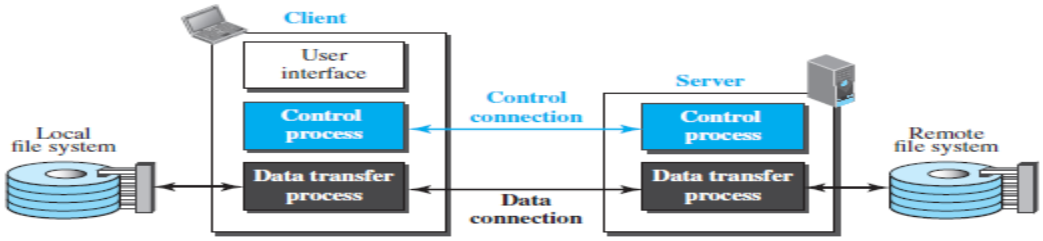
- ***The user interface,***
- ***The client control process, and***
- ***The client data transfer process.***

The server has two components:

- ***The server control process and***
- ***The server data transfer process.***

The control connection is made between **the control processes**. The data connection is made between the **data transfer processes**. Separation of commands and data transfer makes FTP more efficient. The control connection uses very simple rules of communication. We need to transfer only a line of command or a line of response at a time. **The data connection**, on the other hand, needs more complex rules due to the variety of data types transferred.

Figure 26.10 *FTP*



5.4.1 Two Connections

The two connections in FTP have different lifetimes. The control connection remains connected during the entire interactive **FTP session**. The data connection is opened and then closed for each file transfer activity. It opens each time commands that involve transferring files are used, and it closes when the file is transferred. When a user starts an FTP session, the control connection opens. While the control connection is open, the data connection can be opened and closed multiple times if several files are transferred. FTP uses two well-known TCP ports: ***port 21 is used for the control connection, and port 20 is used for the data connection***.

5.4.2 Control Connection

For control communication, FTP uses the same approach as TELNET (discussed later). It uses the NVT ASCII character set as used by TELNET. Communication is achieved through commands and responses. This simple method is adequate for the control connection because we send one command (or response) at a time. Each line is terminated with a two-character (carriage return and line feed) end-of-line token.

During this control connection, commands are sent from the client to the server and responses are sent from the server to the client. Commands, which are sent from the FTP client control process, are in the form of ASCII uppercase, which may or may not be followed by an argument. Some of the most common commands are shown in Table 26.4.

Command	Argument(s)	Description
ABOR		Abort the previous command
CDUP		Change to parent directory
CWD	Directory name	Change to another directory
DELE	File name	Delete a file
LIST	Directory name	List subdirectories or files
MKD	Directory name	Create a new directory
PASS	User password	Password
PASV		Server chooses a port
PORT	Port identifier	Client chooses a port
PWD		Display name of current directory
QUIT		Log out of the system
RETR	File name(s)	Retrieve files; files are transferred from server to client
RMD	Directory name	Delete a directory
RNFR	File name (old)	Identify a file to be renamed
RNTO	File name (new)	Rename the file
STOR	File name(s)	Store files; file(s) are transferred from client to server
STRU	F, R, or P	Define data organization (F: file, R: record, or P: page)
TYPE	A, E, I	Default file type (A: ASCII, E: EBCDIC, I: image)
USER	User ID	User information
MODE	S, B, or C	Define transmission mode (S: stream, B: block, or C: compressed

Every FTP command generates at least one response. A response has two parts: a three-digit number followed by text. The numeric part defines the code; the text part defines needed parameters or further explanations. The first digit defines the status of the command. The second digit defines the area in which the status applies. The third digit provides additional information. Table 26.5 shows some common responses.

Table 26.5 *Some responses in FTP*

Code	Description	Code	Description
125	Data connection open	250	Request file action OK
150	File status OK	331	User name OK; password is needed
200	Command OK	425	Cannot open data connection
220	Service ready	450	File action not taken; file not available
221	Service closing	452	Action aborted; insufficient storage
225	Data connection open	500	Syntax error; unrecognized command
226	Closing data connection	501	Syntax error in parameters or arguments
230	User login OK	530	User not logged in

5.4.3 Data Connection

The data connection uses the well-known port 20 at the server site. However, the creation of a data connection is different from the control connection. The following shows the steps:

1. The client, not the server, issues a passive open using an ephemeral port. This must be done by the client because it is the client that issues the commands for transferring files.

2. Using the PORT command the client sends this port number to the server.

3. The server receives the port number and issues an active open using the well-known port 20 and the received ephemeral port number.

Communication over Data Connection

The purpose and implementation of the data connection are different from those of the control connection. We want to transfer files through the data connection. The client must define the type of file to be transferred, the

structure of the data, and the transmission mode. Before sending the file through the data connection, we prepare for transmission through the control connection. The heterogeneity problem is resolved by defining three attributes of communication:

- ***File type,***
- ***Data structure, and***
- ***Transmission mode.***

File Type

FTP can transfer one of the following file types across the data connection: ASCII file, EBCDIC file, or image file.

Data Structure

FTP can transfer a file across the data connection using one of the following interpretations of the structure of the data: *file structure, record structure,* or *page structure*. The file structure format (used by default) has no structure. It is a continuous stream of bytes. In the record structure, the file is divided into *records*. This can be used only with text files. In the page structure, the file is divided into pages, with each page having a page number and a page header. The pages can be stored and accessed randomly or sequentially.

Transmission Mode

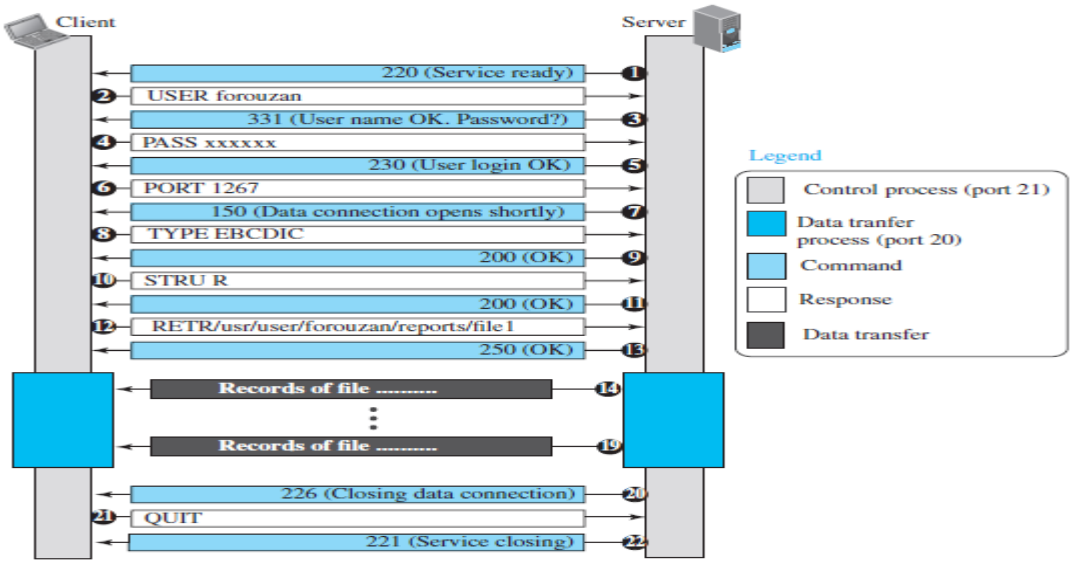
FTP can transfer a file across the data connection using one of the following three transmission modes: ***stream mode:*** The stream mode is the default mode; data are delivered from FTP to TCP as a continuous stream of bytes.

Block mode OR* compressed *mode*:** In the block mode, data can be delivered from FTP to TCP in blocks. In this case, each block is preceded by a **3-byte** header. The first byte is called the ***block descriptor; the next two **bytes define the size of the block in bytes.**

File Transfer

File transfer occurs over the data connection under the control of the commands sent over the control connection. However, we should remember that file transfer in FTP means one of three things: *retrieving a file* (server to client), *storing a file* (client to server), and *directory listing* (server to client).

EXAMPLE:



5.4.4 Security for FTP

The FTP protocol was designed when security was not a big issue. Although FTP requires a password, the password is sent in plaintext (unencrypted), which means it can be intercepted and used by an attacker. The data transfer connection also transfers data in plaintext, which is insecure. To be secure, one can add a Secure Socket Layer between the FTP application layer and the TCP layer. In this case FTP is called SSL-FTP.

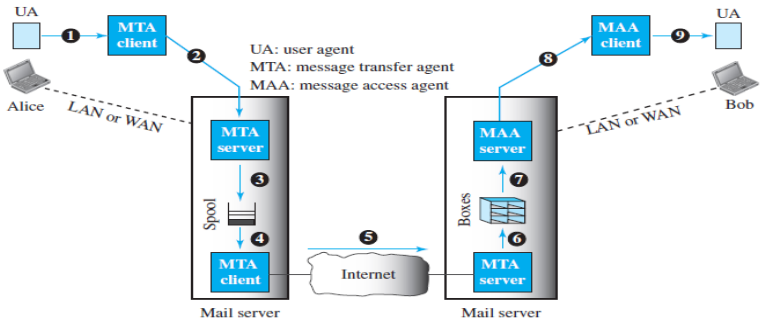
5.5 ELECTRONIC MAIL

- Electronic mail (or e-mail) allows users to exchange messages. The nature of this application is different from other applications discussed so far.
- In an application such as HTTP or FTP, the server program is running all the time, waiting for a request from a client.
- When the request arrives, the server provides the service. There is a request and there is a response. In the case of electronic mail, the situation is different.
- First, e-mail is considered a one-way transaction. When Alice sends an email to Bob, she may expect a response, but this is not a mandate. Bob may or may not respond. If he does respond, it is another one-way transaction.
- Second, it is neither feasible nor logical for Bob to run a server program and wait until someone sends an e-mail to him. Bob may turn off his computer when he is not using it.
- This means that the idea of client/server programming should be implemented in another way: using some intermediate computers (servers).

5.5.1 Architecture

The architecture of e-mail, we give a common scenario, as shown in Figure 26.12. Another possibility is the case in which Alice or Bob is directly connected to the corresponding mail server, in which LAN or WAN connection is not required, but this variation in the scenario does not affect

Figure 26.12 *Common scenario*



In the common scenario, the sender and the receiver of the e-mail, Alice and Bob respectively, are connected via a LAN or a WAN to two mail servers. The administrator has created one mailbox for each user where the received messages are stored. A ***mailbox*** is part of a server hard drive, a special file with permission restrictions. Only the owner of the mailbox has access to it. The administrator has also created a queue (spool) to store messages waiting to be sent.

A simple e-mail from Alice to Bob takes nine different steps, as shown in the figure.

Alice and Bob use three different *agents*:

- ***A user agent (UA),***
- ***A message transfer agent (MTA), and***
- ***A message access agent (MAA).***

- When Alice needs to send a message to Bob, she runs a UA program to prepare the message and send it to her mail server. The mail server at her site uses a queue (spool) to store messages waiting to be sent. The message, however, needs to be sent through the Internet from Alice's site to Bob's site using an MTA.
- Here two message transfer agents are needed: one client and one server.
- Like most client-server programs on the Internet, the server needs to run all the time because it does not know when a client will ask for a connection.
- The client, on the other hand, can be triggered by the system when there is a message in the queue to be sent. The user agent at the Bob site allows Bob to read the received message.
- Bob later uses an MAA client to retrieve the message from an MAA server running on the second server. There are two important points we need to emphasize here.
- First, Bob cannot bypass the mail server and use the MTA server directly. To use the MTA server directly, Bob would need to run the MTA server all the time because he does not know when a message will arrive. This implies that Bob must keep his computer on all the time if he is connected to his system through a LAN. If he is connected through a WAN, he must keep the connection up all the time. Neither of these situations is feasible today.
- Second, note that Bob needs another pair of client-server programs: message access programs. This is because an MTA client-server program is a *push* program: the client pushes the message to the server. Bob needs a *pull* program. The client needs to pull the message from the server. We discuss more about MAAs shortly.

User Agent

The first component of an electronic mail system is the **user agent (UA).** It provides service to the user to make the process of sending and receiving a message easier. A user agent is a software package (program) that composes reads, replies to, and forwards messages. It also handles local mailboxes on the user computers. There are two types of user agents:

1. ***Command-driven:*** Command-driven user agents belong to the early days of electronic mail. They are still present as the underlying user agents. A command-driven user agent normally accepts a one character command from the keyboard to perform its task.

For example: A user can type the character *r*, at the command prompt, to reply to the sender of the message, or type the character *R* to reply to the sender and all recipients. Some examples of command-driven user agents are *mail*, *pine*, and *elm*.

