Unit I

Introduction: Data Communications, Networks, Network Types, Internet History, Standards and Administration. Network Models: 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

DATA COMMUNICATIONS

When we communicate, we are sharing information. This sharing can be local or remote. Between individuals, local communication usually occurs face to face, while remote communication takes place over distance.

The term telecommunication, which includes telephony, telegraphy, and television, means communication at a distance (tele is Greek for "far").

The word data refers to information presented in whatever form is agreed upon by the parties creating and using the data.

Data communications are the exchange of data between two devices via some form of transmission medium such as a wire cable.

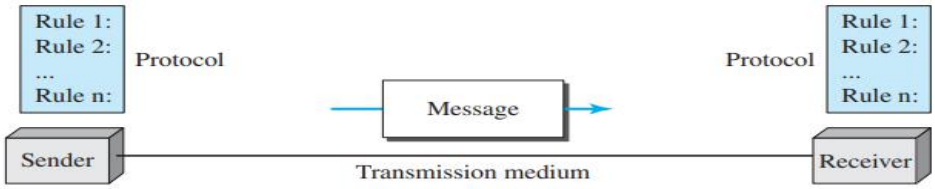
For data communications to occur, the communicating devices must be part of a communication system made up of a combination of hardware (physical equipment) and software (programs).

The effectiveness of a data communications system depends on four fundamental characteristics: delivery, accuracy, timeliness, and jitter.

1. Delivery. The system must deliver data to the correct destination. Data must be received by the intended device or user and only by that device or user.
2. Accuracy. The system must deliver the data accurately. Data that have been altered in transmission and left uncorrected are unusable.
3. Timeliness. The system must deliver data in a timely manner. Data delivered late are useless. In the case of video and audio, timely delivery means delivering data as they are produced, in the same order that they are produced, and without significant delay. This kind of delivery is called real-time transmission.
4. Jitter. Jitter refers to the variation in the packet arrival time. It is the uneven delay in the delivery of audio or video packets. For example, let us assume that video packets are sent every 3D ms. If some of the packets arrive with 3ms delay and others with 4ms delay, an uneven quality in the video is the result.

Components

A data communications system has five components.



1. Message. The message is the information (data) to be communicated. Popular forms of information include text, numbers, pictures, audio, and video.
2. Sender. The sender is the device that sends the data message. It can be a computer, workstation, telephone handset, video camera, and so
3. Receiver. The receiver is the device that receives the message. It can be a computer, workstation, telephone handset, television, and so on.
4. Transmission medium. The transmission medium is the physical path by which a message travels from sender to receiver. Some examples of transmission media include twisted-pair wire, coaxial cable, fiber-optic cable, and radio waves.
5. Protocol. A protocol is a set of rules that govern data communications. It represents an agreement between the communicating devices. Without a protocol, two devices may be connected but not communicating, just as a person speaking French cannot be understood by a person who speaks only Japanese.

Data Representation

Information today comes in different forms such as text, numbers, images, audio, and video.

Text

In data communications, text is represented as a bit pattern, a sequence of bits (0s or 1s). Different sets of bit patterns have been designed to represent text symbols. Each set is called a code, and the process of representing symbols is called coding. Today, the prevalent coding system is called Unicode, which uses 32 bits to represent a symbol or character used in any language in the world. The American Standard Code for Information Interchange (ASCII), developed some decades ago in the United States, now constitutes the first 127 characters in Unicode and is also referred to as Basic Latin.

Numbers

Numbers are also represented by bit patterns. However, a code such as ASCII is not used to represent numbers; the number is directly converted to a binary number to simplify mathematical operations.

Images

Images are also represented by bit patterns. In its simplest form, an image is composed of a matrix of pixels (picture elements), where each pixel is a small dot. The size of the pixel depends on the resolution. For example, an image can be divided into 1000 pixels or 10,000 pixels. In the second case, there is a better representation of the image (better resolution), but more memory is needed to store the image. After an image is divided into pixels, each pixel is assigned a bit pattern. The size and the value of the pattern depend on the image. For an image made of only blackand-white dots (e.g., a chessboard), a 1-bit pattern is enough to represent a pixel. If an image is not made of pure white and pure black pixels, we can increase the size of the bit pattern to include gray scale. For example, to show four levels of gray scale, we can use 2-bit patterns. A black pixel can be represented by 00, a dark gray pixel by 01, a light gray pixel by 10, and a white pixel by 11. There are several methods to represent color images. One method is called RGB, so called because each color is made of a combination of three primary colors: red, green, and blue. The intensity of each color is measured, and a bit pattern is assigned to it. Another method is called YCM, in which a color is made of a combination of three other primary colors: yellow, cyan, and magenta.

Audio

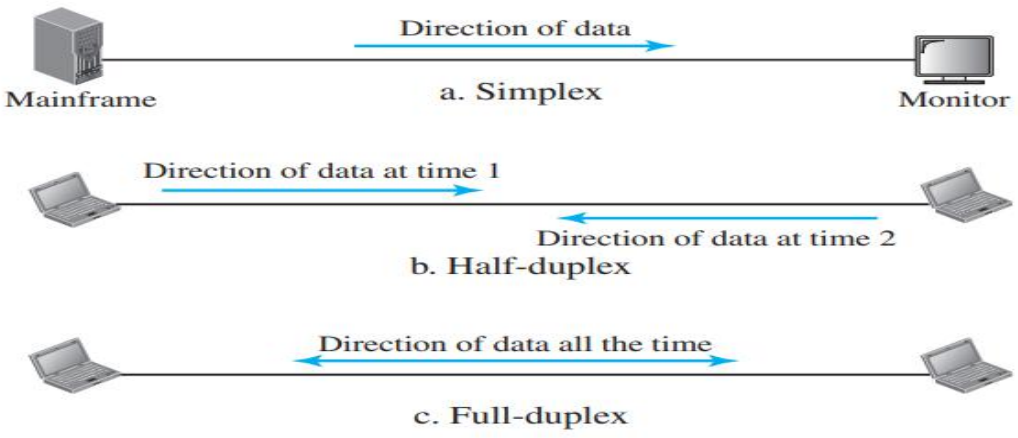
Audio refers to the recording or broadcasting of sound or music. Audio is by nature different from text, numbers, or images. It is continuous, not discrete. Even when we use a microphone to change voice or music to an electric signal, we create a continuous signal.

Video

Video refers to the recording or broadcasting of a picture or movie. Video can either be produced as a continuous entity (e.g., by a TV camera), or it can be a combination of images, each a discrete entity, arranged to convey the idea of motion.

Data Flow

Communication between two devices can be simplex, half-duplex, or full-duplex.



Simplex

In simplex mode, the communication is unidirectional, as on a one-way street. Only one of the two devices on a link can transmit; the other can only receive. Keyboards and traditional monitors are examples of simplex devices. The keyboard can only introduce input; the monitor can only accept output. The simplex mode can use the entire capacity of the channel to send data in one direction.

Half-Duplex

In half-duplex mode, each station can both transmit and receive, but not at the same time. When one device is sending, the other can only receive, and vice versa.

The half-duplex mode is like a one-lane road with traffic allowed in both directions. When cars are traveling in one direction, cars going the other way must wait. In a half-duplex transmission, the entire capacity of a channel is taken over by whichever of the two devices is transmitting at the time. Walkie-talkies and CB (citizens band) radios are both half-duplex systems.

The half-duplex mode is used in cases where there is no need for communication in both directions at the same time; the entire capacity of the channel can be utilized for each direction.

Full-Duplex

In full-duplex mode (also called duplex), both stations can transmit and receive simultaneously.

The full-duplex mode is like a two-way street with traffic flowing in both directions at the same time. In full-duplex mode, signals going in one direction share the capacity of the link with signals going in the other direction. This sharing can occur in two ways: Either the link must contain two physically separate transmission paths, one for sending and the other for receiving; or the capacity of the channel is divided between signals traveling in both directions.

One common example of full-duplex communication is the telephone network. When two people are communicating by a telephone line, both can talk and listen at the same time.

The full-duplex mode is used when communication in both directions is required all the time. The capacity of the channel, however, must be divided between the two directions.

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, and so on. These devices in a network are connected using wired or wireless transmission media such as cable or air. When we connect two computers at home using a plug-and-play router, we have created a network, although very small.

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 throughput and less delay. However, these two criteria are often contradictory. If we try to send more data to the network, we may increase throughput but we increase the delay because of traffic congestion in the network.

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.

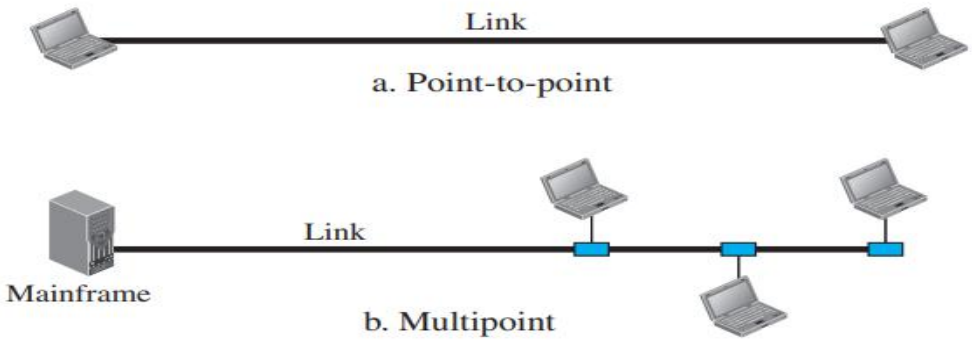
For communication to occur, two devices must be connected in some way to the same link at the same time. There are two possible types of connections: point-to-point and multipoint.

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, but other options, such as microwave or satellite links, are also possible. When we change television channels by infrared remote control, we are establishing a point-to-point connection between the remote control and the television's control system.

Multipoint

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



In a multipoint environment, the capacity of the channel is shared, either spatially or temporally. If several devices can use the link simultaneously, it is a spatially shared connection. If users must take turns, it is a timeshared connection.

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, star, bus, and ring.

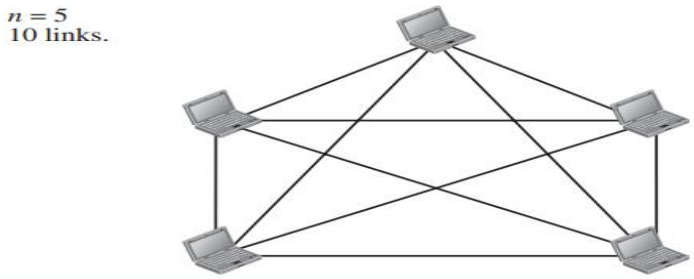
Mesh Topology

In a mesh topology, every device has a dedicated point-to-point link to every other device. The term dedicated means that the link carries traffic only between the two devices it connects. To find the number of physical links in a fully connected mesh network with n nodes, we first consider that each node must be connected to every other node. Node 1 must be connected to n – 1 nodes, node 2 must be connected to n – 1 nodes, and finally node n must be connected to n – 1 nodes. We need n (n – 1) physical links. However, if each physical link allows communication in both directions (duplex mode), we can divide the number of links by 2. In other words, we can say that in a mesh topology, we need n (n – 1) / 2 duplex-mode links. To accommodate that many links, every device on the network must have n – 1 input/output (I/O) ports to be connected to the other n – 1 stations.

A mesh offers several advantages over other network topologies:

1. The use of dedicated links guarantees that each connection can carry its own data load, thus eliminating the traffic problems that can occur when links must be shared by multiple devices.
2. A mesh topology is robust. If one link becomes unusable, it does not incapacitate the entire system.
3. There is the advantage of privacy or security. When every message travels along a dedicated line, only the intended recipient sees it. Physical boundaries prevent other users from gaining access to messages.
4. Point-to-point links make fault identification and fault isolation easy. Traffic can be routed to avoid links with suspected problems. This facility enables the network manager to discover the precise location of the fault and aids in finding its cause and solution.

Figure 1.4 *A fully connected mesh topology (five devices)*



The main disadvantages of a mesh are related to the amount of cabling and the number of I/O ports required.

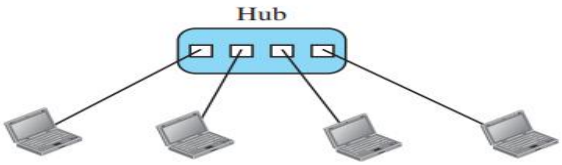
1. Every device must be connected to every other device, installation and reconnection are difficult.
2. The sheer bulk of the wiring can be greater than the available space (in walls, ceilings, or floors) can accommodate.
3. The hardware required to connect each link (I/O ports and cable) can be prohibitively expensive.

For these reasons a mesh topology is usually implemented in a limited fashion, for example, as a backbone connecting the main computers of a hybrid network that can include several other topologies. One practical example of a mesh topology is the connection of telephone regional offices in which each regional office needs to be connected to every other regional office.

Star Topology

In a star topology, each device has a dedicated point-to-point link only to a central controller, usually called a hub. The devices are not directly linked to one another. Unlike a mesh topology, a star topology does not allow direct traffic between devices. The controller acts as an exchange: If one device wants to send data to another, it sends the data to the controller, which then relays the data to the other connected device.

Figure 1.5 *A star topology connecting four stations*



A star topology is less expensive than a mesh topology.

In a star, each device needs only one link and one I/O port to connect it to any number of others. This factor also makes it easy to install and reconfigure. Far less cabling needs to be housed, and additions, moves, and deletions involve only one connection: between that device and the hub.

Other advantages include robustness. If one link fails, only that link is affected. All other links remain active. This factor also lends itself to easy fault identification and fault isolation. As long as the hub is working, it can be used to monitor link problems and bypass defective links.

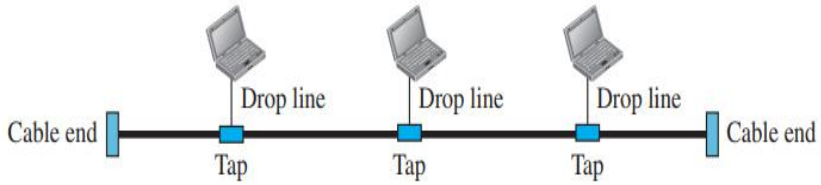
One big disadvantage of a star topology is the dependency of the whole topology on one single point, the hub. If the hub goes down, the whole system is dead. Although a star requires far less cable than a mesh, each node must be linked to a central hub. For this reason, often more cabling is required in a star than in some other topologies (such as ring or bus).

The star topology is used in local-area networks (LANs). High-speed LANs often use a star topology with a central hub.

Bus Topology

A bus topology, on the other hand, is multipoint. One long cable acts as a backbone to link all the devices in a network.

Figure 1.6 *A bus topology connecting three stations*



Nodes are connected to the bus cable by drop lines and taps. A drop line is a connection running between the device and the main cable. A tap is a connector that either splices into the main cable or punctures the sheathing of a cable to create a contact with the metallic core. As a signal travels along the backbone, some of its energy is transformed into heat. Therefore, it becomes weaker and weaker as it travels farther and farther. For this reason there is a limit on the number of taps a bus can support and on the distance between those taps.

Advantages of a bus topology include ease of installation. Backbone cable can be laid along the most efficient path, then connected to the nodes by drop lines of various lengths. In this way, a bus uses less cabling than mesh or star topologies. In a star, for example, four network devices in the same room require four lengths of cable reaching all the way to the hub. In a bus, this redundancy is eliminated. Only the backbone cable stretches through the entire facility. Each drop line has to reach only as far as the nearest point on the backbone.

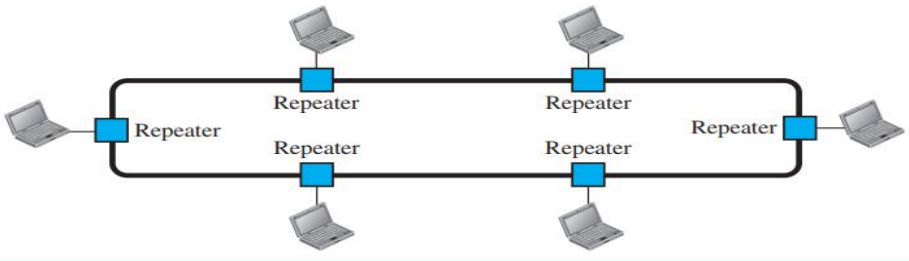
Disadvantages include difficult reconnection and fault isolation. A bus is usually designed to be optimally efficient at installation. It can therefore be difficult to add new devices. Signal reflection at the taps can cause degradation in quality. This degradation can be controlled by limiting the number and spacing of devices connected to a given length of cable. Adding new devices may therefore require modification or replacement of the backbone.

In addition, a fault or break in the bus cable stops all transmission, even between devices on the same side of the problem. The damaged area reflects signals back in the direction of origin, creating noise in both directions.

Ring Topology

In a ring topology, each device has a dedicated point-to-point connection with only the two devices on either side of it. A signal is passed along the ring in one direction, from device to device, until it reaches its destination. Each device in the ring incorporates a repeater. When a device receives a signal intended for another device, its repeater regenerates the bits and passes them along.

Figure 1.7 *A ring topology connecting six stations*



A ring is relatively easy to install and reconfigure. Each device is linked to only its immediate neighbors (either physically or logically). To add or delete a device requires changing only two connections. The only constraints are media and traffic considerations (maximum ring length and number of devices). In addition, fault isolation is simplified. Generally, in a ring a signal is circulating at all times. If one device does not receive a signal within a specified period, it can issue an alarm. The alarm alerts the network operator to the problem and its location.

However, unidirectional traffic can be a disadvantage. In a simple ring, a break in the ring (such as a disabled station) can disable the entire network. This weakness can be solved by using a dual ring or a switch capable of closing off the break.

NETWORK TYPES

The criteria of distinguishing one type of network from another is size, geographical coverage, and ownership.

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.

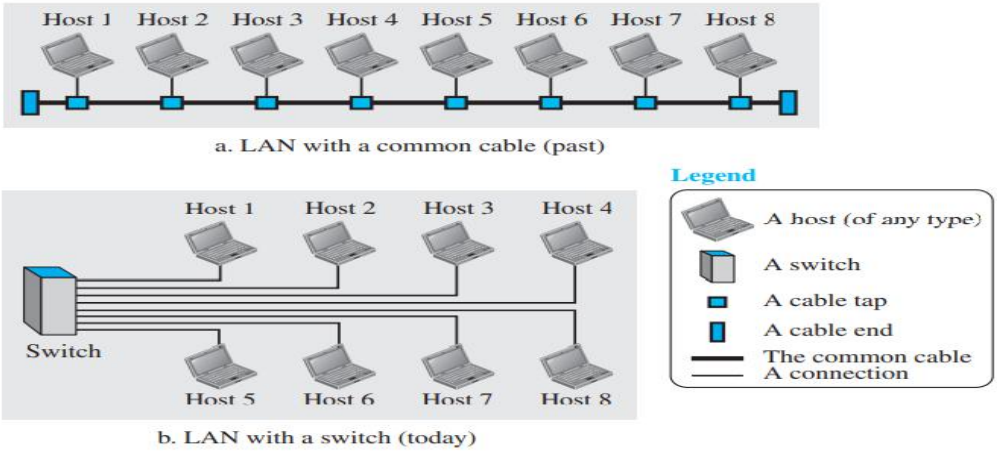
A packet sent by a host to another host carries both the source host's and the destination host's addresses.

In the past, 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. The intended recipient kept the packet; the others dropped the packet.

Today, most LANs use a smart connecting switch, which is able to recognize the destination address of the packet and guide the packet to its destination without sending it to all other hosts.

The switch alleviates the traffic in the LAN and allows more than one pair to communicate with each other at the same time if there is no common source and destination among them.

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



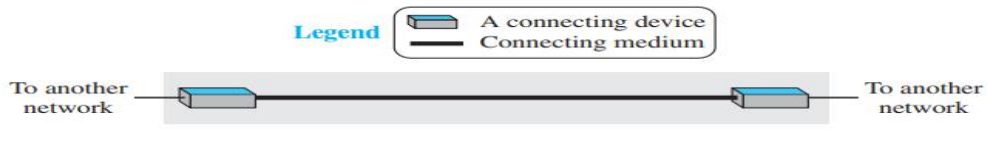
Wide Area Network

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.

Point-to-Point WAN

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

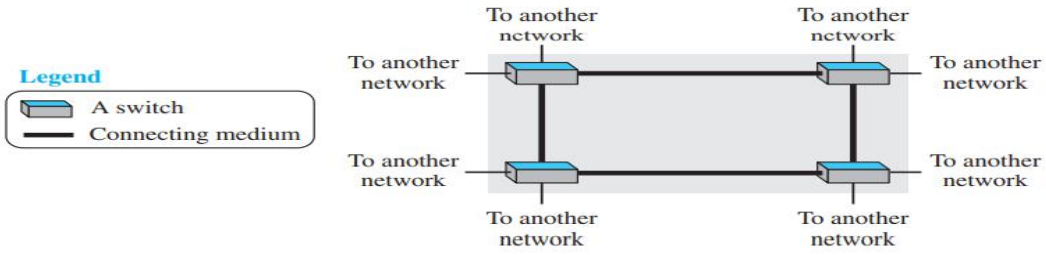
Figure 1.9 *A point-to-point WAN*



Switched WAN

A switched WAN is a network with more than two ends. A switched WAN, 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*



Internetwork

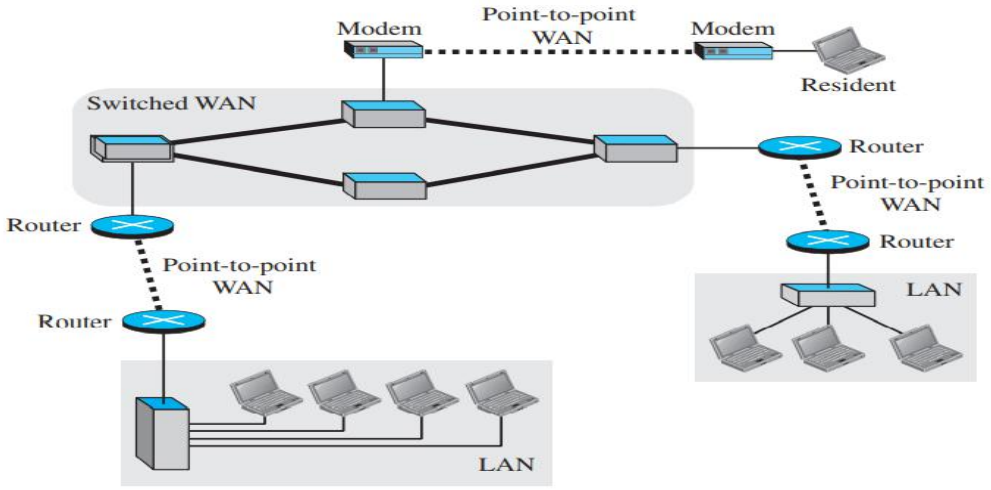
When two or more networks are connected, they make an internetwork, or internet.

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.

Figure 1.12 *A heterogeneous network made of four WANs and three LANs*



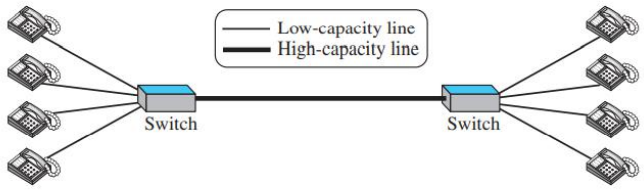
Switching

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.

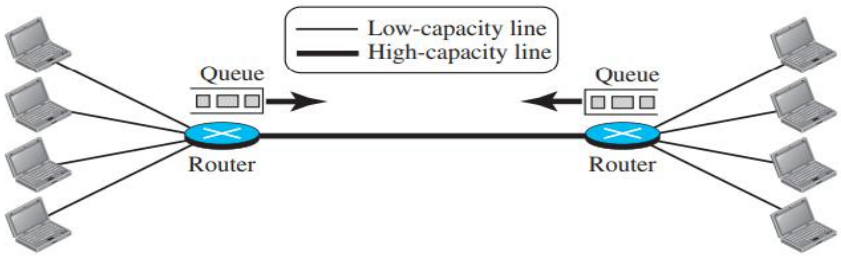
Figure 1.13 *A circuit-switched network*



Packet-Switched Network

In a computer network, the communication between the two ends is done in blocks of data called packets. In other words, 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 *A packet-switched network*

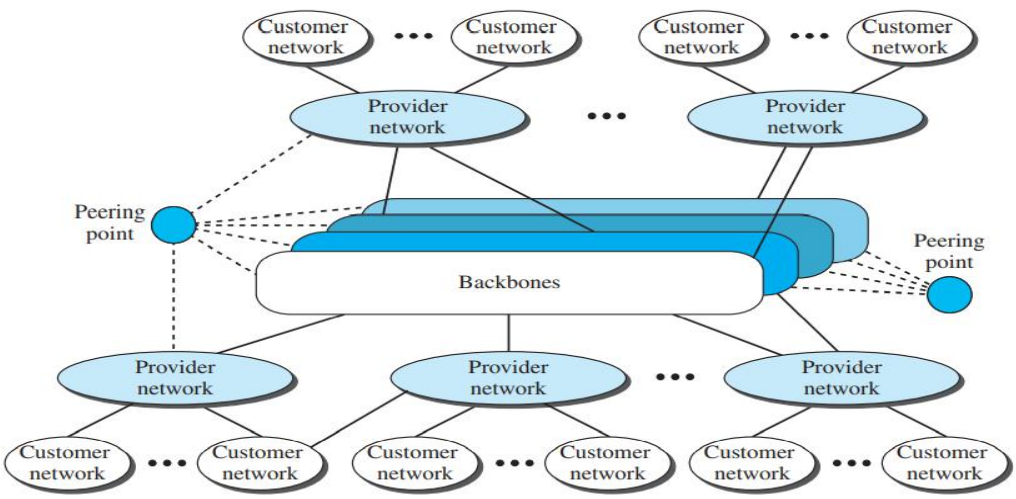


A router in a packet-switched network has a queue that can store and forward the packet. Now assume that the capacity of the thick line is only twice the capacity of the data line connecting the computers to the routers. If only two computers (one at each site) need to communicate with each other, there 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 Internet

The most notable internet is called the Internet (uppercase I), and is composed of thousands of interconnected networks.

Figure 1.15 *The Internet today*



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.

INTERNET HISTORY

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.

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 1981, under a Defence Department contract, UC Berkeley modified the UNIX operating system to include TCP/IP. This inclusion of network software along with a popular operating system did much for the popularity of internetworking. The open (non-manufacturer-specific) implementation of the Berkeley UNIX gave every manufacturer a working code base on which they could build their products.
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.

NSFNET
With the success of CSNET, the NSF in 1986 sponsored the National Science Foundation Network (NSFNET), a backbone that connected five supercomputer centers located throughout the United States. Community networks were allowed access to this backbone, a T-1 line with a 1.544-Mbps data rate, thus providing connectivity throughout the United States. In 1990, ARPANET was officially retired and replaced by NSFNET. In 1995, NSFNET reverted back to its original concept of a research network.

ANSNET
In 1991, the U.S. government decided that NSFNET was not capable of supporting the rapidly increasing Internet traffic. Three companies, IBM, Merit, and Verizon, filled the void by forming a nonprofit organization called Advanced Network & Services (ANS) to build a new, high-speed Internet backbone called Advanced Network Services Network (ANSNET).

Internet Today
Today, we witness a rapid growth both in the infrastructure and new applications. The Internet today is a set of pier networks that provide services to the whole world. What has made the Internet so popular is the invention of new applications.
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.

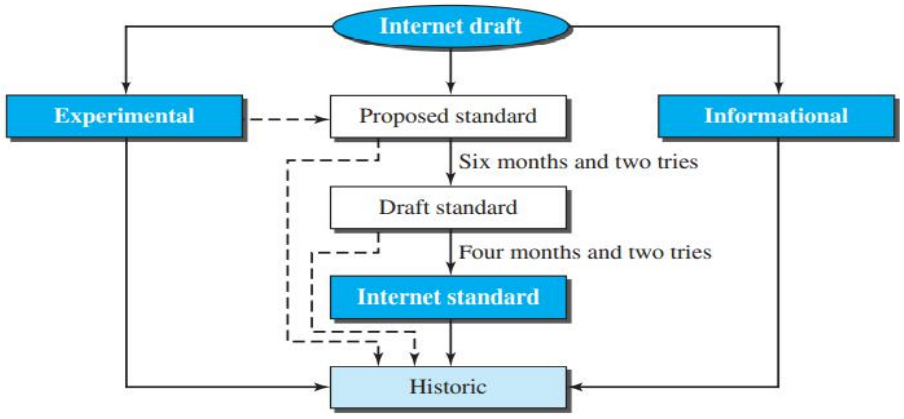
Internet Standards

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.

Figure 1.16 *Maturity levels of an RFC*



- 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

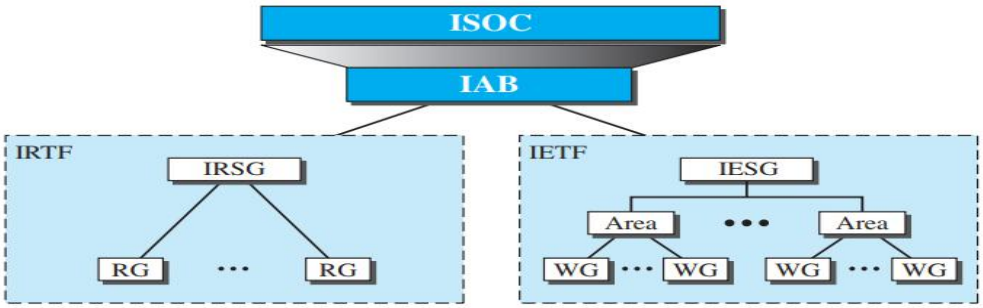
RFCs are classified into five requirement levels: required, recommended, elective, limited use, and not recommended.

- Required. An RFC is labeled required if it must be implemented by all Internet systems to achieve minimum conformance. For example, IP and ICMP 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 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.

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.

Figure 1.17 *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. 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 and 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

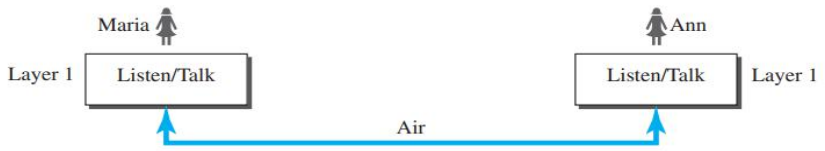
The Internet Research Task Force (IRTF) is a forum of working groups managed by the Internet Research Steering Group (IRSG). IRTF focuses on long-term research topics related to Internet protocols, applications, architecture, and technology.

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.

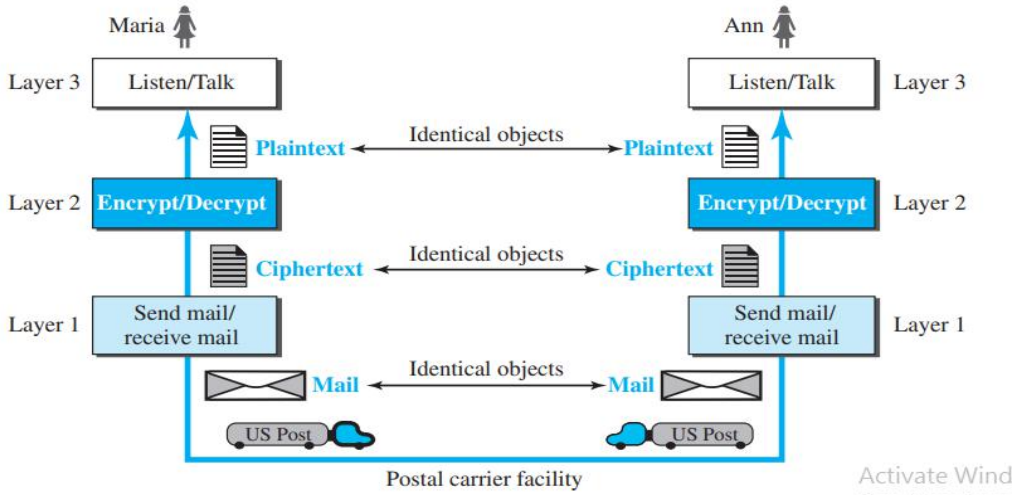
When the communication is simple, we can use only one simple protocol.

Figure 2.1 *A single-layer protocol*



When the communication is complex, we must divide the task between different layers, so, we need to follow a protocol at each layer, this technique we used to call protocol layering. This layering allows us to separate the services from the implementation.

Figure 2.2 *A three-layer protocol*



Each layer needs to receive a set of services from the lower layer and to give the services to the upper layer. The modification done in any one layer will not affect the other layers.

Reasons

The reasons for using layered protocols are −

- Layering of protocols provides well-defined interfaces between the layers, so that a change in one layer does not affect an adjacent layer.
- The protocols of a network are extremely complicated and designing them in layers makes their implementation more feasible.

Advantages

The advantages of layered protocols are −

- Assists in protocol style, as a result of protocols that operate at a particular layer have outlined information that they work and a defined interface to the layers on top of and below.
- Promotes competition because products from completely different vendors will work along.
- Prevents technology or capability changes in one layer from touching different layers above and below.

- Provides a typical language to explain networking functions and capabilities.
- Separates the services from the implementation.
- Communication does not always use only two end systems; there are intermediate systems that need only some layers, but not all layers
- Enables us to divide a complex task into several smaller and simpler tasks

Disadvantages

The disadvantages of layered protocols are −

- The main disadvantages of layered systems consist primarily of overhead each in computation and in message headers caused by the abstraction barriers between layers. Because a message typically should pass through several (10 or more) protocol layers the overhead of those boundaries is commonly more than the computation being done.
- The upper-level layers cannot see what is within the lower layers, implying that an application cannot correct where in an exceedingly connection a problem is or precisely what the matter is.
- The higher-level layers cannot control all aspects of the lower layers, so that they cannot modify the transfer system if helpful (like controlling windowing, header compression, CRC/parity checking, et cetera), nor specify routing, and should rely on the lower protocols operating, and cannot specify alternatives when there are issues.

Principles of Protocol Layering

The basic elements of the layered architecture are as follows −

- Service − Set of actions or services provided from one layer to the higher layer.
- Protocol − It defines a set of rules where a layer uses to exchange the information with its peer entity. It is concerned about both the contents and order of the messages used.
- Interface − It is a way through that the message is transferred from one layer to another layer.

First Principle

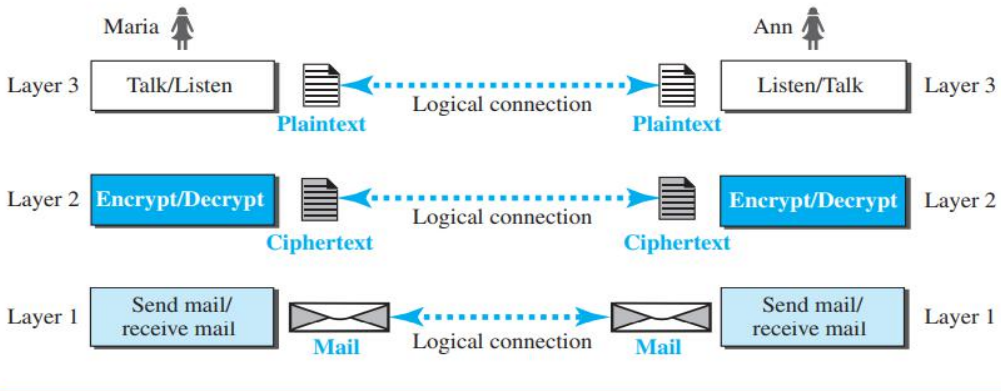
The first principle dictates that if we want bidirectional communication, we need to make each layer so that it is able to perform two opposite tasks, one in each direction. For example, the third layer task is to listen (in one direction) and talk (in the other direction). The second layer needs to 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. For example, 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

Figure 2.3 *Logical connection between peer layers*



Maria and Ann can think that there is a logical (virtual or imaginary) connection at each layer through which they can send the object created from that layer.

TCP/IP PROTOCOL SUITE

(Transmission Control Protocol/Internet Protocol)

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. Today, however, TCP/IP is thought of as a five-layer model.

Layered Architecture

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

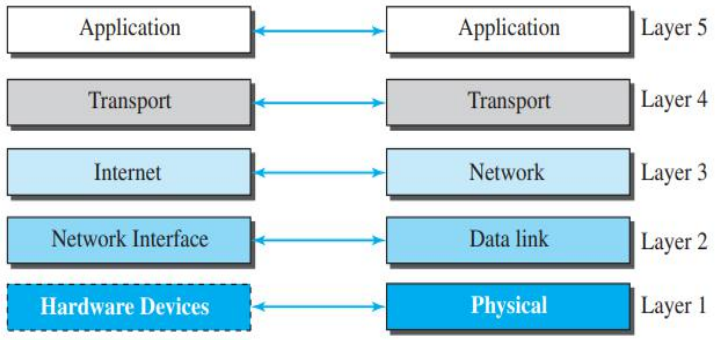
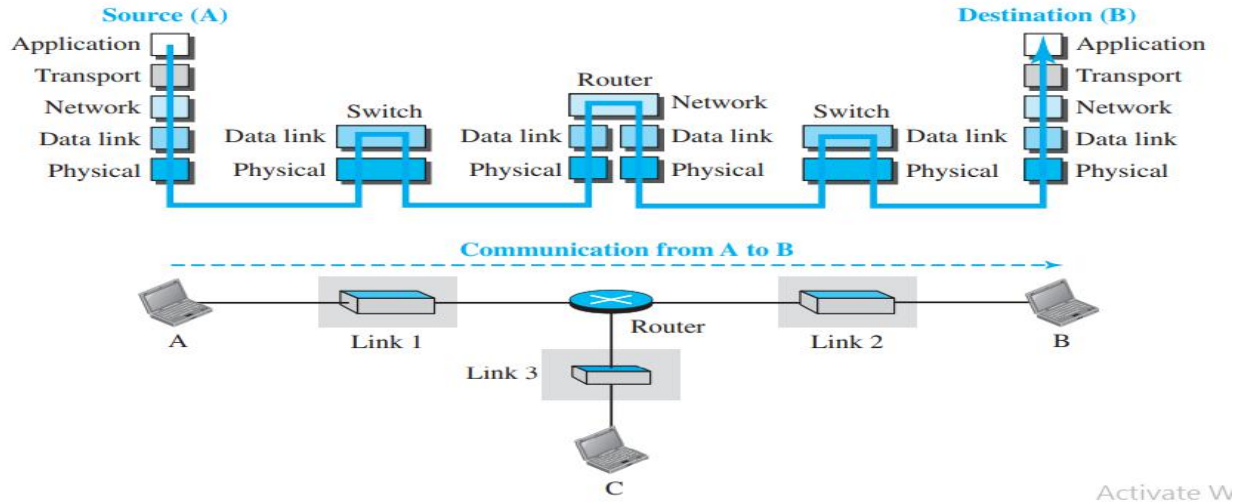


Figure 2.5 *Communication through an internet*



Let us assume that computer A communicates with computer B.

As the figure shows, we have five communicating devices in this communication: source host (computer A), the link-layer switch in link 1, the router, the link-layer switch in link 2, and 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. Although 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. For example, in the above figure, the router is involved in three links, but the message sent from source A to destination B is involved in two links. Each link may be using different link-layer and physical-layer protocols; the router

needs to receive a packet from link 1 based on one pair of protocols and deliver it to link 2 based on another pair of protocols.

A link-layer switch in a link, however, is involved only in two layers, data-link and physical. Although each switch in the above figure has two different connections, the connections are in the same link, which uses only one set of protocols. This means that, unlike a router, a link-layer switch is involved only in one data-link and one physical layer

Layers in the TCP/IP Protocol Suite

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

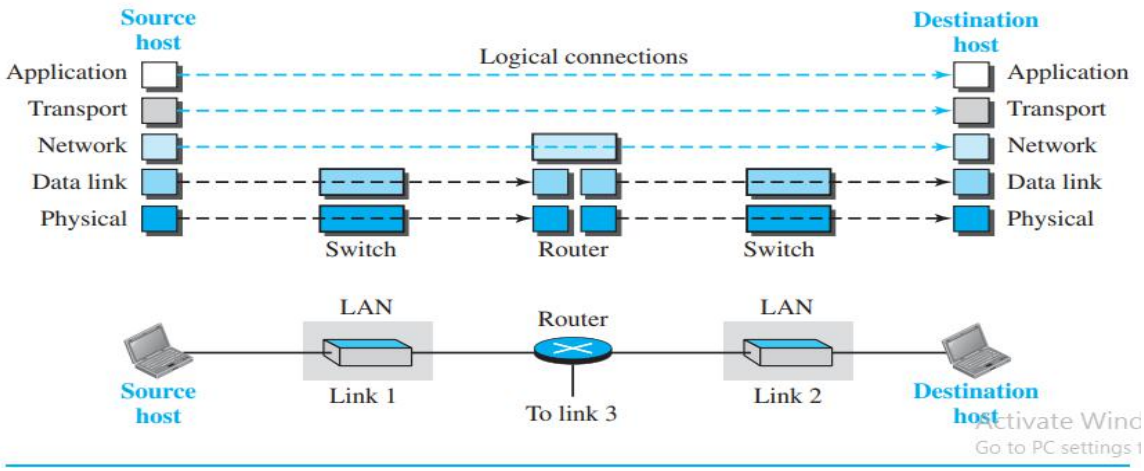
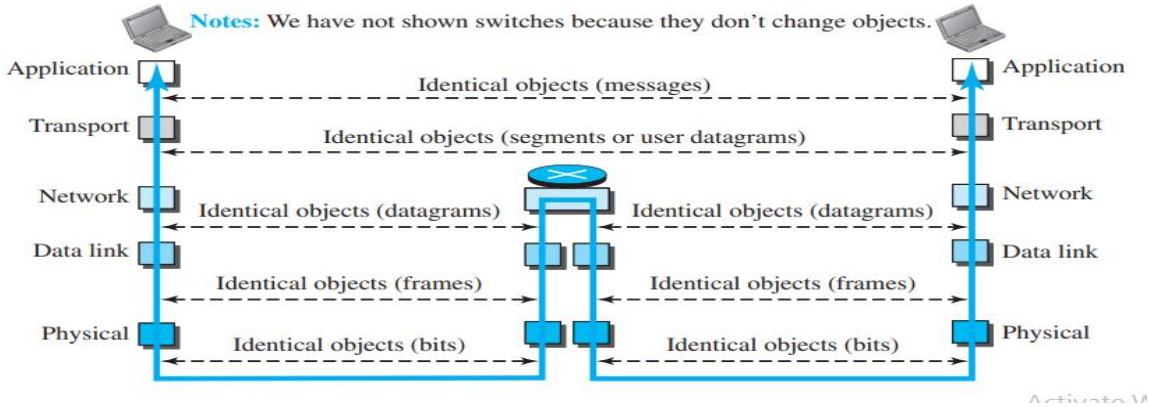


Figure 2.7 *Identical objects in the TCP/IP protocol suite*



Physical Layer

The physical layer is responsible for carrying individual bits in a frame across the link.