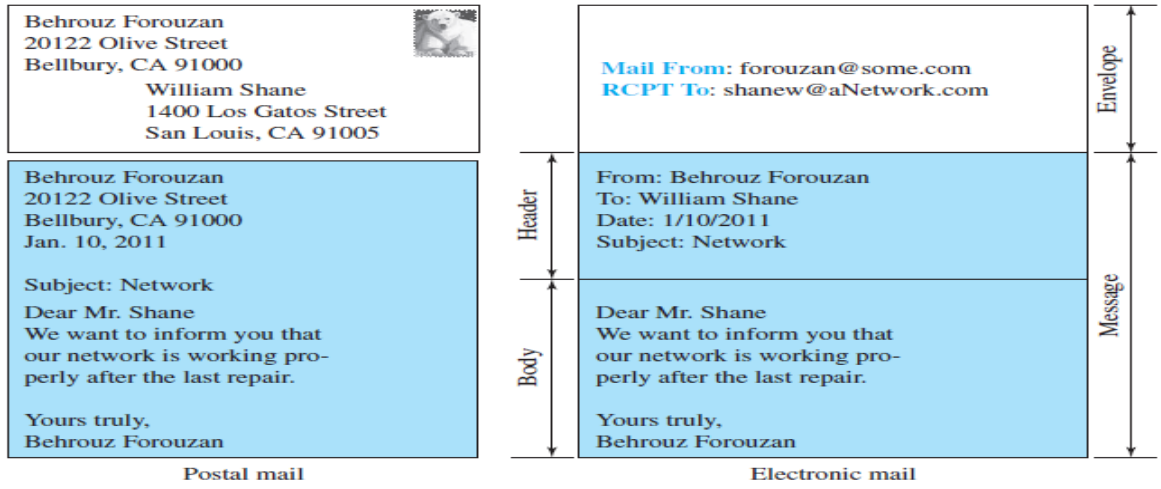
2. ***GUI-based.:*** Modern user agents are GUI-based. They contain graphical user interface (GUI) components that allow the user to interact with the software by using both the keyboard and the mouse. They have graphical components such as icons, menu bars, and windows that make the services easy to access.

Examples of GUI-based user agents are *Eudora* and *Outlook.*

Sending Mail

To send mail, the user, through the UA, creates mail that looks very similar to postal mail. It has an *envelope* and a *message* (see Figure 26.13). The envelope usually contains **the sender address, the receiver address, and other information**. The message contains **the *header* and the *body***. The header of the message defines the sender, the receiver, the subject of the message, and some other information. The body of the message contains the actual information to be read by the recipient.

Figure 26.13 *Format of an e-mail*



Receiving Mail

The user agent is triggered by the user (or a timer). If a user has mail, the UA informs the user with a notice. If the user is ready to read the mail, a list is displayed in which each line contains a summary of the information about a particular message in the mailbox. The summary usually includes the sender mail address,

the subject, and the time the mail was sent or received. The user can select any of the messages and display its contents on the screen.

Addresses

To deliver mail, a mail handling system must use an addressing system with unique addresses. In the Internet, the address consists of two parts:

A local part: The local part defines the name of a special file, called the user mailbox, where all the mail received for a user is stored for retrieval by the message access agent.

A domain name: The second part of the address is the domain name. An organization usually selects one or more hosts to receive and send e-mail; they are sometimes called *mail servers* or *exchangers*. The domain name assigned to each mail exchanger either comes from the DNS database or is a logical name (for example, the name of the organization).

Figure 26.14 *E-mail address*

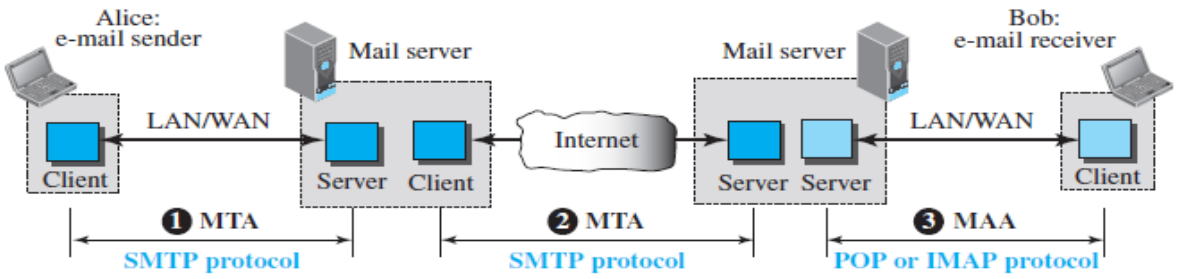
Mailing List or Group List

Electronic mail allows one name, an *alias*, to represent several different e-mail addresses; this is called a mailing list. Every time a message is to be sent, the system checks the recipient's name against the alias database; if there is a mailing list for the defined alias, separate messages, one for each entry in the list, must be prepared and handed to the MTA.

Message Transfer Agent: SMTP

Based on the common scenario (Figure 26.12), we can say that the e-mail is one of those applications that needs three uses of client-server paradigms to accomplish its task. It is important that we distinguish these three when we are dealing with e-mail. Figure 26.15 shows these three client-server applications. We refer to the first and the second as Message Transfer Agents (MTAs), the third as Message Access Agent (MAA).

Figure 26.15 *Protocols used in electronic mail*



The formal protocol that defines the MTA client and server in the Internet is called ***Simple Mail Transfer Protocol (SMTP).*** SMTP is used two times, between the sender and the sender's mail server and

between the two mail servers. As we will see shortly, another protocol is needed between the mail server and the receiver. SMTP simply defines how commands and responses must be sent back and forth.

Commands and Responses

SMTP uses commands and responses to transfer messages between an MTA client and an MTA server. The command is from an MTA client to an MTA server; the response is from an MTA server to the MTA client. Each command or reply is terminated by a two-character (carriage return and line feed) end-of-line token.

Commands: Commands are sent from the client to the server. The format of a command is shown below:

Keyword: argument(s)

It consists of a keyword followed by zero or more arguments. SMTP defines 14 commands, listed in Table 26.6.

Table 26.6 *SMTP commands*

Keyword	*Argument(s)*	*Description*
HELO	Sender's host name	Identifies itself
MAIL FROM	Sender of the message	Identifies the sender of the message
RCPT TO	Intended recipient	Identifies the recipient of the message
DATA	Body of the mail	Sends the actual message
QUIT		Terminates the message
RSET		Aborts the current mail transaction
VRFY	Name of recipient	Verifies the address of the recipient
NOOP		Checks the status of the recipient
TURN		Switches the sender and the recipient
EXPN	Mailing list	Asks the recipient to expand the mailing list
HELP	Command name	Asks the recipient to send information about the command sent as the argument
SEND FROM	Intended recipient	Specifies that the mail be delivered only to the terminal of the recipient, and not to the mailbox
SMOL FROM	Intended recipient	Specifies that the mail be delivered to the terminal *or* the mailbox of the recipient
SMAL FROM	Intended recipient	Specifies that the mail be delivered to the terminal *and* the mailbox of the recipient

Responses: Responses are sent from the server to the client. A response is a three digit code that may be followed by additional textual information. Table 26.7 shows the most common response types.

Table 26.7 *Responses*

Code	*Description*
	Positive Completion Reply
211	System status or help reply
214	Help message
220	Service ready
221	Service closing transmission channel
250	Request command completed
251	User not local; the message will be forwarded
	Positive Intermediate Reply
354	Start mail input
	Transient Negative Completion Reply
421	Service not available
450	Mailbox not available
451	Command aborted: local error
452	Command aborted; insufficient storage
	Permanent Negative Completion Reply
500	Syntax error; unrecognized command

Code	*Description*
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command temporarily not implemented
550	Command is not executed; mailbox unavailable
551	User not local
552	Requested action aborted; exceeded storage location
553	Requested action not taken; mailbox name not allowed
554	Transaction failed

Mail Transfer Phases

The process of transferring a mail message occurs in three phases: connection establishment, mail transfer, and connection termination.

A***. Connection Establishment :*** After a client has made a TCP connection to the well known port 25, the SMTP server starts the connection phase. This phase involves the following three steps:

1. The server sends code 220 (service ready) to tell the client that it is ready to receive mail. If the server is not ready, it sends code 421 (service not available).

2. The client sends the HELO message to identify itself, using its domain name address. This step is necessary to inform the server of the domain name of the client.

3. The server responds with code 250 (request command completed) or some other code depending on the situation.

B. Message Transfer: After connection has been established between the SMTP client and server, a single message between a sender and one or more recipients can be exchanged. This phase involves eight steps. Steps 3 and 4 are repeated if there is more than one recipient.

1. The client sends the MAIL FROM message to introduce the sender of the message. It includes the mail address of the sender (mailbox and the domain name). This step is needed to give the server the return mail address for returning errors and reporting messages.

2. The server responds with code 250 or some other appropriate code.

3. The client sends the RCPT TO (recipient) message, which includes the mail address of the recipient.

4. The server responds with code 250 or some other appropriate code.

5. The client sends the DATA message to initialize the message transfer.

6. The server responds with code 354 (start mail input) or some other appropriate Message

7. The client sends the contents of the message in consecutive lines. Each line is terminated by a two-character end-of-line token (carriage return and line feed). The message is terminated by a line containing just one period.

8. The server responds with code 250 (OK) or some other appropriate code.

C. Connection Termination After the message is transferred successfully, the client terminates the connection. This phase involves two steps.

1. The client sends the QUIT command.

2. The server responds with code 221 or some other appropriate code.

Message Access Agent: POP and IMAP

The first and second stages of mail delivery use SMTP. However, SMTP is not involved in the third stage because SMTP can have two types of protocols

***Push* protocol**: it pushes the message from the client to the server. The direction of the bulk data (messages) is from the client to the server.

***Pull* protocol**; the client must pull messages from the server. The direction of the bulk data is from the server to the client.

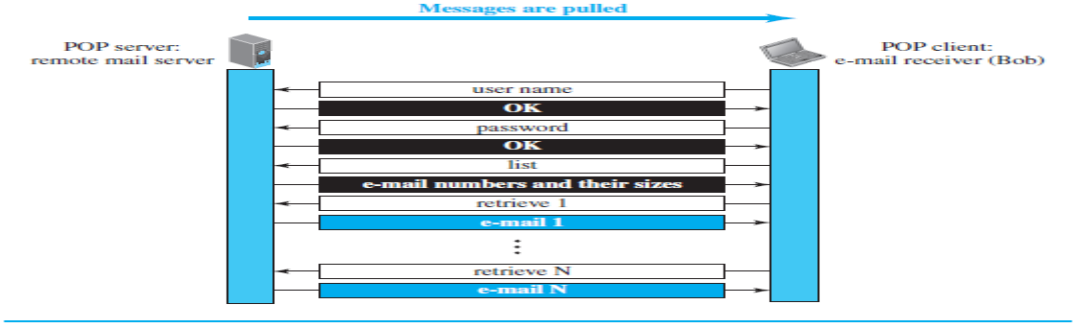
The third stage uses a message access agent. Currently two **message access protocols** are available:

- ***Post Office Protocol, version 3 (POP3) and***
- ***Internet Mail Access Protocol, version 4 (IMAP4)***

POP3

- **Post Office Protocol, version 3 (POP3)** is simple but limited in functionality. The client POP3 software is installed on the recipient computer; the server POP3 software is installed on the mail server.
- Mail access starts with the client when the user needs to download its e-mail from the mailbox on the mail server. The client opens a connection to the server on TCP port 110.
- It then sends its user name and password to access the mailbox. The user can then list and retrieve the mail messages, one by one.

Figure 26.17 *POP3*



POP3 has two modes:

The *delete* mode: In the delete mode, the mail is deleted from the mailbox after each retrieval. In the keep mode, the mail remains in the mailbox after retrieval.

The *keep* mode: The delete mode is normally used when the user is working at her permanent computer and can save and organize the received mail after reading or replying. The keep mode is normally used when the user accesses her mail away from her primary computer (for example, from a laptop). The mail is read but kept in the system for later retrieval and organizing.

IMAP4

Another mail access protocol is **Internet Mail Access Protocol, version 4 (IMAP4).** IMAP4 is similar to POP3, but it has more features; IMAP4 is more powerful and more complex. POP3 is deficient in several ways. It does not allow the user to organize her mail on the server; the user cannot have different folders on the server. In addition, POP3 does not allow the user to partially check the contents of the mail before downloading.

IMAP4 provides the following extra functions:

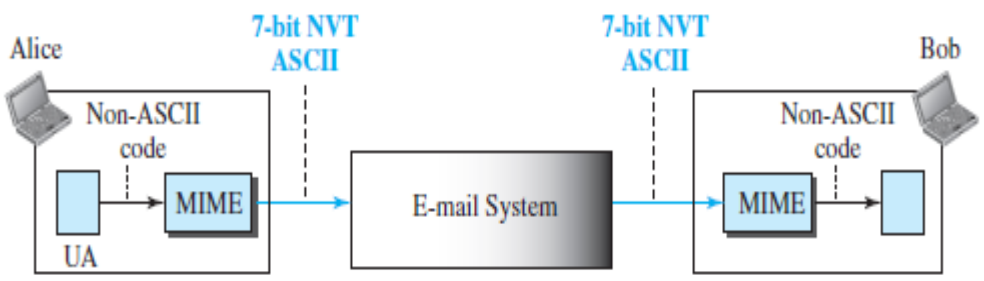
- A user can check the e-mail header prior to downloading.
- A user can search the contents of the e-mail for a specific string of characters prior to downloading.
- A user can partially download e-mail. This is especially useful if bandwidth is limited and the e-mail contains multimedia with high bandwidth requirements.
- A user can create, delete, or rename mailboxes on the mail server.
- A user can create a hierarchy of mailboxes in a folder for e-mail storage.