Although the physical layer is the lowest level in the TCP/IP protocol suite, the communication between two devices at the physical layer is still a logical communication because there is another, hidden layer, the transmission media, under the physical layer.

Two devices are connected by a transmission medium (cable or air). We need to know that the transmission medium does not carry bits; it carries electrical or optical signals. So the bits received in a frame from the data-link layer are transformed and sent through the transmission media, but we can think that the logical unit between two physical layers in two devices is a bit. There are several protocols that transform a bit to a signal.

Data-link Layer

We have seen that an internet is made up of several links (LANs and WANs) connected by routers. There may be several overlapping sets of links that a datagram can travel from the host to the destination. The routers are responsible for choosing the best links. However, when the next link to travel is determined by the router, the data-link layer is responsible for taking the datagram and moving it across the link. The link can be a wired LAN with a link-layer switch, a wireless LAN, a wired WAN, or a wireless WAN. We can also have different protocols used with any link type. In each case, the data-link layer is responsible for moving the packet through the link. TCP/IP does not define any specific protocol for the data-link layer. It

supports all the standard and proprietary protocols. Any protocol that can take the datagram and carry it through the link suffices for the network layer. The data-link layer takes a datagram and encapsulates it in a packet called a frame. Each link-layer protocol may provide a different service. Some link-layer protocols provide complete error detection and correction, some provide only error correction.

Network Layer

The network layer is responsible for creating a connection between the source computer and the destination computer.

The communication at the network layer is host-to-host. However, since there can be several routers from the source to the destination, the routers in the path are responsible for choosing the best route for each packet.

The network layer is responsible for host-to-host communication and routing the packet through possible routes.

Why we need the network layer? The Reasons are:

1. In a network, there are a number of routes available from the source to the destination. The network layer specifies some strategies which find out the best possible route. This process is referred to as routing.

2. The routers do not need the application and transport layers. Separating the tasks allows us to use fewer protocols on the routers.

Features of Network Layer

1. The main responsibility of the Network layer is to carry the data packets from the source to the destination without changing or using them.
2. If the packets are too large for delivery, they are fragmented i.e., broken down into smaller packets.
3. It decides the route to be taken by the packets to travel from the source to the destination among the multiple routes available in a network (also called routing).
4. The source and destination addresses are added to the data packets inside the network layer.

Services Offered by Network Layer

The **services** which are offered by the network layer protocol are as follows:

1. Packetizing
2. Routing
3. Forwarding

Protocols used in this layer are: IP, ICMP, IGMP, DHCP, and ARP.

Transport Layer

The logical connection at the transport layer is also end-to-end. The transport layer at the source host gets the message from the application layer, encapsulates it in a transportlayer packet (called a segment or a user datagram in different protocols) and sends it, through the logical (imaginary) connection, to the transport layer at the destination host. In other words, the transport layer is responsible for giving services to the application layer: to get a message from an application program running on the source host and deliver it to the corresponding application program on the destination host

Responsibilities of a Transport Layer:

- The Process to Process Delivery
- End-to-End Connection between Hosts
- Multiplexing and Demultiplexing
- Congestion Control
- Data integrity and Error correction
- Flow control

Protocols used in this layer are: TCP, UDP, and SCTP

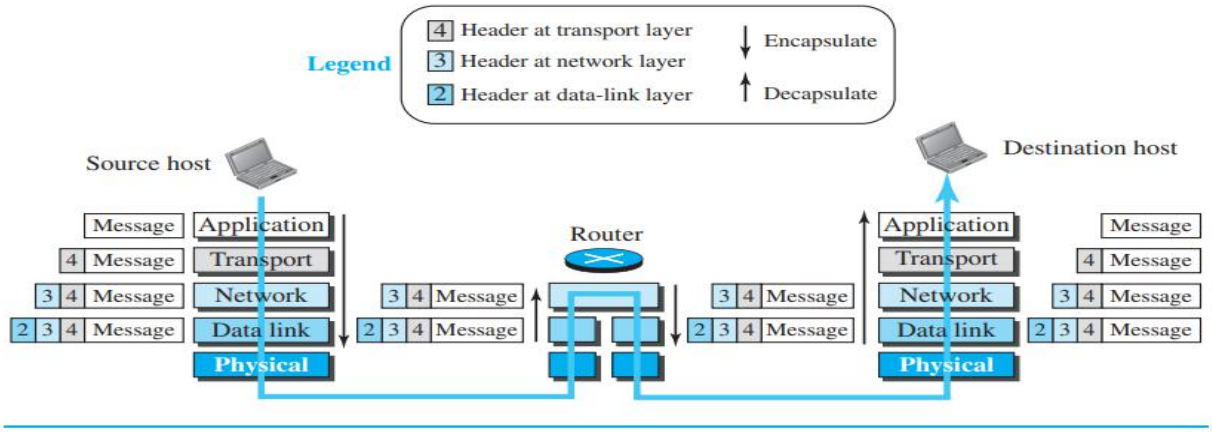
Application Layer

Communication at the application layer is between two processes (two programs running at this layer). To communicate, a process sends a request to the other process and receives a response. Process-to-process communication is the duty of the application layer. The application layer in the Internet includes many predefined protocols, but a user can also create a pair of processes to be run at the two hosts.

Protocols used in this layer are: HTTP, FTP, SMTP, TELNET, SSH, SNMP

Encapsulation and Decapsulation

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, but if it does, we refer to the whole as the message. 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-toend 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.

Decapsulation at the Destination Host

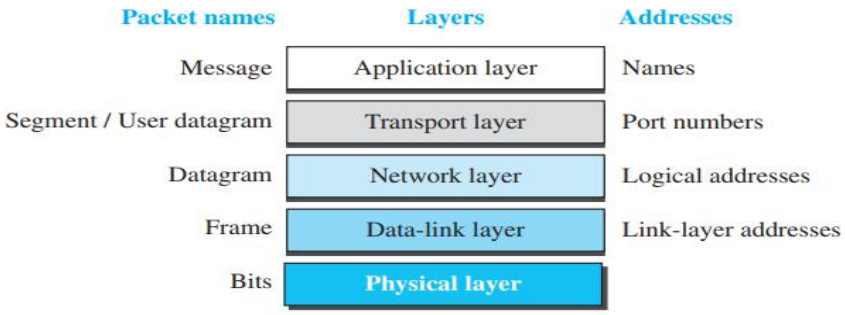
At the destination host, each layer only decapsulates the packet received, removes the payload, and delivers the payload to the next-higher layer protocol until the message reaches the application layer. It is necessary to say that Decapsulation in the host involves error checking.

Addressing

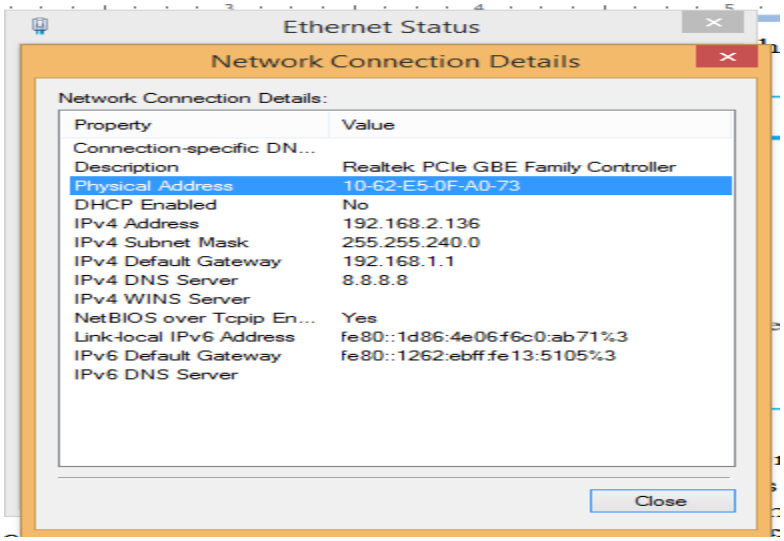
Any communication that involves two parties needs two addresses: source address and destination address.

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*



Names define the site that provides services, such as someorg.com, or the e-mail address.
Port numbers are local addresses that distinguish between several programs running at the same time.
A network-layer address uniquely defines the connection of a device to the Internet like 192.168.2.136
The link-layer addresses, sometimes called MAC addresses, are locally defined addresses, each of which defines a specific host or router in a network (LAN or WAN) like

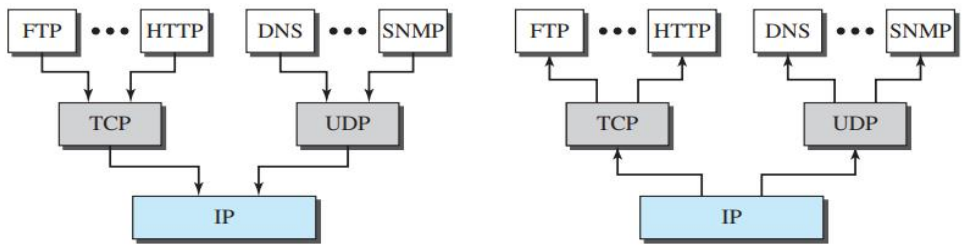


Multiplexing and Demultiplexing

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*



To be able to multiplex and demultiplex, a protocol needs to have a field in its header to identify to which protocol the encapsulated packets belong.

THE OSI MODEL

An ISO standard that covers all aspects of network communications is the Open Systems Interconnection (OSI) model. It was first introduced in the late 1970s.

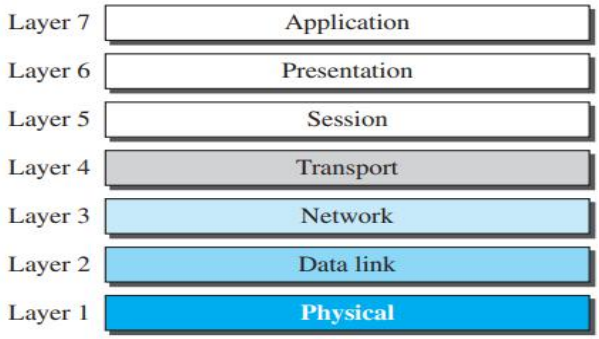
ISO is the organization; OSI is the model.

An open system is a set of protocols that allows any two different systems to communicate regardless of their underlying architecture.

The purpose of the OSI model is to show how to facilitate communication between different systems without requiring changes to the logic of the underlying hardware and software.

The OSI model is not a protocol; it is a model for understanding and designing a network architecture that is flexible, robust, and interoperable. The OSI model was intended to be the basis for the creation of the protocols in the OSI stack. The OSI model is a layered framework for the design of network systems that allows communication between all types of computer systems. It consists of seven separate but related layers, each of which defines a part of the process of moving information across a network.

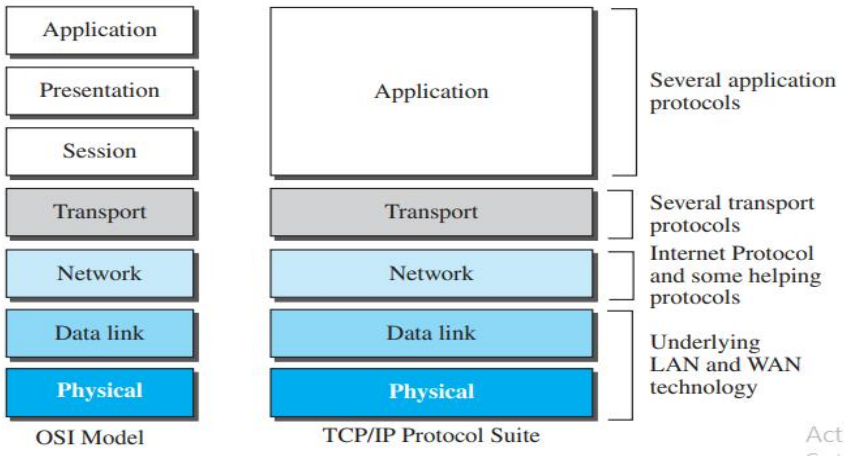
Figure 2.11 *The OSI model*



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.

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.

Introduction to Physical Layer

One of the major functions of the physical layer is to move data in the form of electromagnetic signals across a transmission medium.

Whether you are collecting numerical statistics from another computer, sending animated pictures from a design workstation, or causing a bell to ring at a distant control center, you are working with the transmission of data across network connections.

Generally, the data usable to a person or application are not in a form that can be transmitted over a network. For example, a photograph must first be changed to a form that transmission media can accept.

Transmission media work by conducting energy along a physical path. For transmission, data needs to be changed to signals.

DATA AND SIGNALS

Communication at application, transport, network, or data-link is logical; communication at the physical layer is physical.

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.

Analog and Digital Data

Data can be analog or digital. The term analog data refers to information that is continuous; digital data refers to information that has discrete states. 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. On the other hand, a digital clock that reports the hours and the minutes will change suddenly from 8:05 to 8:06.

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.

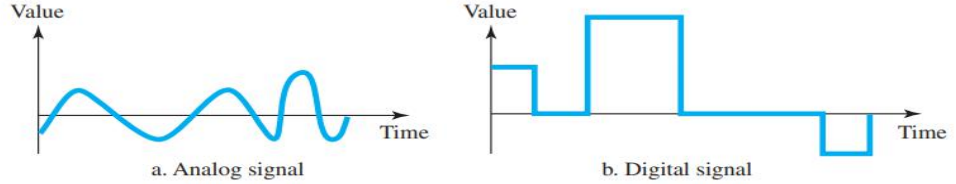
Digital data take on discrete values. For example, data are stored in computer memory in the form of 0s and 1s. They can be converted to a digital signal or modulated into an analog signal for transmission across a medium.

Analog and Digital Signals

Like the data they represent, 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 Nonperiodic

Both analog and digital signals can take one of two forms: periodic or nonperiodic (sometimes referred to as aperiodic; the prefix a in Greek means “non”).

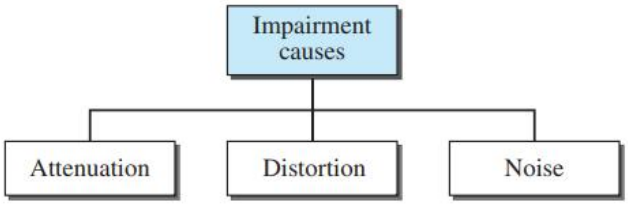
A periodic signal completes a pattern within a measurable time frame, called a period, and repeats that pattern over subsequent identical periods. The completion of one full pattern is called a cycle.
A nonperiodic signal changes without exhibiting a pattern or cycle that repeats over time. Both analog and digital signals can be periodic or nonperiodic.
In data communications, we commonly use periodic analog signals and nonperiodic digital signals.

Definitions:
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.
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.
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.
Frequency and period are the inverse of each other.
Period is formally expressed in seconds. Frequency is formally expressed in Hertz (Hz), which is cycle per second.
If a signal does not change at all, its frequency is zero. If a signal changes instantaneously, its frequency is infinite.
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.
Wavelength binds the period or the frequency of a simple sine wave to the propagation speed of the medium. 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.
The range of frequencies contained in a composite signal is its bandwidth. The bandwidth of a composite signal is the difference between the highest and the lowest frequencies contained in that signal.
The bit rate is the number of bits sent in 1s, expressed in bits per second (bps).
The bit length is the distance one bit occupies on the transmission medium.
Bit length = propagation speed X bit duration

TRANSMISSION IMPAIRMENT

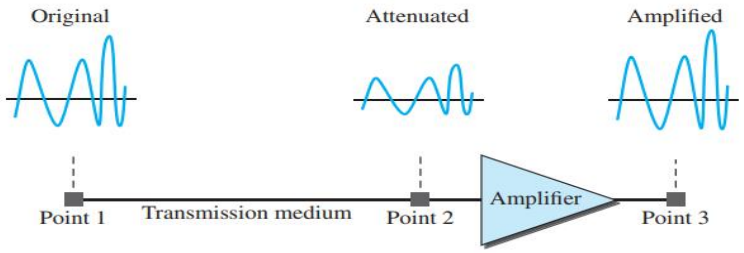
Signals travel through transmission media, which are not perfect. The imperfection causes signal impairment. This means that the signal at the beginning of the medium is not the same as the signal at the end of the medium. What is sent is not what is received.

Figure 3.26 *Causes of impairment*



Attenuation
Attenuation means a loss of energy. When a signal, simple or composite, travels through a medium, it loses some of its energy in overcoming the resistance of the medium. That is why a wire carrying electric signals gets warm, if not hot, after a while. 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*



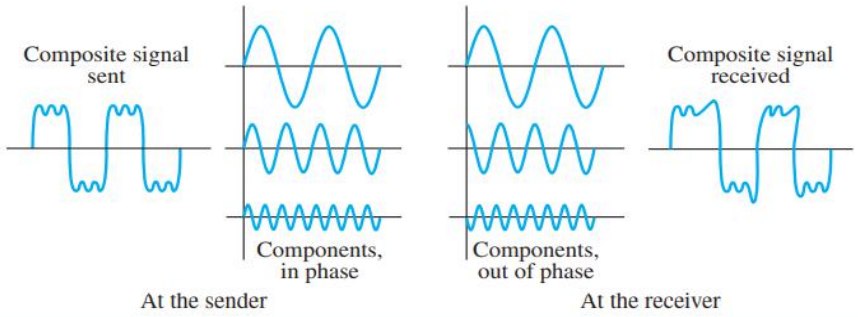
Decibel

To show that a signal has lost or gained strength, engineers use the unit of the 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.

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

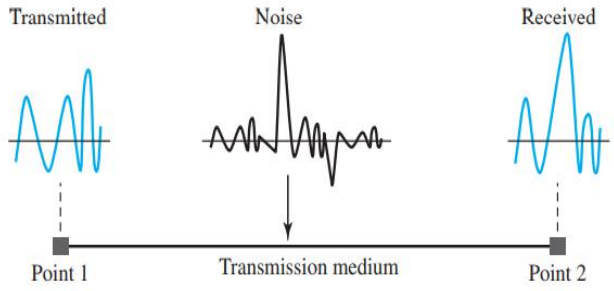
Thermal noise is the random motion of electrons in a wire, which creates an extra signal not originally sent by the transmitter.

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

DATA RATE LIMITS

Data rate governs the speed of data transmission. A very important consideration in data communication is how fast we can send data, in bits per second, over a channel. Data rate depends upon 3 factors:

- The bandwidth available
- Number of levels in digital signal
- The quality of the channel – 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.

1. Noiseless Channel: Nyquist Bit Rate:

For a noiseless channel, the Nyquist bit rate formula defines the theoretical maximum bit rate. ***Nyquist*** proved that if an arbitrary signal has been run through a low-pass filter of bandwidth, the filtered signal can be completely reconstructed by making only 2*Bandwidth (exact) samples per second. Sampling the line faster than 2*Bandwidth times per second is pointless because the higher-frequency components that such sampling could recover have already been filtered out. If the signal consists of L discrete levels, Nyquist's theorem states:

BitRate = 2 * Bandwidth * $\log_2$(L) bits/sec

In the above equation, bandwidth is the bandwidth of the channel, L is the number of signal levels used to represent data, and BitRate is the bit rate in bits per second.

Bandwidth is a fixed quantity, so it cannot be changed. Hence, the data rate is directly proportional to the number of signal levels.

2. Noisy Channel Shannon Capacity:

In reality, we cannot have a noiseless channel; the channel is always noisy. Shannon capacity is used, to determine the theoretical highest data rate for a noisy channel: Capacity = bandwidth * $\log_2$(1 + SNR) bits/sec

In the above equation, 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. Bandwidth is a fixed quantity, so it cannot be changed. Hence, the channel capacity is directly proportional to the power of the signal, as SNR = (Power of signal) / (power of noise).

The signal-to-noise ratio (S/N) is usually expressed in decibels (dB) given by the formula:

10 * $\log_{10}$(S/N)

The maximum data rate, also known as the channel capacity, is the theoretical limit of the amount of information that can be transmitted over a communication channel.

3. Using Both Limits:

In practice, we need to use both methods to find the limits and signal levels. The Shannon capacity gives us the upper limit; the Nyquist formula tells us how many signal levels we need.

PERFORMANCE

One important issue in networking is the performance of the network—how good is it?

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.

2. The throughput is a measure of how fast we can actually send data through a network.

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.

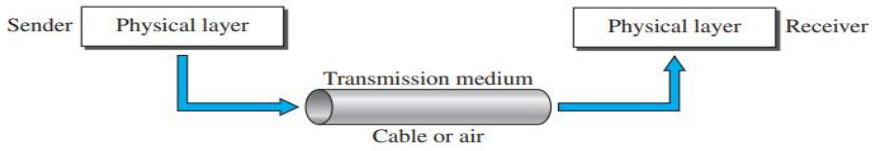
Latency = propagation time + transmission time + queuing time + processing delay

- Propagation time 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 transmission time of a message depends on the size of the message and the bandwidth of the channel.
  Transmission time = (Message size) / Bandwidth
- 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.

Transmission media

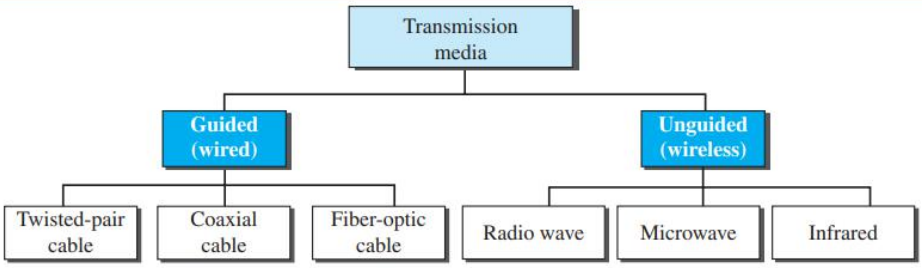
Transmission media are actually located below the physical layer and are directly controlled by the physical layer.

Figure 7.1 *Transmission medium and 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 dinner conversation is the air. The air can also be used to convey the message in a smoke signal or semaphore. For a written message, the transmission medium might be a mail carrier, a truck, or an airplane.

Figure 7.2 *Classes of transmission media*



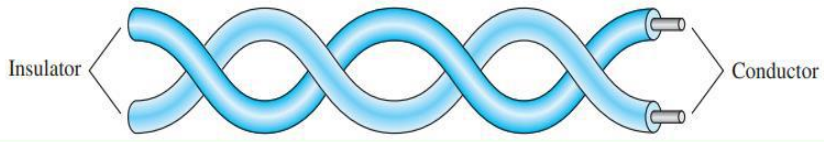
GUIDED MEDIA

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

A twisted pair consists of two conductors (normally copper), each with its own plastic insulation, twisted together.

Figure 7.3 *Twisted-pair cable*



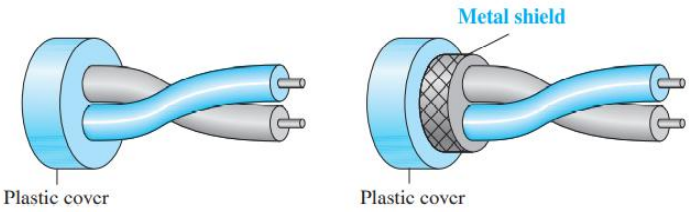
One of the wires is used to carry signals to the receiver, and the other is used only as a ground reference. The receiver uses the difference between the two. In addition to the signal sent by the sender on one of the wires, interference (noise) and crosstalk may affect both wires and create unwanted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

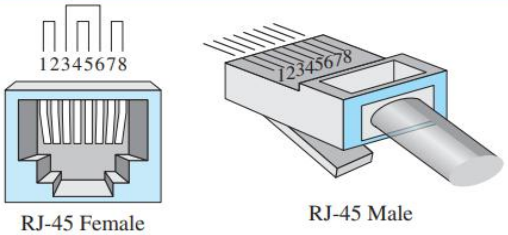
UTP Categories - Copper Cable

UTP Category	Data Rate	Max. Length	Cable Type	Application
CAT1	Up to 1Mbps	-	Twisted Pair	Old Telephone Cable
CAT2	Up to 4Mbps	-	Twisted Pair	Token Ring Networks
CAT3	Up to 10Mbps	100m	Twisted Pair	Token Rink & 10BASE-T Ethernet
CAT4	Up to 16Mbps	100m	Twisted Pair	Token Ring Networks
CAT5	Up to 100Mbps	100m	Twisted Pair	Ethernet, FastEthernet, Token Ring
CAT5e	Up to 1 Gbps	100m	Twisted Pair	Ethernet, FastEthernet, Gigabit Ethernet
CAT6	Up to 10Gbps	100m	Twisted Pair	GigabitEthernet, 10G Ethernet (55 meters)
CAT6a	Up to 10Gbps	100m	Twisted Pair	GigabitEthernet, 10G Ethernet (55 meters)
CAT7	Up to 10Gbps	100m	Twisted Pair	GigabitEthernet, 10G Ethernet (100 meters)

Connectors

The most common UTP connector is RJ45 (RJ stands for registered jack). 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.

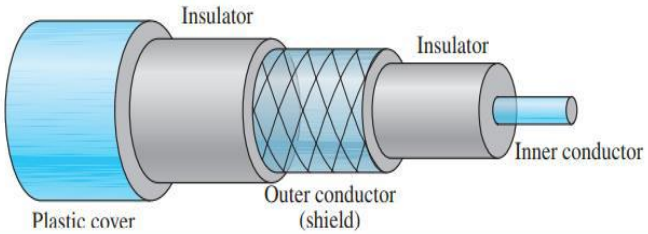
Local-area networks, such as 10Base-T and 100Base-T, also use twisted-pair cables.

Coaxial cable

Coaxial cable (or coax) carries signals of higher frequency ranges than those in twistedpair cable, in part because the two media are constructed quite differently. Instead of having two wires, coax has a central

core conductor of solid or stranded wire (usually copper) enclosed in an insulating sheath, which is, in turn, encased in an outer conductor of metal foil, braid, or a combination of the two. The outer metallic wrapping serves both as a shield against noise and as the second conductor, which completes the circuit. This outer conductor is also enclosed in an insulating sheath, and the whole cable is protected by a plastic cover.

Figure 7.7 *Coaxial cable*



Coaxial Cable Standards

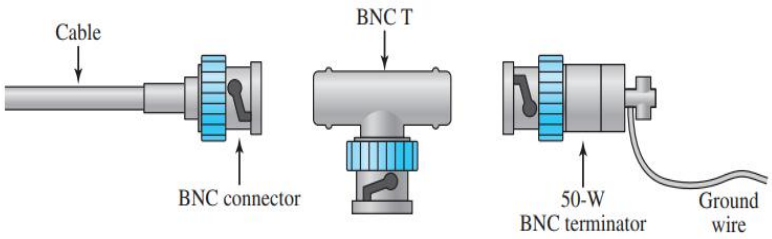
Coaxial cables are categorized by their Radio Government (RG) ratings. Each RG number denotes a unique set of physical specifications, including the wire gauge of the inner conductor, the thickness and type of the inner insulator, the construction of the shield, and the size and type of the outer casing.

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

The most common type of connector used today is the Bayonet Neill-Concelman (BNC) connector.

Figure 7.8 *BNC connectors*



The BNC connector is used to connect the end of the cable to a device, such as a TV set. The BNC T connector is used in Ethernet networks to branch out to a connection to a computer or other device. The BNC terminator is used at the end of the cable to prevent the reflection of the signal.

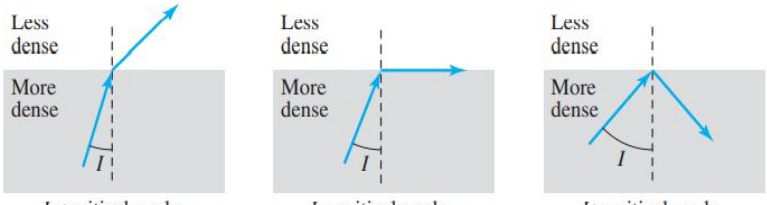
Fiber-Optic Cable

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 more dense to a less dense substance.

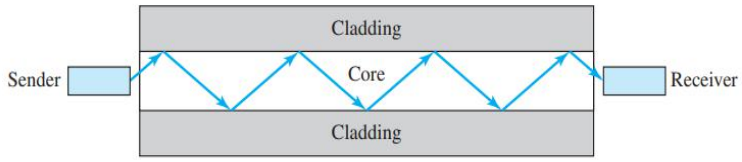
Figure 7.10 *Bending of light ray*



As the figure shows, 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

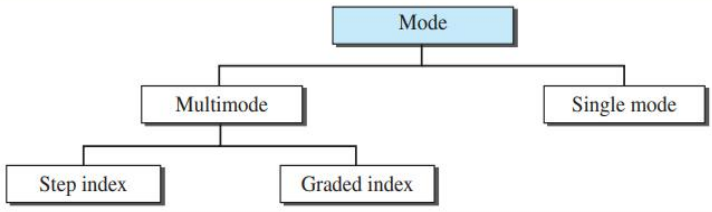
Optical fibers use reflection to guide light through a channel. A glass or plastic core is surrounded by a cladding of less dense glass or plastic. The difference in density of the two materials must be such that a beam of light moving through the core is reflected off the cladding instead of being refracted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

Figure 7.11 *Optical fiber*



Propagation Modes

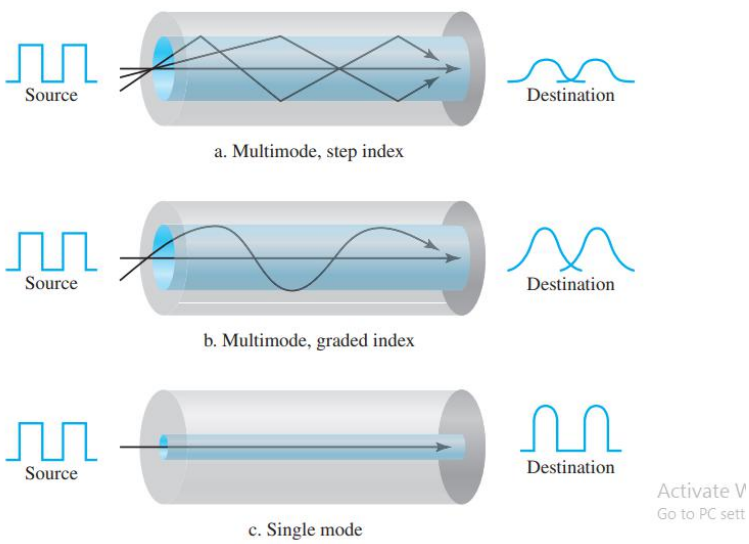
Figure 7.12 *Propagation modes*



Multimode

Multimode is so named because multiple beams from a light source move through the core in different paths. Beams move within the cable depends on the structure of the core.

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.

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.

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.

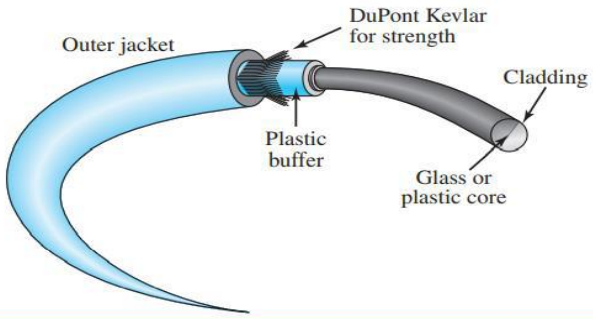
Table 7.3 *Fiber types*

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

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.

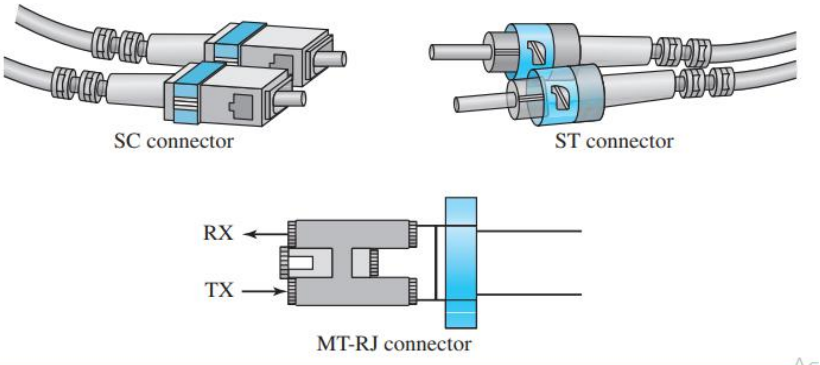
Figure 7.14 *Fiber construction*



Fiber-Optic Cable Connectors

There are three types of connectors for fiber-optic cables. 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*



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

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 *Electromagnetic spectrum for wireless communication*

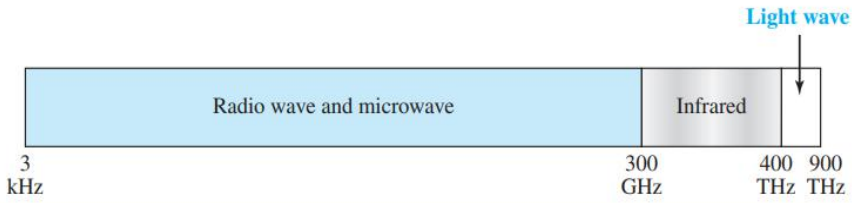
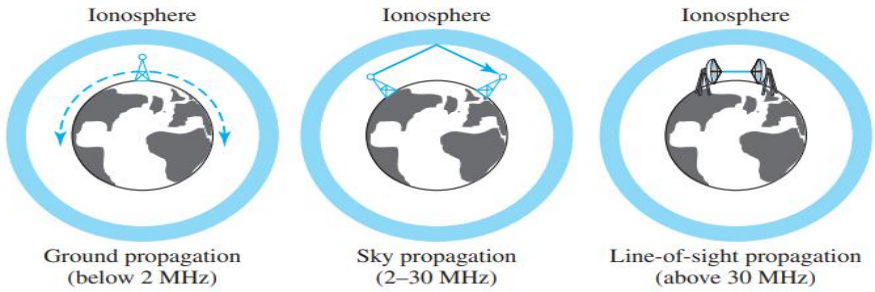


Figure 7.17 shows the part of the electromagnetic spectrum, ranging from 3 kHz to 900 THz, used for wireless communication. Unguided signals can travel from the source to the destination in several ways: ground propagation, sky propagation, and line-of-sight propagation,

Unguided signals can travel from the source to the destination in several ways: ground propagation, sky propagation, and line-of-sight propagation.

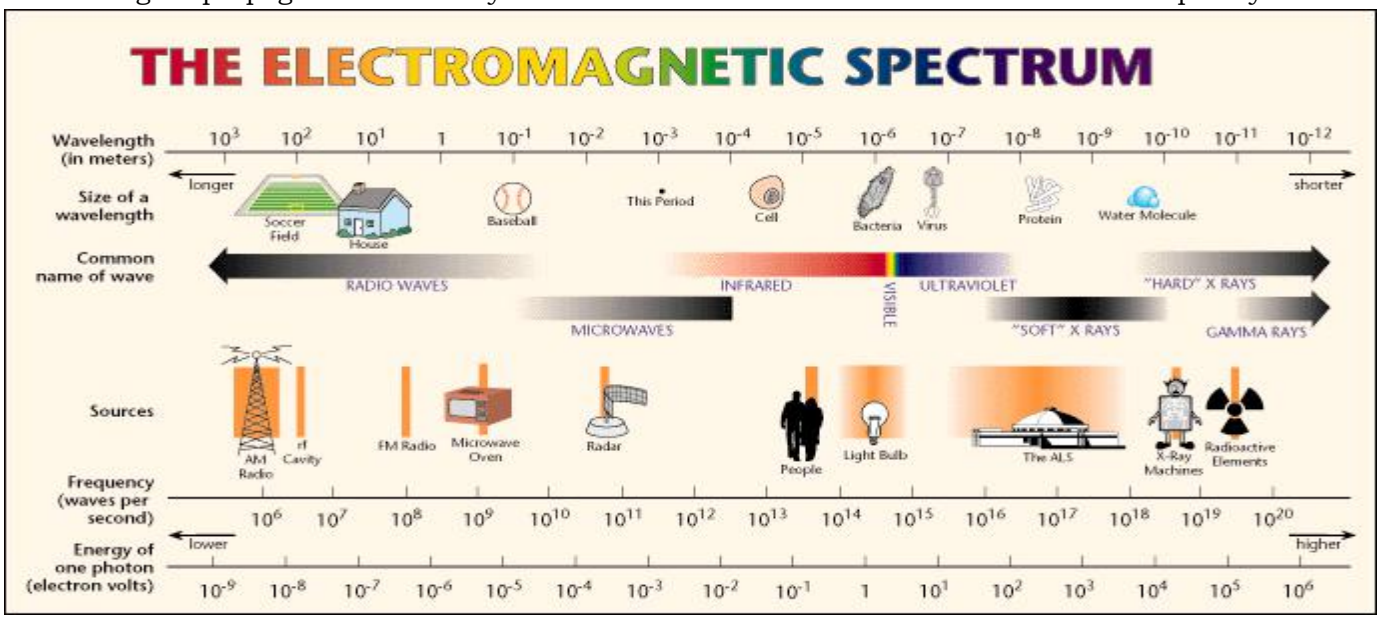
Figure 7.18 *Propagation methods*



In ground propagation, radio waves travel through the lowest portion of the atmosphere, hugging the earth. These low-frequency signals emanate in all directions from the transmitting antenna and follow the curvature of the planet. Distance depends on the amount of power in the signal: The greater the power, the greater the distance.

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. 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.



The section of the electromagnetic spectrum defined as radio waves and microwaves is divided into eight ranges, called bands, each regulated by government authorities. These bands are rated from very low frequency (VLF) to extremely high frequency (EHF).

Band	Range	Propagation	Application
very low frequency (VLF)	3–30 kHz	Ground	Long-range radio navigation
low frequency (LF)	30–300 kHz	Ground	Radio beacons and navigational locators
middle frequency (MF)	300 kHz–3 MHz	Sky	AM radio
high frequency (HF)	3–30 MHz	Sky	Citizens band (CB), ship/aircraft
very high frequency (VHF)	30–300 MHz	Sky and line-of-sight	VHF TV, FM radio
ultrahigh frequency (UHF)	300 MHz–3 GHz	Line-of-sight	UHF TV, cellular phones,

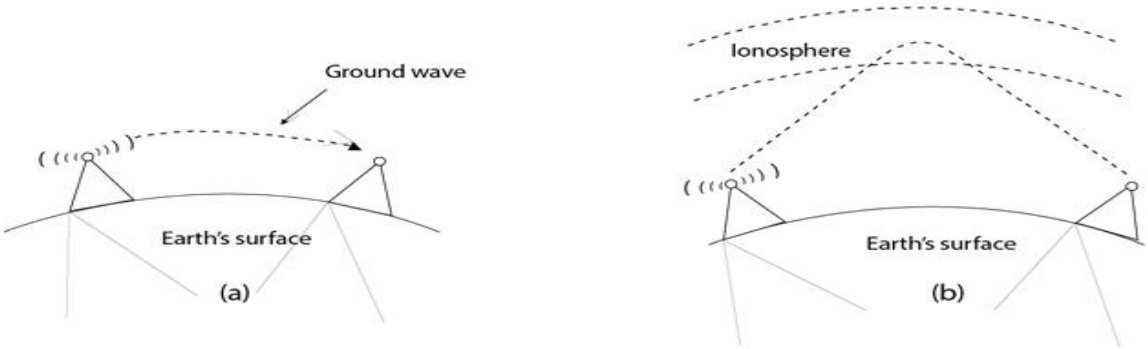
			paging, satellite
superhigh frequency (SF)	3–30 GHz	Line-of-sight	Satellite
extremely high frequency (EHF)	30–300 GHz	Line-of-sight	Radar, satellite

We can divide wireless transmission into three broad groups: radio waves, microwaves, and infrared waves.
An unguided transmission transmits the electromagnetic waves without using any physical medium. Therefore it is also known as wireless transmission. In unguided media, air is the media through which the electromagnetic energy can flow easily.

Unguided transmission is broadly classified into three categories:

Radio waves

- Radio waves are the electromagnetic waves that are transmitted in all the directions of free space.
- Radio waves are omnidirectional, i.e., the signals are propagated in all the directions.
- The range in frequencies of radio waves is from 3Khz to 1 khz.
- In the case of radio waves, the sending and receiving antenna are not aligned, i.e., the wave sent by the sending antenna can be received by any receiving antenna.
- An example of the radio wave is FM radio.



Applications Of Radio waves:

- A Radio wave is useful for multicasting when there is one sender and many receivers.
- An FM radio, television, cordless phones are examples of a radio wave.

Advantages Of Radio transmission:

- Radio transmission is mainly used for wide area networks and mobile cellular phones.
- Radio waves cover a large area, and they can penetrate the walls.
- Radio transmission provides a higher transmission rate.

Figure 7.19 *Omnidirectional antenna*

Applications
The omnidirectional characteristics of radio waves make them useful for multicasting, in which there is one sender but many receivers. AM and FM radio, television, maritime radio, cordless phones, and paging are examples of multicasting.

Microwaves
Waves ranging in 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.

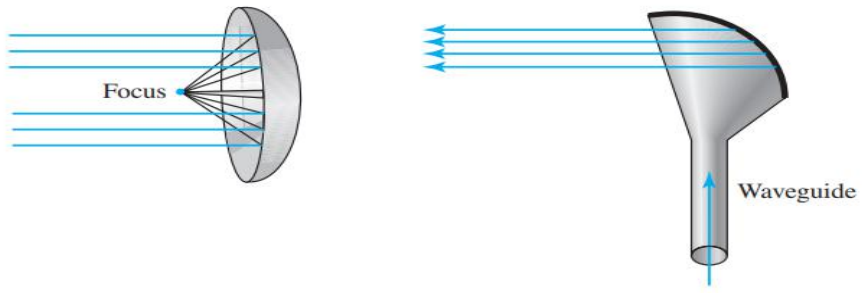
The following describes some characteristics of microwave propagation:

- 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 longdistance 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*



a. Parabolic dish antenna b. Horn antenna

A parabolic dish antenna is based on the geometry of a parabola: Every line parallel to the line of symmetry (line of sight) reflects off the curve at angles such that all the lines intersect in a common point called the focus. The parabolic dish works as a funnel, catching a wide range of waves and directing them to a common point. In this way, more of the signal is recovered than would be possible with a single-point receiver. Outgoing transmissions are broadcast through a horn aimed at the dish. The microwaves hit the dish and are deflected outward in a reversal of the receipt path. A horn antenna looks like a gigantic scoop.

Outgoing transmissions are broadcast up a stem (resembling a handle) and deflected outward in a series of narrow parallel beams by the curved head. Received transmissions are collected by the scooped shape of the horn, in a manner similar to the parabolic dish, and are deflected down into the stem.

Infrared

- An infrared transmission is a wireless technology used for communication over short ranges.
- The frequency of the infrared in the range from 300 GHz to 400 THz.
- It is used for short-range communication such as data transfer between two cell phones, TV remote operation, data transfer between a computer and cell phone resides in the same closed area.

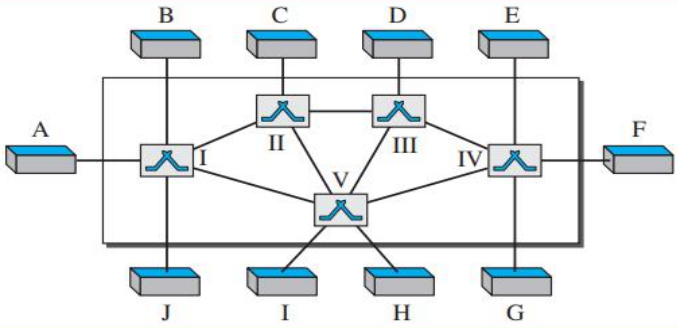
Characteristics Of Infrared:

- It supports high bandwidth, and hence the data rate will be very high.
- Infrared waves cannot penetrate the walls. Therefore, the infrared communication in one room cannot be interrupted by the nearby rooms.
- An infrared communication provides better security with minimum interference.
- Infrared communication is unreliable outside the building because the sun rays will interfere with the infrared waves.
- Infrared signals can be used for short-range communication in a closed area using line-of-sight propagation

Switching

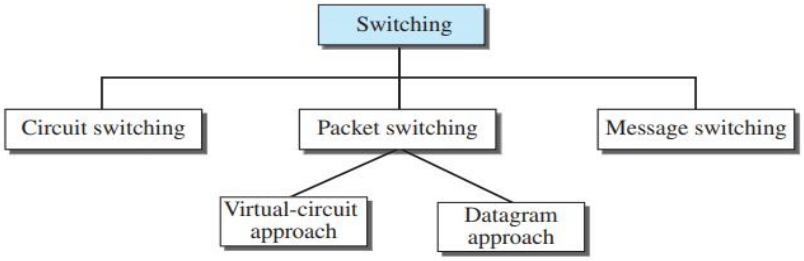
A switched network consists of a series of interlinked nodes, called switches. Switches are devices capable of creating temporary connections between two or more devices linked to the switch. In a switched network, some of these nodes are connected to the end systems (computers or telephones, for example). Others are used only for routing.

Figure 8.1 *Switched network*



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.

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.

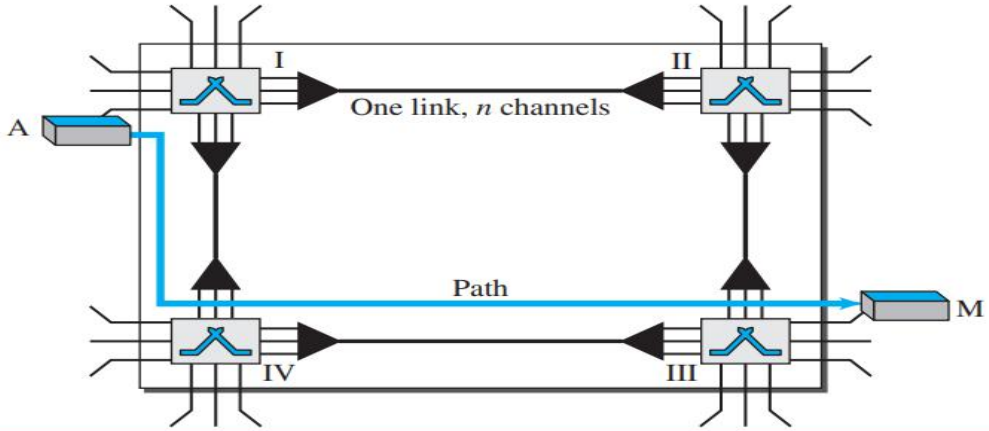
Switching at Application Layer

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. However, 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 *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).

In circuit switching, the resources need to be reserved during the setup phase; the resources remain dedicated for the entire duration of data transfer until the teardown phase.

Three Phases
The actual communication in a circuit-switched network requires three phases: connection setup, data transfer, and connection teardown.

Advantages of Circuit Switching:

- Guaranteed bandwidth: Circuit switching provides a dedicated path for communication, ensuring that bandwidth is guaranteed for the duration of the call.
- Low latency: Circuit switching provides low latency because the path is predetermined, and there is no need to establish a connection for each packet.
- Predictable performance: Circuit switching provides predictable performance because the bandwidth is reserved, and there is no competition for resources.
- Suitable for real-time communication: Circuit switching is suitable for real-time communication, such as voice and video, because it provides low latency and predictable performance.

Disadvantages of Circuit Switching:

- Inefficient use of bandwidth: Circuit switching is inefficient because the bandwidth is reserved for the entire duration of the call, even when no data is being transmitted.
- Limited scalability: Circuit switching is limited in its scalability because the number of circuits that can be established is finite, which can limit the number of simultaneous calls that can be made.
- High cost: Circuit switching is expensive because it requires dedicated resources, such as hardware and bandwidth, for the duration of the call.

PACKET SWITCHING

- The packet switching is a switching technique in which the message is sent in one go, but it is divided into smaller pieces, and they are sent individually.
- The message splits into smaller pieces known as packets and packets are given a unique number to identify their order at the receiving end.
- Every packet contains some information in its headers such as source address, destination address and sequence number.
- Packets will travel across the network, taking the shortest path as possible.
- All the packets are reassembled at the receiving end in correct order.
- If any packet is missing or corrupted, then the message will be sent to resend the message.
- If the correct order of the packets is reached, then the acknowledgment message will be sent.

Approaches Of Packet Switching:

There are two approaches to Packet Switching:

1. Datagram Packet switching:
   - It is a packet switching technology in which packet is known as a datagram, is considered as an independent entity. Each packet contains the information about the destination and switch uses this information to forward the packet to the correct destination.
   - The packets are reassembled at the receiving end in correct order.
   - In Datagram Packet Switching technique, the path is not fixed.
   - Intermediate nodes take the routing decisions to forward the packets.
   - Datagram Packet Switching is also known as connectionless switching.
   - A switch in a datagram network uses a routing table that is based on the destination address.
   - The destination address in the header of a packet in a datagram network remains the same during the entire journey of the packet.
2. Virtual Circuit Switching
   - Virtual Circuit Switching is also known as connection-oriented switching.
   - In the case of Virtual circuit switching, a preplanned route is established before the messages are sent.
   - Call request and call accept packets are used to establish the connection between sender and receiver.
   - In this case, the path is fixed for the duration of a logical connection.
   - In virtual-circuit switching, all packets belonging to the same source and destination travel the same path, but the packets may arrive at the destination with different delays if resource allocation is on demand.

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. However, a global address in virtual-circuit networks is used only to create a virtual-circuit identifier.

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.

Unit II
The 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 LANs: Connecting Devices.

Data Link Layer is second layer of OSI Layered Model. This layer is one of the most complicated layers and has complex functionalities and liabilities. Data link layer hides the details of underlying hardware and represents itself to upper layer as the medium to communicate.

Data link layer works between two hosts which are directly connected in some sense. This direct connection could be point to point or broadcast. Systems on broadcast network are said to be on same link. The work of data link layer tends to get more complex when it is dealing with multiple hosts on single collision domain.

Data link layer is responsible for converting data stream to signals bit by bit and to send that over the underlying hardware. At the receiving end, Data link layer picks up data from hardware which are in the form of electrical signals, assembles them in a recognizable frame format, and hands over to upper layer.

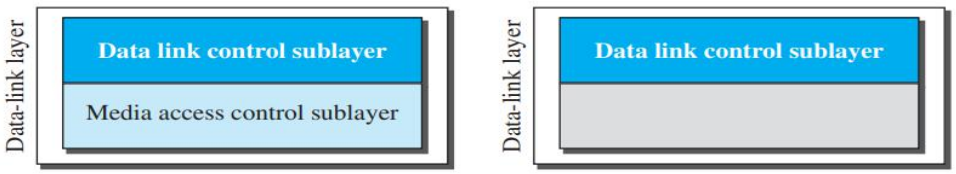
Data link layer has two sub-layers:

- **Data or Logical Link Control:** It deals with protocols, flow-control, and error control
- **Media Access Control:** It deals with actual control of media

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.

In a point-to-point link, the link is dedicated to the two devices; 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; when the same two friends use their cellular phones, they are using a broadcast link (the air is shared among many cell phone users).

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

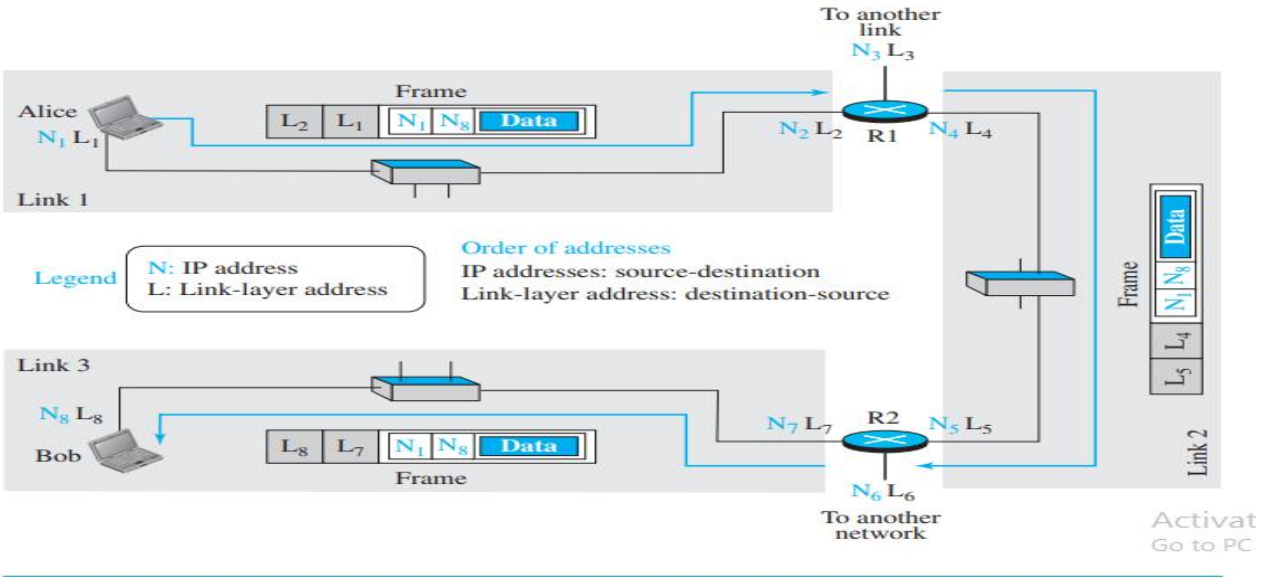


a. Data-link layer of a broadcast link b. Data-link layer of a point-to-point link

LINK-LAYER ADDRESSING
A link-layer address is sometimes called a link address, sometimes 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*



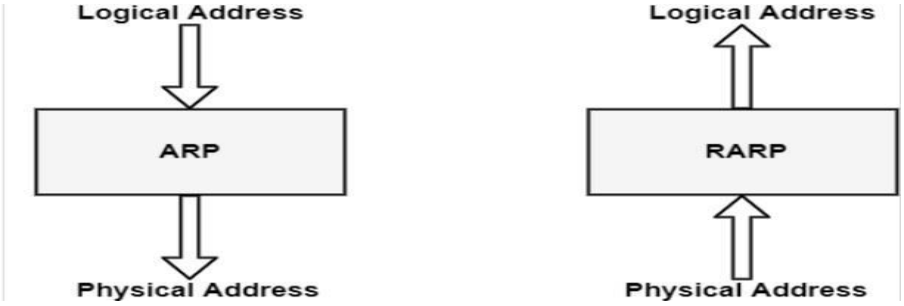
We have three links and two routers. We also have shown only two hosts:
Alice (source) and Bob (destination). For each host, we have shown two addresses, the IP addresses (N) and the link-layer addresses (L). Note that a router has as many pairs of addresses as the number of links the router is connected to. We have shown 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. Note that the IP addresses and the link-layer addresses are not in the same order. For IP addresses, the source address comes before the destination address; for link-layer addresses, the destination address comes before the source.

Three Types of addresses
1. 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. A3:34:45:11:92:F1
2. Multicast Address
Some link-layer protocols define multicast addresses. Multicasting means one-to-many communication. However, the jurisdiction is local (inside the link). A2:34:45:11:92:F1
3. 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. FF:FF:FF:FF:FF:FF

Address Resolution Protocol (ARP)

Address Resolution Protocol (ARP) is a communication protocol used to find the MAC (Media Access Control) address of a device from its IP address. This protocol is used when a device wants to communicate with another device on a Local Area Network or Ethernet.

Address Resolution Protocol (ARP) is a network-specific standard protocol. The Address Resolution Protocol is important for changing the higher-level protocol address (IP addresses) to physical network addresses. It is described in RFC 826.



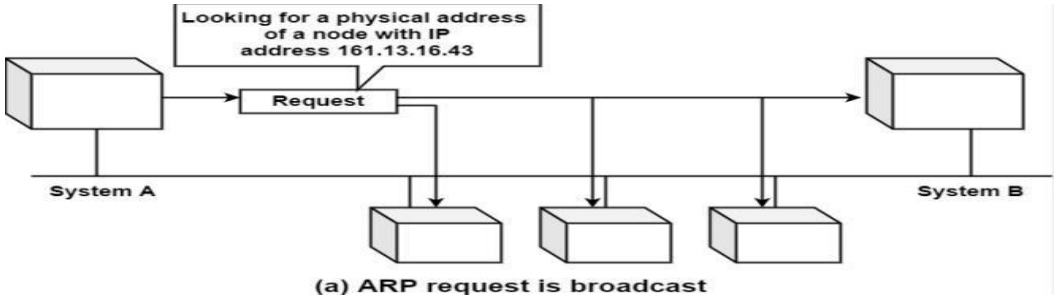
ARP relates an IP address with the physical address. On a typical physical network such as LAN, each device on a link is identified by a physical address, usually printed on the network interface card (NIC). A physical address can be changed easily when NIC on a particular machine fails.

The IP Address cannot be changed. ARP can find the physical address of the node when its internet address is known. ARP provides a dynamic mapping from an IP address to the corresponding hardware address.

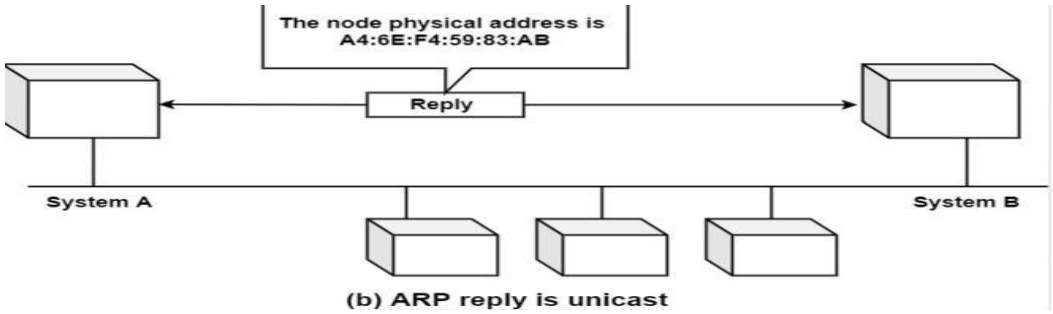
When one host wants to communicate with another host on the network, it needs to resolve the IP address of each host to the host's hardware address.

This process is as follows−

- When a host tries to interact with another host, an ARP request is initiated. If the IP address is for the local network, the source host checks its ARP cache to find out the hardware address of the destination computer.
- If the correspondence hardware address is not found, ARP broadcasts the request to all the local hosts.
- All hosts receive the broadcast and check their own IP address. If no match is discovered, the request is ignored.
- The destination host that finds the matching IP address sends an ARP reply to the source host along with its hardware address, thus establishing the communication. The ARP cache is then updated with the hardware address of the destination host.



(a) ARP request is broadcast



(b) ARP reply is unicast

ARP Packet Generation

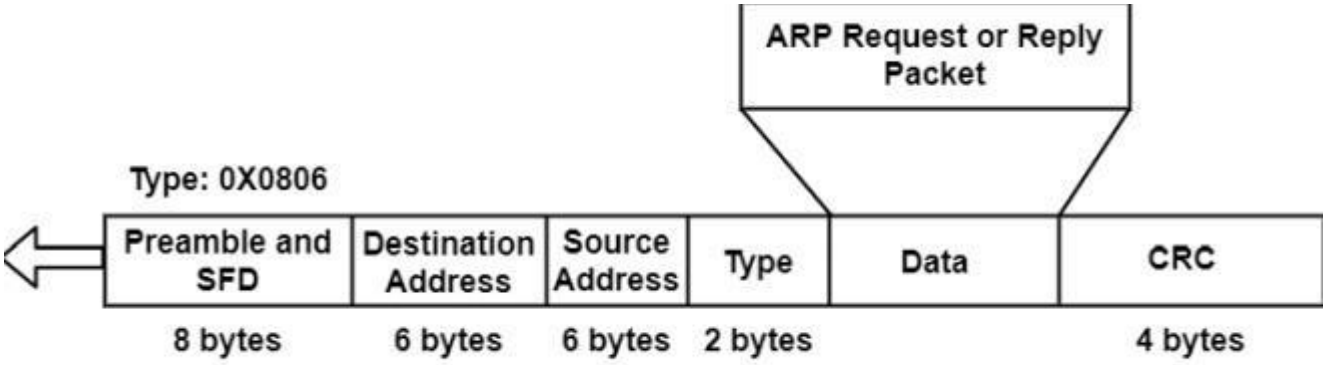
If an application needs to send information to a specific IP destination address, the IP routing structure first determines the IP address of the next-hop of the packet (it should be the destination host itself or a router) and the hardware tool on which it should be transmitted.

If it is an IEEE 802.3/4/5 network, the ARP structure should be considered to design the <protocol type target protocol address> to a physical address.

The ARP module attempts to find the address in this ARP cache. If it is to find the connecting pair, it provides the equivalent 48-bit physical location back to the caller (the device driver), which then shares the packet.

If it does not discover the pair in its table, it removes the packet (the assumption is that a higher-level protocol will resend) and creates a network broadcast of an ARP request.

- **Hardware address space:** It specifies the type of hardware such as Ethernet or Packet Radio net.
- **Protocol address space:** It specifies the type of protocol, same as the Ether type field in the IEEE 802 header (IP or ARP).
- **Hardware Address Length:** It determines the length (in bytes) of the hardware addresses in this packet. For IEEE 802.3 and IEEE 802.5, this is 6.
- **Protocol Address Length:** It specifies the length (in bytes) of the protocol addresses in this packet. For IP, this is 4 byte.
- **Operation Code:** It specifies whether this is an ARP request (1) or reply (2).
- **Source/target hardware address:** It contains the physical network hardware addresses. For IEEE 802.3, these are 48-bit addresses.
- For the ARP request packet, the target hardware address is the only undefined field in the packet.



Encapsulation of ARP Packet

Error detection and Correction: Cyclic codes, Checksum, Forward error correction

Networks must be able to transfer data from one device to another with acceptable accuracy. For most applications, a system must guarantee that the data received are identical to the data transmitted. Any time data are transmitted from one node to the next, they can become corrupted in passage. Many factors can alter one or more bits of a message. Some applications require a mechanism for detecting and correcting errors. Some applications can tolerate a small level of error. For example, random errors in audio or video transmissions may be tolerable, but when we transfer text, we expect a very high level of accuracy. At the data-link layer, if a frame is corrupted between the two nodes, it needs to be corrected before it continues its journey to other nodes. However, most link-layer protocols simply discard the frame and let the upper-layer protocols handle the retransmission of the frame. Some multimedia applications, however, try to correct the corrupted frame.

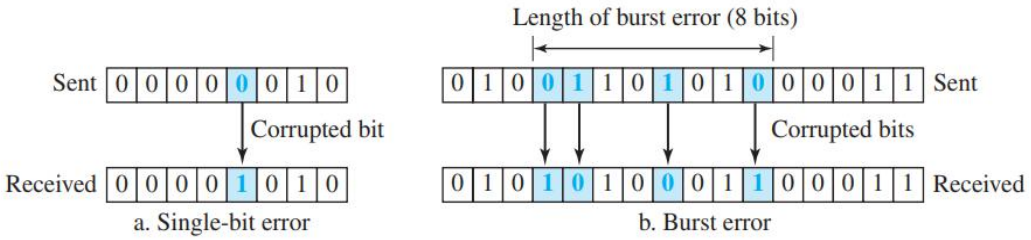
Types of Errors

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 shows the effect of a single-bit and a burst error on a data unit.

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.

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. We are not even interested in the number of corrupted bits. 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. 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. The trick in error detection is the existence of these invalid codes. 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?

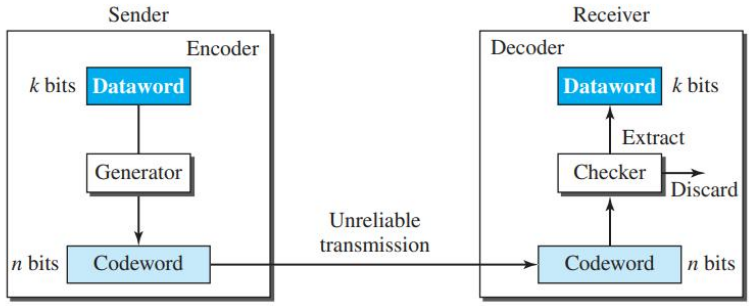
If the following two conditions are met, the receiver can detect a change in the original codeword.

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 (discussed later). 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*



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.

An error-detecting code can detect only the types of errors for which it is designed; other types of errors may remain undetected.

Hamming Distance

One of the central concepts in coding for error control is the idea of the 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 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 d(00000, 01101) = 3.

In other words, if the Hamming distance between the sent codeword and the received codeword is not zero, then 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.

Note that the Hamming distance is a value greater than or equal to zero.

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

1. The Hamming distance d(000, 011) is 2 because (000 ⊕ 011) is 011 (two 1s).
2. The Hamming distance d(10101, 11110) is 3 because (10101 ⊕ 11110) is 01011 (three 1s).

Minimum Hamming Distance for Error Detection

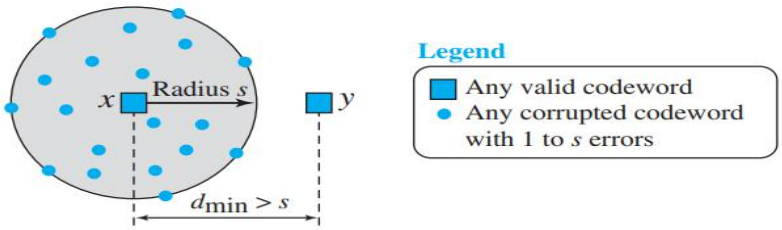
In a set of codewords, the minimum Hamming distance is the smallest Hamming distance between all possible pairs of codewords. Now let us find the minimum Hamming distance in a code if we want to be

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 dmin must be an integer greater than s or d_{min} = s + 1.

Figure 10.3 *Geometric concept explaining d_{min} in error detection*



A code scheme has a Hamming distance d_{min} = 4. This code guarantees the detection of up to three errors (d = s + 1 or s = 3).

Linear Block Codes

Almost all block codes used today belong to a subset of block codes called linear block codes. The minimum Hamming distance is the number of 1s in the nonzero valid codeword with the smallest number of 1s.

Parity-Check Code

The most familiar error-detecting code is the parity-check code. This code is a linear block code.

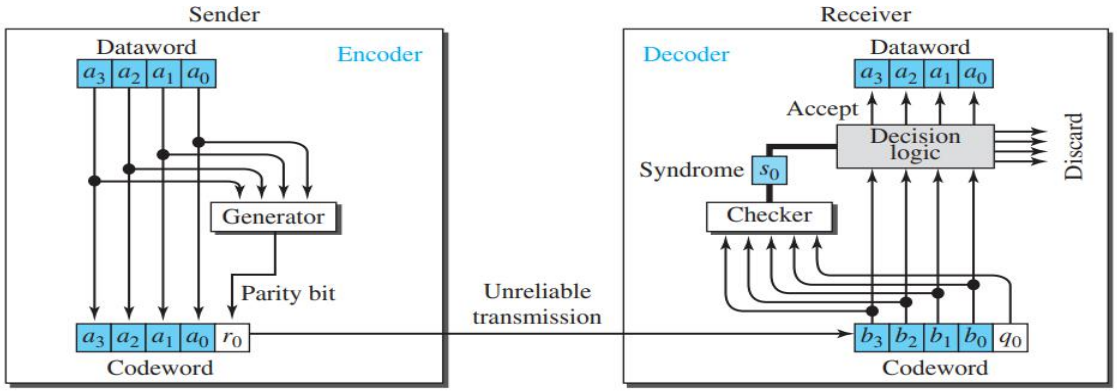
In this code, a k-bit dataword is changed to an n-bit codeword where n = k + 1. The extra bit, called the parity bit, is selected to make the total number of 1s in the codeword even. Although some implementations specify an odd number of 1s. The minimum Hamming distance for this category is d_{min} = 2, which means that the code is a single-bit error-detecting code.

Table 10.2 *Simple parity-check code C(5, 4)*

Dataword	Codeword	Dataword	Codeword
0000	**00000**	1000	**10001**
0001	**00011**	1001	**10010**
0010	**00101**	1010	**10100**
0011	**00110**	1011	**10111**
0100	**01001**	1100	**11000**
0101	**01010**	1101	**11011**
0110	**01100**	1110	**11101**
0111	**01111**	1111	**11110**

Figure 10.4 shows a possible structure of an encoder (at the sender) and a decoder (at the receiver).

Figure 10.4 *Encoder and decoder for simple parity-check code*



The calculation is done in modular arithmetic. The encoder uses a generator that takes a copy of a 4-bit dataword (a_0, a_1, a_2, and a_3) and generates a parity bit r_0. The dataword bits and the parity bit create the 5-bit codeword. The parity bit that is added makes the number of 1s in the codeword even. This is normally done by adding the 4 bits of the dataword (modulo-2); the result is the parity bit.
In other words, $r_0 = a_3 + a_2 + a_1 + a_0$ (modulo-2)
If the number of 1s is even, the result is 0; if the number of 1s is odd, the result is 1.
In both cases, the total number of 1s in the codeword is even.
The sender sends the codeword, which may be corrupted during transmission. The receiver receives a 5-bit word. The checker at the receiver does the same thing as the generator in the sender with one exception: The addition is done over all 5 bits. The result which is called the syndrome, is just 1 bit. The syndrome is 0 when the number of 1s in the received codeword is even; otherwise, it is 1.
$s0 = b_3 + b_2 + b_1 + b_0 + q_0$ (modulo-2)

The syndrome is passed to the decision logic analyzer. If the syndrome is 0, there is no detectable error in the received codeword; the data portion of the received codeword is accepted as the dataword; if the syndrome is 1, the data portion of the received codeword is discarded. The dataword is not created.

A parity-check code can detect an odd number of errors.

Example 10.7

Let us look at some transmission scenarios. Assume the sender sends the dataword 1011. The codeword created from this dataword is 10111, which is sent to the receiver. We examine five cases:

1. No error occurs; the received codeword is 10111. The syndrome is 0. The dataword 1011 is created.
2. One single-bit error changes a_1. The received codeword is 10011. The syndrome is 1. No dataword is created.
3. One single-bit error changes r_0. The received codeword is 10110. The syndrome is 1. No dataword is created. Note that although none of the dataword bits are corrupted, no dataword is created because the code is not sophisticated enough to show the position of the corrupted bit.
4. An error changes r_0 and a second error changes a_3. The received codeword is 00110. The syndrome is 0. The dataword 0011 is created at the receiver. Note that here the dataword is wrongly created due to the syndrome value. The simple parity-check decoder cannot detect an even number of errors. The errors cancel each other out and give the syndrome a value of 0.
5. Three bits—a_3, a_2, and a_1—are changed by errors. The received codeword is 01011. The syndrome is 1. The dataword is not created. This shows that the simple parity check, guaranteed to detect one single error, can also find any odd number of errors.

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:

b1 = a0 b2 = a1 b3 = a2 b4 = a3 b5 = a4 b6 = a5 b0 = a6

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

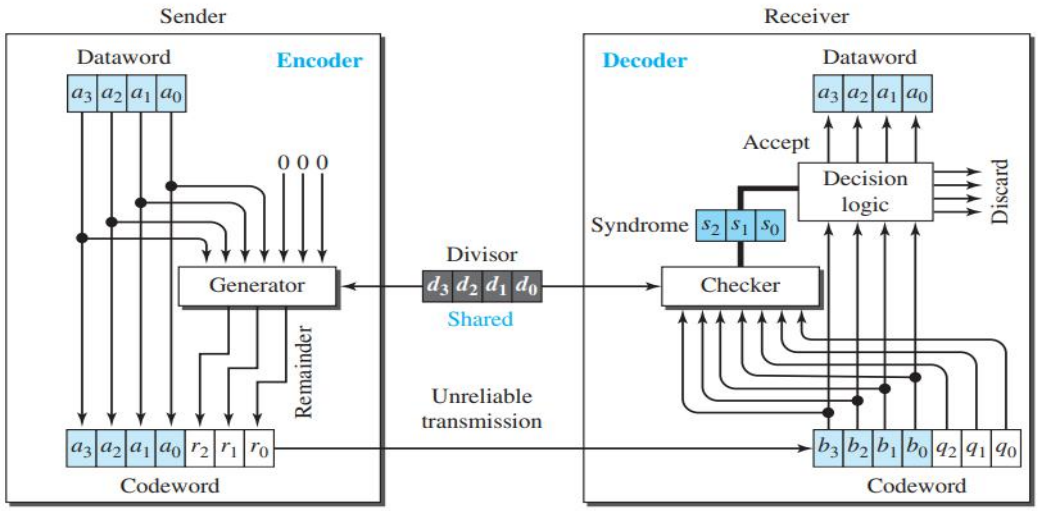
Table 10.3 shows an example of a CRC code. We can see both the linear and cyclic properties of this code.

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*



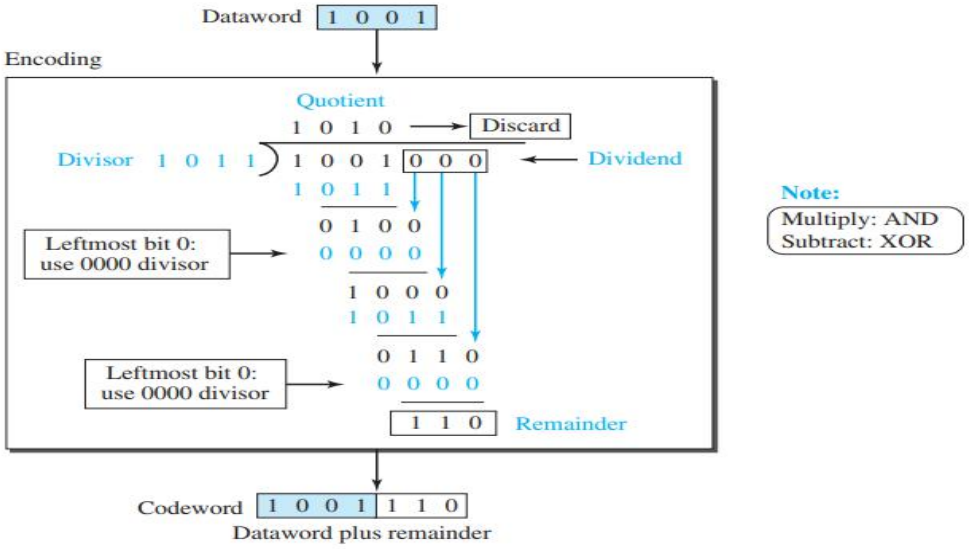
In the encoder, the dataword has k bits (4 here); the codeword has n bits (7 here). The size of the dataword is augmented by adding n − k (3 here) 0s to the right-hand side of the word. The n-bit result is fed into the generator. The generator uses a divisor of size n − k + 1 (4 here), predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder (r2r1r0) 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

The encoder takes a dataword and augments it with n − k number of 0s. It then divides the augmented dataword by the divisor.

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.

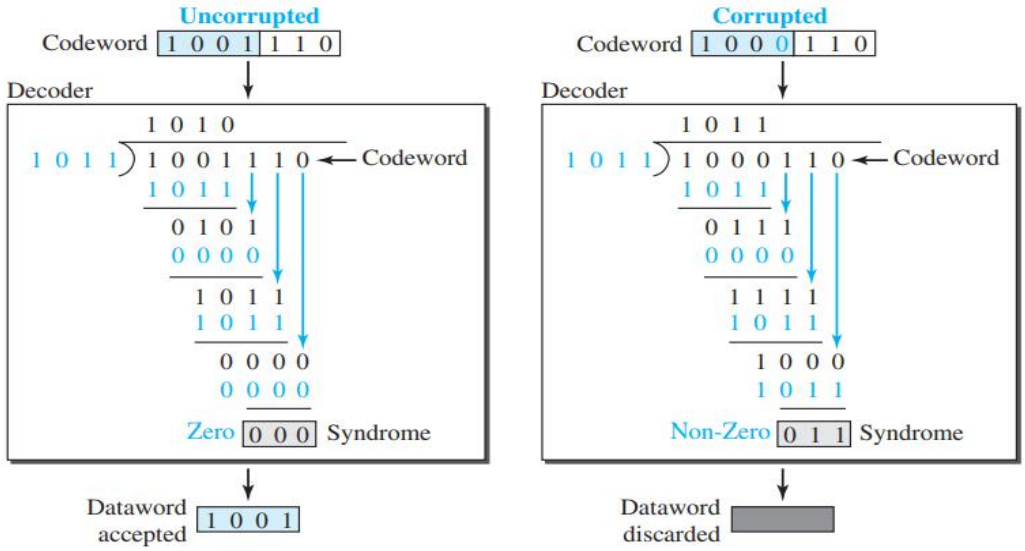
When there are no bits left to pull down, we have a result. The 3-bit remainder forms the check bits (r2, r1, and r0). They are appended to the dataword to create the codeword.

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.

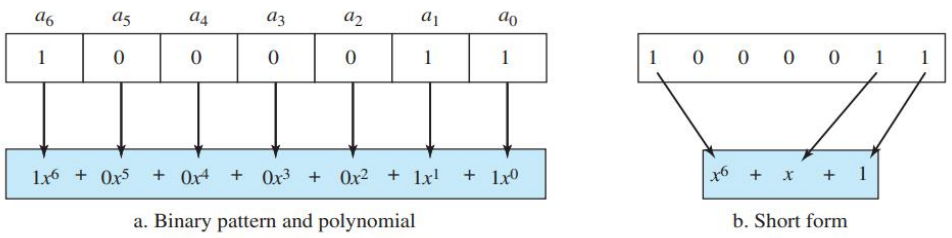
Polynomials

A better way 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 shows one immediate benefit; a 7-bit pattern can be replaced by three terms. The benefit is even more conspicuous when we have a polynomial such as $x^{23} + x^3 + 1$. Here the bit pattern is 24 bits in length (three 1s and twenty-one 0s) while the polynomial is just three terms.

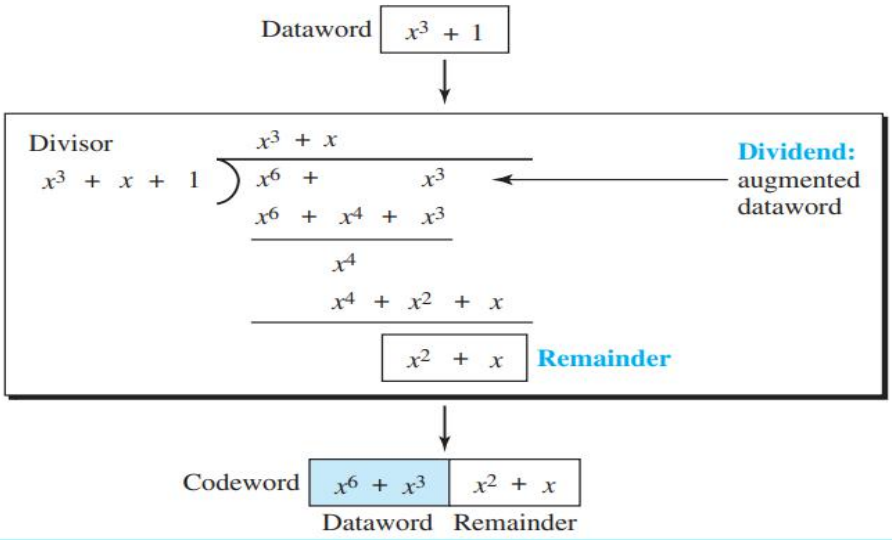
Figure 10.8 *A polynomial to represent a binary word*



Cyclic Code Encoder Using Polynomials

The dataword 1001 is represented as $x^3 + 1$. The divisor 1011 is represented as $x^3 + x + 1$.
To find the augmented dataword, we have left-shifted the dataword 3 bits (multiplying by x3). The result is $x^6 + x^3$.

Figure 10.9 *CRC division using polynomials*



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.

First find the relationship among the sent codeword, error, received codeword, and the generator.

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.

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

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.

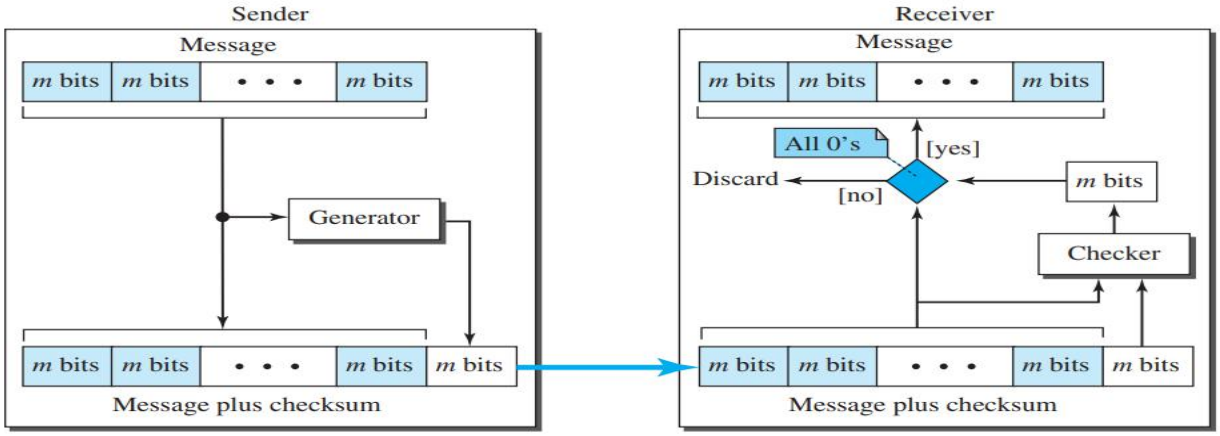
CHECKSUM

Checksum is an error-detecting technique that can be applied to a message of any length.

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.

Figure 10.15 *Checksum*



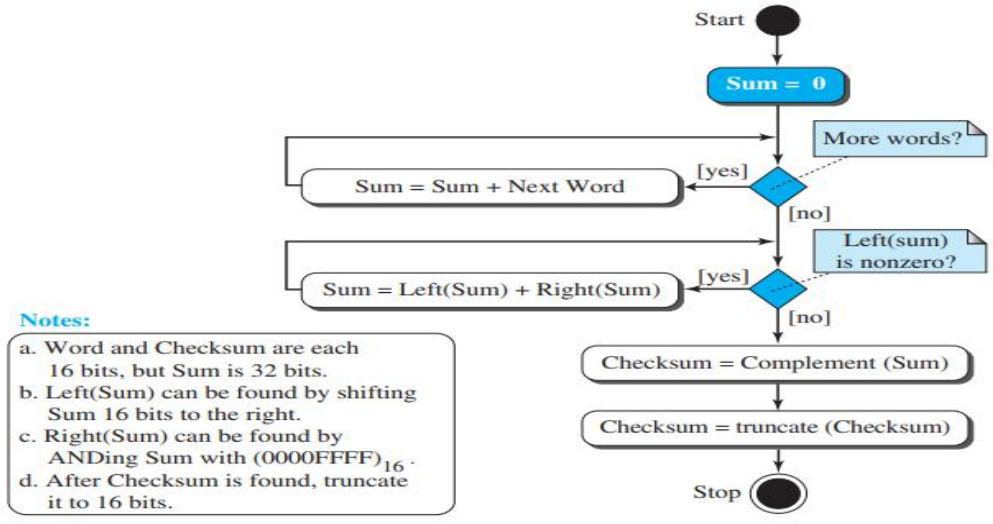
Internet Checksum

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.