MIME

Electronic mail has a simple structure. Its simplicity, however, comes with a price. It can send messages only in NVT 7-bit ASCII format. In other words, it has some limitations. It cannot be used for languages other than English (such as French, German, Hebrew, Russian, Chinese, and Japanese). Also, it cannot be used to send binary files or video or audio data.

Multipurpose Internet Mail Extensions (MIME) is a supplementary protocol that allows non-ASCII data to be sent through e-mail. MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers it to the client MTA to be sent through the Internet. The message at the receiving site is transformed back to the original data. We can think of MIME as a set of software functions that transforms non-ASCII data to ASCII data and vice versa, as shown in Figure 26.18.

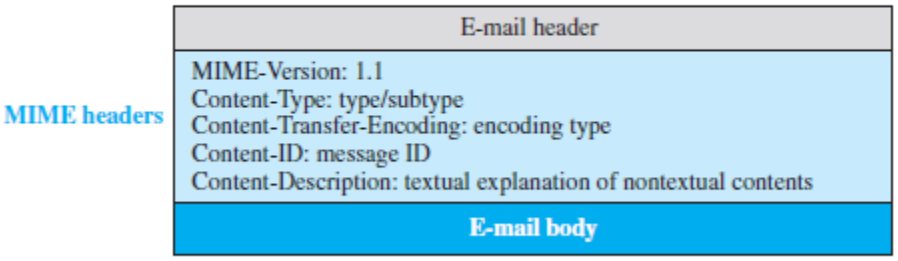
Figure 26.18 *MIME*



MIME Headers

MIME defines five headers, as shown in Figure 26.19, which can be added to the original e-mail header section to define the transformation parameters:

Figure 26.19 *MIME header*



MIME-Version This header defines the version of MIME used. The current version is 1.1.

Content-Type This header defines the type of data used in the body of the message. The content type and the content subtype are separated by a slash. Depending on the subtype, the header may contain other parameters. MIME allows seven different types of data, listed in Table 26.8.

Table 26.8 *Data types and subtypes in MIME*

Type	*Subtype*	*Description*
Text	Plain	Unformatted
	HTML	HTML format (see Appendix C)
Multipart	Mixed	Body contains ordered parts of different data types
	Parallel	Same as above, but no order
	Digest	Similar to Mixed, but the default is message/RFC822
	Alternative	Parts are different versions of the same message
Message	RFC822	Body is an encapsulated message
	Partial	Body is a fragment of a bigger message
	External-Body	Body is a reference to another message
Image	JPEG	Image is in JPEG format
	GIF	Image is in GIF format
Video	MPEG	Video is in MPEG format
Audio	Basic	Single channel encoding of voice at 8 KHz
Application	PostScript	Adobe PostScript
	Octet-stream	General binary data (eight-bit bytes)

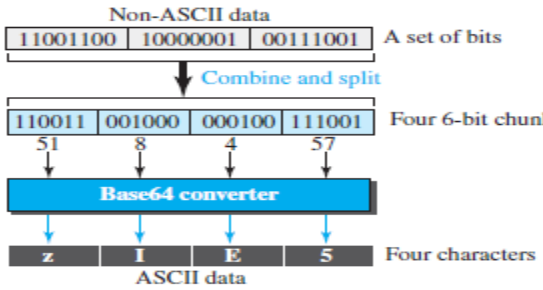
Content-Transfer-Encoding This header defines the method used to encode the messages into 0s and 1s for transport. The five types of encoding methods are listed in Table 26.9.

Table 26.9 *Methods for Content-Transfer-Encoding*

Type	*Description*
7-bit	NVT ASCII characters with each line less than 1000 characters
8-bit	Non-ASCII characters with each line less than 1000 characters
Binary	Non-ASCII characters with unlimited-length lines
Base64	6-bit blocks of data encoded into 8-bit ASCII characters
Quoted-printable	Non-ASCII characters encoded as an equal sign plus an ASCII code

The last two encoding methods are interesting. In the Base64 encoding, data, as a string of bits, is first divided into 6-bit chunks as shown in Figure 26.20.

Figure 26.20 *Base64 conversion*



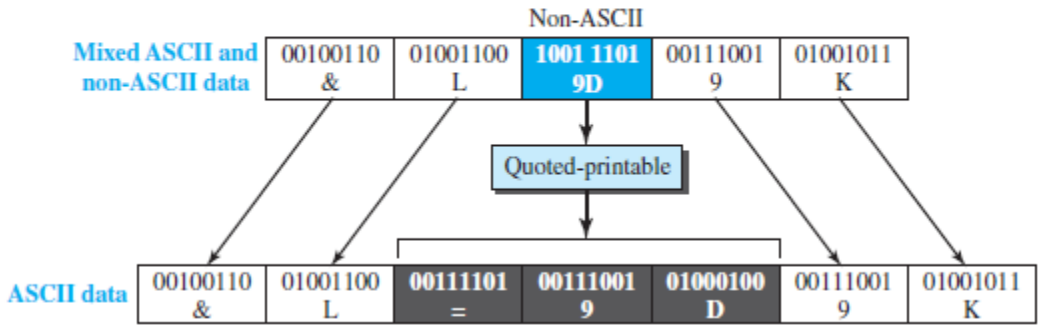
Each 6-bit section is then converted into an ASCII character according to Table 26.10.

Table 26.10 *Base64 converting table*

Value	*Code*	*Value*	*Code*	*Value*	*Code*	*Value*	*Code*	*Value*	*Code*	*Value*	*Code*
0	A	11	L	22	W	33	h	44	s	55	3
1	B	12	M	23	X	34	i	45	t	56	4
2	C	13	N	24	Y	35	j	46	u	57	5
3	D	14	O	25	Z	36	k	47	v	58	6
4	E	15	P	26	a	37	l	48	w	59	7
5	F	16	Q	27	b	38	m	49	x	60	8
6	G	17	R	28	c	39	n	50	y	61	9
7	H	18	S	29	d	40	o	51	z	62	+
8	I	19	T	30	e	41	p	52	0	63	/
9	J	20	U	31	f	42	q	53	1		
10	K	21	V	32	g	43	r	54	2		

Base64 is a redundant encoding scheme; that is, every six bits become one ASCII character and are sent as eight bits. We have an overhead of 25 percent. If the data consist mostly of ASCII characters with a small non-ASCII portion, we can use quoted-printable encoding. In quoted-printable, if a character is ASCII, it is sent as is. If a character is not ASCII, it is sent as three characters. The first character is the equal sign (=). The next two characters are the hexadecimal representations of the byte. Figure 26.21 shows an example. In the example, the third character is a non-ASCII because it starts with bit 1. It is interpreted as two hexadecimal digits ($9D_{16}$), which is replaced by three ASCII characters (=, 9, and D).

Figure 26.21 *Quoted-printable*



Content-ID This header uniquely identifies the whole message in a multiple message environment.

Content-Description This header defines whether the body is image, audio, or video.

5.5.2 Web-Based Mail

E-mail is such a common application that some websites today provide this service to anyone who accesses the site. Three common sites are Hotmail, Yahoo, and Google mail. The idea is very simple. Figure 26.22 shows two cases:

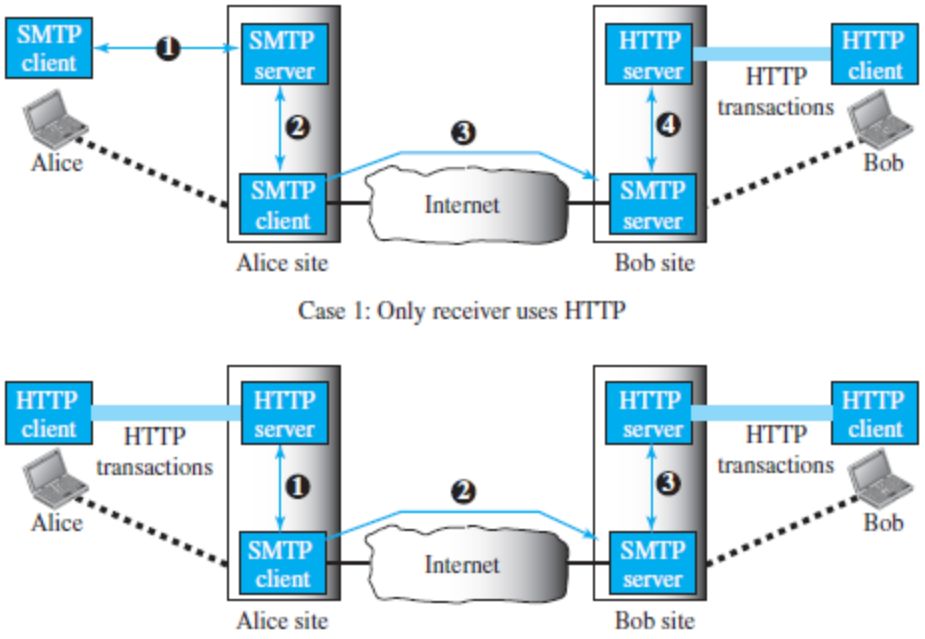
Case I

In the first case, Alice, the sender, uses a traditional mail server; Bob, the receiver, has an account on a web-based server. Mail transfer from Alice's browser to her mail server is done through SMTP. The transfer of the message from the sending mail server to the receiving mail server is still through SMTP. However, the message from the receiving server (the web server) to Bob's browser is done through HTTP. In other words, instead of using POP3 or IMAP4, HTTP is normally used. When Bob needs to retrieve his e-mails, he sends a request HTTP message to the website (Hotmail, for example). The website sends a form to be filled in by Bob, which includes the log-in name and the password. If the log-in name and password match, the list of e-mails is transferred from the web server to Bob's browser in HTML format. Now Bob can browse through his received e-mails and then, using more HTTP transactions, can get his e-mails one by one.

Case II

In the second case, both Alice and Bob use web servers, but not necessarily the same server. Alice sends the message to the web server using HTTP transactions. Alice sends an HTTP request message to her web server using the name and address of Bob's mailbox as the URL. The server at the Alice site passes the message to the SMTP client and sends it to the server at the Bob site using SMTP protocol. Bob receives the message using HTTP transactions. However, the message from the server at the Alice site to the server at the Bob site still takes place using SMTP protocol.

Figure 26.22 *Web-based e-mail, cases I and II*



5.5.3 E-Mail Security

E-mail exchanges can be secured using two application-layer securities designed in particular for e-mail systems. Two of these protocols,

Pretty Good Privacy (PGP) and *Secure/Multipurpose Internet Mail Extensions* (S/MIME).

5.6 TELNET

A server program can provide a specific service to its corresponding client program.

For example:

The FTP server is designed to let the FTP client store or retrieve files on the server site. However, it is impossible to have a client/server pair for each type of service we need; the number of servers soon becomes intractable. The idea is not scalable.

Another solution is to have a specific client/server program for a set of common scenarios, but to have some generic client/server programs that allow a user on the client site to log into the computer at the server site and use the services available there.

For example, if a student needs to use the Java compiler program at her university lab, there is no need for a Java compiler client and a Java compiler server. The student can use a client logging program to log into the university server and use the compiler program at the university. We refer to these generic client/server pairs as ***remote logging*** applications.

One of the original remote logging protocols is **TELNET,** which is an abbreviation for *TErminaL NETwork.* Although TELNET requires a logging name and password, it is vulnerable to hacking because it sends all data including the password in plaintext (not encrypted).

A hacker can eavesdrop and obtain the logging name and password. Because of this security issue, the use of TELNET has diminished in favor of another protocol, Secure Shell (SSH), which we describe in the next section. Although TELNET is almost replaced by SSH, we briefly discuss TELNET here for two reasons:

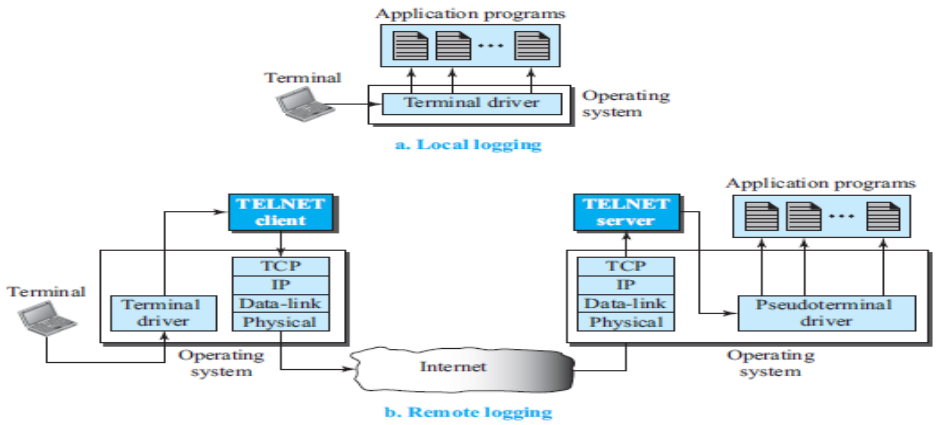
1. The simple plaintext architecture of TELNET allows us to explain the issues and challenges related to the concept of remote logging, which is also used in SSH when it serves as a remote logging protocol.