Figure 10.17 *Algorithm to calculate a traditional checksum*



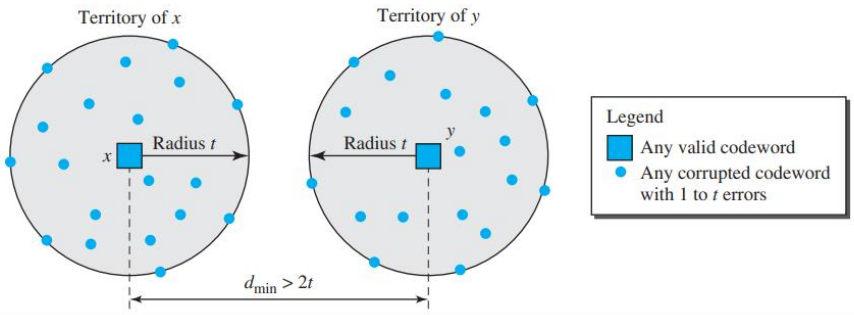
FORWARD ERROR CORRECTION

1. Using Hamming Distance We earlier discussed the Hamming distance for error detection. We said that to detect s errors, the minimum Hamming distance should be $d_{min} = s + 1$. For error detection, we definitely need more distance. It can be shown that to detect t errors, we need to have $d_{min} = 2t + 1$.

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. To give an 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.

Figure 10.20 shows the geometrical representation of this concept.

Figure 10.20 *Hamming distance for error correction*



2. Using XOR

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

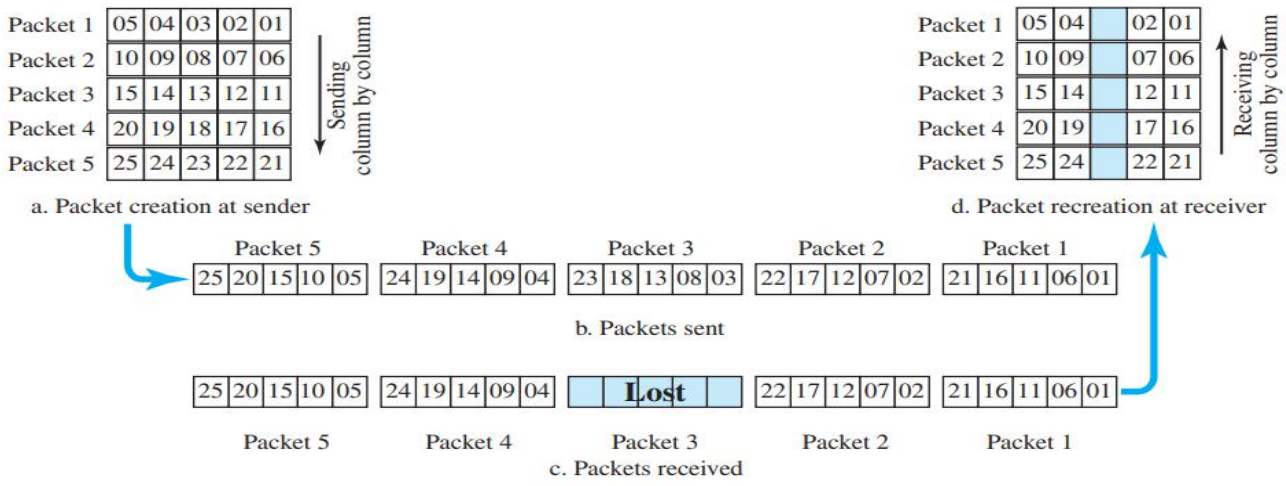
In other words, if we apply the exclusive OR operation on N data items (P1 to PN), we can recreate any of the data items by exclusive-OR ing all of the items, replacing the one to be created by the result of the previous operation (R). This means that we can divide a packet into N chunks, create the 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.

3. 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*



4. 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 × t-bit errors

5. 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.

Data link control: DLC Services, Data link layer protocols, HDLC, Point to Point Protocol

The data link control (DLC) deals with procedures for communication between two adjacent nodes—node-to-node communication—no matter whether the link is dedicated or broadcast. Data link control functions include framing and flow and error control.

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.

Although the whole message could be packed in one frame, that is not normally done. 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

Frames can be of fixed or variable size.

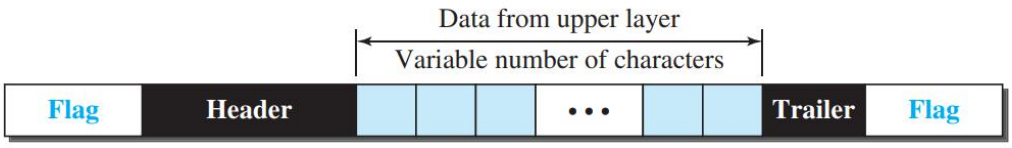
In fixed-size framing, there is no need for defining the boundaries of the frames; the size itself can be used as a delimiter. An example of this type of framing is the ATM WAN, which uses frames of fixed size called cells.

The variable-size framing is prevalent in local-area networks. In variable-size framing, we need a way to define the end of one frame and the beginning of the next. Historically, two approaches were used for this purpose: 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, and the trailer, which carries error detection redundant bits, are 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.

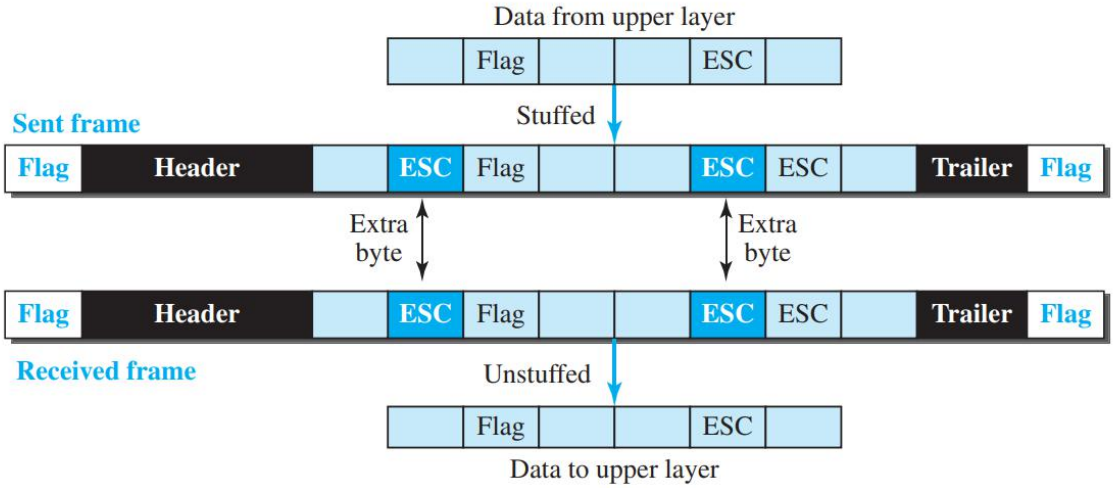
Figure 11.1 *A frame in a character-oriented protocol*



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. Now, however, we send 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 is the process of adding one extra byte whenever there is a flag or escape character in the text.

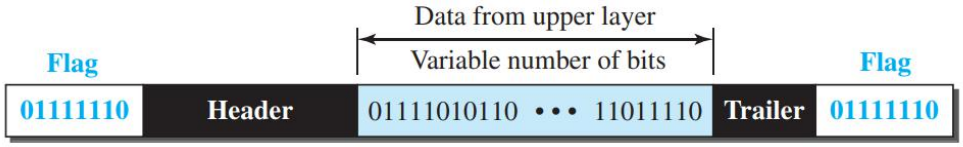
Byte stuffing by the escape character allows the presence of the flag in the data section of the frame, but it creates another problem. What happens if the text contains one or more escape characters followed by a byte with the same pattern as the flag? The receiver removes the escape character, but 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. However, 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.

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.

Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data, so that

the receiver does not mistake the pattern 0111110 for a flag.

Figure 11.4 *Bit stuffing and unstuffing*

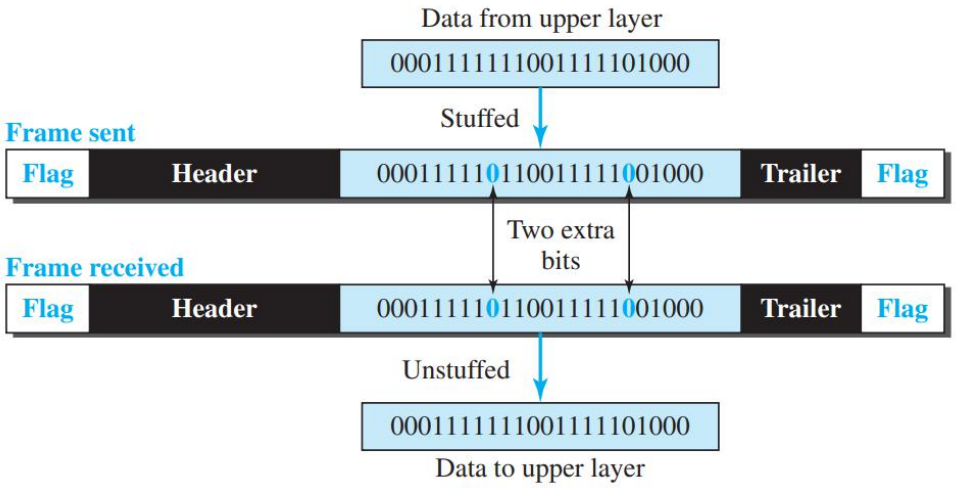
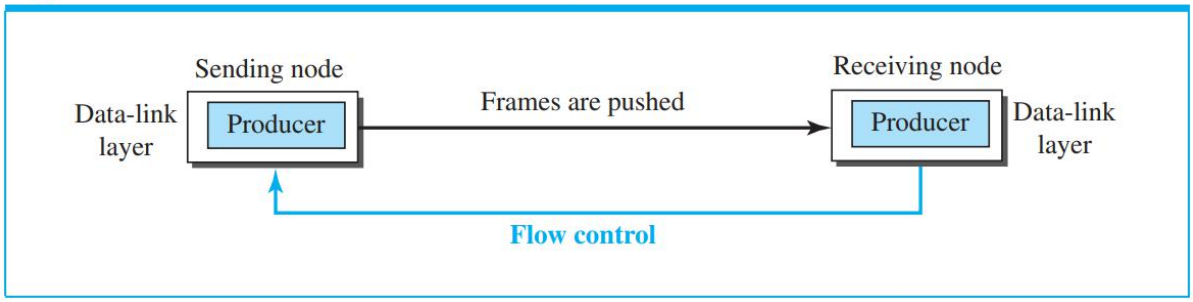


Figure 11.4 shows bit stuffing at the sender and bit removal at the receiver. Note that even if we have a 0 after five 1s, we still stuff a 0. The 0 will be removed by the receiver.
This means that if the flaglike pattern 01111110 appears in the data, it will change to 011111010 (stuffed) and is not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

2. Flow and Error Control
One of the responsibilities of the data-link control sublayer is flow and error control at the data-link layer.

Flow Control
Whenever an entity produces items and another entity consumes them, there should be a balance between production and consumption rates. If the items are produced faster than they can be consumed, the consumer can be overwhelmed and may need to discard some items. If the items are produced more slowly than they can be consumed, the consumer must wait, and the system becomes less efficient. Flow control is related to the first issue. We need to prevent losing the data items at the consumer site. 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. Although we can have a complex relationship with more than one producer and consumer, we ignore the relationships between networks and data-link layers and concentrate on the relationship between two data-link layers.

Figure 11.5 *Flow control at the data-link layer*



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
Although 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

Since the underlying technology at the physical layer is not fully reliable, we need to implement error control at the data-link layer to prevent the receiving node from delivering corrupted packets to its network layer.

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. We show this situation when we discuss some simple protocols in the next section. A frame that carries an acknowledgment is normally called an ACK to distinguish it from the data frame.

3. Connectionless and Connection-Oriented

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

a. 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.

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

DATA-LINK LAYER PROTOCOLS

Traditionally 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 Selective-Repeat.

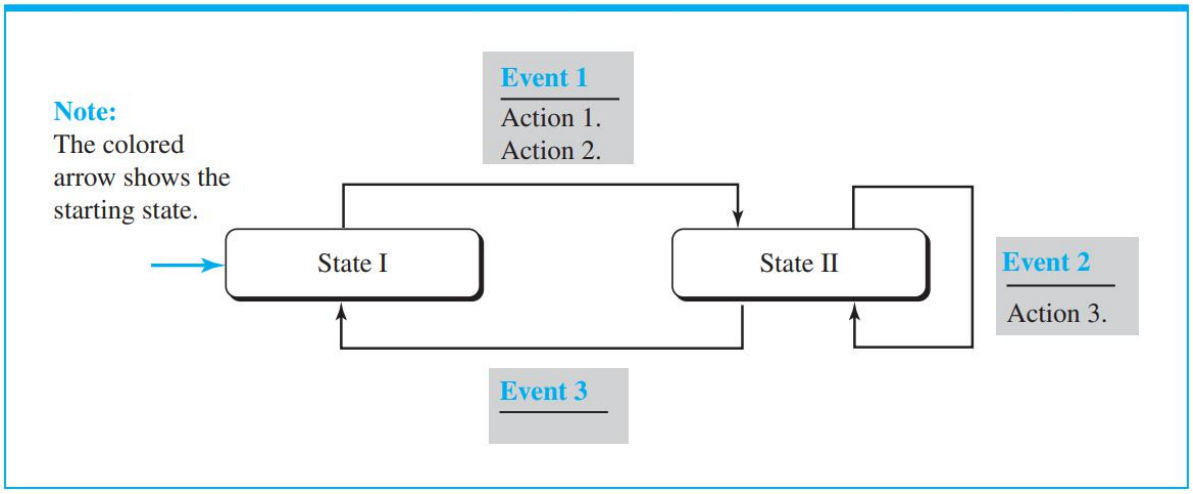
The behavior of a data-link-layer protocol can be better shown as a finite state machine (FSM). An FSM is thought of as a machine with a finite number of states. The machine is always in one of the states until an event occurs. Each event is associated with two reactions: defining the list (possibly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

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.

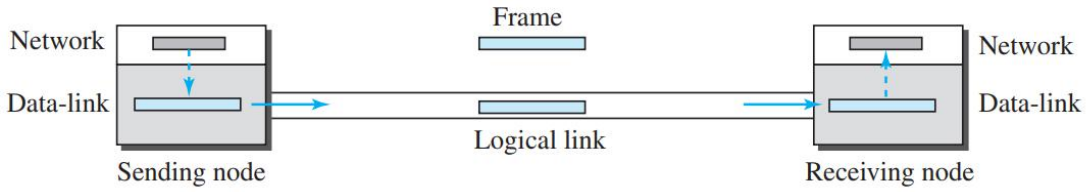
Figure 11.6 *Connectionless and connection-oriented service represented as FSMs*



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. In other words, the receiver can never be overwhelmed with incoming frames.

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 *FSMs for the simple protocol*

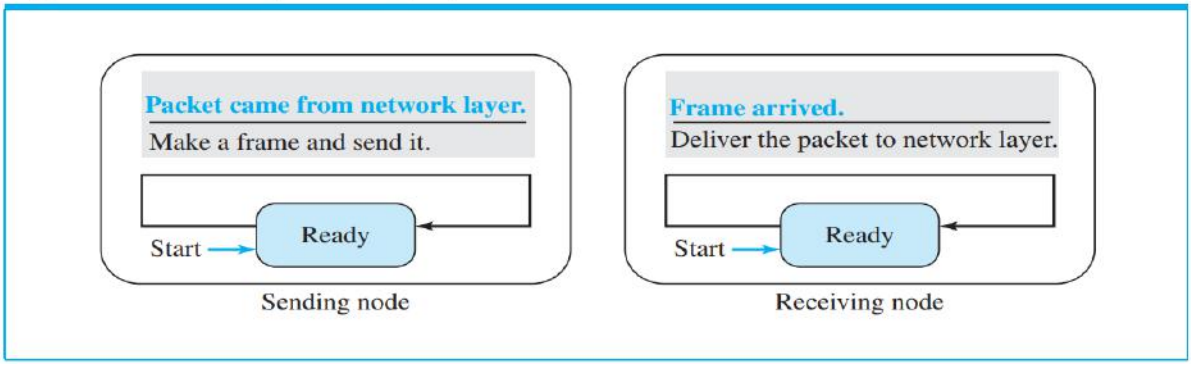
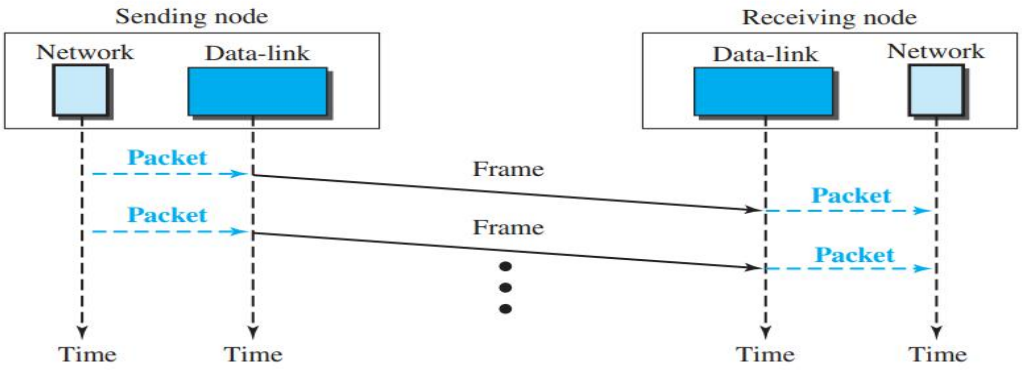


Figure 11.9 shows an example of communication using this protocol. It is very simple. The sender sends frames one after another without even thinking about the receiver.

Figure 11.9 *Flow diagram for Example 11.2*

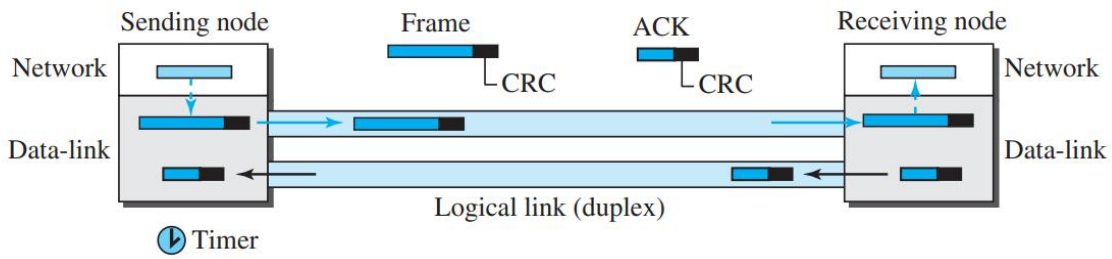


2. Stop-and-Wait Protocol

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.

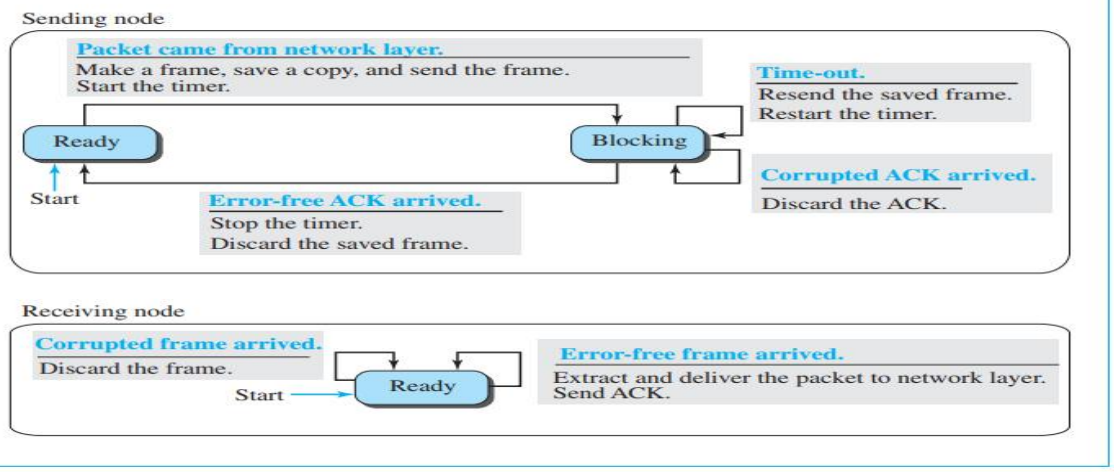
Figure 11.10 shows the outline for the Stop-and-Wait protocol. Note that only one frame and one acknowledgment can be in the channels at any time.

Figure 11.10 *Stop-and-Wait protocol*



FSMs

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



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

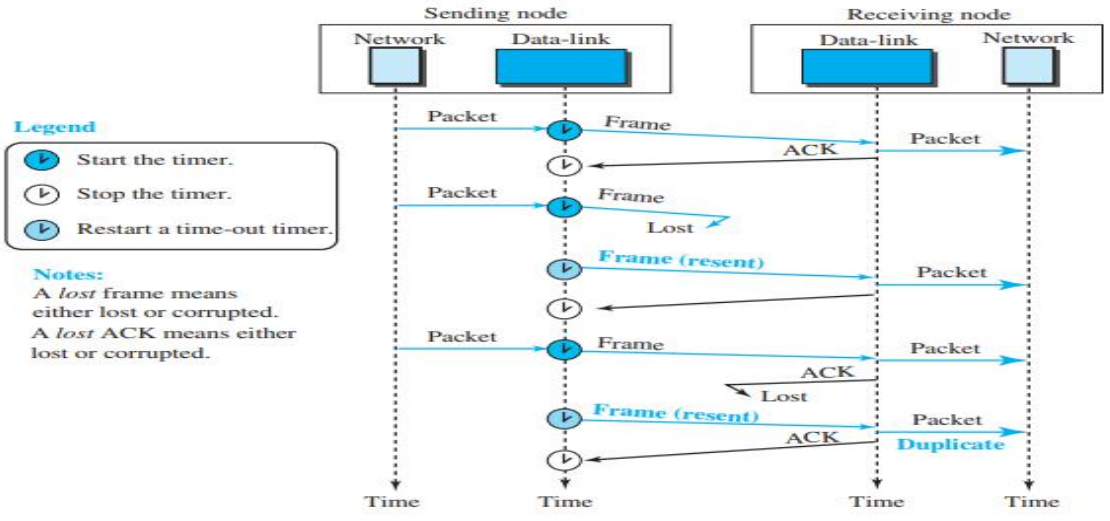
The receiver is always in the ready state. Two events may occur:

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.

Example:

The first frame is sent and acknowledged. The second frame is sent, but lost. After time-out, it is resent. The third frame is sent and acknowledged, but the acknowledgment is lost. The frame is resent. However, there is a problem with this scheme. The network layer at the receiver site receives two copies of the third packet, which is not right.

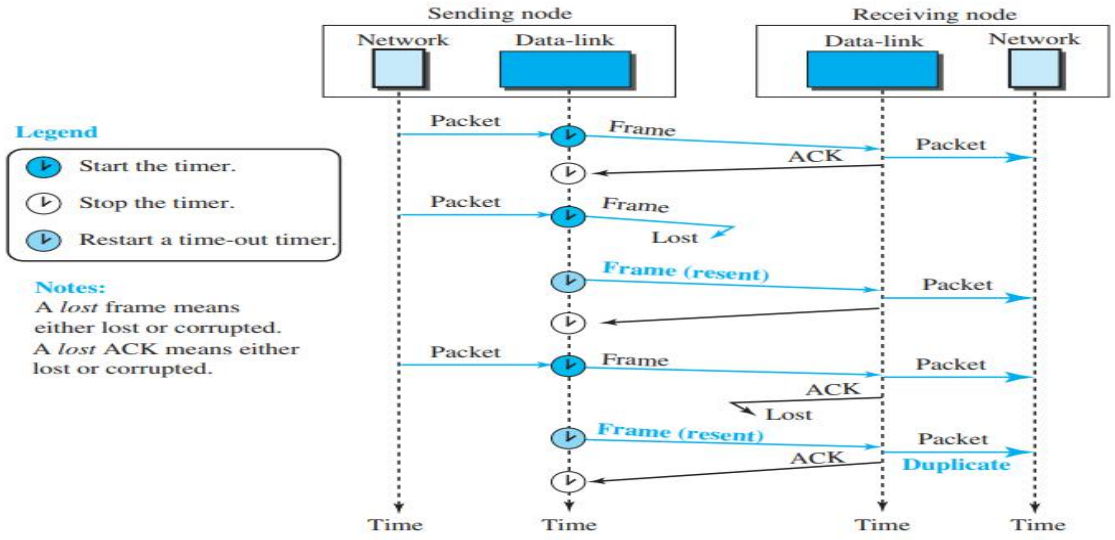
Figure 11.12 *Flow diagram for Example 11.3*



Sequence and Acknowledgment Numbers

We saw a problem in Example that needs to be addressed and corrected.

Duplicate packets, as much as corrupted packets, need to be avoided.
As an example, assume we are ordering some item online. If each packet defines the specification of an item to be ordered, duplicate packets mean ordering an item more than once. To correct the problem in Example, we need to add sequence numbers to the data frames and acknowledgment numbers to the ACK frames. However, numbering in this case is very simple.
Sequence numbers are 0, 1, 0, 1, 0, 1, . . . ; the acknowledgment numbers can also be 1, 0, 1, 0, 1, 0, ... In other words, the sequence numbers start with 0, the acknowledgment numbers start with 1. An acknowledgment number always defines the sequence number of the next frame to receive.



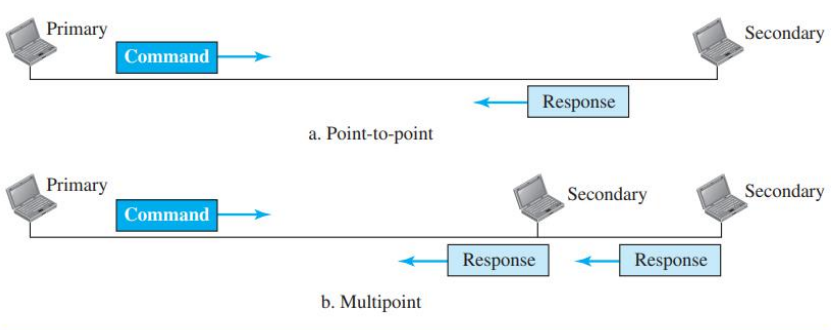
3. Piggybacking
The two protocols we discussed in this section are designed for unidirectional communication, in which data is flowing only in one direction although the acknowledgment may travel in the other direction. Protocols have been designed in the past to allow data to flow in both directions. However, to make the communication more efficient, the data in one direction is piggybacked with the acknowledgment in the other direction. In other words, 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 datalink layer more complicated, it is not a common practice.

HDLC

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

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.

Figure 11.14 *Normal response mode*



In 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 11.15 *Asynchronous balanced mode*

Framing

To provide the flexibility necessary to support all the options possible in the modes and configurations, HDLC defines three types of frames: information frames (I-frames), supervisory frames (S-frames), and unnumbered frames (U-frames). Each type of frame serves as an envelope for the transmission of a different type of message.

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

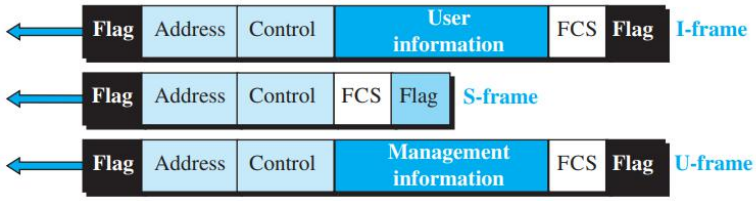
S-frames are used only to transport control information.

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.

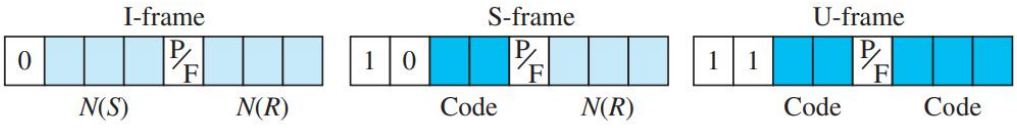
Figure 11.16 *HDLC frames*



- 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.

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 S-frame. 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.

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. PPP is by far the most common.

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

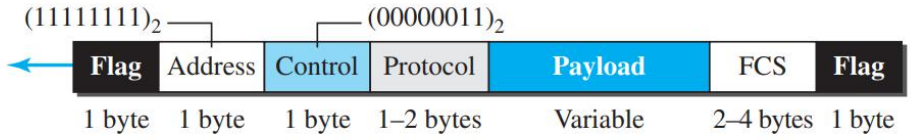
PPP does not provide flow control. A sender can send several frames one after another with no concern about overwhelming the receiver. PPP has a very simple mechanism for error control. A CRC field is used

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.

Framing

PPP uses a character-oriented (or byte-oriented) frame.

Figure 11.20 *PPP frame format*



The description of each field follows:

- 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.

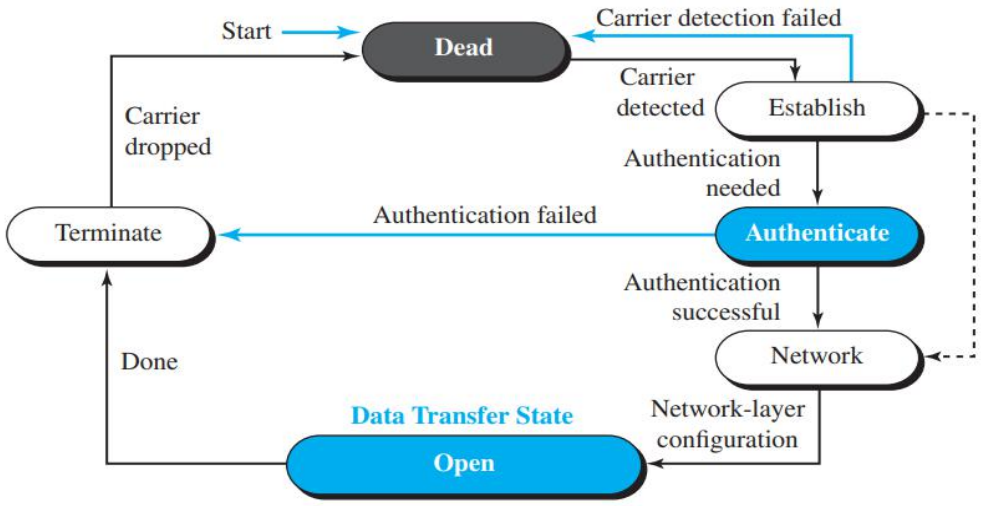
Byte Stuffing

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 every time the flaglike 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.

Transition Phases

A PPP connection goes through phases which can be shown in a transition phase diagram (see Figure 11.21). 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.

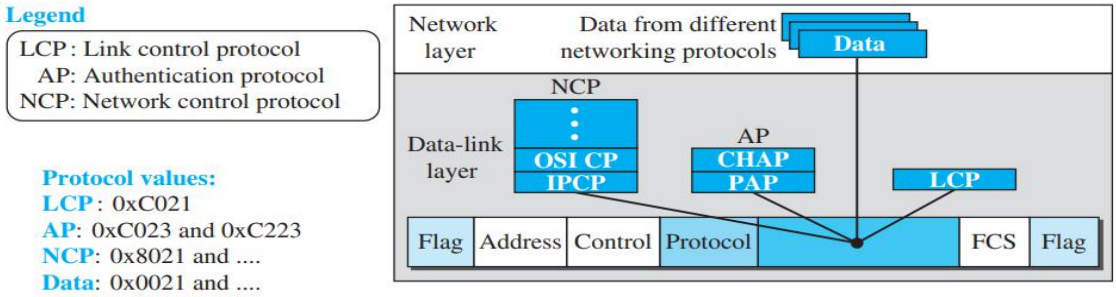
Figure 11.21 *Transition phases*



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. Note that there are one LCP, two APs, and several NCPs.

Figure 11.22 *Multiplexing in PPP*

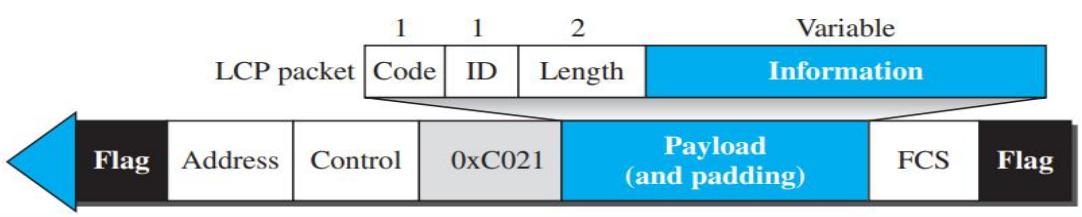


Link Control Protocol

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.

All LCP packets are carried in the payload field of the PPP frame with the protocol field set to C021 in hexadecimal.

Figure 11.23 *LCP packet encapsulated in a frame*



The code field defines the type of LCP packet. There are 11 types of packets.

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.

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 *PAP packets encapsulated in a PPP frame*

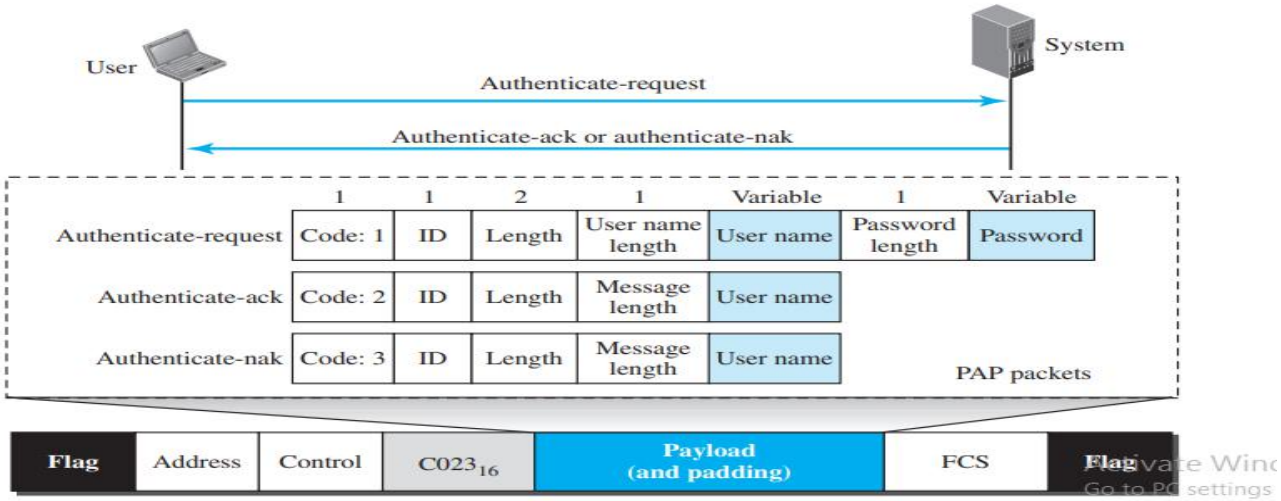


Figure 11.24 shows the three types of packets used by PAP and how they are actually exchanged. When a PPP frame is carrying any PAP packets, the value of the protocol field is 0xC023. The three PAP packets are authenticate-request, authenticate-ack, and authenticate-nak. The first packet is used by the user to send the user name and password. The second is used by the system to allow access. The third is used by the system to deny access.

CHAP

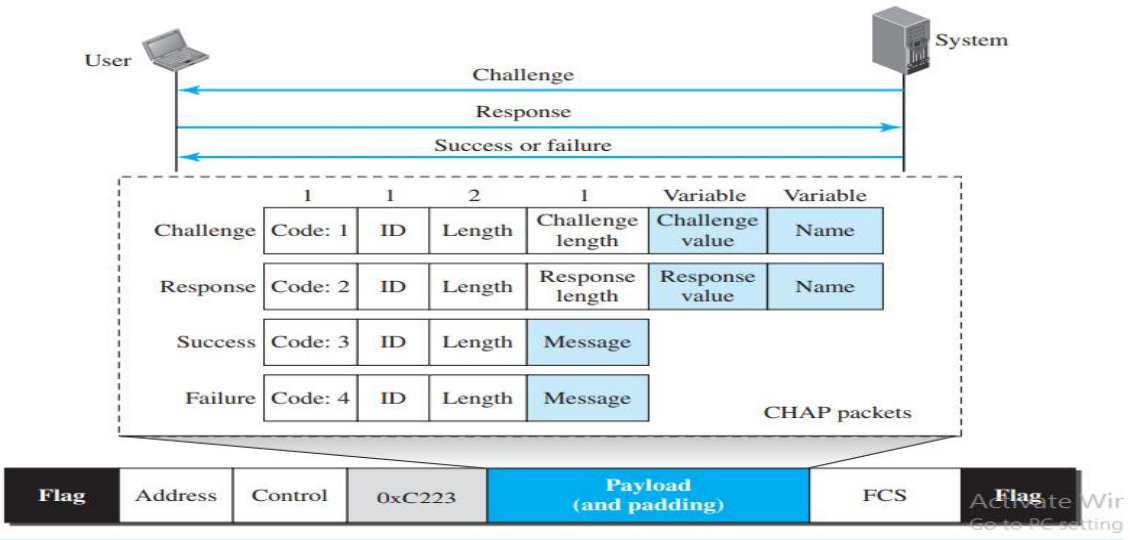
The Challenge Handshake Authentication Protocol (CHAP) is a three-way handshaking authentication protocol that provides greater security than PAP. In this method, the password is kept secret; it is never sent online.

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 *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, response, success, and failure. The first packet is used by the system to send the challenge value. The second is used by the user to return the result of the calculation. The third

is used by the system to allow access to the system. The fourth is used by the system to deny access to the system.

Network Control Protocols

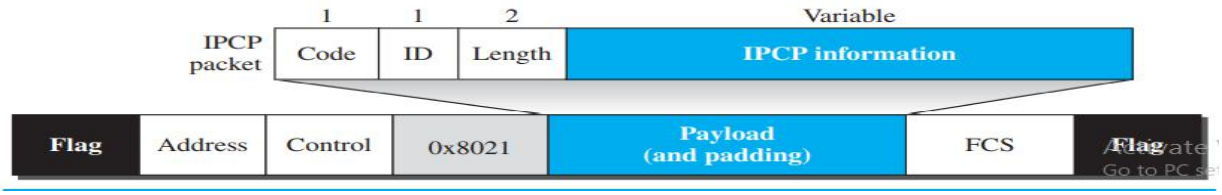
PPP is a multiple-network-layer protocol. It can carry a network-layer data packet from protocols defined by the Internet, OSI, Xerox, DECnet, AppleTalk, Novel, and so on. To do this, PPP has defined a specific Network Control Protocol for each network protocol.

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.

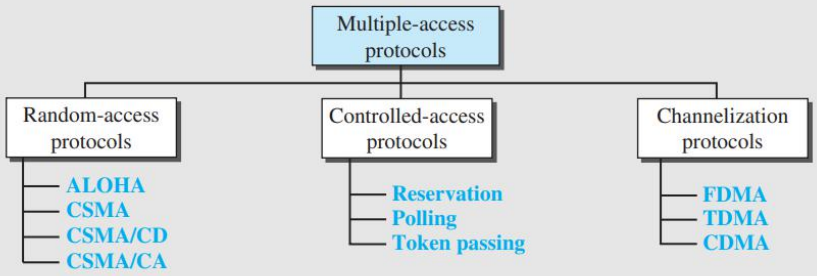
Table 11.3 *Code value for IPCP packets*

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

Media Access Control (MAC)

When nodes or stations are connected and use a common link, called a multipoint or broadcast link, we need a multiple-access protocol to coordinate access to the link. The problem of controlling the access to the medium is similar to the rules of speaking in an assembly. The procedures guarantee that the right to speak is upheld and ensure that two people do not speak at the same time, do not interrupt each other, do not monopolize the discussion, and so on. Many protocols have been devised to handle access to a shared link. All of these protocols belong to a sublayer in the data-link layer called media access control (MAC).

Figure 12.1 *Taxonomy of multiple-access protocols*



RANDOM ACCESS

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). In other words, each station can transmit when it desires on the condition that it follows the predefined procedure, including testing the state of the medium.

Two 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.

That is why 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. However, 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 random-access methods have evolved from a very interesting protocol known as ALOHA, which used a very simple procedure called multiple access (MA). 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). This method later evolved into two parallel methods: carrier sense multiple access with collision detection (CSMA/CD), which tells the station what to do when a collision is detected, and carrier sense multiple access with collision avoidance (CSMA/CA), which tries to avoid the collision.

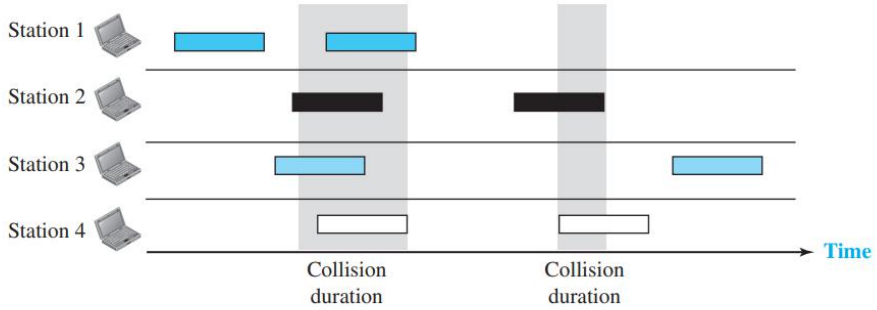
ALOHA

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.

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). However, 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*



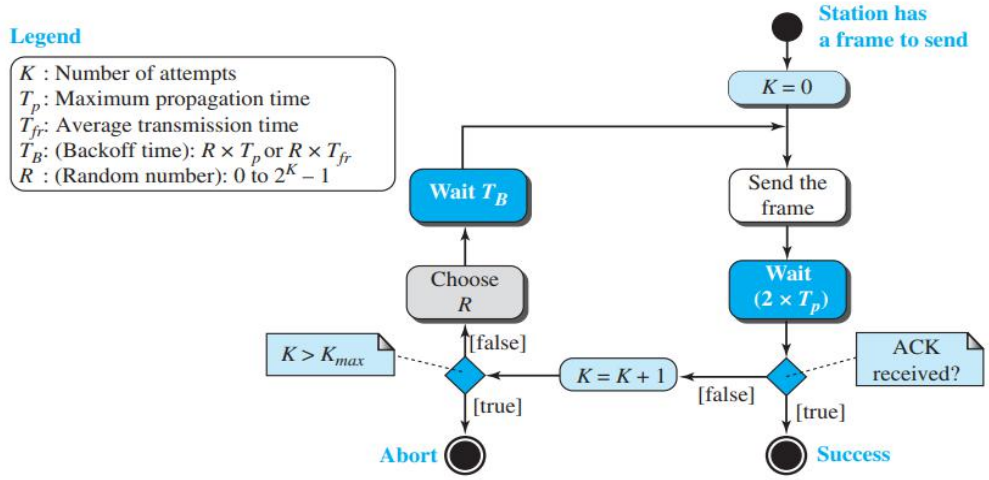
There are four stations (unrealistic assumption) that contend with one another for access to the shared channel. The figure shows that each station sends two frames; there are a total of eight frames on the shared medium. Some of these frames collide because multiple frames are in contention for the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

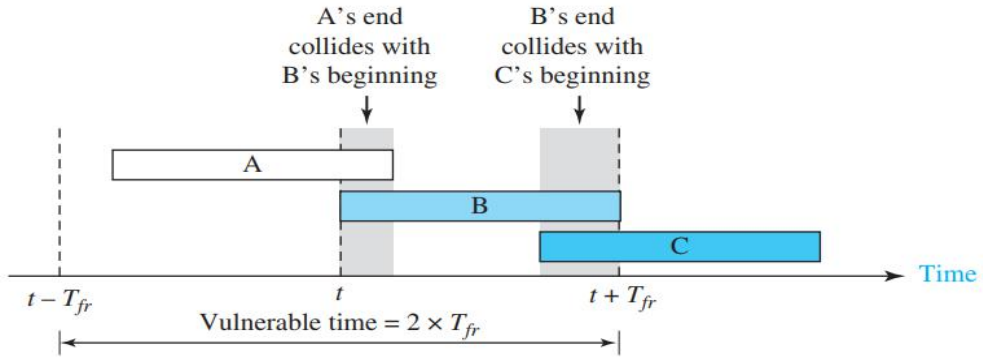


The time-out period is equal to the maximum possible round-trip propagation delay, which is twice the amount of time required to send a frame between the two most widely separated stations ($2 \times T_p$). The backoff time T_B is a random value that normally depends on K (the number of attempted unsuccessful 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.

Example 12.1
The stations on a wireless ALOHA network are a maximum of 600 km apart. If we assume that signals propagate at 3 × 108 m/s, we find Tp = (600 × 103) / (3 × 108) = 2 ms. For K = 2, the range of R is {0, 1, 2, 3}. This means that TB can be 0, 2, 4, or 6 ms, based on the outcome of the random variable R.

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 *Vulnerable time for pure ALOHA protocol*



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 $t + T_{fr}$. Here, there is also a collision between frames from station B and station C.
The vulnerable time during which a collision may occur in pure ALOHA is 2 times the frame transmission time.
Pure ALOHA vulnerable time = 2 x T_{fr}

Example 12.2
A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is the requirement to make this frame collision-free?
Solution
Average frame transmission time Tfr is 200 bits/200 kbps or 1 ms. The vulnerable time is 2 × 1 ms = 2 ms. This means no station should send later than 1 ms before this station starts transmission and no station should start sending during the period (1 ms) that this station is sending.

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 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).
In other words, 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.

The throughput for pure ALOHA is $S = G \times e^{-2G}$.
The maximum throughput $S_{max} = 1/(2e) = 0.184$ when G = (1/2).

Example 12.3 A pure ALOHA network transmits 200-bit frames on a shared channel of 200 kbps. What is the throughput if the system (all stations together) produces
a. 1000 frames per second?
b. 500 frames per second?

c. 250 frames per second?

Solution

The frame transmission time is 200/200 kbps or 1 ms.

a. If the system creates 1000 frames per second, or 1 frame per millisecond, then G = 1. In this case S = G × e^{-2G} = 0.135 (13.5 percent). This means that the throughput is 1000 × 0.135 = 135 frames. Only 135 frames out of 1000 will probably survive.

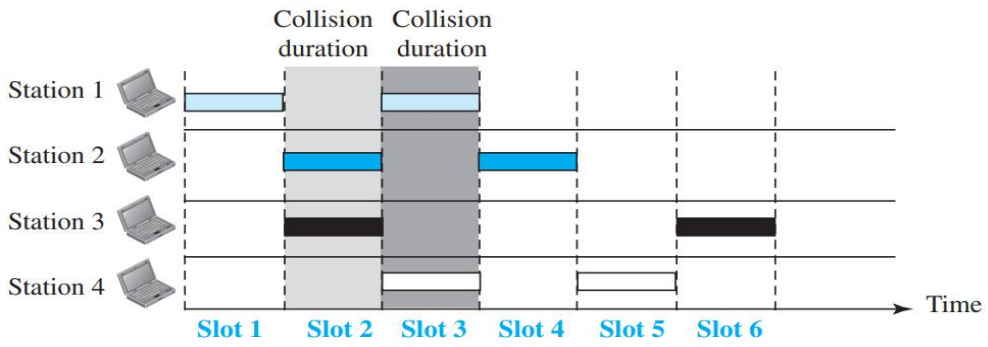
b. If the system creates 500 frames per second, or 1/2 frames per millisecond, then G = 1/2. In this case S = G × e^{-2G} = 0.184 (18.4 percent). This means that the throughput is 500 × 0.184 = 92 and that only 92 frames out of 500 will probably survive. Note that this is the maximum throughput case, percentagewise

c. If the system creates 250 frames per second, or 1/4 frames per millisecond, then G = 1/4. In this case S = G × e^{-2G} = 0.152 (15.2 percent). This means that the throughput is 250 × 0.152 = 38. Only 38 frames out of 250 will probably survive.

Slotted ALOHA

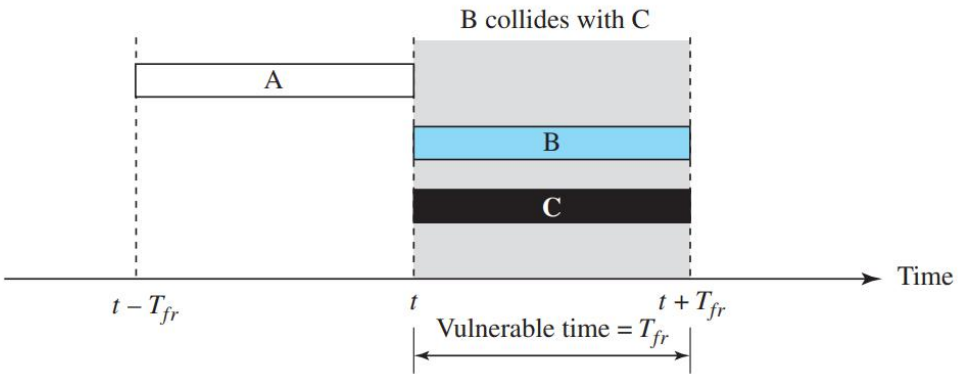
Pure ALOHA has a vulnerable time of 2 × T_{fr}. This is so because there is no rule that defines when the station can send. A station may send soon after another station has started or just before another station has finished. Slotted ALOHA was invented to improve the efficiency of pure ALOHA. In slotted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

Figure 12.6 *Vulnerable time for slotted ALOHA protocol*



Slotted ALOHA vulnerable time = T_{fr}

Throughput

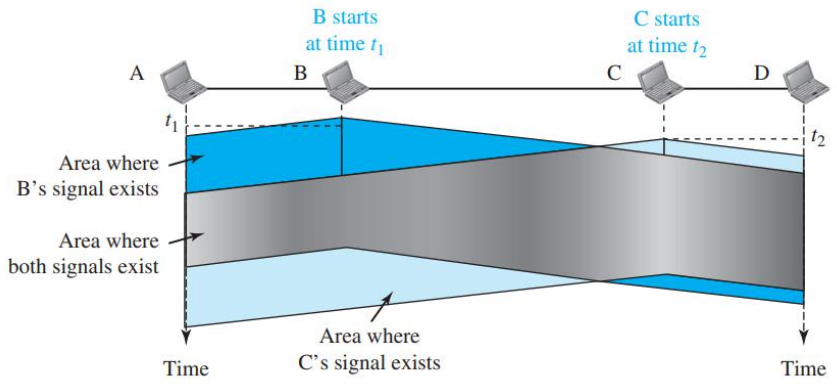
It can be proven that the average number of successful transmissions for slotted ALOHA is $S = G \times e^{-G}$. The maximum throughput S_{max} is 0.368, when G = 1. In other words, if one frame is generated during one frame transmission time, then 36.8 percent of these frames reach their destination 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.

The throughput for slotted ALOHA is $S = G \times e^{-G}$.
The maximum throughput $S_{max} = 0.368$ when G = 1.

Example 12.4
A slotted ALOHA network transmits 200-bit frames using a shared channel with a 200-kbps bandwidth. Find the throughput if the system (all stations together) produces
a. 1000 frames per second.
b. 500 frames per second.
c. 250 frames per second.
Solution
This situation is similar to the previous exercise except that the network is using slotted ALOHA instead of pure ALOHA. The frame transmission time is 200/200 kbps or 1 ms.
a. In this case G is 1. So $S = G \times e^{-G} = 0.368$ (36.8 percent). This means that the throughput is 1000 × 0.0368 = 368 frames. Only 368 out of 1000 frames will probably survive. Note that this is the maximum throughput case, percentagewise.
b. Here G is 1/2. In this case $S = G \times e^{-G} = 0.303$ (30.3 percent). This means that the throughput is 500 × 0.0303 = 151. Only 151 frames out of 500 will probably survive.
c. Now G is 1/4. In this case $S = G \times e^{-G} = 0.195$ (19.5 percent). This means that the throughput is 250 × 0.195 = 49. Only 49 frames out of 250 will probably survive.

CSMA
To minimize the chance of collision and, therefore, increase the performance, the CSMA method was developed. The chance of collision can be reduced if a station senses the medium before trying to use it. Carrier sense multiple access (CSMA) requires that each station first listen to the medium (or check the state of the medium) before sending.
In other words, CSMA is based on the principle "sense before transmit" or "listen before talk."
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).

Figure 12.7 *Space/time model of a collision in CSMA*



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.

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.

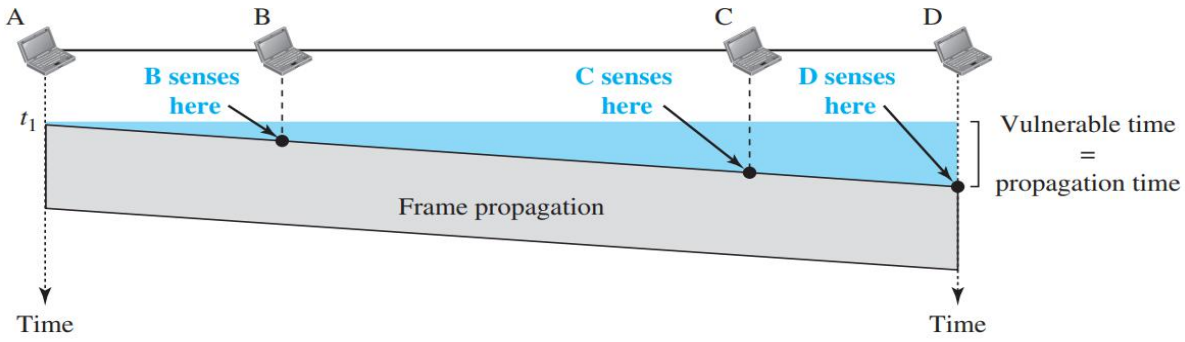
Vulnerable Time

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.

Figure 12.8 *Vulnerable time in CSMA*



Persistence Methods

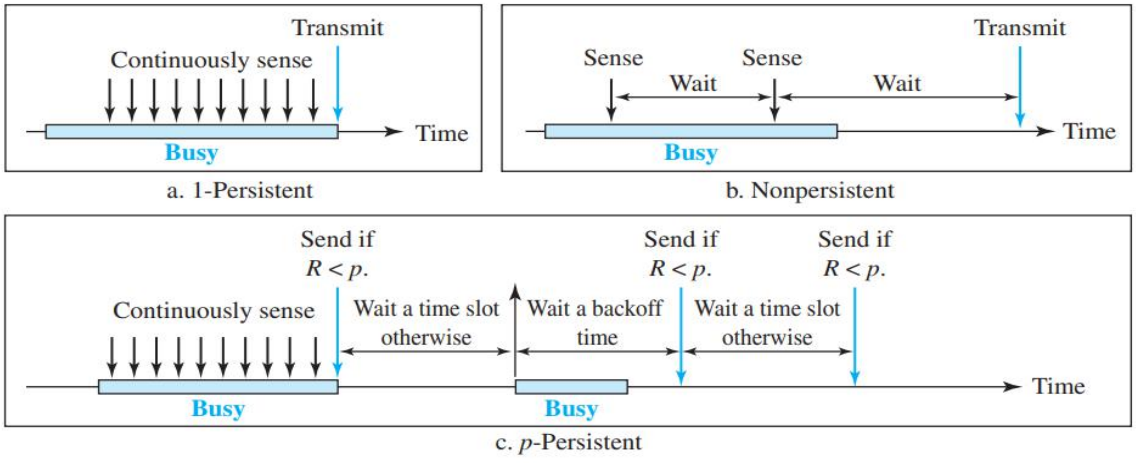
What should a station do if the channel is busy? What should a station do if the channel is idle?

Three methods have been devised to answer these questions:

the 1-persistent method, the nonpersistent method, and the p-persistent method.

Figure 12.9 shows the behavior of three persistence methods when a station finds a channel busy.

Figure 12.9 *Behavior of three persistence methods*



a. 1-Persistent

The 1-persistent method is simple and straightforward. In this method, after the station finds the line idle, it sends its frame immediately (with probability 1). This method has the highest chance of collision because two or more stations may find the line idle and send their frames immediately. We will see later that Ethernet uses this method.

b. Nonpersistent

In the nonpersistent method, a station that has a frame to send senses the line. If the line is idle, it sends immediately. If the line is not idle, it waits a random amount of time and then senses the line again. The nonpersistent approach reduces the chance of collision because it is unlikely that two or more stations will wait the same amount of time and retry to send simultaneously. However, this method reduces the efficiency of the network because the medium remains idle when there may be stations with frames to

send.

c. p-Persistent

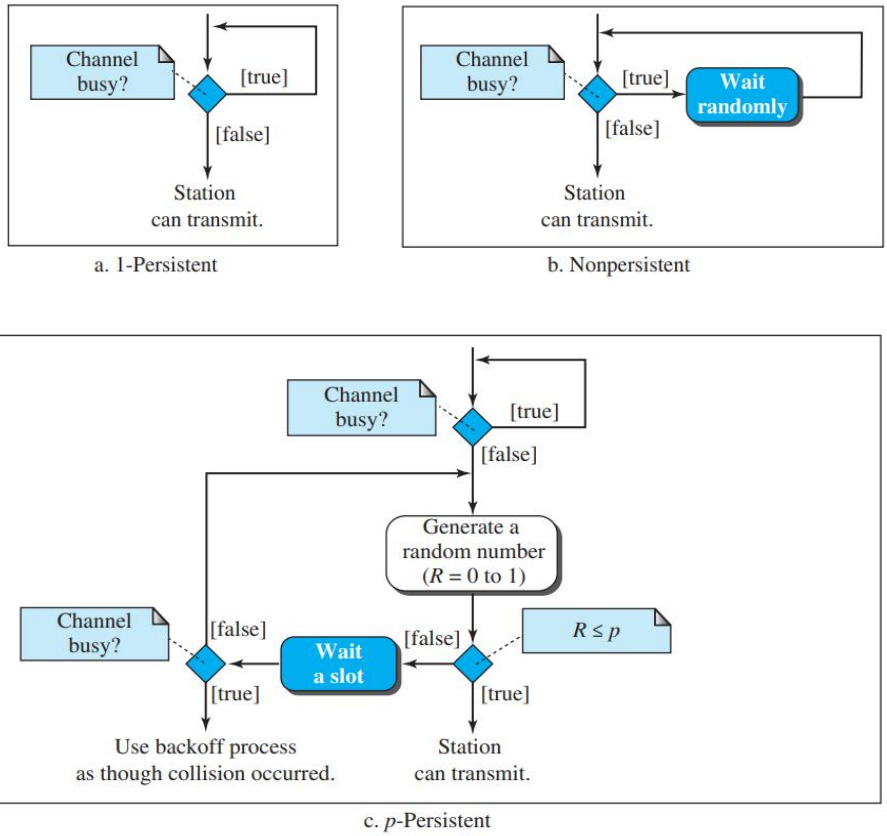
The p-persistent method is used if the channel has time slots with a slot duration equal to or greater than the maximum propagation time. The p-persistent approach combines the advantages of the other two strategies. It reduces the chance of collision and improves efficiency. In this method, after the station finds the line idle it follows these steps:

1. With probability p, the station sends its frame.
2. With probability q = 1 − p, the station waits for the beginning of the next time slot and checks the line again.

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 backoff procedure.

Figure 12.10 *Flow diagram for three persistence methods*

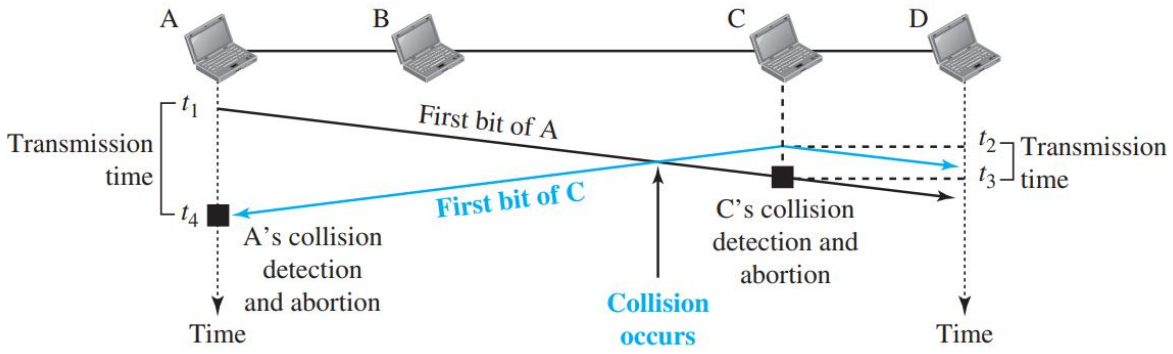


CSMA/CD

The CSMA method does not specify the procedure following a collision. 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. To better understand CSMA/CD, let us look at 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.

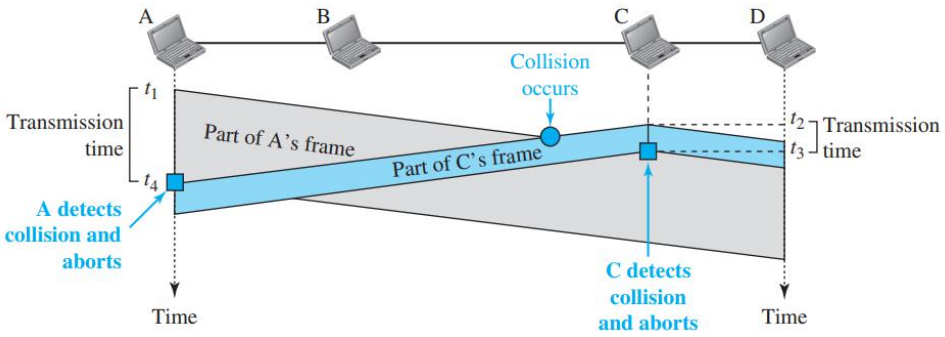
Figure 12.11 *Collision of the first bits in CSMA/CD*



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.

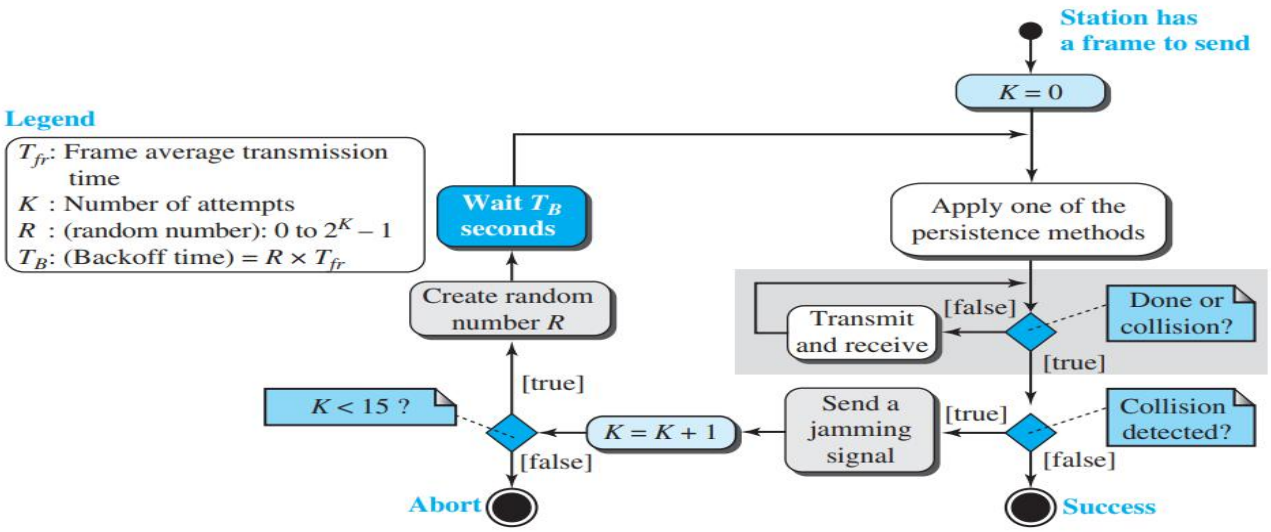
Figure 12.12 *Collision and abortion in CSMA/CD*



Minimum Frame Size

For CSMA/CD to work, we need a restriction on the frame size. Before sending the last bit of the frame, the sending station must detect a collision, if any, and abort the transmission. This is so because the station, once the entire frame is sent, does not keep a copy of the frame and does not monitor the line for collision detection. Therefore, the frame transmission time T_{fr} must be at least two times the maximum propagation time T_p. To understand the reason, let us think about the worst-case scenario. If the two stations involved in a collision are the maximum distance apart, the signal from the first takes time T_p to reach the second, and the effect of the collision takes another time T_P to reach the first. So the requirement is that the first station must still be transmitting after $2T_p$.

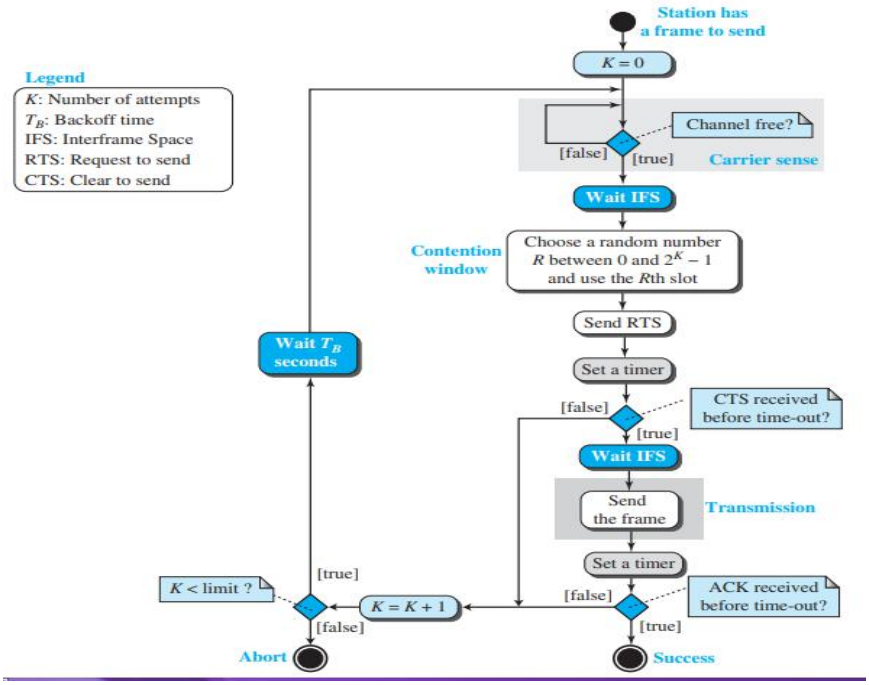
Example 12.5: A network using CSMA/CD has a bandwidth of 10 Mbps. If the maximum propagation time (including the delays in the devices and ignoring the time needed to send a jamming signal, as we see later) is 25.6 μs, what is the minimum size of the frame?
Solution: The minimum frame transmission time is Tfr = 2 × Tp = 51.2 μs. This means, in the worst case, a station needs to transmit for a period of 51.2 μs to detect the collision. The minimum size of the frame is 10 Mbps × 51.2 μs = 512 bits or 64 bytes. This is actually the minimum size of the frame for Standard Ethernet.



The first difference is the addition of the persistence process. We need to sense the channel before we start sending the frame by using one of the persistence processes (nonpersistent, 1-persistent, or p-persistent).
The second difference is the frame transmission. In ALOHA, we first transmit the entire frame and then wait for an acknowledgment. In CSMA/CD, transmission and collision detection are continuous processes. We do not send the entire frame and then look for a collision. The station transmits and receives continuously and simultaneously (using two different ports or a bidirectional port). We use a loop to show that transmission is a continuous process. We constantly monitor in order to detect one of two conditions: either transmission is finished or a collision is detected. Either event stops transmission. When we come out of the loop, if a collision has not been detected, it means that transmission is complete; the entire frame is transmitted. Otherwise, a collision has occurred.
The third difference is the sending of a short jamming signal to make sure that all other stations become aware of the collision.

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.



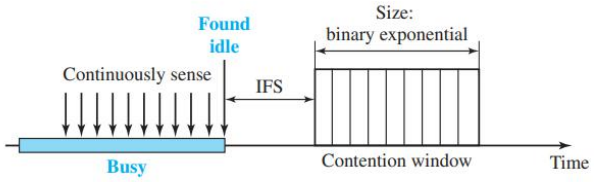
- Interframe Space (IFS).

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 a 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.

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.

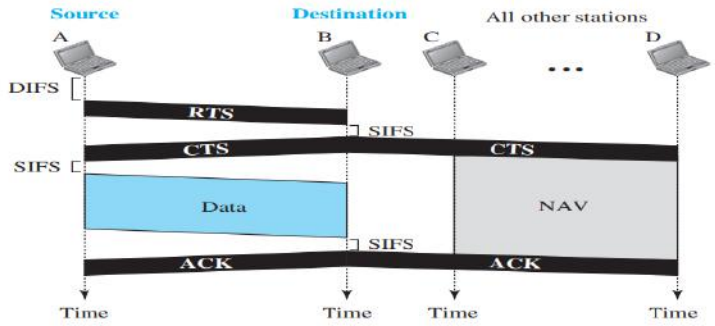
b. After the station is found to be idle, the station waits for a period of time called the DCF interframe space (DIFS); then the station sends a control frame called the request to send (RTS).

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.

Figure 12.17 *CSMA/CA and NAV*



Network Allocation Vector

How do other stations defer sending their data if one station acquires access? In other words, how is the collision avoidance aspect of this protocol accomplished? The key is a feature called NAV.

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. In other words, 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.

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.

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.

There are three controlled-access methods.

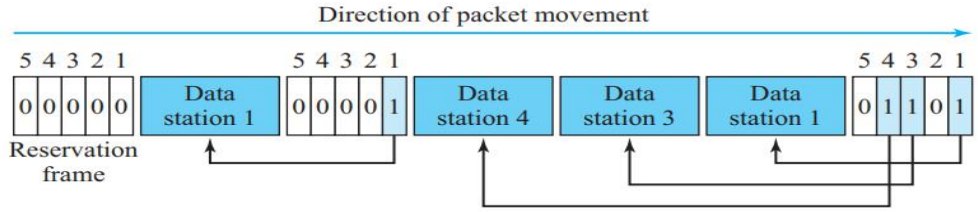
1. Reservation

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.

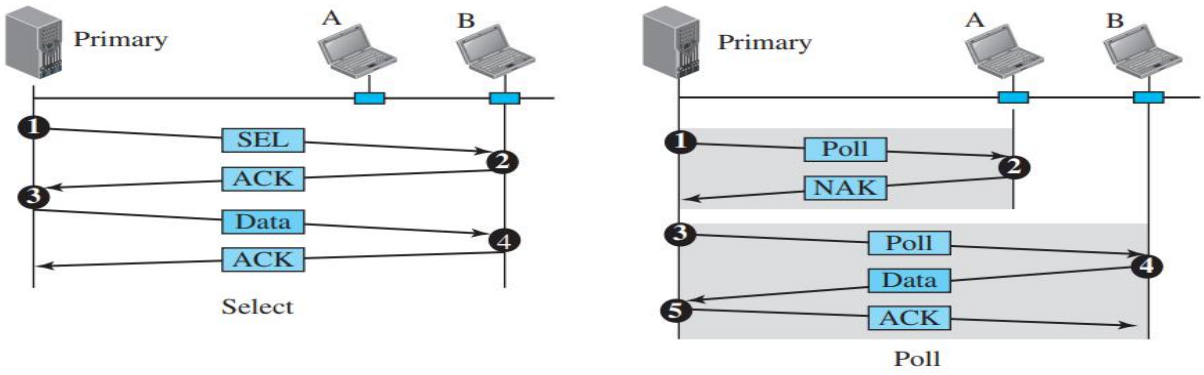
Figure 12.18 *Reservation access method*



2. 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.

3. Token Passing

In the token-passing method, the stations in a network are organized in a logical ring. In other words, 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. But how is the right to access the channel passed from one station to another? In this method, a special packet called a token circulates through the ring. The possession of the token

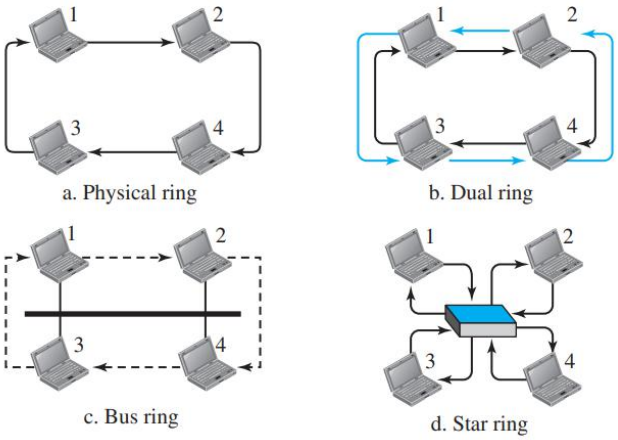
gives the station the right to access the channel and send its data. When a station has some data to send, it waits until it receives the token from its predecessor. It then holds the token and sends its data. When the station has no more data to send, it releases the token, passing it to the next logical 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

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 star ring 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.

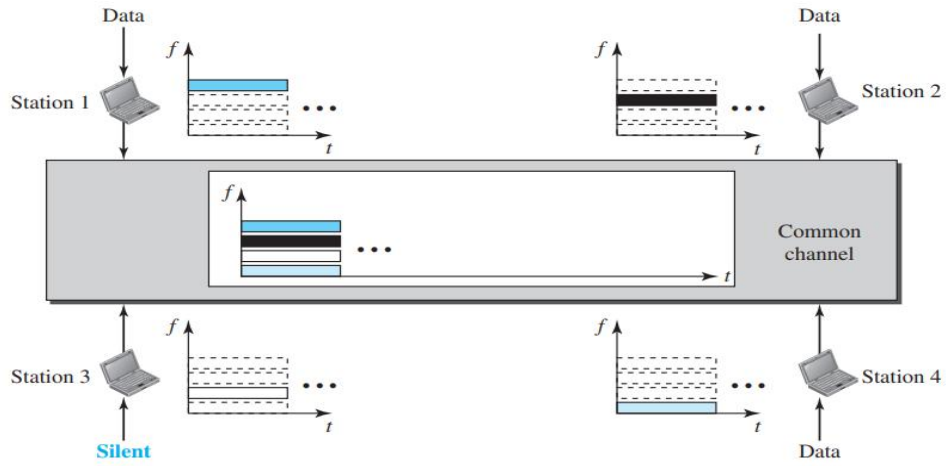
CHANNELIZATION

Channelization (or channel partition, as it is sometimes called) 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, and CDMA.

FDMA

In frequency-division multiple access (FDMA), the available bandwidth is divided into frequency bands. Each station is allocated a band to send its data. In other words, each band is reserved for a specific station, and it belongs to the station all the time. Each station also uses a bandpass filter to confine the transmitter frequencies. To prevent station interferences, the allocated bands are separated from one another by small guard bands.

Figure 12.21 *Frequency-division multiple access (FDMA)*

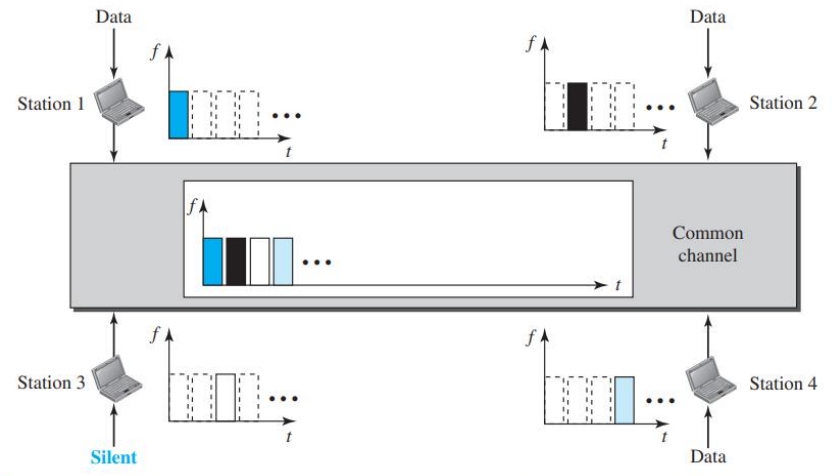


FDMA specifies a predetermined frequency band for the entire period of communication. This means that stream data (a continuous flow of data that may not be packetized) can easily be used with FDMA.

TDMA

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.

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.

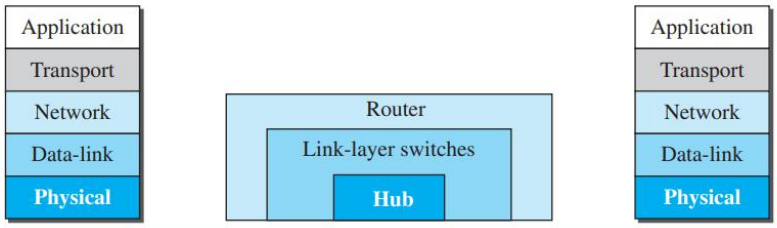
In CDMA, one channel carries all transmissions simultaneously.

CONNECTING DEVICES

Hosts and networks do not normally operate in isolation. We use connecting devices 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, link-layer switches, and routers. Hubs today operate in the first layer of the Internet model. Link-layer switches operate in the first two layers. Routers operate in the first three layers.

Figure 17.1 *Three categories of connecting devices*



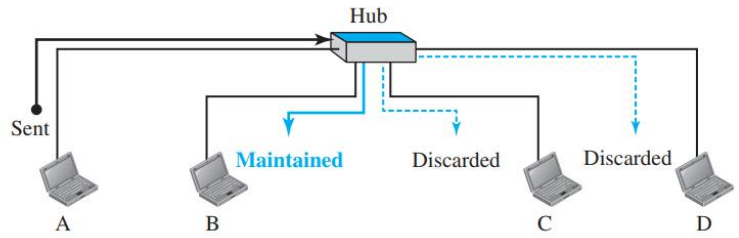
1. Hubs

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. In the past, when Ethernet LANs were using bus topology, 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. Figure 17.2 shows that 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. In other words, 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. Figure 17.2 shows the role of a repeater or a hub in a switched LAN. The figure definitely shows that a hub does not have a filtering capability; it does not have the intelligence to find from which port the frame should be sent out.

A repeater has no filtering capability.

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*



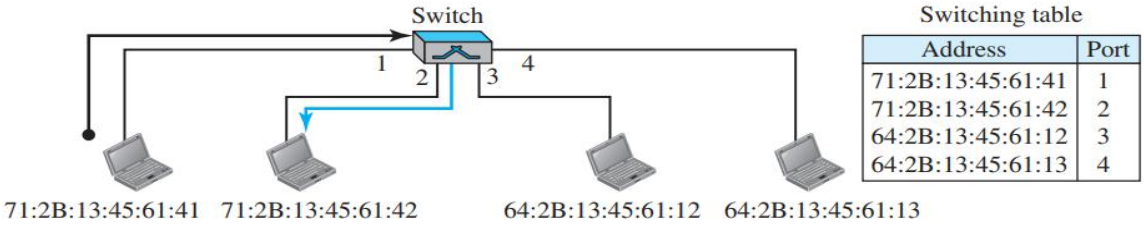
2. Link-Layer Switches

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 One may ask what the difference in functionality is between a link-layer switch and a hub. 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.

A link-layer switch has a table used in filtering decisions.

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

3. 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.

A router changes the link-layer addresses in a packet.

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.

The network layer in the TCP/IP protocol suite is responsible for the host-to-host delivery of datagrams. It provides services to the transport layer and receives services from the data-link layer. It's various functions are:

- Addressing: Maintains the address at the frame header of both source and destination and performs addressing to detect various devices in network.
- Packetizing: This is performed by Internet Protocol. The network layer converts the packets from its upper layer.
- Routing: It is the most important functionality. The network layer chooses the most relevant and best path for the data transmission from source to destination.
- Inter-networking: It works to deliver a logical connection across multiple devices.

Network layer design issues

The network layer comes with some design issues they are described as follows:

1. Store and Forward packet switching:

The host sends the packet to the nearest router. This packet is stored there until it has fully arrived once the link is fully processed by verifying the checksum then it is forwarded to the next router till it reaches the destination. This mechanism is called "Store and Forward packet switching."

2. Services provided to Transport Layer:

Through the network/transport layer interface, the network layer transfers it's services to the transport layer. These services are described below.

But before providing these services to the transfer layer following goals must be kept in mind :-

- Offering services must not depend on router technology.
- The transport layer needs to be protected from the type, number and topology of the available router.
- The network addresses for the transport layer should use uniform numbering pattern also at LAN and WAN connections.

Based on the connections there are 2 types of services provided :

- Connectionless – The routing and insertion of packets into subnet is done individually. No added setup is required.
- Connection-Oriented – Subnet must offer reliable service and all the packets must be transmitted over a single route.

3. Implementation of Connectionless Service:

Packet are termed as "datagrams" and corresponding subnet as "datagram subnets". When the message size that has to be transmitted is 4 times the size of the packet, then the network layer divides into 4 packets and transmits each packet to router via. a few protocol. Each data packet has destination address and is routed independently irrespective of the packets.

4. Implementation of Connection Oriented service:

To use a connection-oriented service, first we establishes a connection, use it and then release it. In connection-oriented services, the data packets are delivered to the receiver in the same order in which they have been sent by the sender.

It can be done in either two ways :

- Circuit Switched Connection – A dedicated physical path or a circuit is established between the communicating nodes and then data stream is transferred.
- Virtual Circuit Switched Connection – The data stream is transferred over a packet switched network, in such a way that it seems to the user that there is a dedicated path from the sender to the receiver. A virtual path is established here. While, other connections may also be using the same path.

Comparison between Datagram switching & Virtual circuit switching :

Datagram Switching	Virtual Circuit
It is connection less service. There is no need for reservation of resources as there is no dedicated path for a connection session.	Virtual circuits are connection-oriented, which means that there is a reservation of resources like buffers, bandwidth, etc. for the time during which the new setup VC is going to be used by a data transfer session.
All packets are free to use any available path. As a result, intermediate routers calculate routes on the go due to dynamically changing routing tables on routers.	The first sent packet reserves resources at each server along the path. Subsequent packets will follow the same path as the first sent packet for the connection time.
Data packets reach the destination in random order, which means they need not reach in the order in which they were sent out.	Packets reach in order to the destination as data follows the same path.
Every packet is free to choose any path, and hence all the packets must be associated with a header containing information about the source and the upper layer data.	All the packets follow the same path and hence a global header is required only for the first packet of connection and other packets will not require it.
Datagram networks are not as reliable as Virtual Circuits.	Virtual Circuits are highly reliable.
Efficiency high, delay more	Efficiency low and delay less
But it is always easy and cost-efficient to implement datagram networks as there is no need of reserving resources and making a dedicated path each time an application has to communicate.	Implementation of virtual circuits is costly as each time a new connection has to be set up with reservation of resources and extra information handling at routers.
A Datagram based network is a true packet switched network. There is no fixed path for transmitting data.	A virtual circuit network uses a fixed path for a particular session, after which it breaks the connection and another path has to be set up for the next session.
Widely used in Internet	Used in X.25, ATM(Asynchronous Transfer Mode)

Routing algorithm

- In order to transfer the packets from source to the destination, the network layer must determine the best route through which packets can be transmitted.
- Whether the network layer provides datagram service or virtual circuit service, the main job of the network layer is to provide the best route. The routing protocol provides this job.
- The routing protocol is a routing algorithm that provides the best path from the source to the destination. The best path is the path that has the "least-cost path" from source to the destination.
- Routing is the process of forwarding the packets from source to the destination but the best route to send the packets is determined by the routing algorithm.

Routing algorithms can be broadly categorized into two types, adaptive and nonadaptive routing algorithms.