2. Network administrators often use TELNET for diagnostic and debugging purposes.

5.6.1 Local versus Remote Logging

We first discuss the concept of local and remote logging as shown in Figure 26.23.

Figure 26.23 *Local versus remote logging*



When a user logs into a local system, it is called *local logging*. As a user types at a terminal or at a workstation running a terminal emulator, the keystrokes are accepted by the terminal driver. The terminal driver passes the characters to the operating system. The operating system, in turn, interprets the combination of characters and invokes the desired application program or utility.

When a user wants to access an application program or utility located on a remote machine, she performs *remote logging.* Here the TELNET client and server programs come into use. The user sends the keystrokes to the terminal driver where the local operating system accepts the characters but does not interpret them. The characters are sent to the TELNET client, which transforms the characters into a universal character set called *Network Virtual Terminal* (NVT) characters (discussed below) and delivers them to the local TCP/IP stack.

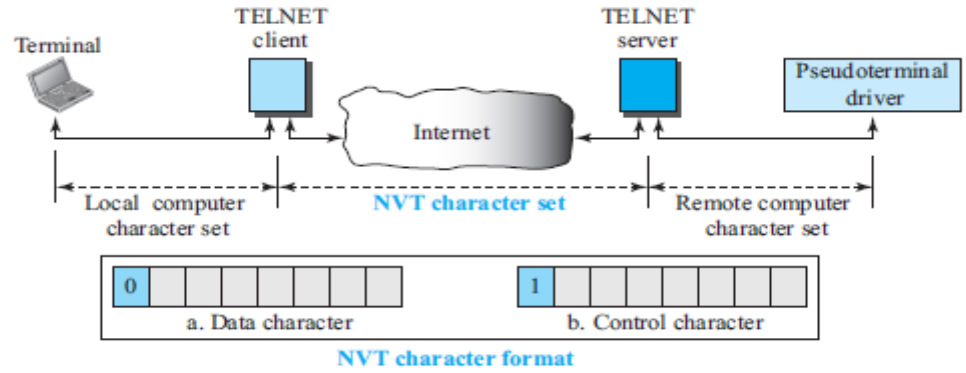
The commands or text, in NVT form, travel through the Internet and arrive at the TCP/IP stack at the remote machine. Here the characters are delivered to the operating system and passed to the TELNET server, which changes the characters to the corresponding characters understandable by the remote computer. However, the characters cannot be passed directly to the operating system because the remote operating system is not designed to receive characters from a TELNET server; it is designed to receive characters from a terminal driver. The solution is to add a piece of software called a *pseudo-terminal driver*, which pretends that the characters are coming from a terminal. The operating system then passes the characters to the appropriate application program.

Network Virtual Terminal (NVT)

The mechanism to access a remote computer is complex. This is because every computer and its operating system accept a special combination of characters as tokens.

For example: The end-of-file token in a computer running the DOS operating system is Ctrl+z, while the UNIX operating system recognizes Ctrl+d. We are dealing with heterogeneous systems. If we want to access any remote computer in the world, we must first know what type of computer we will be connected to, and we must also install the specific terminal emulator used by that computer. TELNET solves this problem by defining a universal interface called the *Network Virtual Terminal (NVT)* character set. Via this interface, the client TELNET translates characters (data or commands) that come from the local terminal into NVT form and delivers them to the network. The server TELNET, on the other hand, translates data and commands from NVT form into the form acceptable by the remote computer. Figure 26.24 shows the concept.

Figure 26.24 *Concept of NVT*



NVT uses two sets of characters, one for data and one for control. Both are 8-bit bytes as shown in Figure 26.24. For data, NVT normally uses what is called *NVT ASCII*. This is an 8-bit character set in which the seven lowest order bits are the same as US ASCII and the highest order bit is 0. To send control characters between computers (from client to server or vice versa), NVT uses an 8-bit character set in which the highest order bit is set to 1.

Options

TELNET lets the client and server negotiate options before or during the use of the service. Options are extra features available to a user with a more sophisticated terminal. Users with simpler terminals can use default features.

User Interface

The operating system (UNIX, for example) defines an interface with user-friendly commands. An example of such a set of commands can be found in Table 26.11.

Table 26.11 *Examples of interface commands*

Command	*Meaning*	*Command*	*Meaning*
open	Connect to a remote computer	**set**	Set the operating parameters
close	Close the connection	**status**	Display the status information
display	Show the operating parameters	**send**	Send special characters
mode	Change to line or character mode	**quit**	Exit TELNET

5.7 SECURE SHELL (SSH)

Secure Shell (SSH) is a secure application program that can be used today for several purposes such as remote logging and file transfer; it was originally designed to replace TELNET. There are two versions of SSH:

- SSH-1 and
- SSH-2

The first version, SSH-1, is now deprecated because of security flaws in it. In this section, we discuss only SSH-2.

5.7.1 Components

SSH is an application-layer protocol with three components, as shown in Figure 26.25.

SSH Transport-Layer Protocol (SSH-TRANS)

Since TCP is not a secured transport-layer protocol, SSH first uses a protocol that creates a secured channel on top of the TCP. This new layer is an independent protocol referred to as SSH-TRANS. When the procedure implementing this protocol is called, the client and server first use the TCP protocol to establish an insecure connection. Then they exchange several security parameters to establish a secure channel on top of the TCP. We discuss transport-layer security in Chapter 32, but here we briefly list the services provided by this protocol:

1. Privacy or confidentiality of the message exchanged

2. Data integrity, which means that it is guaranteed that the messages exchanged between the client and server are not changed by an intruder

3. Server authentication, which means that the client is now sure that the server is the one that it claims to be

4. Compression of the messages, which improves the efficiency of the system and makes attack more difficult

SSH Authentication Protocol (SSH-AUTH)

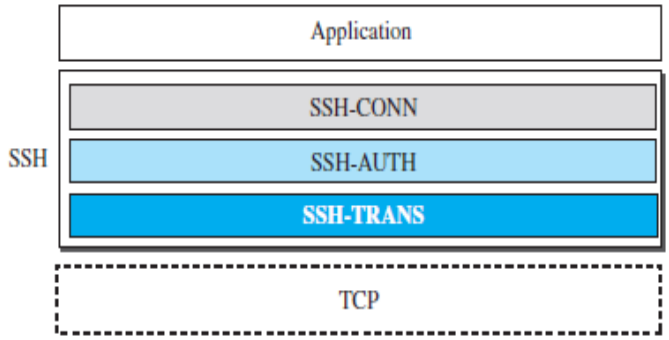
After a secure channel is established between the client and the server and the server is authenticated for the client, SSH can call another procedure that can authenticate the client for the server. The client authentication process in SSH is very similar to what is done in Secure Socket Layer (SSL),

This layer defines a number of authentication tools similar to the ones used in SSL. Authentication starts with the client, which sends a request message to the server. The request includes the user name, server name, the method of authentication, and the required data. The server responds with either a success message, which confirms that the client is authenticated, or a failed message, which means that the process needs to be repeated with a new request message.

SSH Connection Protocol (SSH-CONN)

After the secured channel is established and both server and client are authenticated for each other, SSH can call a piece of software that implements the third protocol, SSHCONN. One of the services provided by the SSH-CONN protocol is multiplexing. SSH-CONN takes the secure channel established by the two previous protocols and lets the client create multiple logical channels over it. Each channel can be used for a different purpose, such as remote logging, file transfer, and so on.

Figure 26.25 *Components of SSH*



5.7.2 Applications

Although SSH is often thought of as a replacement for TELNET, SSH is, in fact, a general-purpose protocol that provides a secure connection between a client and server.

SSH for Remote Logging

Several free and commercial applications use SSH for remote logging. Among them, we can mention PuTTy, by Simon Tatham, which is a client SSH program that can be used for remote logging. Another application program is Tectia, which can be used on several platforms.

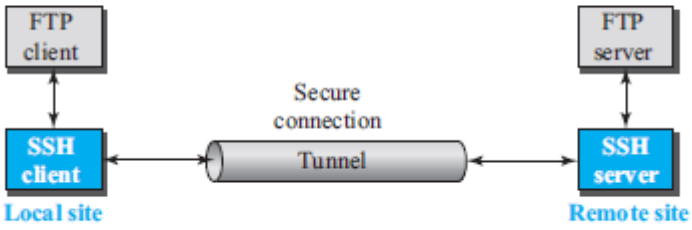
SSH for File Transfer

One of the application programs that is built on top of SSH for file transfer is the *Secure File Transfer Program* (*sftp*). The *sftp* application program uses one of the channels provided by the SSH to transfer files. Another common application is called *Secure Copy* (*scp*). This application uses the same format as the UNIX copy command, *cp*, to copy files.

Port Forwarding

One of the interesting services provided by the SSH protocol is **port forwarding**. We can use the secured channels available in SSH to access an application program that does not provide security services. Applications such as TELNET and Simple Mail Transfer Protocol (SMTP), which are discussed above, can use the services of the SSH port forwarding mechanism. The SSH port forwarding mechanism creates a tunnel through which the messages belonging to other protocols can travel. For this reason, this mechanism is sometimes referred to as SSH *tunneling*. Figure 26.26 shows the concept of port forwarding for securing the FTP application.

Figure 26.26 *Port forwarding*



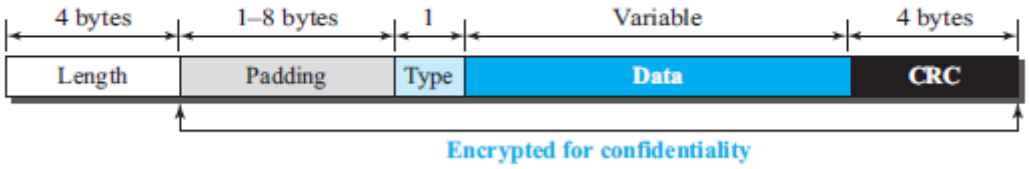
The FTP client can use the SSH client on the local site to make a secure connection with the SSH server on the remote site. Any request from the FTP client to the FTP server is carried through the tunnel provided by the SSH client and server. Any response from the FTP server to the FTP client is also carried through the tunnel provided by the SSH client and server.

Format of the SSH Packets

Figure 26.27 shows the format of packets used by the SSH protocols. The length field defines the length of the packet but does not include the padding. One to eight bytes of padding is added to the packet to make the attack on the security provision more difficult. The *cyclic redundancy check* (CRC) field is used for error detection. The type field designates the type of the packet used in different SSH protocols.

The data field is the data transferred by the packet in different protocols.

Figure 26.27 *SSH packet format*



DOMAIN NAME SYSTEM (DNS)

The last client-server application program we discuss has been designed to help other application programs. To identify an entity, TCP/IP protocols use the IP address, which uniquely identifies the connection of a host to the Internet. People prefer to use names instead of numeric addresses. Therefore, the Internet needs to have a directory system that can map a name to an address. This is analogous to the telephone network.

A telephone network is designed to use telephone numbers, not names. People can either keep a private file to map a name to the corresponding telephone number or can call the telephone directory to do so.

Since the Internet is so huge today, **a central directory system cannot hold all the mapping. In addition, if the central computer fails, the whole communication network will collapse. A better solution is to distribute the information among many computers in the world**.

In this method, the host that needs mapping can contact the closest computer holding the needed information. This method is used by the **Domain Name System (DNS)**.

Figure 26.28 shows how TCP/IP uses a DNS client and a DNS server to map a name to an address. A user wants to use a file transfer client to access the corresponding file transfer server running on a remote host. The user knows only the file transfer server name, such as ***afilesource.com***. However, the TCP/IP suite needs the IP address of the file transfer server to make the connection.

The following six steps map the host name to an IP address:

1. The user passes the host name to the file transfer client.
2. The file transfer client passes the host name to the DNS client.
3. Each computer, after being booted, knows the address of one DNS server. The DNS client sends a message to a DNS server with a query that gives the file transfer server name using the known IP address of the DNS server.
4. The DNS server responds with the IP address of the desired file transfer server.
5. The DNS server passes the IP address to the file transfer client.
6. The file transfer client now uses the received IP address to access the file transfer server.

Note that the purpose of accessing the Internet is to make a connection between **the file transfer client and server,** but before this can happen, another connection needs to be made between **the DNS client and DNS server**. We need at least two connections in this case.

The first is for mapping the name to an IP address;

The second is for transferring files.