Adaptive Routing Algorithms

Adaptive routing algorithms, also known as dynamic routing algorithms, makes routing decisions

dynamically depending on the network conditions. It constructs the routing table depending upon the network traffic and topology. They try to compute the optimized route depending upon the hop count, transit time and distance.

Non – Adaptive Routing Algorithms

Non-adaptive Routing algorithms, also known as static routing algorithms, construct a static routing table to determine the path through which packets are to be sent. The static routing table is constructed based upon the routing information stored in the routers when the network is booted up.

The two types of non – adaptive routing algorithms are −

- Flooding − In flooding, when a data packet arrives at a router, it is sent to all the outgoing links except the one it has arrived on. Flooding may be uncontrolled, controlled or selective flooding.
- Random walks − This is a probabilistic algorithm where a data packet is sent by the router to any one of its neighbours randomly.

The Optimality Principle

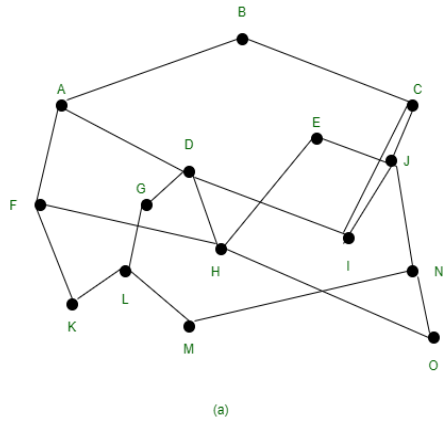
A general statement is made about optimal routes without regard to network topology or traffic. This statement is known as the optimality principle.

It states that if the router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along the same route. Call the route from I to J *r1* and the rest of the route *r2*. it could be concatenated with *r1* to improve the route from I to K, contradicting our statement that *r1r2* is optimal only if a route better than r2 existed from J to K.

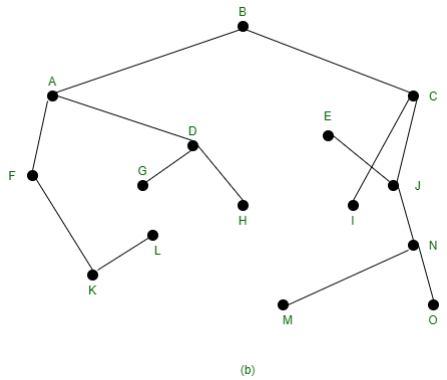
Sink Tree for routers :

We can see that the set of optimal routes from all sources to a given destination from a tree rooted at the destination as a directed consequence of the optimality principle. This tree is called a **sink tree**.

In the given figure the distance metric is the number of hops. Therefore, the goal of all routing algorithms is to discover and use the sink trees for all routers.



(a) A network



(b) A sink tree for router B

The sink tree is not unique also other trees with the same path lengths may exist. If we allow all of the possible paths to be chosen, the tree becomes a more general structure called a **DAG (Directed Acyclic Graph).** DAGs have no loops. We will use sink trees as a convenient shorthand for both cases. we will take technical assumption for both cases that the paths do not interfere with each other so, for example, a traffic jam on one path will not cause another path to divert.

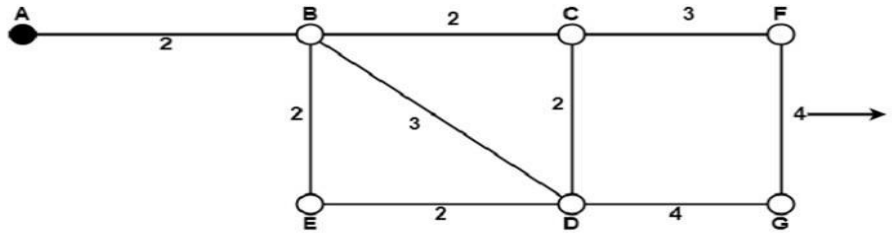
The sink tree does not contain any loops, so each packet will be delivered within a finite and bounded number of hops. In practice, life is not quite easy. Links and routers can go on and come back up during operation, so different routers may have different ideas about the current topology. Also, we have found the issue of whether each router has to individually acquire the information on which to base its sink tree computation or whether this information is collected by some other means.
Sink tree and the optimality principle provide a benchmark against which other routing algorithms can be measured.

Shortest Path Routing
In this algorithm, to select a route, the algorithm discovers the shortest path between two nodes. It can use multiple hops, the geographical area in kilometres or labeling of arcs for measuring path length.
The labeling of arcs can be done with mean queuing, transmission delay for a standard test packet on an hourly basis, or computed as a function of bandwidth, average distance traffic, communication cost, mean queue length, measured delay or some other factors.

In shortest path routing, the topology communication network is defined using a directed weighted graph. The nodes in the graph define switching components and the directed arcs in the graph define communication connection between switching components. Each arc has a weight that defines the cost of sharing a packet between two nodes in a specific direction.

This cost is usually a positive value that can denote such factors as delay, throughput, error rate, financial costs, etc. A path between two nodes can go through various intermediary nodes and arcs. The goal of shortest path routing is to find a path between two nodes that has the lowest total cost, where the total cost of a path is the sum of arc costs in that path.

For example, Dijikstra uses the nodes labeling with its distance from the source node along the better-known route. Initially, all nodes are labelled with infinity, and as the algorithm proceeds, the label may change. The labeling graph is displayed in the figure.



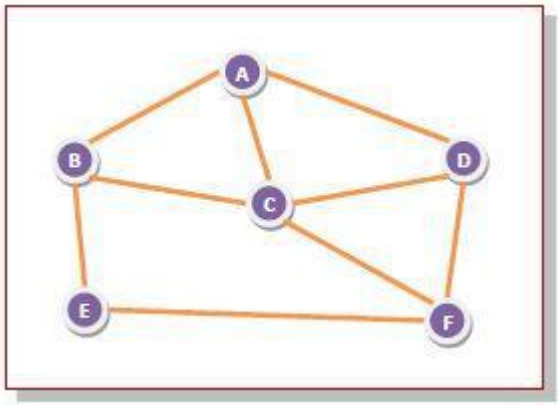
Graphical Representation of Nodes with labeled path

It can be done in various passes as follows, with A as the source.

- Pass 1. B (2, A), C(∞,–), F(∞,–), e(∞,–), d(∞,–), G 60
- Pass 2. B (2, A), C(4, B), D(5, B), E(4, B), F(∞,–),G(∞,–)
- Pass 3. B(2, A), C(4, B), D(5, B), E(4, B), F(7, C), G(9, D)

We can see that there can be two paths between A and G. One follows through ABCFG and the other through ABDG. The first one has a path length of 11, while the second one has 9. Hence, the second one, as G (9, D), is selected. Similarly, Node D has also three paths from A as ABD, ABCD and ABED. The first one has a path length of 5 rest two have 6. So, the first one is selected.
All nodes are searched in various passes, and finally, the routes with the shortest path lengths are made permanent, and the nodes of the path are used as a working node for the next round.

Flooding
Flooding is a non-adaptive routing technique following this simple method: when a data packet arrives at a router, it is sent to all the outgoing links except the one it has arrived on.
For example, let us consider the network in the figure, having six routers that are connected through transmission lines.



Using flooding technique −

- An incoming packet to A, will be sent to B, C and D.
- B will send the packet to C and E.
- C will send the packet to B, D and F.
- D will send the packet to C and F.
- E will send the packet to F.
- F will send the packet to C and E.

Types of Flooding

Flooding may be of three types −

- **Uncontrolled flooding** − Here, each router unconditionally transmits the incoming data packets to all its neighbours.
- **Controlled flooding** − They use some methods to control the transmission of packets to the neighbouring nodes. The two popular algorithms for controlled flooding are Sequence Number Controlled Flooding (SNCF) and Reverse Path Forwarding (RPF).
- **Selective flooding** − Here, the routers don't transmit the incoming packets only along those paths which are heading towards approximately in the right direction, instead of every available paths.

Advantages of Flooding

- It is very simple to setup and implement, since a router may know only its neighbours.
- It is extremely robust. Even in case of malfunctioning of a large number routers, the packets find a way to reach the destination.
- All nodes which are directly or indirectly connected are visited. So, there are no chances for any node to be left out. This is a main criteria in case of broadcast messages.
- The shortest path is always chosen by flooding.

Limitations of Flooding

- Flooding tends to create an infinite number of duplicate data packets, unless some measures are adopted to damp packet generation.
- It is wasteful if a single destination needs the packet, since it delivers the data packet to all nodes irrespective of the destination.
- The network may be clogged with unwanted and duplicate data packets. This may hamper delivery of other data packets.

Distance Vector Routing

Distance Vector Routing is a dynamic routing algorithm. It is mainly used in ARPANET, and RIP. Each router maintains a distance table known as Vector.

It works in the following steps:

Step-01:

Each router prepares its routing table. By their local knowledge. each router knows about-

- All the routers present in the network
- Distance to its neighboring routers

Step-02:

- Each router exchanges its distance vector with its neighboring routers.
- Each router prepares a new routing table using the distance vectors it has obtained from its neighbors.

- This step is repeated for (n-2) times if there are n routers in the network.
- After this, routing tables converge / become stable.

Three Keys to understand the working of Distance Vector Routing Algorithm:

- Knowledge about the whole network: Each router shares its knowledge through the entire network. The Router sends its collected knowledge about the network to its neighbors.
- Routing only to neighbors: The router sends its knowledge about the network to only those routers which have direct links. The router sends whatever it has about the network through the ports. The information is received by the router and uses the information to update its own routing table.
- Information sharing at regular intervals: Within 30 seconds, the router sends the information to the neighboring routers.

Distance Vector Routing Algorithm

Let $d_x(y)$ be the cost of the least-cost path from node x to node y. The least costs are related by Bellman-Ford equation,

$d_x(y) = \min_v\{c(x,v) + d_v(y)\}$ where the $\min_v$ is the equation taken for all x neighbors. After traveling from x to v, if we consider the least-cost path from v to y, the path cost will be $c(x,v) + d_v(y)$. The least cost from x to y is the minimum of $c(x,v) + d_v(y)$ taken over all neighbors.

With the Distance Vector Routing algorithm, the node x contains the following routing information:

- For each neighbor v, the cost c(x,v) is the path cost from x to directly attached neighbor, v.
- The distance vector x, i.e., $D_x = [\ D_x(y) : y \text{ in } N\ ]$, containing its cost to all destinations, y, in N.
- The distance vector of each of its neighbors, i.e., $D_v = [\ D_v(y) : y \text{ in } N\ ]$ for each neighbor v of x.

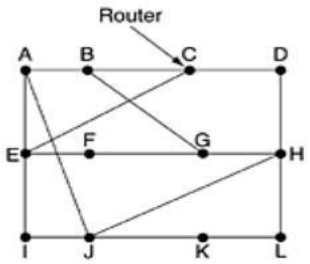
Distance vector routing is an asynchronous algorithm in which node x sends the copy of its distance vector to all its neighbors. When node x receives the new distance vector from one of its neighboring vector, v, it saves the distance vector of v and uses the Bellman-Ford equation to update its own distance vector. The equation is given below:

$d_x(y) = \min_v\{ c(x,v) + d_v(y)\}$ for each node y in N

The node x has updated its own distance vector table by using the above equation and sends its updated table to all its neighbors so that they can update their own distance vectors

The first four columns of part (b) show the delay vectors received from the neighbors of router J. A claims to have a 12-msec delay to B, a 25-msec delay to C, a 40-msec delay to D, etc. Suppose that J has measured or estimated its delay to its neighbors, A, I, H, and K as 8, 10, 12, and 6 msec, respectively.

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



(a)

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

Vectors received from J's four neighbors

(b)

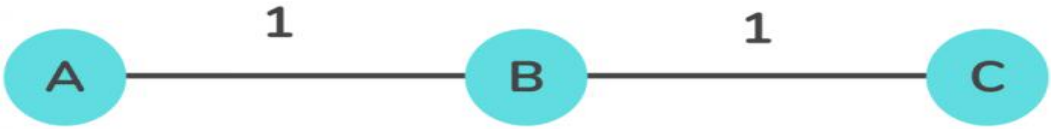
Consider how J computes its new route to router G. It knows that it can get to A in 8 msec, and A claims to be able to get to G in 18 msec, so J knows it can count on a delay of 26 msec to G if it forwards packets bound for G to A. Similarly, it computes the delay to G via I, H, and K as 41 (31 + 10), 18 (6 + 12), and 37

(31 + 6) msec, respectively. The best of these values is 18, so it makes an entry in its routing table that the delay to G is 18 msec and that the route to use is via H. The same calculation is performed for all the other destinations, with the new routing table shown in the last column of the figure.

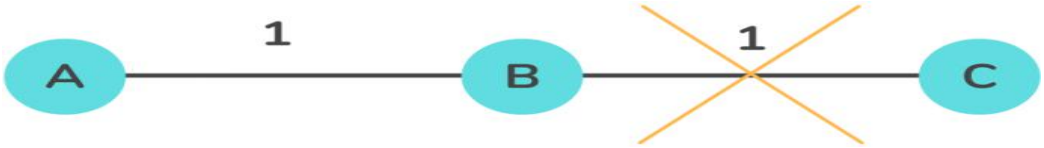
The Count-to-Infinity Problem

The Routing Loops occur when the two neighboring routers in the network send an update simultaneously at the same time to the router. The Routing Loops also occur when an interface (link) goes down between two routers in the network.

If node A tells node B that it has a path somewhere, there is no way for node B to know if the path has node B as a part of it.



Consider the above diagram, for this setup, the Bellman-Ford algorithm will work such that for each router, they will have entries for each other. Router A will infer that it can reach C at a cost of 2 units, and B will infer that it can reach C at a cost of 1 unit.

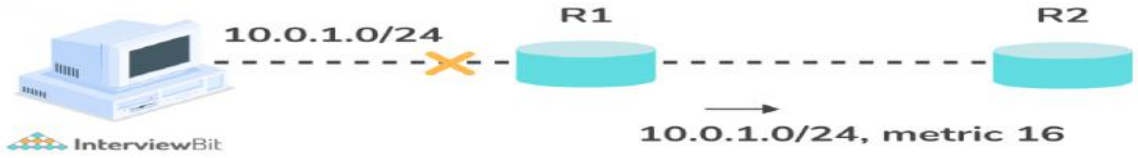


Consider the case in the above diagram, where the connection between B and C gets disconnected. In this case, B will know that it cannot get to C at a cost of 1 anymore and update its table accordingly. However, it can be possible that A sends some information to B that it is possible to reach C from A at a cost of 2. Then, since B can reach A at a cost of 1, B will erroneously update its table that it can reach C via A at a cost of 1 + 2 = 3 units. A will then receive updates from B and update its costs to 4, and so on. Thus, the process enters into a loop of bad feedback and the cost shoots towards infinity. This entire situation is called the Count to Infinity problem.

Solution for the Count to Infinity Problem

1. Route Poisoning:

Route Poisoning is a method used to prevent routers from sending packets through a route that has been deemed invalid within computer networks. Upon failure of a route, Distance Vector Protocols spread the *bad news* about the route failure by poisoning the route. In Route Poisoning, a special metric value called Infinity is used when advertising the route. Routers with a metric of Infinity are considered to have failed. The main disadvantage of this method is that it increases the sizes of routing announcements significantly in many common network topologies.



2. Split Horizon:

In our initial diagram, if the connection between B and C is down, and A sends a route to B, B could use up that route to reach A. Hereafter, A would send the same packet back to B, and the process would end up in an endless loop. However, by Split Horizon Rule, the route to C from A will not be advertised back to B.

Link State Routing

While distance-vector routers use a distributed algorithm to compute their routing tables, link-state routing uses link-state routers to exchange messages that allow each router to learn the entire network

topology. Based on this learned topology, each router is then able to compute its routing table by using the shortest path computation.

Features of Link State Routing Protocols

- Link State Packet: A small packet that contains routing information.
- Link-State Database: A collection of information gathered from the link-state packet.
- Shortest Path First Algorithm (Dijkstra algorithm): A calculation performed on the database results in the shortest path
- Routing Table: A list of known paths and interfaces.

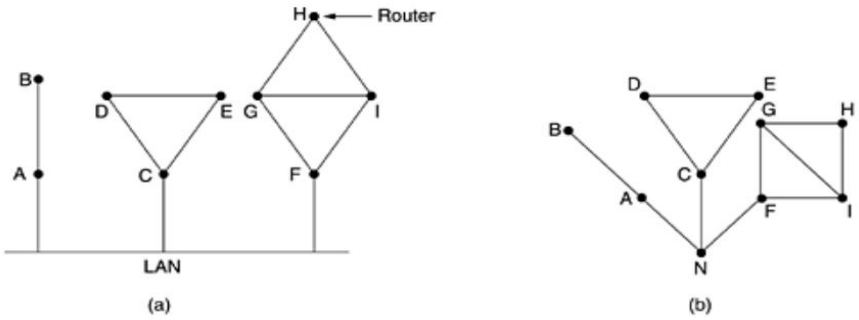
The idea behind link state routing is simple and can be stated as five parts. Each router must do the following:

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

1. Learning about the Neighbors

When a router is booted, its first task is to learn who its neighbors are. It accomplishes this goal by sending a special HELLO packet on each point-to-point line. The router on the other end is expected to send back a reply telling who it is. These names must be globally unique because when a distant router later hears that three routers are all connected to F, it is essential that it can determine whether all three mean the same F.

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



One way to model the LAN is to consider it as a node itself, as shown in Fig. 5-11(b). Here we have introduced a new, artificial node, *N*, to which *A*, *C*, and *F* are connected. The fact that it is possible to go from *A* to *C* on the LAN is represented by the path *ANC* here.

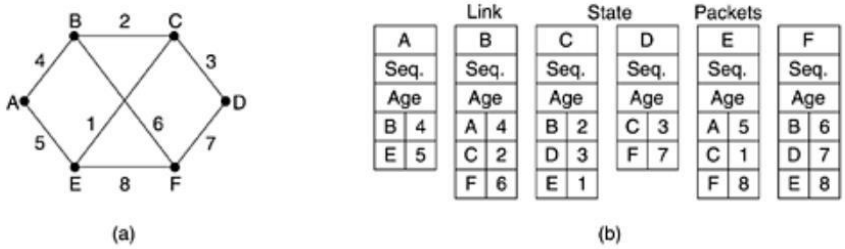
2. Measuring Line Cost

The link state routing algorithm requires each router to know, or at least have a reasonable estimate of, the delay to each of its neighbors. The most direct way to determine this delay is to send over the line a special ECHO packet that the other side is required to send back immediately. By measuring the round-trip time and dividing it by two, the sending router can get a reasonable estimate of the delay. For even better results, the test can be conducted several times, and the average used.

3. Building Link State Packets

Once the information needed for the exchange has been collected, the next step is for each router to build a packet containing all the data. The packet starts with the identity of the sender, followed by a sequence number and age, and a list of neighbors. For each neighbor, the delay to that neighbor is given.

Figure 5-13. (a) A subnet. (b) The link state packets for this subnet.



Building the link state packets is easy. The hard part is determining when to build them. One possibility is to build them periodically, that is, at regular intervals. Another possibility is to build them when some significant event occurs, such as a line or neighbor going down or coming back up again or changing its properties appreciably.

4. Distributing the Link State Packets

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

This algorithm has a few problems, but they are manageable.

First, if the sequence numbers wrap around, confusion will reign. The solution here is to use a 32-bit sequence number. With one link state packet per second, it would take 137 years to wrap around, so this possibility can be ignored.

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), packets 5 through 65,540 will be rejected as obsolete, since the current sequence number is thought to be 65,540.

The solution to all these problems is to include the age of each packet after the sequence number and decrement it once per second. When the age hits zero, the information from that router is discarded. Normally, a new packet comes in, say, every 10 sec, so router information only times out when a router is down (or six consecutive packets have been lost, an unlikely event). The Age field is also decremented by each router during the initial flooding process, to make sure no packet can get lost and live for an indefinite period of time (a packet whose age is zero is discarded).

Some refinements to this algorithm make it more robust. When a link state packet comes in to a router for flooding, it is not queued for transmission immediately. Instead it is first put in a holding area to wait a short while. If another link state packet from the same source comes in before the first packet is transmitted, their sequence numbers are compared. If they are equal, the duplicate is discarded. If they are different, the older one is thrown out. To guard against errors on the router-router lines, all link state packets are acknowledged. When a line goes idle, the holding area is scanned in round-robin order to select a packet or acknowledgement to send.

5. Computing the New Routes

Once a router has accumulated a full set of link state packets, it can construct the entire subnet graph because every link is represented. Every link is, in fact, represented twice, once for each direction. The two values can be averaged or used separately.

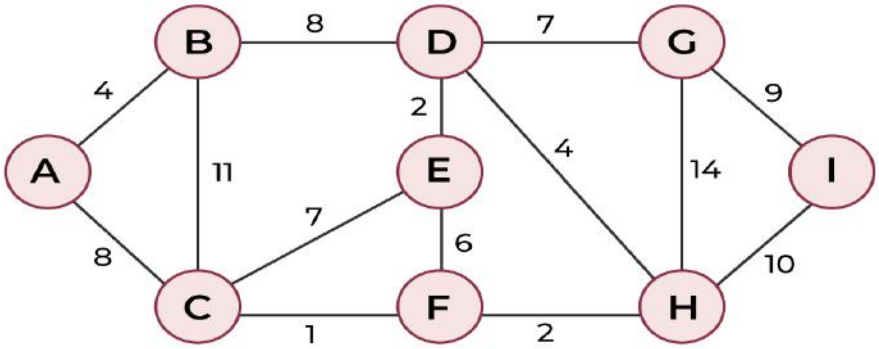
Now Dijkstra's algorithm can be run locally to construct the shortest path to all possible destinations. The results of this algorithm can be installed in the routing tables, and normal operation resumed.

Calculation of Shortest Path

To find the shortest path, each node needs to run the famous Dijikstra algorithm. Let us understand how can we find the shortest path using an example.

Illustration:
To understand the Dijkstra Algorithm, let's take a graph and find the shortest path from the source to all nodes.
Note: We use a boolean array **sptSet[]** to represent the set of vertices included in SPT. If a value **sptSet[v]** is true, then vertex v is included in SPT, otherwise not. Array **dist[]** is used to store the shortest distance values of all vertices.

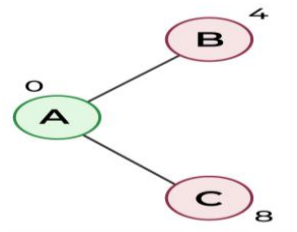
Consider the below graph and src = 0.



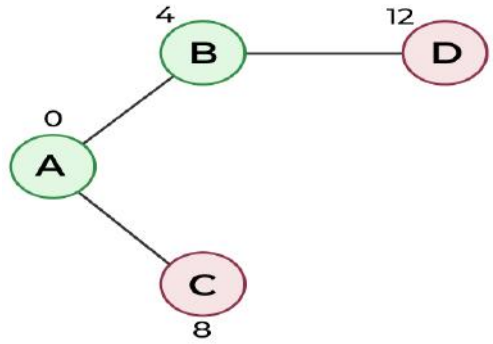
Shortest Path Calculation – Step 1

STEP 1: The set sptSet is initially empty and distances assigned to vertices are {0, INF, INF, INF, INF, INF, INF, INF} where INF indicates infinite. Now pick the vertex with a minimum distance value. The vertex 0 is picked and included in sptSet. So sptSet becomes {0}. After including 0 to sptSet, update the distance values of its adjacent vertices. Adjacent vertices of 0 are 1 and 7. The distance values of 1 and 7 are updated as 4 and 8.

The following subgraph shows vertices and their distance values. Vertices included in SPT are included in GREEN color.

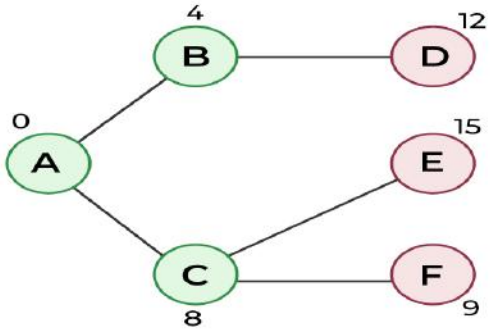


Shortest Path Calculation – Step 2
STEP 2: Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). The vertex 1 is picked and added to sptSet. So sptSet now becomes {0, 1}. Update the distance values of adjacent vertices of 1. The distance value of vertex 2 becomes 12.



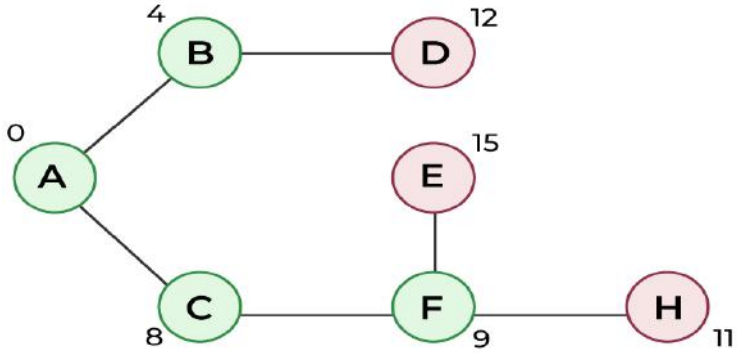
Shortest Path Calculation – Step 3

STEP 3: Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). Vertex 7 is picked. So sptSet now becomes {0, 1, 7}. Update the distance values of adjacent vertices of 7. The distance value of vertex 6 and 8 becomes finite (15 and 9 respectively).



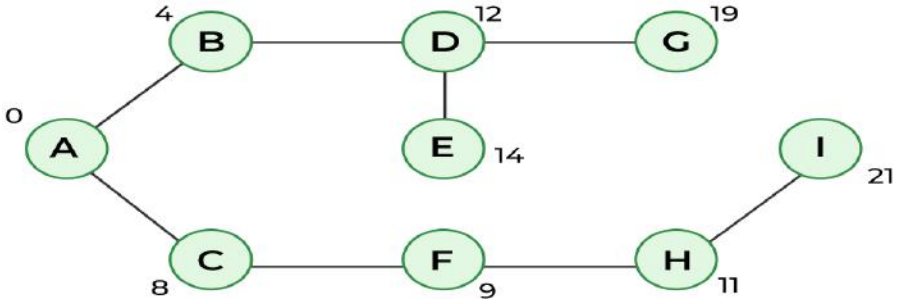
Shortest Path Calculation – Step 4

STEP 4: Pick the vertex with minimum distance value and not already included in SPT (not in sptSET). Vertex 6 is picked. So sptSet now becomes {0, 1, 7, 6}. Update the distance values of adjacent vertices of 6. The distance value of vertex 5 and 8 are updated.



Shortest Path Calculation – Step 5

We repeat the above steps until sptSet includes all vertices of the given graph. Finally, we get the following Shortest Path Tree (SPT).



Shortest Path Calculation – Step 6

Hierarchical Routing

In hierarchical routing, the routers are divided into regions. Each router has complete details about how to route packets to destinations within its own region. But it does not have any idea about the internal structure of other regions.

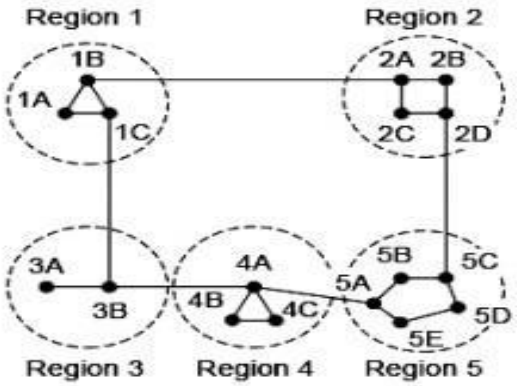
As we know, in both LS and DV algorithms, every router needs to save some information about other routers. When network size is growing, the number of routers in the network will increase. Therefore, the size of routing table increases, then routers cannot handle network traffic as efficiently. To overcome this problem we are using hierarchical routing.

In hierarchical routing, routers are classified in groups called regions. Each router has information about the routers in its own region and it has no information about routers in other regions. So, routers save one record in their table for every other region.

For huge networks, a two-level hierarchy may be insufficient hence, it may be necessary to group the regions into clusters, the clusters into zones, the zones into groups and so on.

Example

Consider an example of two-level hierarchy with five regions as shown in figure



Let see the full routing table for router 1A which has 17 entries, as shown below −

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

When routing is done hierarchically then there will be only 7 entries as shown below −

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

Unfortunately, this reduction in table space comes with the increased path length.

Explanation

Step 1 − For example, the best path from 1A to 5C is via region 2, but hierarchical routing of all traffic to region 5 goes via region 3 as it is better for most of the other destinations of region 5.

Step 2 − Consider a subnet of 720 routers. If no hierarchy is used, each router will have 720 entries in its routing table.

Step 3 − Now if the subnet is partitioned into 24 regions of 30 routers each, then each router will require 30 local entries and 23 remote entries for a total of 53 entries.

Example
If the same subnet of 720 routers is partitioned into 8 clusters, each containing 9 regions and each region containing 10 routers. Then what will be the total number of table entries in each router.
Solution
10 local entries + 8 remote regions + 7 clusters = 25 entries.

Broadcast Routing
In some applications, hosts need to send messages to many or all other hosts.
For example, a service distributing weather reports, stock market updates, or live radio programs might work best by broadcasting to all machines and letting those that are interested read the data. Sending a packet to all destinations simultaneously is called broadcasting;

One broadcasting method that requires no special features from the subnet is for the source to simply send a distinct packet to each destination.

Flooding is another obvious candidate. Although flooding is ill-suited for ordinary point-to-point communication, for broadcasting it might rate serious consideration, especially if none of the methods described below are applicable. The problem with flooding as a broadcast technique is the same problem it has as a point-to-point routing algorithm: it generates too many packets and consumes too much bandwidth.

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

A fourth broadcast algorithm makes explicit use of the sink tree for the router initiating the broadcast—or any other convenient spanning tree for that matter. A spanning tree is a subset of the subnet that includes all the routers but contains no loops. If each router knows which of its lines belong to the spanning tree, it can copy an incoming broadcast packet onto all the spanning tree lines except the one it arrived on.

Fifth Algorithm is Reverse Path Forwarding (RPF).
The algorithm forces the router to forward a multicast packet from one specific interface: the one which has come through the shortest path from the source to the router.
The router uses the first property of the shortest-path tree (we discussed in unicast routing), which says that the shortest path from A to B is also the shortest path from B to A. The router does not know the shortest path from the source to itself, but it can find which is the next router in the shortest path from itself to the source (reverse path). The router simply consults its unicast forwarding table, pretending that it wants to send a packet to the source; the forwarding table gives the next router and the interface the message that the packet should be sent out in this reverse direction. The router uses this information to accept a multicast packet only if it arrives from this interface. This is needed to prevent looping. In multicasting, a packet may arrive at the same router that has forwarded it. If the router does not drop all arrived packets except the one, multiple copies of the packet will be circulating in the internet. Of course, the router may add a tag to the packet when it arrives the first time and discard packets that arrive with the same tag, but the RPF strategy is simpler

Multicast Routing
Some applications require that widely-separated processes work together in groups, for example, a group of processes implementing a distributed database system.

- We need a way to send messages to well-defined groups that 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 its routing algorithm is called multicast routing.
- Multicasting requires group management.
- Some way is needed to create and destroy groups, and to allow processes to join and leave groups.
- When a process joins a group, it informs its host of this fact. It is important that routers know which of their hosts belong to which groups.
- Either host must inform their routers about changes in group membership, or routers must query their hosts periodically
- Either way, routers learn about which of their hosts are in which groups.
- Routers tell their neighbors, so the information propagates through the subnet.
- To do multicast routing, each router computes a spanning tree covering all other routers.
  For example, in Fig. we have two groups, 1 and 2. Some routers are attached to hosts that belong to one or both of these groups, as indicated in the figure. A spanning tree for the leftmost router is shown in Fig. (b).

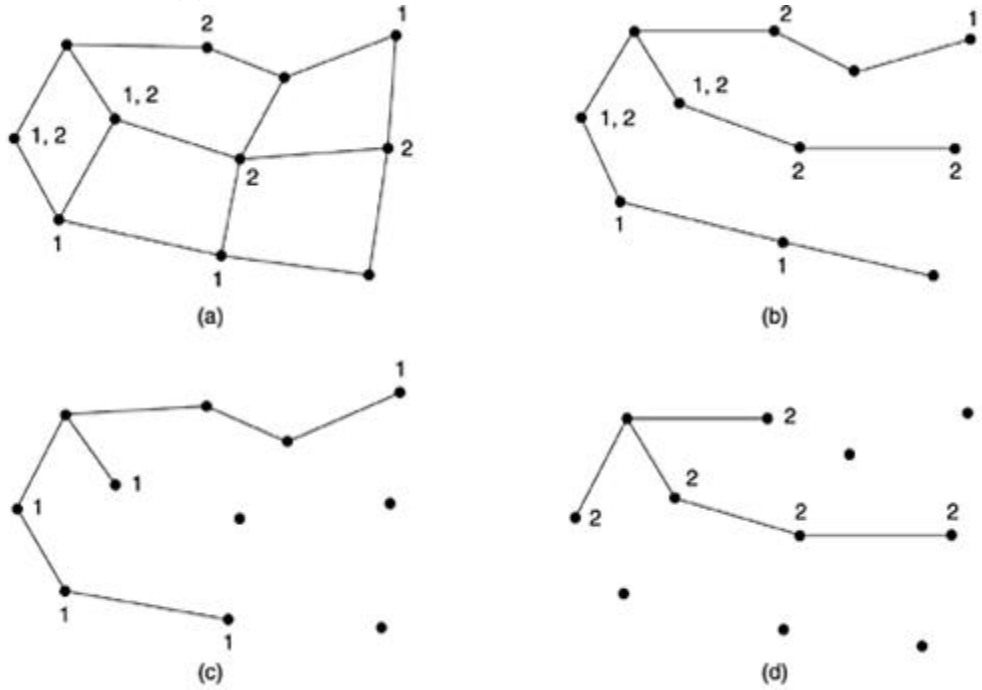


Figure (a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.

- When a process sends a multicast packet to a group, the first router examines its spanning tree and prunes it, removing all lines that do not lead to hosts that are members of the group.
- In our example, Fig. (c) shows the pruned spanning tree for group 1.
- Similarly, Fig. 3.11(d) shows the pruned spanning tree for group 2.
- Multicast packets are forwarded only along the appropriate spanning tree.

Ways of Pruning Spanning Tree:

- Various ways of pruning the spanning tree are possible.
- The simplest one can be used if link state routing is used and each router is aware of the complete topology, including which hosts belong to which groups. Then the spanning tree can be pruned, starting at the end of each path, working toward the root, and removing all routers that do not belong to the group in question.
- With distance vector routing, a different pruning strategy can be followed. The basic algorithm is reverse path forwarding. However, whenever a router with no hosts interested in a particular group and no connections to other routers receives a multicast message for that group, it responds with a PRUNE message, telling the sender not to send it any more multicasts for that group.

- When a router with no group members among its own hosts has received such messages on all its lines, it, too, can respond with a PRUNE message. In this way, the subnet is recursively pruned.

Advantage:

- It scales poorly to large networks.

Routing Information Protocol (RIP)

RIP stands for Routing Information Protocol. RIP is an intra-domain routing protocol used within an autonomous system. Here, intra-domain means routing the packets in a defined domain, for example, web browsing within an institutional area.

- RIP is based on the distance vector-based strategy, so we consider the entire structure as a graph where nodes are the routers, and the links are the networks.
- In a routing table, the first column is the destination, or we can say that it is a network address.
- The cost metric is the number of hops to reach the destination. The number of hops available in a network would be the cost. The hop count is the number of networks required to reach the destination.
- In RIP, infinity is defined as 16, which means that the RIP is useful for smaller networks or small autonomous systems. The maximum number of hops that RIP can contain is 15 hops, i.e., it should not have more than 15 hops as 16 is infinity.
- The next column contains the address of the router to which the packet is to be sent to reach the destination.

Distance Vector Multicast Routing Protocol (DVMRP)

The Distance Vector Multicast Routing Protocol (DVMRP) is the extension of the Routing Information Protocol (RIP) which is used in unicast routing.

It uses the sourcebased tree approach to multicasting.

It is worth mentioning that each router in this protocol that receives a multicast packet to be forwarded implicitly creates a source-based multicast tree in three steps:

1. The router uses an algorithm called reverse path forwarding (RPF) to simulate creating part of the optimal source-based tree between the source and itself.
2. The router uses an algorithm called reverse path broadcasting (RPB) to create a broadcast (spanning) tree whose root is the router itself and whose leaves are all networks in the internet.
3. The router uses an algorithm called reverse path multicasting (RPM) to create a multicast tree by cutting some branches of the tree that end in networks with no member in the group.

Multicast Open Shortest Path First (MOSPF)

Multicast Open Shortest Path First (MOSPF) is the extension of the Open Shortest Path First (OSPF) protocol, which is used in unicast routing.

It also uses the source based tree approach to multicasting.

If the internet is running a unicast link-state routing algorithm, the idea can be extended to provide a multicast link-state routing algorithm.

Each router in the internet has a link-state database (LSDB) that can be used to create a shortest-path tree.

To extend unicasting to multicasting, each router needs to have another database, as with the case of unicast distance-vector routing, to show which interface has an active member in a particular group.

Now a router goes through the following steps to forward a multicast packet received from source S and to be sent to destination G (a group of recipients):

1. The router uses the Dijkstra algorithm to create a shortest-path tree with S as the root and all destinations in the internet as the leaves. The root of the tree is the source of the packet defined in the source address of the packet. The router is capable of creating this tree because it has the LSDB, the whole topology of the internet; the Dijkstra algorithm can be used to create a tree with any root, no matter which router is using it. The point we need to remember is that the shortest-path tree created this way depends on the specific source. For each source we need to create a different tree.
2. The router finds itself in the shortest-path tree created in the first step. In other words, the router creates a shortest-path subtree with itself as the root of the subtree.
3. The shortest-path subtree is actually a broadcast subtree with the router as the root and all networks as the leaves. The router now uses a strategy similar to the one in the case of DVMRP to prune the

broadcast tree and to change it to a multicast tree. The IGMP protocol is used to find the information at the leaf level. MOSPF has added a new type of link state update packet that floods the membership to all routers. The router can use the information it receives in this way and prune the broadcast tree to make the multicast tree.
4. The router can now forward the received packet out of only those interfaces that correspond to the branches of the multicast tree. We need to make certain that a copy of the multicast packet reaches all networks that have active members of the group and that it does not reach those networks that do not.

Protocol Independent Multicast (PIM)

Protocol Independent Multicast (PIM) is the name given to a common protocol that needs a unicast routing protocol for its operation, but the unicast protocol can be either a distance-vector protocol or a link-state protocol.
In other words, PIM needs to use the forwarding table of a unicast routing protocol to find the next router in a path to the destination, but it does not matter how the forwarding table is created.
PIM has another interesting feature: it can work in two different modes: dense and sparse.
The term dense here means that the number of active members of a group in the internet is large; the probability that a router has a member in a group is high. This may happen, for example, in a popular teleconference that has a lot of members.
The term sparse, on the other hand, means that only a few routers in the internet have active members in the group; the probability that a router has a member of the group is low. This may happen, for example, in a very technical teleconference where a number of members are spread somewhere in the internet.
When the protocol is working in the dense mode, it is referred to as PIM-DM; when it is working in the sparse mode, it is referred to as PIM-SM.
Protocol Independent Multicast-Dense Mode (PIM-DM):
1. A router that has received a multicast packet from the source S destined for the group G first uses the RPF strategy to avoid receiving a duplicate of the packet. It consults the forwarding table of the underlying unicast protocol to find the next router if it wants to send a message to the source S (in the reverse direction). If the packet has not arrived from the next router in the reverse direction, it drops the packet and sends a prune message in that direction to prevent receiving future packets related to (S, G).
2. If the packet in the first step has arrived from the next router in the reverse direction, the receiving router forwards the packet from all its interfaces except the one from which the packet has arrived and the interface from which it has already received a prune message related to (S, G). Note that this is actually a broadcasting instead of a multicasting if the packet is the first packet from the source S to group G. However, each router downstream that receives an unwanted packet sends a prune message to the router upstream, and eventually the broadcasting is changed to multicasting. Note that DVMRP behaves differently: it requires that the prune messages (which are part of DV packets) arrive and the tree is pruned before sending any message through unpruned interfaces. PIM-DM does not care about this precaution because it assumes that most routers have an interest in the group (the idea of the dense mode).
Protocol Independent Multicast-Sparse Mode (PIM-SM):
In this environment, the use of a protocol that broadcasts the packets until the tree is pruned is not justified; PIM-SM uses a group-shared tree approach to multicasting. The core router in PIM-SM is called the rendezvous point (RP). Multicast communication is achieved in two steps. Any router that has a multicast packet to send to a group of destinations first encapsulates the multicast packet in a unicast packet (tunneling) and sends it to the RP. The RP then decapsulates the unicast packet and sends the multicast packet to its destination

Distance Vector Multicast Routing Protocol

• The distance vector multicast routing protocol is multicast routing protocol that takes the routing decision based upon the source address of the packet.
• This algorithm constructs the routing tree for a network.
• Whenever a router receives a packet, it forwards it to some of its ports based on the source address of packet.
• The rest of the routing tree is made by downstream routers.
• In this way, routing tree is created from destination to source.
• The protocol must achieve the following tasks:
1. It must prevent the formation of loops in the network.

2. It must prevent the formation of duplicate packets.
3. It must ensure that the path traveled by a packet is the shortest from its source to the router.
4. It should provide dynamic membership.
To accomplish this, the DVMR algorithm uses a process based on following decision making strategies:

1. Reverse Path Forwarding (RPF)
• In this strategy, the router only forwards those packets that have traveled the shortest path from source to destination.
• To achieve this, the router pretends that it has a packet to send to the source from where the packet has arrived.
• In this way, the shortest path to the sender of the packet is computed.
• If the same route is followed by the received packet, it is forwarded to the next router and it is discarded otherwise.
• The reverse path forwarding ensures that the network receives a copy of the packet without formation of loops. A loop occurs when a packet that has left the router may come back again from another interface or the same interface and be forwarded again.
• RPF does not guarantee that there would be no duplicate packets in the network *i.e.* the network may receive two or more copies.
• The reason for this is that the routing is based on the source address and not on the destination address.