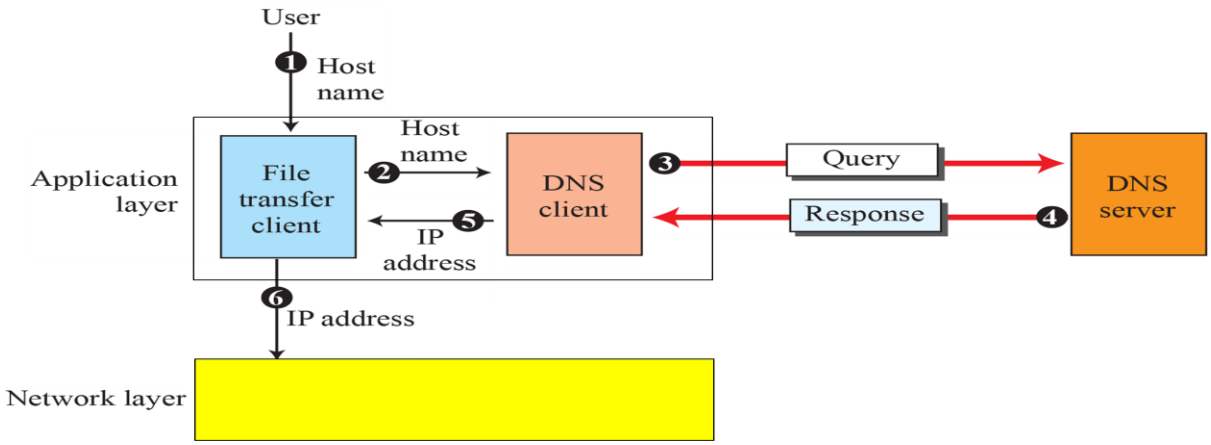


Fig: Purpose of DNS

1. Name Space

The names assigned to machines must be carefully selected from a name space with complete control over the binding between the names and IP addresses. The names must be unique because the addresses are unique.

A **name space** that maps each address to a unique name can be organized in two ways:

1) Flat or
2) Hierarchical.

In a ***flat name space***, a name is assigned to an address. A name in this space is a sequence of characters without structure. The names may or may not have a common section; if they do, it has no meaning. The main disadvantage of a flat name space is that it cannot be used in a large system such as the Internet because it must be centrally controlled to avoid ambiguity and duplication.

In a ***hierarchical name space***, each name is made of several parts.

a) The first part can define the nature of the organization,

b) The second part can define the name of an organization.

c) The third part can define departments in the organization, and so on.

In this case, **the authority to assign and control the name spaces can be decentralized**. A **central authority** can assign the part of the name that defines the nature of the organization and the name of the organization. The responsibility for the rest of the name can be given to the organization itself. The organization can add suffixes (or prefixes) to the name to define its host or resources. The management of the organization need not worry that the prefix chosen for a host is taken by another organization because, even if part of an address is the same, the whole address is different.

For example:

Assume two organizations call one of their computers *caesar*. The first organization is given a name by the central authority, such as *first.com,* the second organization is given the name *second.com.* When each of these organizations adds the name *caesar* to the name they have already been given, the end result is two distinguishable names: ***ceasar.first.com*** and ***ceasar.second.com***. The names are unique.

Domain Name Space

To have a hierarchical name space, a **domain name space** was designed. In this design the names are defined in an inverted-tree structure with the root at the top. The tree can have only 128 levels: level 0 (root) to level 127.

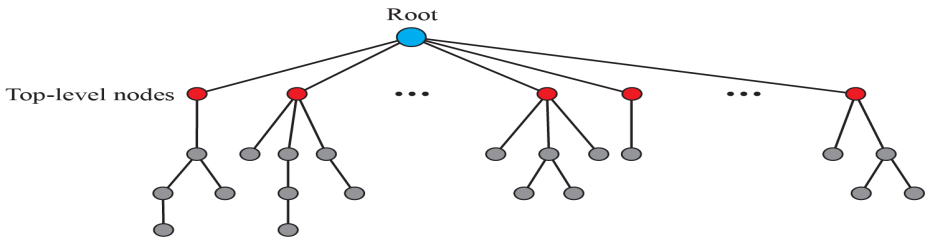


Fig:Domain Name Space

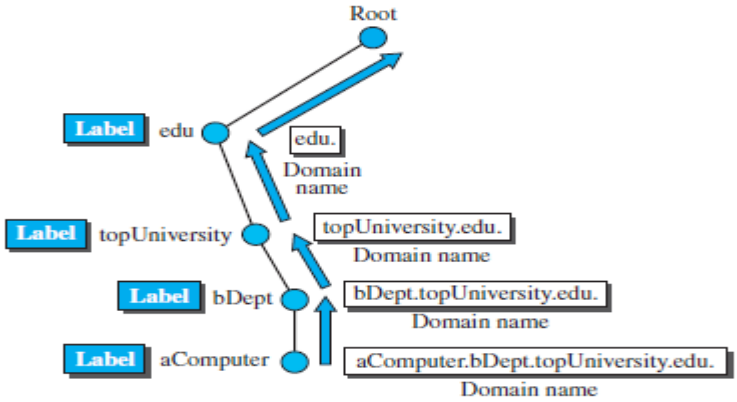
Label

Each node in the tree has a **label**, which is a string with a maximum of 63 characters. The root label is a null string (empty string). DNS requires that children of a node (nodes that branch from the same node) have different labels, which guarantees the uniqueness of the domain names.

Domain Name

Each node in the tree has a domain name. A full **domain name** is a sequence of labels separated by dots (**.**). The domain names are always read from the node up to the root. The last label is the label of the root (null). This means that a full domain name always ends in a null label, which means the last character is a dot because the null string is nothing. Figure 26.30 shows some domain names.

Figure 26.30 *Domain names and labels*

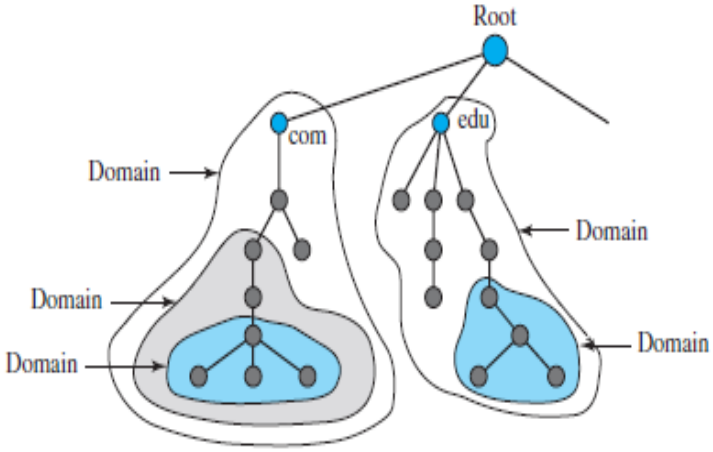


If a label is terminated by a null string, it is called a **fully qualified domain name (FQDN)**. The name must end with a null label, but because null means nothing, the label ends with a dot. If a label is not terminated by a null string, it is called a **partially qualified domain name (PQDN).** A PQDN starts from a node, but it does not reach the root. It is used when the name to be resolved belongs to the same site as the client. Here the resolver can supply the missing part, called the *suffix*, to create an FQDN.

Domain

A **domain** is a subtree of the domain name space. The name of the domain is the name of the node at the top of the subtree. Figure 26.31 shows some domains. Note that a domain may itself be divided into domains.

Figure 26.31 *Domains*



Distribution of Name Space

The information contained in the domain name space must be stored. It is **very inefficient** and also not reliable to have just one computer store such a huge amount of information. It is **inefficient** because responding

to requests from all over the world places a heavy load on the system. It is not reliable because any failure makes the data inaccessible.

Hierarchy of Name Servers

The solution to these problems is to distribute the information among many computers called ***DNS servers.*** One way to do this is to divide the whole space into many domains based on the first level. We let the root stand alone and create as many domains (subtrees) as there are first-level nodes. Because a domain created this way could be very large, DNS allows domains to be divided further into smaller domains (sub-domains). Each server can be responsible (authoritative) for either a large or small domain. In other words, we have a hierarchy of servers in the same way that we have a hierarchy of names.

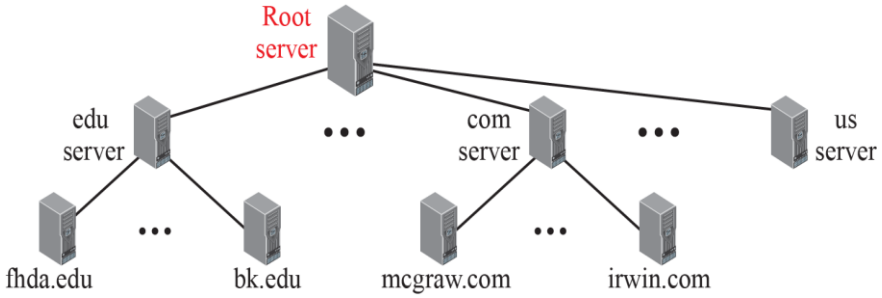


Fig Hierarchy of name servers

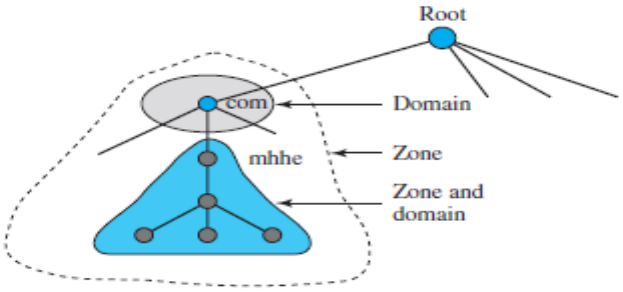
Zone

Since the complete domain name hierarchy cannot be stored on a single server, it is divided among many servers. What a server is responsible for or has authority over is called a ***zone***.

We can define a zone as a contiguous part of the entire tree. If a server accepts responsibility for a domain and does not divide the domain into smaller domains, the ***"domain"*** and the ***"zone"*** refer to the same thing. The server makes a database called a *zone file* and keeps all the information for every node under that domain.

However, if a server divides its domain into sub-domains and delegates part of its authority to other servers, ***"domain"*** and ***"zone"*** refer to different things. The information about the nodes in the sub-domains is stored in the servers at the lower levels, with the original server keeping some sort of reference to these lower-level servers. Of course, the original server does not free itself from responsibility totally. It still has a zone, but the detailed information is kept by the lower-level servers.

Figure 26.33 *Zone*



Root Server

A **root server** is a server whose zone consists of the whole tree. A root server usually does not store any information about domains but delegates its authority to other servers, keeping references to those servers.

There are several root servers, each covering the whole domain name space. The root servers are distributed all around the world.

Primary and Secondary Servers

DNS defines two types of servers:

a) ***Primary and***

b) ***Secondary.***

A ***primary server*** is a server that stores a file about the zone for which it is an authority. It is responsible for creating, maintaining, and updating the zone file. It stores the zone file on a local disk.

A ***secondary server*** is a server that transfers the complete information about a zone from another server (primary or secondary) and stores the file on its local disk. The secondary server neither creates nor updates the zone files. If updating is required, it must be done by the primary server, which sends the updated version to the secondary.

The primary and secondary servers are both authoritative for the zones they serve. The idea is not to put the secondary server at a lower level of authority but to create redundancy for the data so that if one server fails, the other can continue serving clients.

2. ***DNS in the Internet***

DNS is a protocol that can be used in different platforms. In the Internet, the domain name space (tree) was originally divided into three different sections:

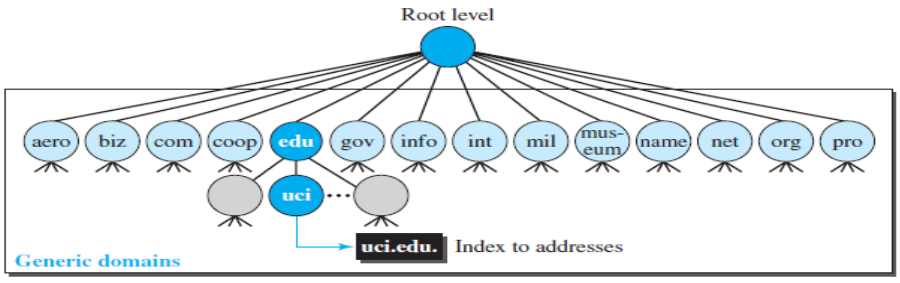
- ***Generic domains,***
- ***Country domains, and***

Due to the rapid growth of the Internet, it became extremely difficult to keep track of the inverse domains, which could be used to find the name of a host when given the IP address. The inverse domains are now deprecated (see RFC 3425). We, therefore, concentrate on the first two.

Generic Domains

The **generic domains** define registered hosts according to their generic behavior. Each node in the tree defines a domain, which is an index to the domain name space database (see Figure 26.34).

Figure 26.34 *Generic domains*



Looking at the tree, we see that the first level in the **generic domains** section allows 14 possible labels. These labels describe the organization types as listed in Table 26.12.

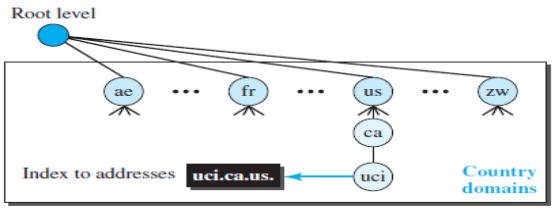
Table 26.12 *Generic domain labels*

Label	*Description*	*Label*	*Description*
aero	Airlines and aerospace	**int**	International organizations
biz	Businesses or firms	**mil**	Military groups
com	Commercial organizations	**museum**	Museums
coop	Cooperative organizations	**name**	Personal names (individuals)
edu	Educational institutions	**net**	Network support centers
gov	Government institutions	**org**	Nonprofit organizations
info	Information service providers	**pro**	Professional organizations

Country Domains

The **country domains** section uses two-character country abbreviations (e.g., us for United States). Second labels can be organizational, or they can be more specific national designations. The United States, for example, uses state abbreviations as a subdivision of us (e.g., ca.us.). Figure 26.35 shows the country domains section. The address ***uci.ca.us.*** can be translated to University of California, Irvine, in the state of California in the United States.

Figure 26.35 *Country domains*



3. *Resolution*

Mapping a name to an address is called ***name-address resolution***. DNS is designed as a client-server application. A host that needs to map an address to a name or a name to an address calls a DNS client called a ***resolver.*** The resolver accesses the closest DNS server with a mapping request. If the server has the information, it satisfies the resolver; otherwise, it either refers the resolver to other servers or asks other servers to provide the information. After the resolver receives the mapping, it interprets the response to see if it is a real resolution or an error, and finally delivers the result to the process that requested it. A resolution can be either **recursive or iterative.**

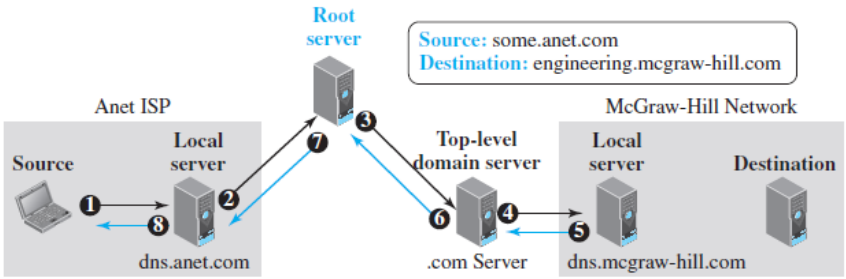
Recursive Resolution

Figure 26.36 shows a simple example of a **recursive resolution.** We assume that an application program running on a host named ***some.anet.com*** needs to find the IP address of another host named ***engineering.mcgraw-hill.com*** to send a message to. The source host is connected to the Anet ISP; the destination host is connected to the McGraw-Hill network.

The application program on the source host calls the DNS resolver (client) to find the IP address of the destination host. The resolver, which does not know this address, sends the query to the local DNS server (for example, *dns.anet.com*) running at the Anet ISP site (event 1). We assume that this server does not know the IP address of the destination host either. It sends the query to a root DNS server, whose IP address is supposed to be known to this local DNS server (event 2). Root servers do not normally keep the mapping between names and IP addresses, but a root server should at least know about one server at each top level domain (in this case, a server responsible for *com* domain). The query is sent to this top-level-domain server (event 3). We assume that

this server does not know the name-address mapping of this specific destination, but it knows the IP address of the local DNS server in the McGraw-Hill company (for example, *dns.mcgraw-hill.com*). The query is sent to this server (event 4), which knows the IP address of the destination host. The IP address is now sent back to the top-level DNS server (event 5), then back to the root server (event 6), then back to the ISP DNS server, which may cache it for the future queries (event 7), and finally back to the source host (event 8).

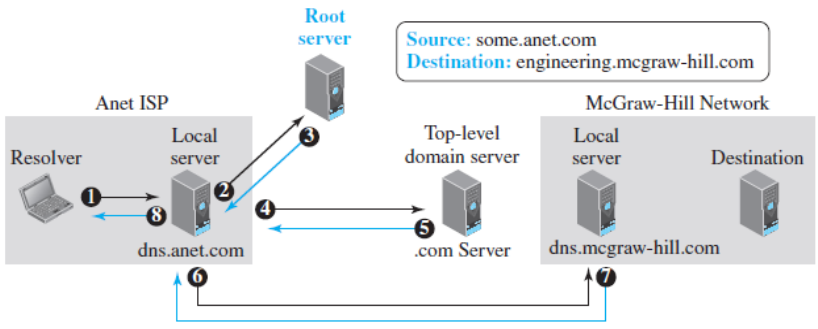
Figure 26.36 *Recursive resolution*



Iterative Resolution

In **iterative resolution,** each server that does not know the mapping sends the IP address of the next server back to the one that requested it. Figure 26.37 shows the flow of information in an iterative resolution in the same scenario as the one depicted in Figure 26.36. Normally the iterative resolution takes place between two local servers; the original resolver gets the final answer from the local server. Note that the messages shown by events 2, 4, and 6 contain the same query. However, the message shown by event 3 contains the IP address of the top-level domain server, the message shown by event 5 contains the IP address of the McGraw-Hill local DNS server, and the message shown by event 7 contains the IP address of the destination. When the Anet local DNS server receives the IP address of the destination, it sends it to the resolver (event 8).

Figure 26.37 *Iterative resolution*



4. Caching

Each time a server receives a query for a name that is not in its domain, it needs to search its database for a server IP address. Reduction of this search time would increase efficiency. DNS handles this with a mechanism called ***caching***.

When a server asks for a mapping from another server and receives the response, it stores this information in its cache memory before sending it to the client. If the same or another client asks for the same mapping, it can check its cache memory and resolve the problem. However, to inform the client that the

response is coming from the cache memory and not from an authoritative source, the server marks the response as ***unauthoritative.***

Caching speeds up resolution, but it can also be problematic. If a server caches a mapping for a long time, it may send an outdated mapping to the client.

To counter this, two techniques are used.

First, the authoritative server always adds information to the mapping called ***time to live (TTL).*** *It defines the time in seconds that the receiving server can cache the information. After that time, the mapping is invalid and any query must be sent again to the authoritative server.*

Second, DNS requires that each server keep a TTL counter for each mapping it caches. The cache memory must be searched periodically and those mappings with an expired TTL must be purged.

5. Resource Records

The zone information associated with a server is implemented as a set of ***resource records***. In other words, a name server stores a database of resource records. A ***resource record*** is a 5-tuple structure, as shown below:

(Domain Name, Type, Class, TTL, Value)

The domain name field is what identifies the resource record. The value defines the information kept about the domain name. The TTL defines the number of seconds for which the information is valid. The class defines the type of network; we are only interested in the class IN (Internet). The type defines how the value should be interpreted. Table 26.13 lists the common types and how the value is interpreted for each type.

Table 26.13 *Types*

Type	*Interpretation of value*
A	A 32-bit IPv4 address (see Chapter 18)
NS	Identifies the authoritative servers for a zone
CNAME	Defines an alias for the official name of a host
SOA	Marks the beginning of a zone
MX	Redirects mail to a mail server
AAAA	An IPv6 address (see Chapter 22)

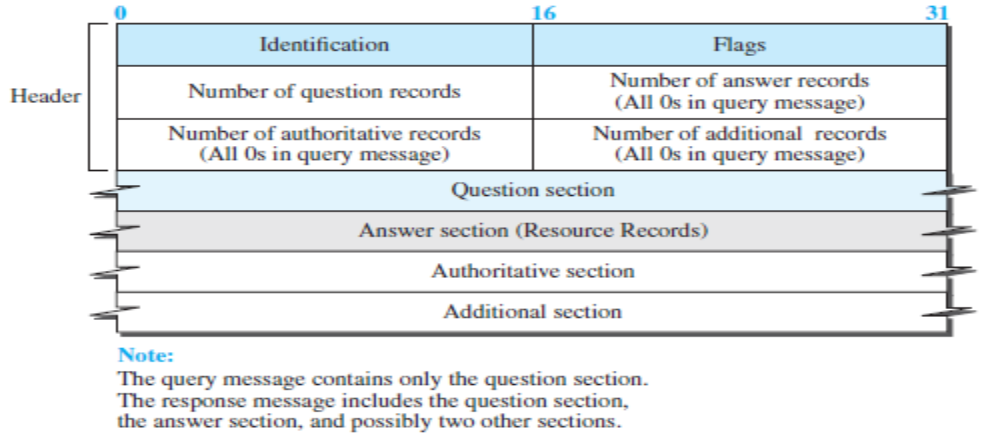
6. DNS Messages

To retrieve information about hosts, DNS uses two types of messages:

- ***Query and***
- ***Response.***

Both types have the same format as shown in Figure 26.38.

Figure 26.38 *DNS message*



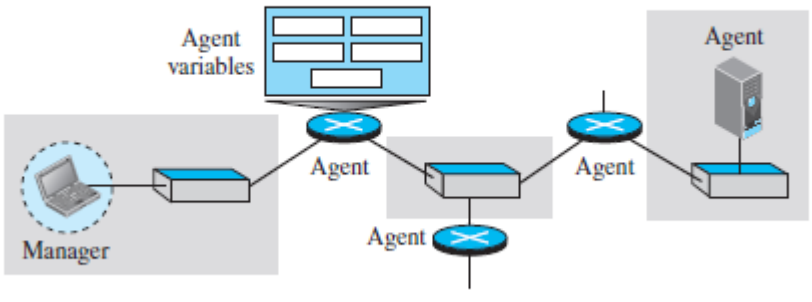
We briefly discuss the fields in a DNS message. The **identification** field is used by the client to match the response with the query. The **flag** field defines whether the message is a query or response. It also includes status of error. The next four fields in the header define the number of each record type in the message. The question section consists of one or more question records. It is present in both query and response messages.

The answer section consists of one or more resource records. It is present only in response messages. The authoritative section gives information (domain name) about one or more authoritative servers for the query. The additional information section provides additional information that may help the resolver.

SNMP

Several network management standards have been devised during the last few decades. The most important one is **Simple Network Management Protocol (SNMP),** used by the Internet. SNMP is a framework for managing devices in an internet using the TCP/IP protocol suite. It provides a set of fundamental operations for monitoring and maintaining an internet. SNMP uses the concept of manager and agent. That is, a manager, usually a host, controls and monitors a set of agents; usually routers or servers (see Figure 27.2).

Figure 27.2 *SNMP concept*



SNMP is an **application-level protocol** in which a few manager stations control a set of agents. The protocol is designed at the application level so that it can monitor devices made by different manufacturers and installed on different physical networks. SNMP frees management tasks from both the physical characteristics

of the managed devices and the underlying networking technology. It can be used in **a heterogeneous internet** made of different LANs and WANs connected by routers made by different manufacturers.

1. Managers and Agents

A management station, called a ***manager***, is a host that runs the SNMP client program. A managed station, called an ***agent***, is a router (or a host) that runs the SNMP server program. Management is achieved through simple interaction between a manager and an agent. The agent keeps performance information in a database. The manager has access to the values in the database.

For example: A router can store in appropriate variables the number of packets received and forwarded. The *manager can fetch* and compare the values of these *two variables* to see if the router is *congested or not*. The manager can also make the router perform certain actions.

For example: A router periodically checks the value of a reboot counter to see when it should reboot itself.

For example: if the value of the counter is 0. The manager can use this feature to reboot the agent remotely at any time. It simply sends a packet to force a 0 value in the counter.

Agents can also contribute to the management process. The server program running on the agent can check the environment and, if it notices something unusual, it can send a warning message (called a *Trap*) to the manager.

In other words, management with SNMP is based on three basic ideas:

1. A manager checks an agent by requesting information that reflects the behaviour of the agent.

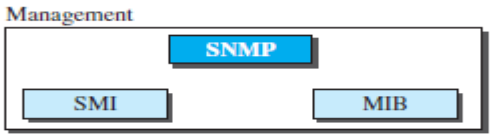
2. A manager forces an agent to perform a task by resetting values in the agent database.

3. An agent contributes to the management process by warning the manager of an unusual situation.

2 Management Components

To do management tasks, SNMP uses two other protocols: **Structure of Management Information (SMI)** and **Management Information Base (MIB).** In other words, management on the Internet is done through the cooperation of three protocols: SNMP, SMI, and MIB, as shown in Figure 27.3.

Figure 27.3 *Components of network management on the Internet*



Let us elaborate on the interactions between these protocols.

Role of SNMP

SNMP has some very specific roles in network management. It defines the format of the packet to be sent from a manager to an agent and vice versa. It also interprets the result and creates statistics (often with the help of other management software). The packets exchanged contain the object (variable) names and their status (values). SNMP is responsible for reading and changing these values.

Role of SMI

To use SNMP, we need rules for naming objects. This is particularly important because the objects in SNMP form a hierarchical structure (an object may have a parent object and some child objects). Part of a name

can be inherited from the parent. We also need rules to define the types of objects. What types of objects are handled by SNMP? Can SNMP handle simple types or structured types? How many simple types are available? What are the sizes of these types? What is the range of these types? In addition, how are each of these types encoded? We need these universal rules because we do not know the architecture of the computers that send, receive, or store these values. The sender may be a powerful computer in which an integer is stored as 8-byte data; the receiver may be a small computer that stores an integer as 4-byte data.

SMI is a protocol that defines these rules. However, we must understand that SMI only defines the rules; it does not define how many objects are managed in an entity or which object uses which type. SMI is a collection of general rules to name objects and to list their types. The association of an object with the type is not done by SMI.