2. Reverse Path Broadcasting (RPB)
In order to solve the problem, RPB is used.
• In this method, one parent router is defined for each network.
• The network could accept the multicast packets from this parent router only.
• This router sends packets to those ports for which it is designated as parent.
• Thus, RPB principle' allows a router to broadcast the packet in the network.
This creates duplicate packets on the network and reduces the network efficiency.

3. Reverse Path Multicasting (RPM)
• To overcome the problem of broadcasting in RPB, Reverse Path Multicasting in used.
• In this the desired multicast network tree is created by using two different methods: Pruning and grafting.
• A router can send a prune message to its upstream router whenever it finds that its network is not interested in a multicast packet. In this way a router prunes (cuts) its network from multicasting.
• If a router receives prune message from all the downstream routers, it in turn, sends a prune message to its upstream router.
• A router can also send a graft message to its upstream router if it finds that its network is again interested in receiving the multicast packet. In this way, graft message forces the upstream router to resume sending the multicast message. The network is again grafted (joined).

4. Multicast Open Shortest Path First (MOSPF)
• Multicast open shortest path first is the multicast version of open shortest path first protocol.
• It is an extension of OSPF that uses multicast link state routing method to create source based trees.
• The method used by MOSPF is different from DVMRP.
• The first difference is· that in this method, the tree is least cost tree instead of shortest path tree.
• The second .difference is that the tree is not made gradually. It is made immediately it is prepruned and ready to use.

Routing for Mobile Hosts
Millions of people use computers while on go, from the truly mobile situations with a wireless device in moving cars, to nomadic situations in which laptop computers are used in a series of a different location. We use the term mobile hosts to mean either category, as distinct from stationary hosts that never move. The mobile hosts introduce a new complication to route packets to the mobile hosts, the network first has to find it.

Assumed model :

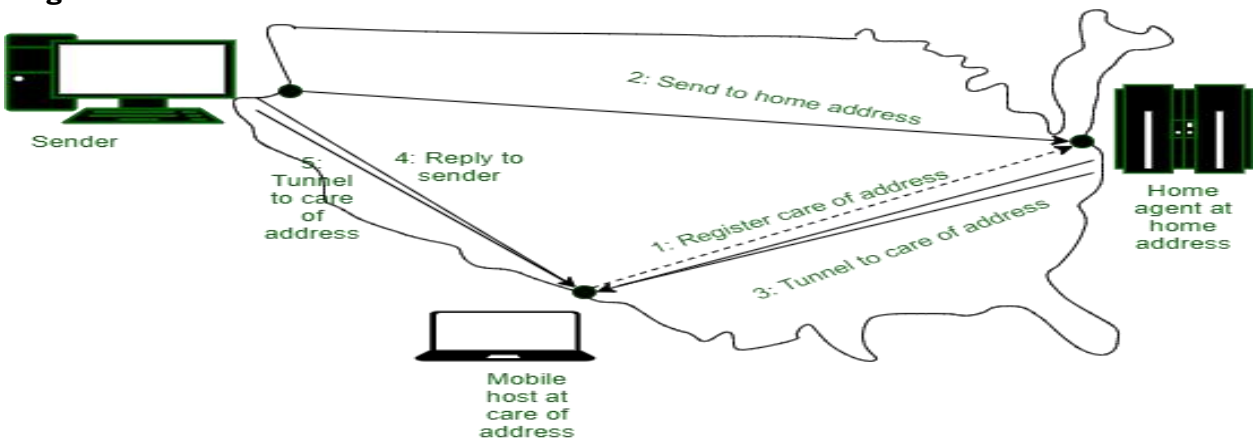
The model of the world that we will consider is one in which all hosts are assumed to have a permanent home location that never changes. Each host has a permanent home address that can be used to determine home location.

Feature :

- The basic idea used for mobile routing on the internet and cellular network is for the mobile hosts to tell the host at the home location.
- This host, which acts on behalf of the mobile host called a home agent.
- Once it knows where the mobile host currently located, it can forward packets so that they are delivered. The figure shows mobile routing in action.
- The local address called care-of address.
- Once it has this address it can tell its home agent where it is now. It does this by sending registration message to home agent with care of address.

Description of Diagram :
The message is shown with a dashed line in the figure indicate that it is a control message, not a data message. The sender sends a data packet to the mobile host using its permanent address. This packet is routed by the network to the host home location because the home addresses belong there. It encapsulates the packet with a new header and sends this bundle to the care-of address. This mechanism is called tunneling. It is very important on the internet

Diagram :



- When the encapsulated packet arrives at the care-of address, the mobile host unwraps it and retrieves the packet from the sender.
- The overall route is called triangle routing because it way is circuitous if the remote location is far from the home location.
- As part of the step, 4 senders learns the current care-of address.
- Subsequent packets can be routed directly to the mobile host by tunneling them to the care-of address (step 5) bypassing the home location.
- If connectivity lost for any reason as the mobile moves, the home address can always be used to reach the mobile.

Routing in AdHoc Networks
There are some extreme cases in which the routers themselves are mobile. Some of the possibilities are Emergency workers at an earthquake that destroyed the infrastructure; A gathering of people with notebook computers in an area lacking 802.11.The following section will give an overview of Ad hoc networks and routing algorithm for it.

Ad hoc networks or MANETs:

- In Such cases, and others, each node consists of a router and a host, usually on the same computer.
- Networks of nodes that just happen to be near each other are called ad hoc networks or MANETs (Mobile Ad hoc NETworks).

- What makes ad hoc networks different from wired networks is that all the usual rules about fixed topologies, fixed and known neighbors, fixed relationship between IP address and location, and more are suddenly tossed out the window.
- Routers can come and go or appear in new places at the drop of a bit. With a wired network, if a router has a valid path to some destination, that path continues to be valid indefinitely (barring a failure somewhere in the system).
- With an ad hoc network, the topology may be changing all the time, so desirability and even validity of paths can change spontaneously, without warning.

Routing algorithm for ad hoc Networks:
A variety of routing algorithms for ad hoc networks have been proposed.

- One of the more interesting ones is the AODV (Ad hoc On-demand Distance Vector) routing algorithm.
- It is a distant relative of the Bellman-Ford distance vector algorithm but adapted to work in a mobile environment and takes into account the limited bandwidth and low battery life found in this environment.
- Another unusual characteristic is that it is an on-demand algorithm, that is, it determines a route to some destination only when somebody wants to send a packet to that destination.

Route Discovery:

- At any instant of time, an ad hoc network can be described by a graph of the nodes (routers hosts).
- Two nodes are connected (i.e., have an arc between them in the graph) if they can communicate directly using their radios.
- Since one of the two may have a more powerful transmitter than the other, it is possible that A is connected to B but B is not connected to A.

Figure 5-20. (a) Range of A's broadcast. (b) After B and D have received A's broadcast. (c) After C, F, and G have received A's broadcast. (d) After E, H, and I have received A's broadcast. The shaded nodes are new recipients. The arrows show the possible reverse routes.

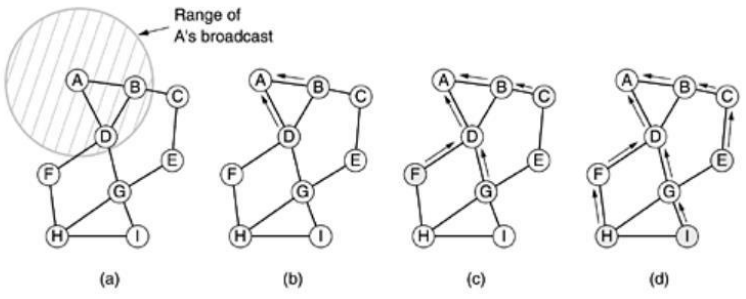
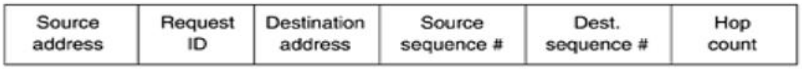


Figure (a) Range of A's broadcast. (b) After B and D have received A's broadcast. (c) After C, F, and G have received A's broadcast. (d) After E, H, and I have received A's broadcast. The shaded nodes are new recipients. The arrows show the possible reverse routes.

- However, for simplicity, we will assume all connections are symmetric.
- To describe the algorithm, consider the ad hoc network of Fig., in which a process at node A wants to send a packet to node I.
- The AODV algorithm maintains a table at each node, keyed by destination, giving information about that destination, including which neighbor to send packets to in order to reach the destination

The format of the ROUTE REQUEST packet:
It contains the source and destination addresses, typically their IP addresses, which identify who is looking for whom. It also contains a Request ID, which is a local counter maintained separately by each node and incremented each time a ROUTE REQUEST is broadcast. Together, the Source address and Request ID fields uniquely identify the ROUTE REQUEST packet to allow nodes to discard any duplicates they may receive.

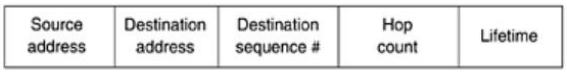
Figure 5-21. Format of a ROUTE REQUEST packet.



In addition to the Request ID counter, each node also maintains a second sequence counter incremented whenever a ROUTE REQUEST is sent (or a reply to someone else's ROUTE REQUEST). It functions a little bit like a clock and is used to tell new routes from old routes. The fourth field of Fig. 5-21 is A's sequence counter; the fifth field is the most recent value of I's sequence number that A has seen (0 if it has never seen it). The use of these fields will become clear shortly. The final field, Hop count, will keep track of how many hops the packet has made. It is initialized to 0.

Format of a ROUTE REPLY packet.
The Source address, Destination address, and Hop count are copied from the incoming request, but the Destination sequence number taken from its counter in memory. The Hop count field is set to 0. The Lifetime field controls how long the route is valid. This packet is unicast to the node that the ROUTE REQUEST packet came from, in this case, G. It then follows the reverse path to D and finally to A. At each node, Hop count is incremented so the node can see how far from the destination (I) it is.

Figure 5-22. Format of a ROUTE REPLY packet.



At each intermediate node on the way back, the packet is inspected. It is entered into the local routing table as a route to I if one or more of the following three conditions are met:
1. No route to I is known.
2. The sequence number for I in the ROUTE REPLY packet is greater than the value in the routing table.
3. The sequence numbers are equal but the new route is shorter.

Route Maintenance
This phase performs the maintenance work of the route as the topology in the mobile ad-hoc network is dynamic in nature and hence, there are many cases of link breakage resulting in the network failure between the mobile nodes.

Node Lookup in Peer-to-Peer Networks
A peer-to-peer networks, is in which a large number of people, usually with permanent wired connections to the Internet, are in contact to share resources. The following section will give an overview of Node Lookup in Peer to peer Network.
Basic Features:

- Peer-to-peer systems are totally distributed systems.
- All nodes are symmetric and there is no central control or hierarchy.
- In a typical peer-to-peer system the users each have some information that may be of interest to other users.
- This information may be free software, (public domain) music, photographs, and so on
- If there are large numbers of users, they will not know each other and will not know where to find what they are looking for.
- One solution is a big central database, but this may not be feasible for some reason (e.g., nobody is willing to host and maintain it).

How a user finds a node that contains what he is looking for in the absence of a centralized database or even a centralized index?

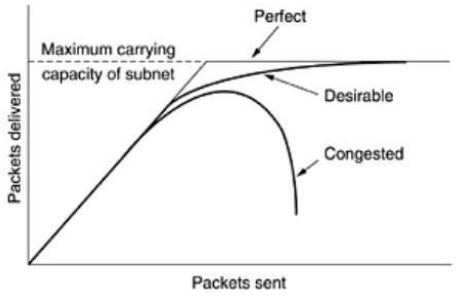
- Let us assume that each user has one or more data items such as songs, photographs, programs, files, and so on that other users might want to read.
- Each item has an ASCII string naming it.
- A potential user knows just the ASCII string and wants to find out if one or more people have copies and, if so, what their IP addresses are.

Congestion Control Algorithms

When too many packets are present in (a part of) the subnet, performance degrades. This situation is called congestion.

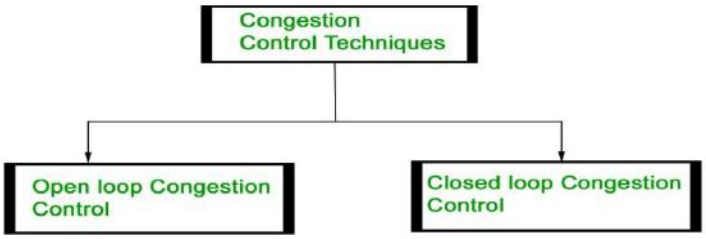
When the number of packets dumped into the subnet by the hosts is within its carrying capacity, they are all delivered (except for a few that are afflicted with transmission errors) and the number delivered is proportional to the number sent. However, as traffic increases too far, the routers are no longer able to cope and they begin losing packets. This tends to make matters worse. At very high traffic, performance collapses completely and almost no packets are delivered.

Figure 5-25. When too much traffic is offered, congestion sets in and performance degrades sharply.



Congestion Prevention Policies

Congestion control refers to the techniques used to control or prevent congestion. Congestion control techniques can be broadly classified into two categories:



Open Loop Congestion Control

Open loop congestion control policies are applied to prevent congestion before it happens. The congestion control is handled either by the source or the destination.

Policies adopted by open loop congestion control –

1. Retransmission Policy :
   It is the policy in which retransmission of the packets are taken care of. If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted. This transmission may increase the congestion in the network.
   To prevent congestion, retransmission timers must be designed to prevent congestion and also able to optimize efficiency.
2. Window Policy :
   The type of window at the sender's side may also affect the congestion. Several packets in the Go-back-n window are re-sent, although some packets may be received successfully at the receiver side. This duplication may increase the congestion in the network and make it worse.
   Therefore, Selective repeat window should be adopted as it sends the specific packet that may have been lost.
3. Discarding Policy :
   A good discarding policy adopted by the routers is that the routers may prevent congestion and at the same time partially discard the corrupted or less sensitive packages and also be able to maintain the quality of a message.
   In case of audio file transmission, routers can discard less sensitive packets to prevent congestion and also maintain the quality of the audio file.
4. Acknowledgment Policy :
   Since acknowledgements are also the part of the load in the network, the acknowledgment policy imposed by the receiver may also affect congestion. Several approaches can be used to prevent

congestion related to acknowledgment.
The receiver should send acknowledgement for N packets rather than sending acknowledgement for a single packet. The receiver should send an acknowledgment only if it has to send a packet or a timer expires.

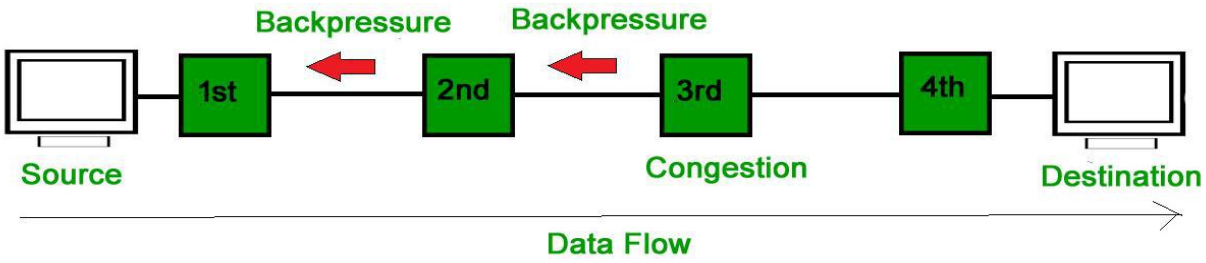
5. Admission Policy :
In admission policy a mechanism should be used to prevent congestion. Switches in a flow should first check the resource requirement of a network flow before transmitting it further. If there is a chance of a congestion or there is a congestion in the network, router should deny establishing a virtual network connection to prevent further congestion.

Closed Loop Congestion Control

Closed loop congestion control techniques are used to treat or alleviate congestion after it happens. Several techniques are used by different protocols; some of them are:

1. Backpressure :

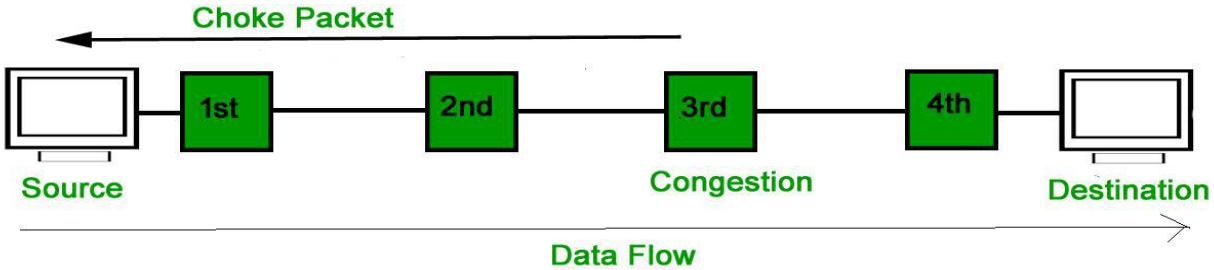
Backpressure is a technique in which a congested node stops receiving packets from upstream node. This may cause the upstream node or nodes to become congested and reject receiving data from above nodes. Backpressure is a node-to-node congestion control technique that propagate in the opposite direction of data flow. The backpressure technique can be applied only to virtual circuit where each node has information of its above upstream node.



In above diagram the 3rd node is congested and stops receiving packets as a result 2nd node may be get congested due to slowing down of the output data flow. Similarly 1st node may get congested and inform the source to slow down.

2. Choke Packet Technique :

Choke packet technique is applicable to both virtual networks as well as datagram subnets. A choke packet is a packet sent by a node to the source to inform it of congestion. Each router monitors its resources and the utilization at each of its output lines. Whenever the resource utilization exceeds the threshold value which is set by the administrator, the router directly sends a choke packet to the source giving it a feedback to reduce the traffic. The intermediate nodes through which the packets has traveled are not warned about congestion.



3. Implicit Signaling :

In implicit signaling, there is no communication between the congested nodes and the source. The source guesses that there is congestion in a network. For example when sender sends several packets and there is no acknowledgment for a while, one assumption is that there is a congestion.

4. Explicit Signaling :

In explicit signaling, if a node experiences congestion it can explicitly sends a packet to the source or destination to inform about congestion. The difference between choke packet and explicit signaling is that the signal is included in the packets that carry data rather than creating a different packet as in case of choke packet technique.

Explicit signaling can occur in either forward or backward direction.

- Forward Signaling : In forward signaling, a signal is sent in the direction of the congestion. The destination is warned about congestion. The receiver in this case adopt policies to prevent further congestion.
- Backward Signaling : In backward signaling, a signal is sent in the opposite direction of the congestion. The source is warned about congestion and it needs to slow down.

Figure 5-26. 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

Congestion Control in Virtual-Circuit Subnets

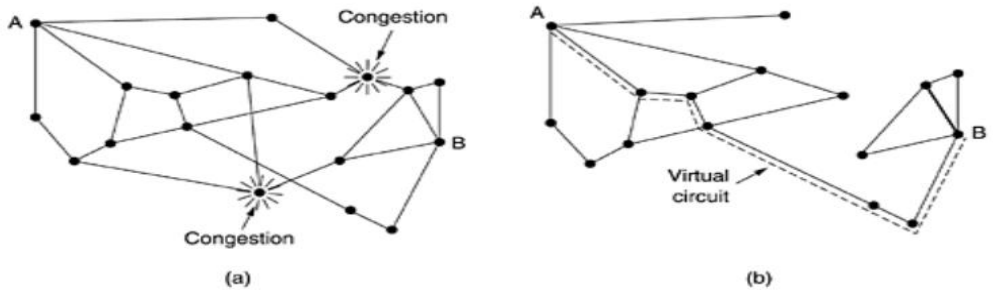
Some approaches to dynamically controlling congestion in virtual circuit subnets:

1. Admission control

Once congestion has been signaled, no more virtual circuits are set up until the problem has gone away. Thus, attempts to set up new transport layer connections fail. Letting more people in just makes matters worse. While this approach is crude, it is simple and easy to carry out. In the telephone system, when a switch gets overloaded, it also practices admission control by not giving dial tones.

2. Another strategy relating to virtual circuits is to negotiate an agreement between the host and subnet when a virtual circuit is set up. This agreement normally specifies the volume and shape of the traffic, quality of service required, and other parameters. To keep its part of the agreement, the subnet will typically reserve resources along the path when the circuit is set up. These resources can include table and buffer space in the routers and bandwidth on the lines. In this way, congestion is unlikely to occur on the new virtual circuits because all the necessary resources are guaranteed to be available.

Figure 5-27. (a) A congested subnet. (b) A redrawn subnet that eliminates the congestion. A virtual circuit from A to B is also shown.



Suppose that a host attached to router *A* wants to set up a connection to a host attached to router *B*. Normally, this connection would pass through one of the congested routers. To avoid this situation, we can redraw the subnet as shown in Fig. 5-27(b), omitting the congested routers and all of their lines. The dashed line shows a possible route for the virtual circuit that avoids the congested routers.

Load Shedding

It is the ultimate medicine for controlling the load. A router overloaded by packets that it can't handle just throws them away.

Load Shedding Technique:

- Random selection in which the packets are randomly selected.
- Depending on the application old packets can be thrown away or new packets can be thrown away.

- Priority can be marked in packets by the source to indicate how important the packet is. Depending on this packet may or may not be selected for throwing away.

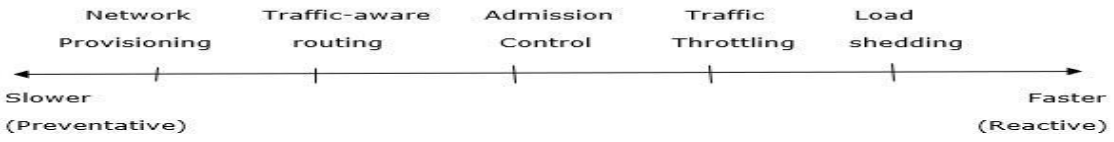
Random Early Detection

The basic concept of RED method is to drop packet before situation becomes hopeless. So, in this method, the situation is tackled beforehand. The router maintains an average queue length for the outgoing lines. When the average queue length exceeds a threshold the line is said to be Congested.

The router then drops the packet at random from the queue and does not report it to the source. The source will notice a lack of acknowledgement and it'll slow down the transmission rate assuming that a lost packet generally occurs due to congestion and discard.

Jitter

Variation or standard deviation in packet arrival time is called jitter. For good quality of service low jitter is expected.



Time scale of approaches to congestion control

Quality of Service

A stream of packets from a source to destination is called a flow. Quality of Service is defined as something a flow seeks to attain. In connection oriented network, all the packets belonging to a flow follow the same order. In a connectionless network, all the packets may follow different routes.

The needs of each flow can be characterized by four primary parameters :

- Reliability, Lack of reliability means losing a packet or acknowledgement which entertains retransmission.
- Delay, Increase in delay means destination will find the packet later than expected, Importance of delay changes according to the various application.
- Jitter, Variation of the delay is jitter, If the delay is not at a constant rate, it may result in poor quality.
- Bandwidth, Increase in bandwidth means increase in the amount of data which can be transferred in given amount of time, Importance of bandwidth also varies according to various application.

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

Techniques for achieving good Quality of Service :

1. Overprovisioning – The logic of overprovisioning is to provide greater router capacity, buffer space and bandwidth. It is an expensive technique as the resources are costly. Eg: Telephone System.
2. Buffering – Flows can be buffered on the receiving side before being delivered. It will not affect reliability or bandwidth, but helps to smooth out jitter. This technique can be used at uniform intervals.
   i. Traffic Shaping **–** It is defined as about regulating the average rate of data transmission. It smooths the traffic on server side other than client side. When a connection is set up, the user machine and subnet agree on a certain traffic pattern for that circuit called as *Service Level Agreement.* It reduces congestion and thus helps the carrier to deliver the packets in the agreed pattern. Traffic

shaping helps to regulate the rate of data transmission and reduces congestion. There are 2 types of traffic shaping algorithms: Leaky Bucket and Token Bucket

The **leaky bucket algorithm** is a method of temporarily storing a variable number of requests and organizing them into a set-rate output of packets. This basic concept is applied in the case of the Leaky Bucket Algorithm which is nothing but a single server queueing system with constant service time.
Consider a bucket with a hole at the bottom. Whatever the rate at which water enters the bucket, it leaks out of the bucket at a constant rate through the hole. The rate of flow is zero if there is no water in the bucket and if the bucket is full then the excess water spills over and it is lost.

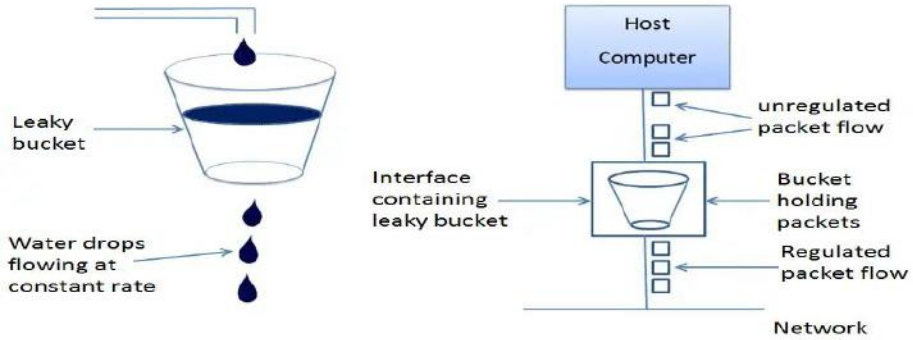


Fig: Leaky Bucket Algorithm

Basic Features of Leaky Bucket Algorithm:
1. An interface containing a leaky bucket is connect each host to the network which is a finite internal queue.
2. When space is available in a queue, a packet will send from an application in store.
3. When the queue is full and a new packet will send from an application is discarded.
4. The host operating system builds or simulates this arrangement in the hardware.
5. The packets are queuing and releasing at regular intervals with the regular amount and that reduces the chances of congestion.
Problem: As the packets carry an uneven amount of data, The packets on the network is generates the actual loading effect that is also uneven.

Byte Counting Leaky Bucket Algorithm:
1. At every clock tick, a counter is initialized to a value n.
2. When the first packet in the queue –
if length < n then transmitter counter = n – length of the first packet
3. When additional packets are in the queue – sent along with the first packet if total length<n
4. For the first packet-
if length = n then transmitted
No other packet can be transmitted till the next clock tick.
5. In the next clock, the tick counter is initialized to n.

The Token Bucket Algorithm

- The leaky bucket algorithm has a rigid output design at an average rate independent of the bursty traffic.
- In some applications, when large bursts arrive, the output is allowed to speed up. This calls for a more flexible algorithm, preferably one that never loses information. Therefore, a token bucket algorithm finds its uses in network traffic shaping or rate-limiting.
- It is a control algorithm that indicates when traffic should be sent. This order comes based on the display of tokens in the bucket.
- The bucket contains tokens. Each of the tokens defines a packet of predetermined size. Tokens in the bucket are deleted for the ability to share a packet.
- When tokens are shown, a flow to transmit traffic appears in the display of tokens.
- No token means no flow sends its packets. Hence, a flow transfers traffic up to its peak burst rate in good tokens in the bucket.

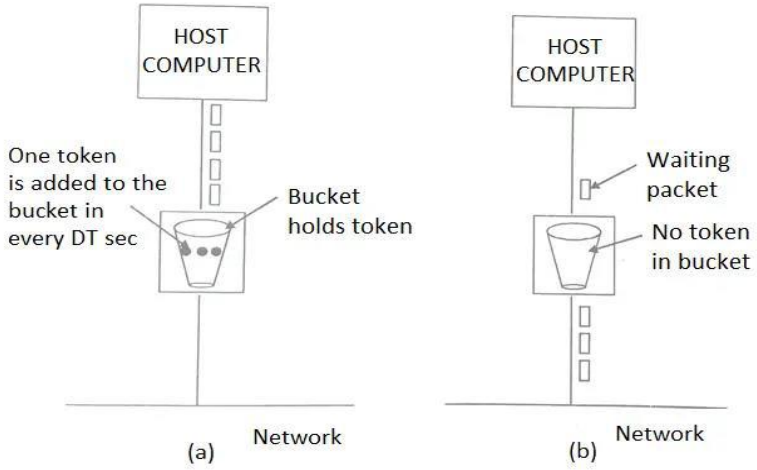


Fig: Token Bucket System (a) Before tokens are captured (b) After tokens are captured

Need of token bucket Algorithm:-

The leaky bucket algorithm enforces output pattern at the average rate, no matter how bursty the traffic is. So in order to deal with the bursty traffic we need a flexible algorithm so that the data is not lost. One such algorithm is token bucket algorithm.

Steps of this algorithm can be described as follows:

1. In regular intervals tokens are thrown into the bucket. *f*
2. The bucket has a maximum capacity. *f*
3. If there is a ready packet, a token is removed from the bucket, and the packet is sent.
4. If there is no token in the bucket, the packet cannot be sent.

The implementation of the basic token bucket algorithm is just a variable that counts tokens.
Counter incremented by one in every Δ second.
The counter is decremented when the packet is sent.
If counter = 0 then no packet is sent.
In the case of the byte-count version:

(i) The counter is incremented by k bytes in every Δ second.(Read as Delta Second)
(ii) The counter is decremented by the length of the packet sent.
(iii) If counter = 0 then no packet is sent.

Length of the burst is a design issue:

Let S = burst length(sec) and C = token bucket capacity(byte)
r = token arrival rate(byte/sec)
M = maximum output rate(byte/sec)
MS = C + rS
S = C/(M-r)

Resource Reservation

Three different kinds of resources can potentially be reserved: 1. Bandwidth. 2. Buffer space. 3. CPU cycles.

Bandwidth: 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.

Buffer space: When a packet arrives, it is usually deposited on the network interface card by the hardware itself. The router software then has to copy it to a buffer in RAM and queue that buffer for transmission on the chosen outgoing line. If no buffer is available, the packet has to be discarded since there is no place to put it. For a good quality of service, some buffers can be reserved for a specific flow so that flow does not have to compete for buffers with other flows. There will always be a buffer available when the flow needs one, up to some maximum.

CPU cycles: It takes router CPU time to process a packet, so a router can process only a certain number of packets per second. Making sure that the CPU is not overloaded is needed to ensure timely processing of each packet.

Admission Control
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.

Figure 5-35. An example flow specification.

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

1. The Token bucket rate, is the number of bytes per second that are put into the bucket. This is the maximum sustained rate the sender may transmit, averaged over a long time interval.
2. The second parameter is the size of the bucket in bytes. If, for example, the Token bucket rate is 1 Mbps and the Token bucket size is 500 KB, the bucket can fill continuously for 4 sec before it fills up (in the absence of any transmissions). Any tokens sent after that are lost.
3. The third parameter, the Peak data rate, is the maximum tolerated transmission rate, even for brief time intervals. The sender must never exceed this rate.
4. The last two parameters specify the minimum and maximum packet sizes, including the transport and network layer headers (e.g., TCP and IP).

Proportional Routing
- To provide a higher quality of service is to split the traffic for each destination over multiple paths.
- To divide the traffic equally or in proportion to the capacity of the outgoing links.

Packet Scheduling
The fair queueing algorithm - Routers have separate queues for each output line, one for each flow. When a line becomes idle, the router scans the queues round robin, taking the first packet on the next queue. In this way, with n hosts competing for a given output line, each host gets to send one out of every n packets. Sending more packets will not improve this fraction.

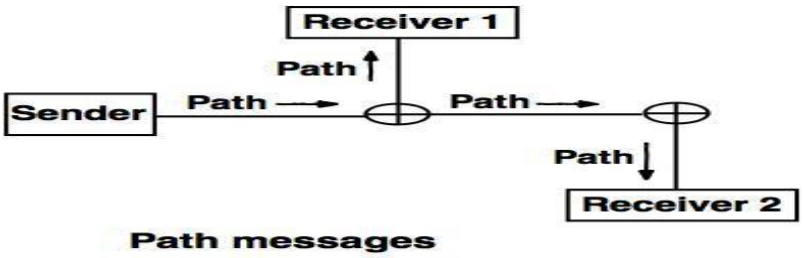
Weighted fair queueing (WFQ) is the data packet queuing algorithm used by network schedulers. This strategy consists of implementations of generalized processor sharing policy (GPS), and a natural generalization of fair queuing (FQ). WFQ lets each flow have a certain ration of link capacity, which is usually specified by the flow itself. Weighted fair queueing is also known as packet-by-packet GPS (PGPS or P-GPS).

Integrated Services
- Aimed at both unicast and multicast applications.

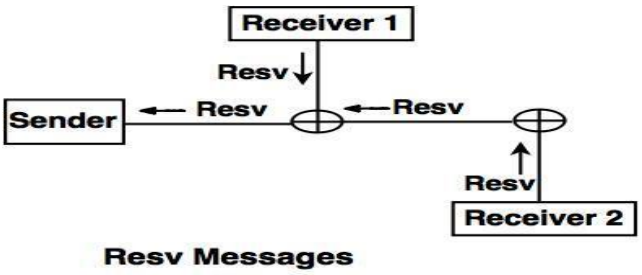
RSVP — The Resource reSerVation Protocol
- The RSVP is a signaling protocol, which helps IP to create a flow and to make resource reservation.
- It is an independent protocol and also can be used in other different model.
- RSVP helps to design multicasting (one to many or many to many distribution), where a data can be sent to group of destination computers simultaneously.
  For example: The IP multicast is technique for one to many communication through an IP infrastructure in the network.
- RSVP can be also used for unicasting (transmitting a data to all possible destination) to provide resource reservation for all types of traffic.

The two important types of RSVP messages are:
1. Path messages:

- The receivers in a flow make the reservation in RSVP, but the receivers do not know the path traveled by the packets before the reservation. The path is required for reservation To solve this problem the RSVP uses the path messages.
- A path message travels from the sender and reaches to all receivers by multicasting and path message stores the necessary information for the receivers.



2. Resv messages:
After receiving path message, the receiver sends a Resv message. The Resv message travels to the sender and makes a resource reservation on the routers which supports for RSVP.



Differentiated services (DiffServ or DS)
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:

- IETF has standardized architecture for it, called differentiated services, which is described in RFCs 2474, 2475, and numerous others.
- Differentiated services (DS) can be offered by a set of routers forming an administrative domain (e.g., an ISP or a Telco).
- The administration defines a set of service classes with corresponding forwarding rules.
- If a customer signs up for DS, customer packets entering the domain may carry a Type of Service field in them, with better service provided to some classes (e.g., premium service) than to others.
- Traffic within a class may be required to conform to some specific shape, such as a leaky bucket with some specified drain rate.
- An operator with a nose for business might charge extra for each premium packet transported or might allow up to N premium packets per month for a fixed additional monthly fee.
- This makes DS relatively easy to implement.
- Class-based service also occurs in other industries. For example, package delivery companies often offer overnight, two-day, and three-day service.
- For packets, the classes may differ in terms of delay, jitter, and probability of being discarded in the event of congestion, among other possibilities (but probably not roomier Ethernet frames).
- To make the difference between flow-based quality of service and class-based quality of service clearer, 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 file transfer class or other classes, but no telephone call gets any private resources reserved for it alone.

Expedited Forwarding

- The simplest class is expedited forwarding.
- The idea behind expedited forwarding is very simple.
- Two classes of service are available: regular and expedited.
- The vast majority of the traffic is expected to be regular, but a small fraction of the packets are expedited.
- The expedited packets should be able to transit the subnet as though no other packets were present.
- A symbolic representation of this "two-tube" system is given in Fig.3.31.
- The two logical pipes shown in the figure represent a way to reserve bandwidth, not a second physical line.

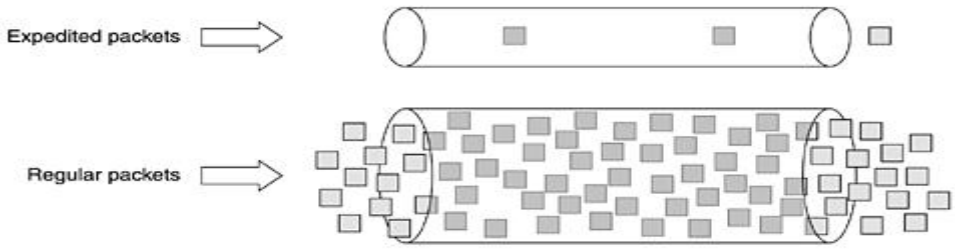


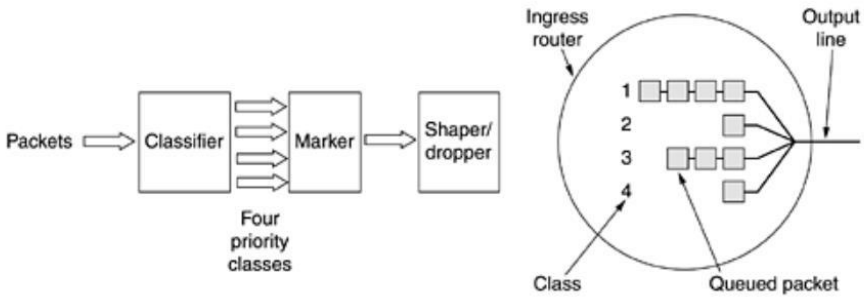
Figure: Expedited packets experience a traffic-free network.

- One way to implement this strategy is to program the routers to 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.
- Packet scheduling use something like weighted fair queuing.
- For example, if 10% of the traffic is expedited and 90% is regular, 20% of the bandwidth could be dedicated to expedited traffic and the rest to regular traffic.
- Doing so would give the expedited traffic twice as much bandwidth as it needs in order to provide low delay for it.
- This allocation can be achieved by transmitting one expedited packet for every four regular packets (assuming the size distribution for both classes is similar).
- In this way, it is hoped that expedited packets see an unloaded subnet, even when there is, in fact, a heavy load.

Assured Forwarding

A somewhat more elaborate scheme for managing the service classes is called assured forwarding. It is described in RFC 2597. It specifies that there shall be four priority classes, each class having its own resources. In addition, it defines three discard probabilities for packets that are undergoing congestion: low, medium, and high. Taken together, these two factors define 12 service classes.

Figure 5-40. A possible implementation of the data flow for assured forwarding.



Step 1 is to classify the packets into one of the four priority classes. This step might be done on the sending host (as shown in the figure) or in the ingress (first) router.

Step 2 is to mark the packets according to their class. A header field is needed for this purpose. Fortunately, an 8-bit Type of service field is available in the IP header, as we will see shortly. RFC 2597 specifies that six of these bits are to be used for the service class, leaving coding room for historical service classes and future ones.

Step 3 is to pass the packets through a shaper/dropper filter that may delay or drop some of them to shape the four streams into acceptable forms, for example, by using leaky or token buckets. If there are too many packets, some of them may be discarded here, by discard category. More elaborate schemes involving metering or feedback are also possible.

Label Switching and MPLS

Several router vendors have worked and focused on adding a label in front of each packet and doing the routing based on the label rather than on the destination address. Making the label an index into an internal table makes finding the correct output line becomes just a matter of table lookup. Using this technique, routing can be done very quickly and any necessary resources can be reserved along the path. The following section explains the MPLS in details.

MPLS (Multiprotocol Label Switching):

- This "new" switching idea goes by various (proprietary) names, including label switching and tag switching. Eventually, IETF began to standardize the idea under the name MPLS (Multiprotocol Label Switching).
- Since IP packets were not designed for virtual circuits, there is no field available for virtual-circuit numbers within the IP header.
- For this reason, a new MPLS header had to be added in front of the IP header.
- On a router-to-router line using PPP as the framing protocol, the frame format, including the PPP, MPLS, IP, and TCP headers, is as shown in Figure.

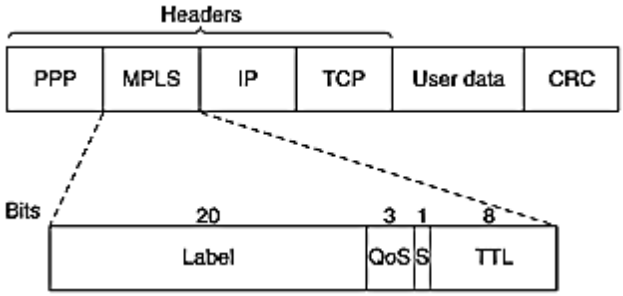


Figure: Transmitting a TCP segment using IP, MPLS, and PPP.

- The generic MPLS header has four fields, the most important of which is the Label field, which holds the index. The QoS field indicates the class of service.
- The **S** field relates to stacking multiple labels in hierarchical networks (discussed below). If it hits 0, the packet is discarded.
- This feature prevents infinite looping in the case of routing instability.

How it works?

- When an MPLS-enhanced packet (or cell) arrives at an MPLS-capable router, the label is used as an index into a table to determine the outgoing line to use and also the new label to use.
- This label swapping is used in all virtual-circuit subnets because labels have only local significance and two different routers can feed unrelated packets with the same label into another router for transmission on the same outgoing line.
- To be distinguishable at the other end, labels have to be remapped at every hop.
- With MPLS, the packets still contain their final destination address, in addition to the label, so that at the end of the labeled route the label header can be removed and forwarding can continue the usual way, using the network layer destination address.

How the forwarding table is constructed?

- There are two ways for the forwarding table entries to be created.

a.) Data Driven approach:

- In the data-driven approach, when a packet arrives, the first router it hits contacts the router downstream where the packet has to go and asks it to generate a label for the flow.
- This method is applied recursively.
- The protocols that do this spreading are very careful to avoid loops.
- They often use a technique called colored threads.
- The backward propagation of an FEC can be compared to pulling a uniquely colored thread back into the subnet.
- If a router ever sees a color it already has, it knows there is a loop and takes remedial action.

- The data-driven approach is primarily used on networks in which the underlying transport is ATM (such as much of the telephone system).

b.) Control Driven approach:

- The other way, used on networks not based on ATM, is the control-driven approach.
- It has several variants.
- One of these works like this.
- When a router is booted, it checks to see for which routes it is the final destination (e.g., which hosts are on its LAN).
- It then creates one or more FECs for them, allocate a label for each one, and pass the labels to its neighbors.
- They, in turn, enter the labels in their forwarding tables and send new labels to their neighbors, until all the routers have acquired the path.
- Resources can also be reserved as the path is constructed to guarantee an appropriate quality of service.