Role of MIB

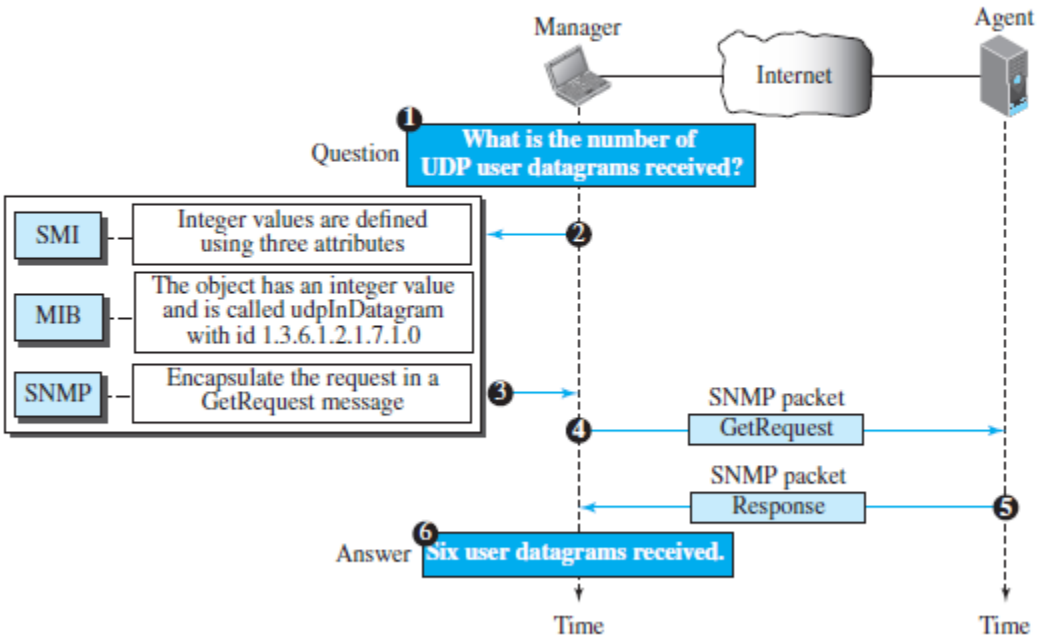
We hope it is clear that we need another protocol. For each entity to be managed, this protocol must define the number of objects, name them according to the rules defined by SMI, and associate a type to each named object. This protocol is MIB. MIB creates a set of objects defined for each entity in a manner similar to that of a database (mostly metadata in a database, names and types without values).

3 An Overview

Before discussing each component in more detail, let us show how each of these components is involved in a simple scenario. This is an overview that will be developed later, at the end of the chapter. A manager station (SNMP client) wants to send a message to an agent station (SNMP server) to find the number of UDP user datagrams received by the agent. Figure 27.5 shows an overview of steps involved.

MIB is responsible for finding the object that holds the number of UDP user datagrams received. SMI, with the help of another embedded protocol, is responsible for encoding the name of the object. SNMP is responsible for creating a message, called a GetRequest message, and encapsulating the encoded message. Of course, things are more complicated than this simple overview, but we first need more details of each protocol.

Figure 27.5 *Management overview*



TWO MARKS

UNIT-I

1. What are the three criteria necessary for an effective and efficient network?

The most important criteria are performance, reliability and security.

Performance of the network depends on number of users, type of transmission medium, and the capabilities of the connected h/w and the efficiency of the s/w.

Reliability is measured by frequency of failure, the time it takes a link to recover from the failure and the network's robustness in a catastrophe.

Security issues include protecting data from unauthorized access and viruses.

2. Group the OSI layers by function?

The seven layers of the OSI model belonging to three subgroups.

Physical, data link and network layers are the *network support layers*; they deal with the physical aspects of moving data from one device to another.

Session, presentation and application layers are the *user support layers*; they allow interoperability among unrelated software systems.

The transport layer ensures *end-to-end reliable data transmission.*

3. What are header and trailers and how do they get added and removed?

Each layer in the sending machine adds its own information to the message it receives from the layer just above it and passes the whole package to the layer just below it. This information is added in the form of headers or trailers. Headers are added to the message at the layers 6,5,4,3, and 2. A trailer is added at layer2. At the receiving machine, the headers or trailers attached to the data unit at the corresponding sending layers are removed, and actions appropriate to that layer are taken.

4. What are the features provided by layering?

Two nice features:

- It decomposes the problem of building a network into more manageable components.

- It provides a more modular design.

5. Why are protocols needed?

In networks, communication occurs between the entities in different systems. Two entities cannot just send bit streams to each other and expect to be understood. For communication, the entities must agree on a protocol. A protocol is a set of rules that govern data communication.

6. What are the two interfaces provided by protocols?

- Service interface
- Peer interface

Service interface- defines the operations that local objects can perform on the protocol.

Peer interface- defines the form and meaning of messages exchanged between protocol peers to implement the communication service.

7. Mention the different physical media?

- Twisted pair(the wire that your phone connects to)
- Coaxial cable(the wire that your TV connects to)
- Optical fiber(the medium most commonly used for high-bandwidth, long-distance links)
- Space(the stuff that radio waves, microwaves and infra red beams propagate through)

8. Define Signals?

Signals are actually electromagnetic waves traveling at the speed of light. The speed of light is, however, medium dependent-electromagnetic waves traveling through copper and fiber do so at about two-thirds the speed of light in vacuum.

9. What is wave's wavelength?

The distance between a pair of adjacent maxima or minima of a wave, typically measured in meters, is called wave's wavelength.

10. Define Modulation?

Modulation -varying the frequency, amplitude or phase of the signal to effect the transmission of information. A simple example of modulation is to vary the power (amplitude) of a single wavelength.

11. Explain the two types of duplex?

- *Full duplex*-two bit streams can be simultaneously transmitted over the links at the same time, one going in each direction.
- *Half duplex*-it supports data flowing in only one direction at a time.

12. What is CODEC?

A device that encodes analog voice into a digital ISDN link is called a CODEC, for *coder/decoder*.

13. What is spread spectrum and explain the two types of spread spectrum?

Spread spectrum is to spread the signal over a wider frequency band than normal in such a way as to minimize the impact of interference from other devices.

- Frequency Hopping
- Direct sequence

14. What are the different encoding techniques?

- NRZ
- NRZI
- Manchester
- 4B/5B

15. How does NRZ-L differ from NRZ-I?

In the NRZ-L sequence, positive and negative voltages have specific meanings: positive for 0 and negative for 1. in the NRZ-I sequence, the voltages are meaningless.

Instead, the receiver looks for changes from one level to another as its basis for recognition of 1s.

16. What are the responsibilities of data link layer?

Specific responsibilities of data link layer include the following. a) Framing b) Physical addressing c) Flow control d) Error control e) Access control.

17. What are the ways to address the framing problem?

- Byte-Oriented Protocols(PPP)
- Bit-Oriented Protocols(HDLC)
- Clock-Based Framing(SONET)

18. Distinguish between peer-to-peer relationship and a primary-secondary relationship. peer -to-peer relationship?

All the devices share the link equally.

Primary-secondary relationship: One device controls traffic and the others must transmit through it.

19. Mention the types of errors and define the terms?

There are 2 types of errors

- Single-bit error.
- Burst-bit error.

Single bit error: The term single bit error means that only one bit of a given data unit (such as byte character/data unit or packet) is changed from 1 to 0 or from 0 to 1.

Burst error: Means that 2 or more bits in the data unit have changed from 1 to 0 from 0 to 1.

20. List out the available detection methods.

There are 4 types of redundancy checks are used in data communication.

- Vertical redundancy checks (VRC).
- Longitudinal redundancy checks (LRC).
- Cyclic redundancy checks (CRC).
- Checksum.

21. Write short notes on VRC.

The most common and least expensive mechanism for error detection is the vertical redundancy check (VRC) often called a parity check. In this technique a redundant bit called a parity bit, is appended to every data unit so, that the total number of 0's in the unit (including the parity bit) becomes even.

22. Write short notes on LRC.

In longitudinal redundancy check (LRC), a block of bits is divided into rows and a redundant row of bits is added to the whole block.

23. Write short notes on CRC.

The third and most powerful of the redundancy checking techniques is the cyclic redundancy checks (CRC) CRC is based on binary division. Here a sequence of redundant bits, called the CRC remainder is appended to the end of data unit.

24. Write short notes on CRC checker.

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

25. Define checksum.

The error detection method used by the higher layer protocol is called checksum. Checksum is based on the concept of redundancy.

26. What are the steps followed in checksum generator?

The sender follows these steps a) the units are divided into k sections each of n bits. b) All sections are added together using 2's complement to get the sum. c) The sum is complemented and become the checksum. d) The checksum is sent with the data.

27. Mention the types of error correcting methods.

There are 2 error-correcting methods.

- Single bit error correction
- Burst error correction.

28. Write short notes on error correction?

It is the mechanism to correct the errors and it can be handled in 2 ways.

- When an error is discovered, the receiver can have the sender retransmit the entire data unit.
- A receiver can use an error correcting coder, which automatically corrects certain errors.

29. What is the purpose of hamming code?

A hamming code can be designed to correct burst errors of certain lengths. So the simple strategy used by the hamming code to correct single bit errors must be redesigned to be applicable for multiple bit correction.

30. What is redundancy?

It is the error detecting mechanism, which means a shorter group of bits or extra bits may be appended at the destination of each unit.

31. Define flow control?

Flow control refers to a set of procedures used to restrict the amount of data. The sender can send before waiting for acknowledgment.

32. Mention the categories of flow control?

There are 2 methods have been developed to control flow of data across communication links. a) Stop and wait- send one from at a time. b) Sliding window- send several frames at a time.

33. What is a buffer?

Each receiving device has a block of memory called a buffer, reserved for storing incoming data until they are processed.

UNIIT-II

1. What are the functions of MAC?

MAC sub layer resolves the contention for the shared media. It contains synchronization, flag, flow and error control specifications necessary to move information from one place to another, as well as the physical address of the next station to receive and route a packet.

2. What are the functions of LLC?

The IEEE project 802 models take the structure of an HDLC frame and divides it into 2 sets of functions. One set contains the end user portion of the HDLC frame – the logical address, control information, and data. These functions are handled by the IEEE 802.2 logical link control (LLC) protocol.

3. What is Ethernet?

Ethernet is a multiple-access network, meaning that a set of nodes send and receive frames over a shared link.

4. Define the term carrier sense in CSMA/CD?

All the nodes can distinguish between idle and a busy-link and "collision detect" means that a node listens as it transmits and can therefore detect when a frame it is transmitting has interfered (collided) with a frame transmitted by another node.

5. Define Repeater?

A repeater is a device that forwards digital signals, much like an amplifier forwards analog signals. However, no more than four repeaters may be positioned between any pairs of hosts, meaning that an Ethernet has a total reach of only 2,500m.

6. Define collision detection?

In Ethernet, all these hosts are competing for access to the same link, and as a consequence, they are said to be in the same collision detection.

7. Why Ethernet is said to be a I-*persistent* protocol?

An adaptor with a frame to send transmits with probability '1 'whenever a busy line goes idle.

8. What is exponential back off?

Once an adaptor has detected a collision and stopped its transmission, it waits a certain amount of time and tries again. Each time it tries to transmit but fails, the adaptor doubles the amount of time it waits before trying again. This strategy of doubling the delay interval between each transmission attempt is a general technique known as exponential back off.

9. What is token holding time (THT)?

It defines that how much data a given node is allowed to transmit each time it possesses the token or equivalently, how long a given node is allowed to hold the token.

10. What are the two classes of traffic in FDDI?

- Synchronous
- Asynchronous

11. What are the four prominent wireless technologies?

- Bluetooth
- Wi-Fi(formally known as 802.11)
- WiMAX(802.16)

- Third generation or 3G cellular wireless.

12. Define Bluetooth?

Bluetooth fills the niche of very short-range communication between mobile phones, PDAs, notebook computers, and other personal or peripheral devices. For example, Bluetooth can be used to connect mobile phones to a headset, or a notebook computer to a printer.

13. What are the four steps involves in scanning?

1. The node sends a Probe frame.
2. All APs within reach reply with a Probe Response frame.
3. The node selects one of the access points, and sends that AP an Association Request frame.
4. The AP replies with an Association Response frame.

14. Explain the term handoff?

If the phone is involved in a call at the time , the call must be transferred to the new base station in what is called a hand off.

15. Define satphones?

Satphones use communication satellites as base stations, communicating on frequency bands that have been reserved internationally for satellite use.

16. How to mediate access to a shared link?

Ethernet,token ring, and several wireless protocols. Ethernet and token ring media access protocols have no central arbitrator of access. Media access in wireless networks is made more complicated by the fact that some nodes may be hidden from each other due to range limitations of radio transmission.

17. Define Aggregation points?

It collects and processes the data they receive from neighboring nodes, and then transmit the processed data. By processing the data incrementally, instead of forwarding all the raw data to the base station, the amount of traffic in the network is reduced.

18. Define Beacons?

Beacon to determine their own absolute locations based on GPS or manual configuration. The majority of nodes can then derive their absolute location by combining an estimate of their position relative to the beacons with the absolute location information provided by the beacons.

19. What is the use of Switch?

It is used to forward the packets between shared media LANs such as Ethernet. Such switches are sometimes known by the obvious name of LAN switches.

20. Explain Bridge?

It is a collection of LANs connected by one or more bridges is usually said to form an extended LAN. In their simplest variants, bridges simply accept LAN frames on their inputs and forward them out on all other outputs.

21. What is Spanning tree?

It is for the bridges to select the ports over which they will forward frames.

22. What are the three pieces of information in the configuration messages?

1. The ID for the bridge that is sending the message.
2. The ID for what the sending bridge believes to the root bridge.
3. The distance, measured in hops, from the sending bridge to the root bridge.

23. What is broadcast?

Broadcast is simple – each bridge forwards a frame with a destination broadcast address out on each active (selected) port other than the one on which the frame was received.

24. What is multicast?

It can be implemented with each host deciding for itself whether or not to accept the message.

25. How does a given bridge learn whether it should forward a multicast frame over a given port?

It learns exactly the same way that a bridge learns whether it should forward a unicast frame over a particular port- by observing the source addresses that it receives over that port.

26. What are the limitations of bridges?

- scale

- heterogeneity

UNIT-III

1. Define packet switching?

A packet switch is a device with several inputs and outputs leading to and from the hosts that the switch interconnects.

2. What is a virtual circuit?

A logical circuit made between the sending and receiving computers. The connection is made after both computers do handshaking. After the connection, all packets follow the same route and arrive in sequence.

3. What are data grams?

In datagram approach, each packet is treated independently from all others. Even when one packet represents just a place of a multi packet transmission, the network treats it although it existed alone. Packets in this technology are referred to as datagram.

4. What is meant by switched virtual circuit?

Switched virtual circuit format is comparable conceptually to dial-up line in circuit switching. In this method, a virtual circuit is created whenever it is needed and exits only for the duration of specific exchange.

5. What is meant by Permanent virtual circuit?

Permanent virtual circuits are comparable to leased lines in circuit switching. In this method, the same virtual circuit is provided between two uses on a continuous basis. The circuit is dedicated to the specific uses.

6. What are the properties in star topology?

- Even though a switch has a fixed number of inputs and outputs, which limits the number of hosts that can be connected to a single switch , large networks can be built by interconnecting a number of switches.
- We can connect switches to each other and to hosts using point-to point links, which typically means that we can build networks of large geographic scope.

7. What is VCI?

A Virtual Circuit Identifier that uniquely identifies the connection at this switch, and which will be carried inside the header of the packets that belongs to this connection.

8. What is hop-by-hop flow control?

Each node is ensured of having the buffers it needs to queue the packets that arrive on that circuit. This basic strategy is usually called hop-by-hop flow control.

9. Explain the term best-effort?

If something goes wrong and the packet gets lost, corrupted, misdelivered, or in any way fails to reach its intended destination, the network does nothing.

10. What is maximum transmission unit?

MTU- which is the largest IP datagram that it can carry in a frame .

11. Define Routing?

It is the process of building up the tables that allow thwe collect output for a packet to be determined.

12. Define ICMP?

Internet Control Message Protocol is a collection of error messages that are sent back to the source host whenever a router or host is unable to process an IP datagram successfully

13. Write the keys for understanding the distance vector routing?

The three keys for understanding the algorithm are,

- Knowledge about the whole networks
- Routing only to neighbors
- Information sharing at regular intervals

14. Write the keys for understanding the link state routing?

The three keys for understanding the algorithm are,

- Knowledge about the neighborhood.
- Routing to all neighbors.

- Information sharing when there is a range.

15. How the packet cost referred in distance vector and link state routing?

In distance vector routing, cost refer to hop count while in case of link state routing, cost is a weighted value based on a variety of factors such as security levels, traffic or the state of the link.

16. Define Reliable flooding?

It is the process of making sure that all the nodes participating in the routing protocol get a copy of the link state information from all the other nodes.

17. What are the features in OSPF?

- Authentication of routing messages.
- Additional hierarchy.
- Load balancing.

18. Define Subnetting?

Subnetting provides an elegantly simple way to reduce the total number of network numbers that are assigned. The idea is to take a single IP network number and allocate the IP address with that network to several physical networks, which are now referred to as subnets.

19. What are the different types of AS?

- Stub AS
- Multi homed AS
- Transit AS

20. What is an Area?

An Area is a set of routers that are administratively configured to exchange link-state information with each other. There is one special area- the backbone area, also known as area 0.

21. What is Source Specific Multicast?

SSM , a receiving host specifies both a multicast group and a specific host .the receiving host would then receive multicast addressed to the specified group, but only if they are from the special sender.

22. What is meant by congestion?

Congestion in a network occurs if user sends data into the network at a rate greater than that allowed by network resources.

23. Why the congestion occurs in network?

Congestion occurs because the switches in a network have a limited buffer size to store arrived packets.

24. What are the rules of non boundary-level masking?

- The bytes in the IP address that corresponds to 255 in the mask will be repeated in the sub network address
- The bytes in the IP address that corresponds to 0 in the mask will change to 0 in the sub network address
- For other bytes, use the bit-wise AND operator.

25. What is LSP?

In link state routing, a small packet containing routing information sent by a router to all other router by a packet called link state packet.

UNIT-IV

1. Explain the main idea of UDP?

The basic idea is for a source process to send a message to a port and for the destination process to receive the message from a port.

2. What are the different fields in pseudo header?

- Protocol number
- Source IP address
- Destination IP addresses.

3. Define TCP?

TCP guarantees the reliable, in order delivery of a stream of bytes. It is a full-duplex protocol, meaning that each TCP connection supports a pair of byte streams, one flowing in each direction.

4. Define Congestion Control?

It involves preventing too much data from being injected into the network, thereby causing switches or links to become overloaded. Thus flow control is an end to an end issue, while congestion control is concerned with how hosts and networks interact.

5. State the two kinds of events trigger a state transition?

- A segment arrives from the peer.
- The local application process invokes an operation on TCP.

6. What is meant by segment?

At the sending and receiving end of the transmission, TCP divides long transmissions into smaller data units and packages each into a frame called a segment.

7. What is meant by segmentation?

When the size of the data unit received from the upper layer is too long for the network layer datagram or data link layer frame to handle, the transport protocol divides it into smaller usable blocks. The dividing process is called segmentation.

8. What is meant by Concatenation?

The size of the data unit belonging to single sessions are so small that several can fit together into a single datagram or frame, the transport protocol combines them into a single data unit. The combining process is called concatenation.

9. What is rate based design?

Rate- based design, in which the receiver tells the sender the rate-expressed in either bytes or packets per second – at which it is willing to accept incoming data.

10. Define Gateway.

A device used to connect two separate networks that use different communication protocols.

11. What is meant by quality of service?

The quality of service defines a set of attributes related to the performance of the connection. For each connection, the user can request a particular attribute each service class is associated with a set of attributes.

12. What are the two categories of QoS attributes?

The two main categories are,

- User Oriented
- Network Oriented

13. List out the user related attributes?

User related attributes are SCR – Sustainable Cell Rate PCR – Peak Cell Rate MCR- Minimum Cell Rate CVDT – Cell Variation Delay Tolerance.

14. What are the networks related attributes?

The network related attributes are, Cell loss ratio (CLR) Cell transfer delay (CTD) Cell delay variation (CDV) Cell error ratio (CER).

15. What is RED?

Random Early Detection in each router is programmed to monitor its own queue length and when it detects that congestion is imminent, to notify the source to adjust its congestion window.

16. What are the three events involved in the connection?

For security, the transport layer may create a connection between the two end ports. A connection is a single logical path between the source and destination that is associated with all packets in a message. Creating a connection involves three steps:

- Connection establishment
- Data transfer
- Connection release

UNIT-V

1. What is the function of SMTP?

The TCP/IP protocol supports electronic mail on the Internet is called Simple Mail Transfer (SMTP). It is a system for sending messages to other computer users based

on e-mail addresses. SMTP provides mail exchange between users on the same or different computers.

2. What is the difference between a user agent (UA) and a mail transfer agent (MTA)?

The UA prepares the message, creates the envelope, and puts the message in the envelope. The MTA transfers the mail across the Internet.

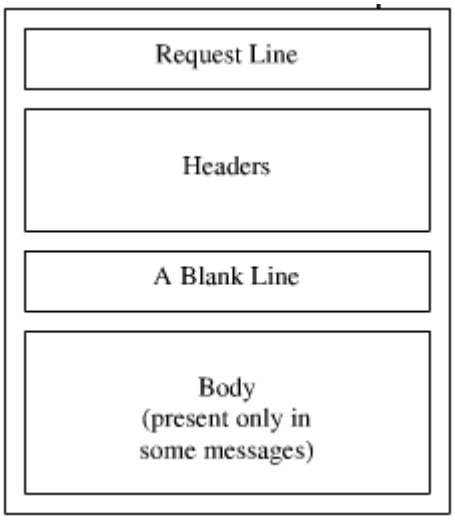
3. How does MIME enhance SMTP?

MIME is a supplementary protocol that allows non-ASCII data to be sent through SMTP. MIME transforms non-ASCII data at the sender site to NVT ASCII data and deliverers it to the client SMTP to be sent through the Internet. The server SMTP at the receiving side receives the NVT ASCII data and delivers it to MIME to be transformed back to the original data.

4. **Why is an application such as POP needed for electronic messaging?**

Workstations interact with the SMTP host, which receives the mail on behalf of every host in the organization, to retrieve messages by using a client-server protocol such as Post Office Protocol, version 3(POP3). Although POP3 is used to download messages from the server, the SMTP client still needed on the desktop to forward messages from the workstation user to its SMTP mail server.

5. Give the format of HTTP request message?



6. **What is the purpose of Domain Name System?**

Domain Name System can map a name to an address and conversely an address to name.

7. Discuss the three main division of the domain name space.

Domain name space is divided into three different sections: generic domains, country domains & inverse domain.

Generic domain: Define registered hosts according to their generic behavior, uses generic suffixes.

Country domain: Uses two characters to identify a country as the last suffix.

Inverse domain: Finds the domain name given the IP address.

8. Discuss the TCP connections needed in FTP.

FTP establishes two connections between the hosts. One connection is used for data transfer, the other for control information. The control connection uses very simple rules of communication. The data connection needs more complex rules due to the variety of data types transferred.

9. Discuss the basic model of FTP.

The client has three components: the user interface, the client control process, and the client data transfer process. The server has two components: the server control process and the server data transfer process. The control connection is made between the control processes. The data connection is made between the data transfer processes.

10. Name four factors needed for a secure network?

Privacy: The sender and the receiver expect confidentiality.

Authentication: The receiver is sure of the sender's identity and that an imposter has not sent the message.

Integrity: The data must arrive at the receiver exactly as it was sent.

Non-Reputation: The receiver must able to prove that a received message came from a specific sender.

11. How is a secret key different from public key?

In secret key, the same key is used by both parties. The sender uses this key and an encryption algorithm to encrypt data; the receiver uses the same key and the corresponding decryption algorithm to decrypt the data. In public key, there are two keys: a private key and a public key. The private key is kept by the receiver. The public key is announced to the public.

12. What is a digital signature?

Digital signature is a method to authenticate the sender of a message. It is similar to that of signing transactions documents when you do business with a bank. In network transactions, you can create an equivalent of an electronic or digital signature by the way you send data.

13. What are the advantages & disadvantages of public key encryption?

Advantages:

a) Remove the restriction of a shared secret key between two entities. Here each entity can create a pair of keys, keep the private one, and publicly distribute the other one.

b) The no. of keys needed is reduced tremendously. For one million users to communicate, only two million keys are needed.

Disadvantage:

If you use large numbers the method to be effective. Calculating the cipher text using the long keys takes a lot of time. So it is not recommended for large amounts of text.

14. What are the advantages & disadvantages of secret key encryption?

Advantage:

Secret Key algorithms are efficient: it takes less time to encrypt a message. The reason is that the key is usually smaller. So it is used to encrypt or decrypt long messages.

Disadvantages:

a) Each pair of users must have a secret key. If N people in world want to use this method, there needs to be N (N-1)/2 secret keys. For one million people to communicate, a half-billion secret keys are needed.

b) The distribution of the keys between two parties can be difficult.

15. Define permutation.

Permutation is transposition in bit level.

Straight permutation: The no. of bits in the input and output are preserved.

Compressed permutation: The no. of bits is reduced (some of the bits are dropped).

Expanded permutation: The no. of bits is increased (some bits are repeated).

16. Define substitution & transposition encryption?

. *Substitution:* A character level encryption in which each character is replaced by another character in the set.

Transposition: A Character level encryption in which the characters retain their plaintext but the position of the character changes.

17. Define CGI?

. CGI is a standard for communication between HTTP servers and executable programs. It is used in crating dynamic documents.

18. What are the requests messages support SNMP and explain it?

- GET
- SET

The former is used to retrieve a piece of state from some node and the latter is used to store a new piece of state in some node.

19. Define PGP?

Pretty Good Privacy is used to provide security for electronic mail. It provides authentication, confidentiality, data integrity, and non repudiation.

20. Define SSH?

Secure Shell is used to provide a remote login, and used to remotely execute commands and transfer files and also provide strong client/server authentication / message integrity.