Internetworking

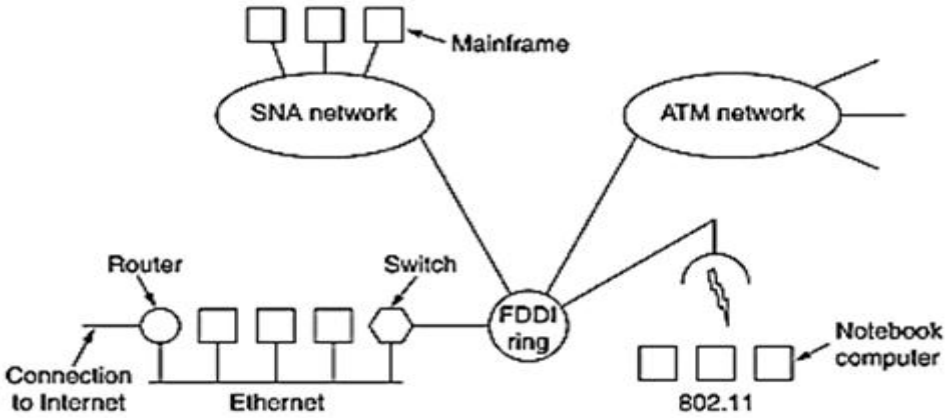
Many different networks exist, including LANs, MANs, and WANs. Numerous protocols are in widespread use in every layer. The following sections we will take a careful look at the issues that arise when two or more networks are connected to form an internet.

Different Networks and Protocols:

- Having different networks invariably means having different protocols.
- We believe that a variety of different networks (and thus protocols) will always be around, for the following reasons.

i. First of all, the installed base of different networks is large.
ii. Nearly all personal computers run TCP/IP.
iii. Many large businesses have mainframes running IBM's SNA.
iv. A substantial number of telephone companies operate ATM networks.
v. Some personal computer LANs still use Novell NCP/IPX or AppleTalk.
vi. Finally, wireless is an up-and-coming area with a variety of protocols.

- In the future, it may be commonplace for the telephone, the television set, and other appliances all to be networked together so that they can be controlled remotely.
- This new technology will undoubtedly bring new networks and new protocols.



- As an example of how different networks might be connected.
- Here we see a corporate network with multiple locations tied together by a wide area ATM network.
- At one of the locations, an FDDI optical backbone is used to connect an Ethernet, an 802.11 wireless LAN, and the corporate data center's SNA mainframe network.
- The purpose of interconnecting all these networks is to allow users on any of them to communicate with users on all the other ones and also to allow users on any of them to access data on any of them.
- Accomplishing this goal means sending packets from one network to another.

How Networks Differ

Networks can differ in many ways. Some of the differences, such as different modulation techniques or frame formats, are in the physical and data link layers, these differences will not concern us here.
Some of the many ways networks can differ:

- It is papering over these differences that makes internetworking more difficult than operating within a single network.

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

Figure: some of the many ways networks can differ

Problems between Networks interface:
When packets sent by a source on one network must transit one or more foreign networks before reaching the destination network many problems can occur at the interfaces between networks.

- To start with, when packets from a connection-oriented network must transit a connectionless one, they may be reordered, something the sender does not expect and the receiver is not prepared to deal with.
- Protocol conversions will often be needed, which can be difficult if the required functionality cannot be expressed.
- Address conversions will also be needed, which may require some kind of directory system.
- Passing multicast packets through a network that does not support multicasting requires generating separate packets for each destination.
- The differing maximum packet sizes used by different networks can be a major nuisance.
- Differing qualities of service is an issue when a packet that has real-time delivery constraints passes through a network that does not offer any real-time guarantees.
- Error, flow, and congestion control often differ among different networks.
- If the source and destination both expect all packets to be delivered in sequence without error but an intermediate network just discards packets whenever it smells congestion on the horizon, many applications will break.
- Also, if packets can wander around aimlessly for a while and then suddenly emerge and be delivered, trouble will occur if this behavior was not anticipated and dealt with.
- Different security mechanisms, parameter settings, and accounting rules, and even national privacy laws also can cause problems

How Networks Can Be Connected

Networks can be interconnected by different devices. The following section will explain the network connection in different way.
Network connection in Different layer:

- In the physical layer, networks can be connected by repeaters or hubs, which just move the bits from one network to an identical network.
    - These are mostly analog devices and do not understand anything about digital protocols (they just regenerate signals).
- In the data link layer we find bridges and switches, which operates.
    - They can accept frames, examine the MAC addresses, and forward the frames to a different network while doing minor protocol translation in the process..
- In the network layer, we have routers that can connect two networks.

- If two networks have dissimilar network layers, the router may be able to translate between the packet formats, although packet translation is now increasingly rare.
- A router that can handle multiple protocols is called a multiprotocol router.

- In the transport layer we find transport gateways, which can interface between two transport connections.
  - For example, a transport gateway could allow packets to flow between a TCP network and an SNA network, which has a different transport protocol, by essentially gluing a TCP connection to an SNA connection.
- In the application layer, application gateways translate message semantics.
  - As an example, gateways between Internet e-mail (RFC 822) and X.400 e-mail must parse the e-mail messages and change various header fields.

How interworking in Network layer differs from switching in data link layer?

- In Fig., consider the source machine, S, wants to send a packet to the destination machine, D. These machines are on different Ethernets, connected by a switch. **S** encapsulates the packet in a frame and sends it on its way. The frame arrives at the switch, which then determines that the frame has to go to LAN 2 by looking at its MAC address. The switch just removes the frame from LAN 1 and deposits it on LAN 2.

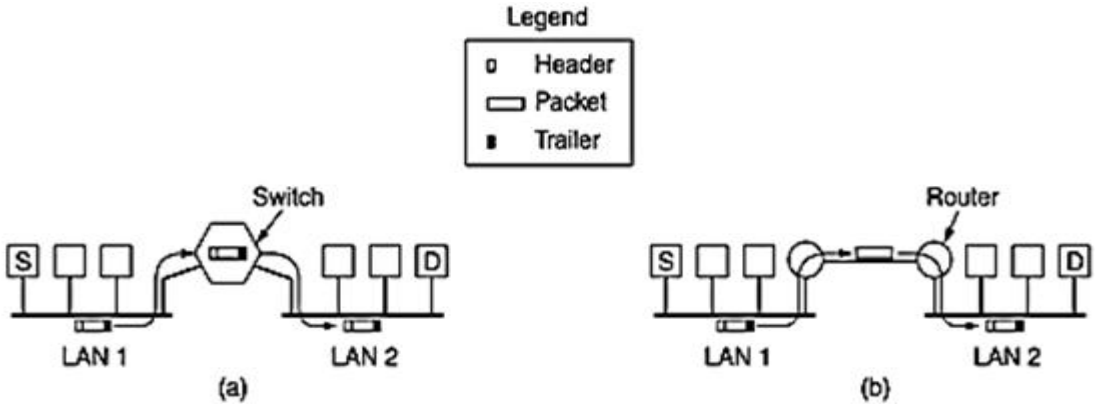


Figure: (a) Two Ethernets connected by a switch. (b) Two Ethernets connected by routers.

Now let us consider the same situation but with the two Ethernets connected by a pair of routers instead of a switch. The routers are connected by a point-to-point line, possibly a leased line thousands of kilometers long. Now the frame is picked up by the router and the packet removed from the frame's data field. The router examines the address in the packet (e.g., an IP address) and looks up this address in its routing table. Based on this address, it decides to send the packet to the remote router, potentially encapsulated in a different kind of frame, depending on the line protocol. At the far end, the packet is put into the data field of an Ethernet frame and deposited onto LAN 2.

An essential difference between the switched (or bridged) case and the routed case is this. With a switch (or bridge), the entire frame is transported on the basis of its MAC address. With a router, the packet is extracted from the frame and the address in the packet is used for deciding where to send it. Switches do not have to understand the network layer protocol being used to switch packets. Routers do.

Concatenated Virtual Circuits

Two styles of internetworking are possible: a connection-oriented concatenation of virtual-circuit subnets, and a datagram internet style.

Concatenated Virtual Circuit Model:

- In the concatenated virtual-circuit model, shown in Fig. 3.4, a connection to a host in a distant network is set up in a way similar to the way connections are normally established.
- The subnet sees that the destination is remote and builds a virtual circuit to the router nearest the destination network. Then it constructs a virtual circuit from that router to an external gateway (multiprotocol router).
- This gateway records the existence of the virtual circuit in its tables and proceeds to build another virtual circuit to a router in the next subnet.
- This process continues until the destination host has been reached.

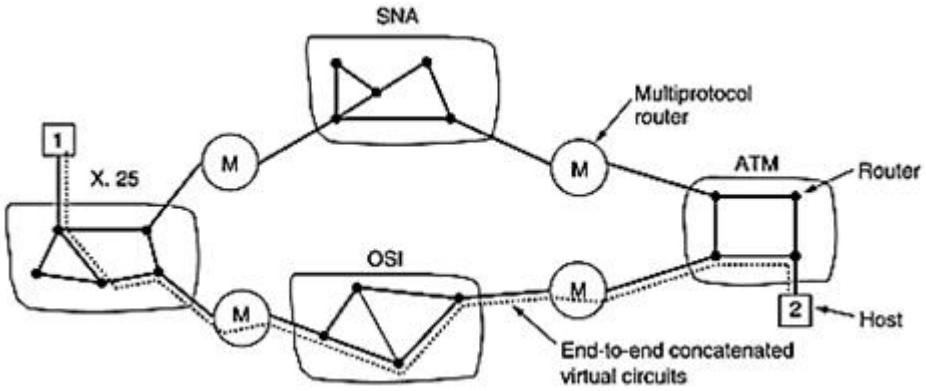


Figure: Internetworking using concatenated virtual circuits.

- Once data packets begin flowing along the path, each gateway relays incoming packets, converting between packet formats and virtual-circuit numbers as needed.
- Clearly, all data packets must traverse the same sequence of gateways
- The essential feature of this approach is that a sequence of virtual circuits is set up from the source through one or more gateways to the destination.
- Each gateway maintains tables telling which virtual circuits pass through it, where they are to be routed, and what the new virtual-circuit number is.
- This scheme works best when all the networks have roughly the same properties.

Connectionless Internetworking

The alternative internetwork model is the datagram model. The following section will give an overview of this model.

Datagram Model:

- In this model, the only service the network layer offers to the transport layer is the ability to inject datagrams into the subnet and hope for the best.
- There is no notion of a virtual circuit at all in the network layer, let alone a concatenation of them.
- This model does not require all packets belonging to one connection to traverse the same sequence of gateways.
- In Figure datagrams from host 1 to host 2 are shown taking different routes through the internetwork.
- A routing decision is made separately for each packet, possibly depending on the traffic at the moment the packet is sent.
- This strategy can use multiple routes and thus achieve a higher bandwidth than the concatenated virtual-circuit model.
- On the other hand, there is no guarantee that the packets arrive at the destination in order, assuming that they arrive at all.

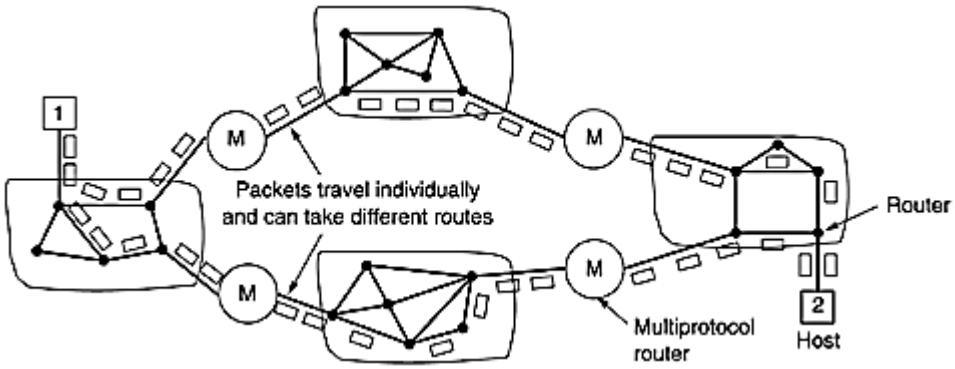


Figure: A connectionless internet

Problems with the Model:

- If each network has its own network layer protocol, it is not possible for a packet from one network to transit another one.

- One could imagine the multiprotocol routers actually trying to translate from one format to another, but unless the two formats are close relatives with the same information fields, such conversions will always be incomplete and often doomed to failure.
- For this reason, conversion is rarely attempted.

- A second, and more serious, problem is addressing.
  - Imagine a simple case: a host on the Internet is trying to send an IP packet to a host on an adjoining SNA network.
  - The IP and SNA addresses are different.
  - One would need a mapping between IP and SNA addresses in both directions.
- Furthermore, the concept of what is addressable is different.
  - In IP, hosts (actually, interface cards) have addresses.
  - In SNA, entities other than hosts (e.g., hardware devices) can also have addresses.
  - At best, someone would have to maintain a database mapping everything to everything to the extent possible, but it would constantly be a source of trouble.

Properties of datagram approach:

- The properties of the datagram approach to internetworking are pretty much the same as those of datagram subnets:
  - more potential for congestion,
  - but also more potential for adapting to it,
  - robustness in the face of router failures,
  - And longer headers needed.

Various adaptive routing algorithms are possible in an internet, just as they are within a single datagram network.

Advantage: A major advantage of the datagram approach to internetworking is that it can be used over subnets that do not use virtual circuits inside.

Tunneling

There is a case where the source and destination hosts are on the same type of network, but there is a different network in between.

Overview of Tunneling:

- As an example, assume an international bank with a TCP/IP-based Ethernet in Paris, a TCP/IP-based Ethernet in London, and a non-IP wide area network (e.g., ATM) in between.

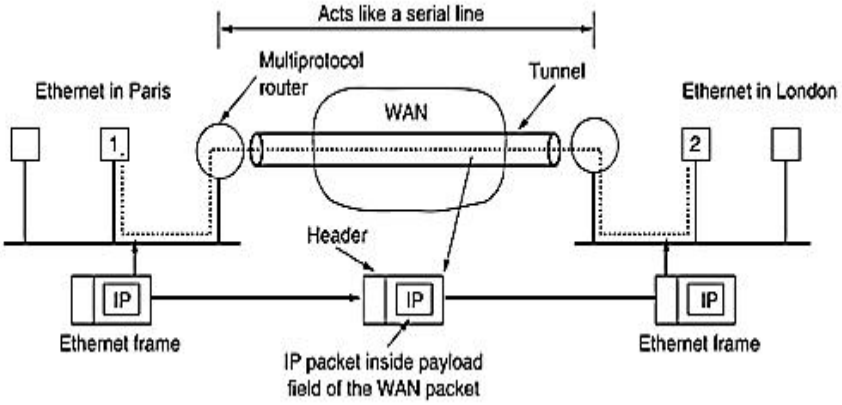


Figure Tunneling a packet from Paris to London

- The solution to this problem is a technique called tunneling.
- To send an IP packet to host 2, host 1 constructs the packet containing the IP address of host 2, inserts it into an Ethernet frame addressed to the Paris multiprotocol router, and puts it on the Ethernet.
- When the multiprotocol router gets the frame, it removes the IP packet, inserts it in the payload field of the WAN network layer packet, and addresses the latter to the WAN address of the London multiprotocol router.

- When it gets there, the London router removes the IP packet and sends it to host 2 inside an Ethernet frame.
- The WAN can be seen as a big tunnel extending from one multiprotocol router to the other.
- The IP packet just travels from one end of the tunnel to the other, snug in its nice box.

Analogy for Tunneling:

- Consider a person driving her car from Paris to London.
- Within France, the car moves under its own power, but when it hits the English Channel, it is loaded into a high-speed train and transported to England through the Chunnel (cars are not permitted to drive through the Chunnel).
- Effectively, the car is being carried as freight.
- At the far end, the car is let loose on the English roads and once again continues to move under its own power. Tunneling of packets through a foreign network works the same way.

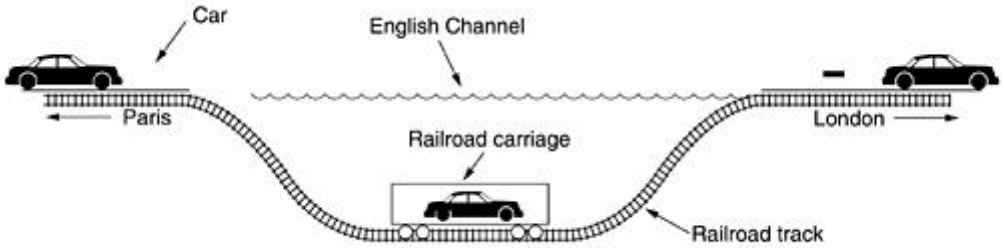


Figure: Tunneling a car from France to England

Internetwork Routing

Routing through an internetwork is similar to routing within a single subnet, but with some added complications. The following section will explain the Internetwork Routing in detail.

Overview of the Internetwork Routing:

- Consider the internetwork of Figure in which five networks are connected by six (possibly multiprotocol) routers.
- Making a graph model of this situation is complicated by the fact that every router can directly access (i.e., send packets to) every other router connected to any network to which it is connected.
- For example, **B** in Fig. can directly access A and C via network 2 and also D via network 3.
- This leads to the graph of Fig.(b).

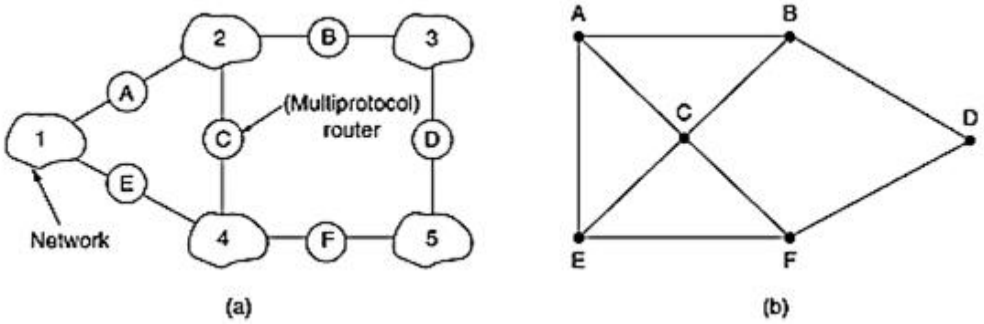


Figure: (a) An internetwork. (b) A graph of the internetwork

- Once the graph has been constructed, known routing algorithms, such as the distance vector and link state algorithms, can be applied to the set of multiprotocol routers.
- This gives a two-level routing algorithm: within each network an interior gateway protocol is used, but between the networks, an exterior gateway protocol is used ("gateway" is an older term for "router").
- In fact, since each network is independent, they may all use different algorithms.

How a packet reaches its destination?

- Each network in internetwork is often referred to as an Autonomous System (AS).
- A typical internet packet starts out on its LAN addressed to the local multiprotocol router (in the MAC layer header).
- After it gets there, the network layer code decides which multiprotocol router to forward the packet to, using its own routing tables.

- If that router can be reached using the packet's native network protocol, the packet is forwarded there directly.
- Otherwise it is tunneled there, encapsulated in the protocol required by the intervening network. This process is repeated until the packet reaches the destination network.

Fragmentation

It is an important function of network layer. It is technique in which gateways break up or divide larger packets into smaller ones called fragments. Each fragment is then sent as a separate internal packet. Each fragment has its separate header and trailer.

Sometimes, a fragmented datagram can also get fragmented further when it encounters a network that handles smaller fragments. Thus, a datagram can be fragmented several times before it reaches final destination. Reverse process of the fragmentation is difficult. Reassembling of fragments is usually done by the destination host because each fragment has become an independent datagram.

There are two different strategies for the recombination or we can say reassembly of fragments : Transparent Fragmentation, and Non-Transparent Fragmentation.

1. Transparent Fragmentation:

This fragmentation is done by one network is made transparent to all other subsequent networks through which packet will pass. Whenever a large packet arrives at a gateway, it breaks the packet into smaller fragments as shown in the following figure i.e the gateway G1 breaks a packet into smaller fragments.

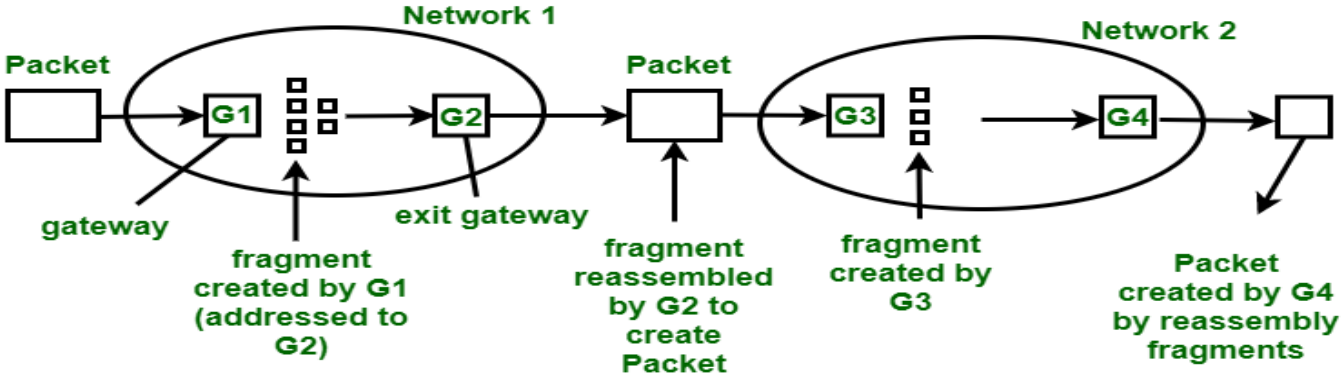


Figure – Transparent Fragmentation

After this, each fragment is going to address to same exit gateway. Exit gateway of a network reassembles or recombines all fragments as shown in above figure. The exit gateway, G2 of network 1 recombines all fragments created by G1 before passing them to network 2. Thus, subsequent network is not aware that fragmentation has occurred. This type of strategy is used by ATM networks . These networks use special hardware that provides transparent fragmentation of packets.

There are some disadvantages of transparency strategy which are as follows :

- Exit fragment that recombines fragments in a network must known when it has received all fragments.
- Some fragments chooses different gateways for exit that results in poor performance.
- It adds considerable overhead in repeatedly fragmenting and reassembling large packet.

2. Non-Transparent Fragmentation:

This fragmentation is done by one network is non-transparent to the subsequent networks through which a packet passes. Packet fragmented by a gateway of a network is not recombined by exit gateway of same network as shown in the below figure.

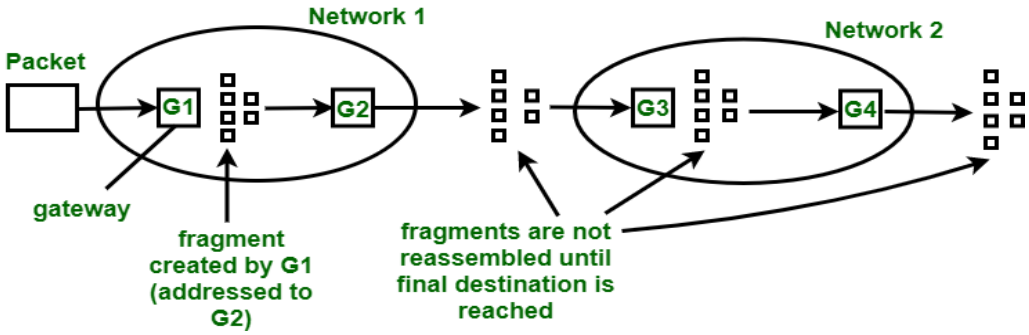


Figure – Non-transparent Fragmentation

Once a packet is fragmented, each fragment is treated as original packet. All fragments of a packet are passed through exit gateway and recombination of these fragments is done at the destination host.

Advantages of Non-Transparent Fragmentation is as follows :

- We can use multiple exit gateways and can improve the network performance.
- It has a higher throughput

Disadvantages of Non-Transparent Fragmentation is as follows :

- Every host has capability of reassembling fragments.
- When a packet is fragmented, fragments should be numbered in such a way that the original data stream can be reconstructed.
- Total overhead increases due to fragmentation as each fragment must have its own header.

The Network Layer in the Internet

The Internet can be viewed as a collection of subnetworks or Autonomous Systems (ASes) that are interconnected at the Network layer.

Quasi-hierarchical organization:

- There is no real structure, but several major backbones exist.
- These are constructed from high-bandwidth lines and fast routers.
- Attached to the backbones are regional (midlevel) networks, and attached to these regional networks are the LANs at many universities, companies, and Internet service providers.
- A sketch of this quasi-hierarchical organization is given in Fig.

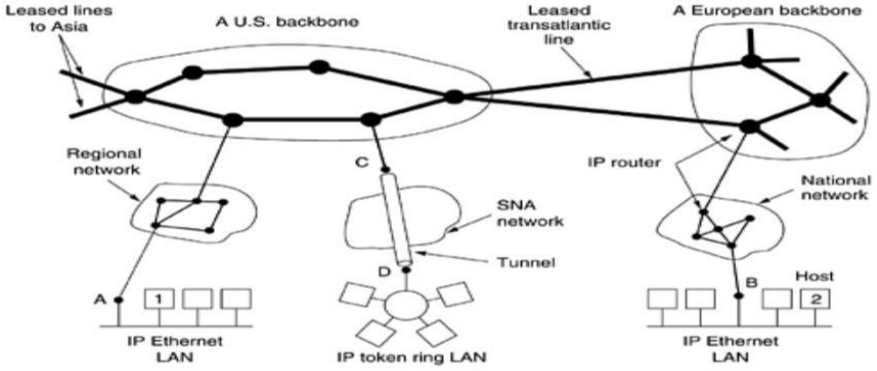


Figure: The Internet is an interconnected collection of many networks.

- The glue that holds the whole Internet together is the network layer protocol, IP (Internet Protocol).
- Unlike older network layer protocols, it was designed from the beginning with internetworking in mind.
- A good way to think of the network layer is this.
- Its job is to provide a best-efforts (i.e., not guaranteed) way to transport datagrams from source to destination, without regard to whether these machines are on the same network or whether there are other networks in between them.

How communication takes place?

- The transport layer takes data streams and breaks them up into datagrams. In theory, datagrams can be up to 64 Kbytes each, but in practice they are usually not more than 1500 bytes (so they fit in one Ethernet frame).
- Each datagram is transmitted through the Internet, possibly being fragmented into smaller units as it goes.
- When all the pieces finally get to the destination machine, they are reassembled by the network layer into the original datagram
- This datagram is then handed to the transport layer, which inserts it into the receiving process' input stream.
- As can be seen from Fig. 3.1, a packet originating at host 1 has to traverse six networks to get to host 2.
- In practice, it is often much more than six.

The IP Protocol

IPv4 is a connectionless protocol used for packet-switched networks. It operates on a best-effort delivery model, in which neither delivery is guaranteed, nor proper sequencing or avoidance of duplicate delivery is assured. Internet Protocol Version 4 (IPv4) is the fourth revision of the Internet Protocol and a widely used protocol in data communication over different kinds of networks. IPv4 is a connectionless protocol used in packet-switched layer networks, such as Ethernet. It provides a logical connection between network devices by providing identification for each device. There are many ways to configure IPv4 with all kinds of devices – including manual and automatic configurations – depending on the network type. IPv4 is defined and specified in IETF publication RFC 791. IPv4 uses 32-bit addresses for Ethernet communication in five classes: A, B, C, D and E. Classes A, B and C have a different bit length for addressing the network host. Class D addresses are reserved for multicasting, while class E addresses are reserved for military purposes. IPv4 uses 32-bit (4-byte) addressing, which gives 2^{32} addresses. IPv4 addresses are written in the dot-decimal notation, which comprises of four octets of the address expressed individually in decimal and separated by periods, for instance, 192.168.1.5.

IPv4 Datagram Header: Size of the header is 20 to 60 bytes.

IP datagram:

- 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.
- It is transmitted in big-endian order: from left to right, with the high-order bit of the Version field going first. (The SPARC is big endian; the Pentium is little-endian.)

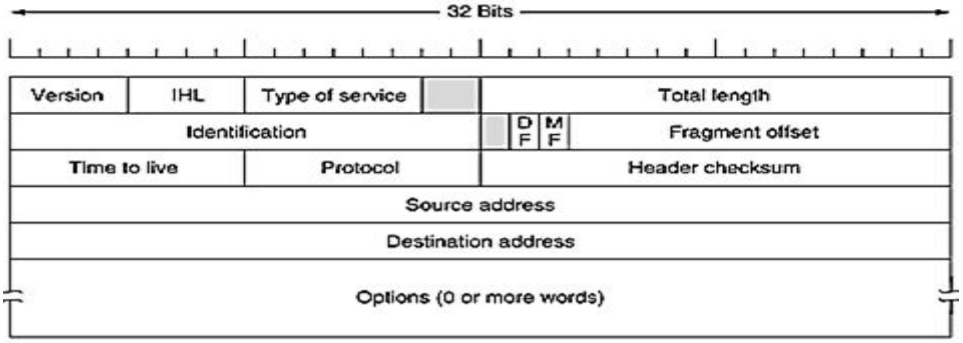


Figure: the IPv4 (Internet Protocol) header.

VERSION: Version of the IP protocol (4 bits), which is 4 for IPv4

HLEN: IP header length (4 bits), which is the number of 32 bit words in the header. The minimum value for this field is 5 and the maximum is 15.

Type of service: Low Delay, High Throughput, Reliability (8 bits)

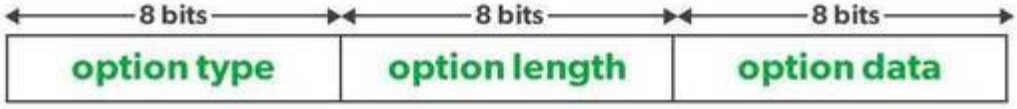
Total Length: Length of header + Data (16 bits), which has a minimum value 20 bytes and the maximum is 65,535 bytes.

Identification: Unique Packet Id for identifying the group of fragments of a single IP datagram (16 bits)

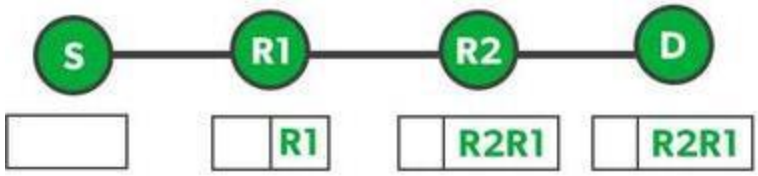
Flags: 3 flags of 1 bit each : reserved bit (must be zero), do not fragment flag, more fragments flag (same order)

Fragment Offset: Represents the number of Data Bytes ahead of the particular fragment in the particular Datagram. Specified in terms of number of 8 bytes, which has the maximum value of 65,528 bytes.
Time to live: Datagram's lifetime (8 bits), It prevents the datagram to loop through the network by restricting the number of Hops taken by a Packet before delivering to the Destination.
Protocol: Name of the protocol to which the data is to be passed (8 bits)
Header Checksum: 16 bits header checksum for checking errors in the datagram header
Source IP address: 32 bits IP address of the sender
Destination IP address: 32 bits IP address of the receiver
Option: Optional information such as source route, record route. Used by the Network administrator to check whether a path is working or not.

The IP datagram may contain zero, one, or more options, which makes the total length of the Options field in the IPv4 header variable. Each of the options can be either a single byte long, or multiple bytes in length, depending on how much information the option needs to convey. When more than one option is included they are just concatenated together and put into the Options field as a whole. All datagrams may not contain this field. This field is optional.

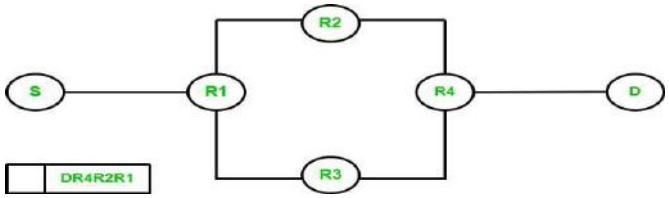


Options Field in IPv4 Header
Here are some functioning supported by this field:
1. Record Route:



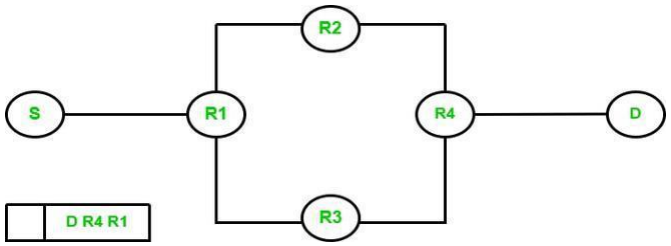
Record Routing
If the packet is going and options record route is set then at router R1, IP address of router R1 say R1 will be recorded on the packet and at router R2, the IP address of router R2 says R2 will get recorded.
Therefore, by the time the destination D receives the packet, it will see that the packet has come through routers R1 and R2.
The maximum size of the Options field in the IPv4 header is 40 bytes and every IP address is 4 bytes. Therefore, the maximum number of IP addresses that can be recorded on any IPv4 datagram when the record route option is set should be 10. But since in the Options field first 8 bits are reserved for option type and the next 8 bits are reserved for option length, we have 38 bytes left for the actual data field. So, in 38 bytes we can record a maximum of 9 IP addresses.
Because of most of the security reasons this record route option is not allowed to be used by the users. Only the network administrator will use it for various purposes like network management and debugging, but an end-user will never be given control over it.

2. Source Routing
If the source does not want to follow pre-defined routing protocols, it can set its own routing protocols and paths. It can specify the route that packet has to take on the path we are sending the packet.



Strict Source Routing
Here for one hop, destination IP address will become IP address of router R1 say R1, and for the next-hop destination, IP address will become IP address of router R2 say R2. Similarly, for the next-hop destination,

the IP address will become the IP address of router R4 say R4 and for the next-hop destination, the IP address will be the IP address of destination D say D. This routing is called strict source routing.
The next source routing option is loose source routing in which we don't have to specify the entire route. We should only specify that the packet should go to router R1 and we don't care whether it goes to router R2 or R3. So, we will skip specifying the IP address of R2 or R3. And then we will specify the IP address of router R4 and destination D.



Loose Source Routing

The network manager at Internet Service Provider will use loose source routing to check if a path is working or not or if they want to test a specific path. Otherwise, if you send a packet it would not always be guaranteed that it has taken a specified path. So, end users cannot use this option. Here, also a maximum of 9 IP addresses can be recorded on the datagram.

3. Padding

Padding is used to ensure that the IP packet header has a length that is a multiple of 32 bits. It is needed because of the varying length of the options field in the IP header. If one or more options are chosen from the Options field, and the number of bits used for them is not a multiple of 32, enough zero bits are added to "pad out" the header to a multiple of 32 bits (4 bytes).

IP Addresses

IP address formats:

- Every host and router on the Internet has an IP address, which encodes its network number and host number.
- No two machines on the Internet have the same IP address.
- All IP addresses are 32 bits long and are used in the Source address and Destination address fields of IP packets.
- For several decades, IP addresses were divided into the five categories listed.
- This allocation has come to be called classful addressing. It is no longer used, but references to it in the literature are still common.

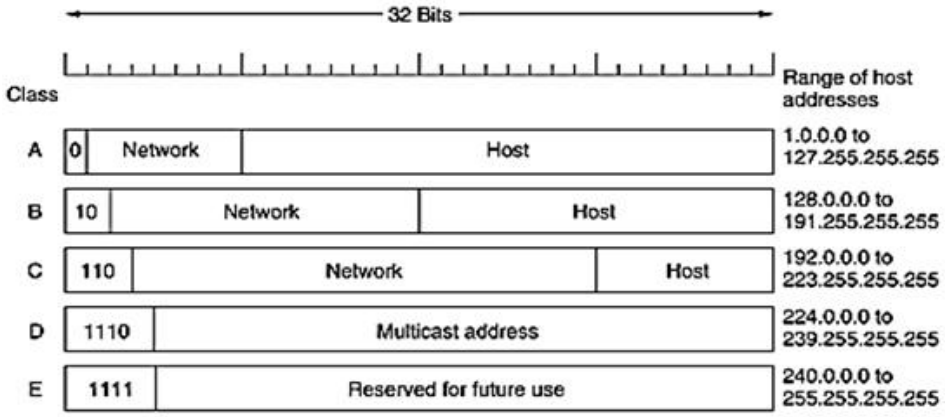


Figure: IP address formats.

The class A, B, C, and D formats allow for up to 128 networks with 16 million hosts each, 16,384 networks with up to 64K hosts, and 2 million networks (e.g., LANs) with up to 256 hosts each (although a few of these are special).

- Also supported is multicast, in which a datagram is directed to multiple hosts. Addresses beginning with 1111 are reserved for future use.
- Over 500,000 networks are now connected to the Internet, and the number grows every year.

Representation:

- Network addresses, which are 32-bit numbers, are usually written in dotted decimal notation.
- In this format, each of the 4 bytes is written in decimal, from 0 to 255.
- For example, the 32-bit hexadecimal address C0290614 is written as 192.41.6.20.
- The lowest IP address is 0.0.0.0 and the highest is 255.255.255.255.
- The values 0 and -1 (all 1s) have special meanings, as shown in Fig. 3.4.
- The value 0 means this network or this host. The value of -1 is used as a broadcast address to mean all hosts on the indicated network.

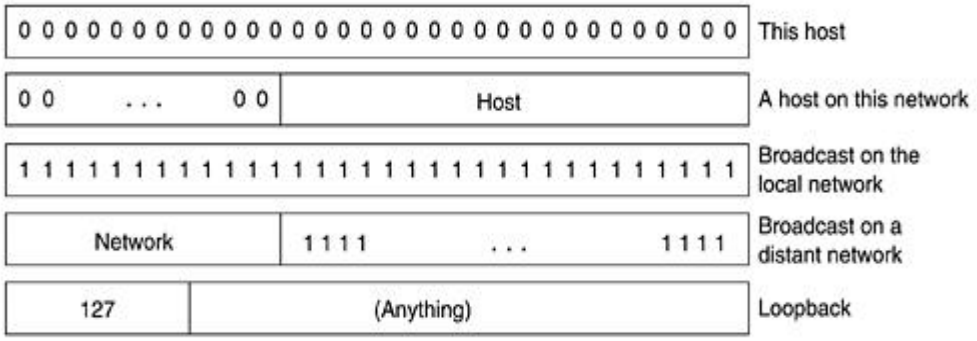


Figure: Special IP addresses

- The IP address **0.0.0.0** is used by hosts when they are being booted.
- IP addresses with **0** as network number refer to the current network.
- These addresses allow machines to refer to their own network without knowing its number (but they have to know its class to know how many 0s to include).
- The address consisting of all 1s allows broadcasting on the local network, typically a **LAN**.
- The addresses with a proper network number and all 1s in the host field allow machines to send broadcast packets to distant LANs anywhere in the Internet (although many network administrators disable this feature).
- Finally, all addresses of the form 127.xx.yy.zz are reserved for loopback testing.
- Packets sent to that address are not put out onto the wire; they are processed locally and treated as incoming packets.
- This allows packets to be sent to the local network without the sender knowing its number.

Subnets

All the hosts in a network must have the same network number. This property of IP addressing can cause problems as networks grow. The solution is to allow a network to be split into several parts for internal use but still act like a single network to the outside world.

- A typical campus network nowadays might look like that of Fig., with a main router connected to an ISP or regional network and numerous Ethernets spread around campus in different departments.
- Each of the Ethernets has its own router connected to the main router (possibly via a backbone LAN, but the nature of the inter-router connection is not relevant here).

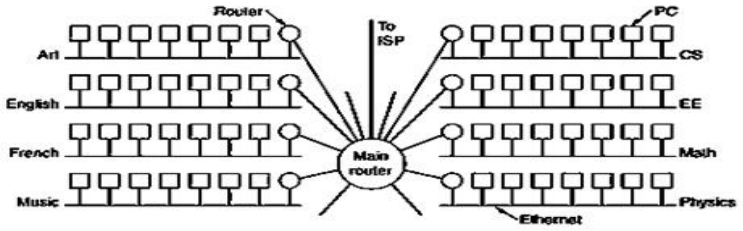


Figure: A campus network consisting of LANs for various departments.

- In the Internet literature, the parts of the network (in this case, Ethernets) are called subnets.
- For example, if the university has 35 departments, it could use a 6-bit subnet number and a 10-bit host number, allowing for up to 64 Ethernets, each with a maximum of 1022 hosts (0 and -1 are not available, as mentioned earlier).
- This split could be changed later if it turns out to be the wrong one.

Subnet Mask:

- To implement sub netting, the main router needs a subnet mask that indicates the split between network subnet number and host, as shown in Fig.
- Subnet masks are also written in dotted decimal notation, with the addition of a slash followed by the number of bits in the network subnet part.
- For the example of Fig. 3.6, the subnet mask can be written as 255.255.252.0. An alternative notation is /22 to indicate that the subnet mask is 22 bits long.

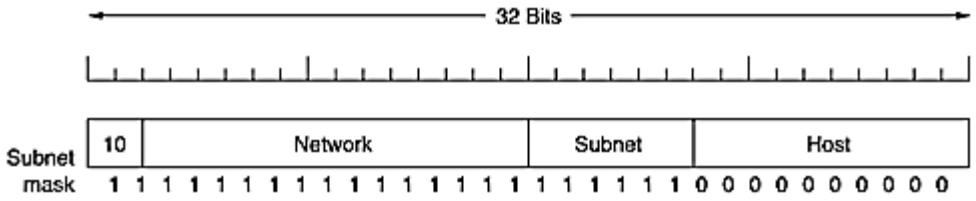


Figure: A class B network sub netted into 64 subnets

- changing any external databases.
- In this example, the first subnet might use IP addresses starting at 130.50.4.1; the second subnet might start at 130.50.8.1; the third subnet might start at 130.50.12.1; and so on.
- To see why the subnets are counting by fours, note that the corresponding binary addresses are as follows:

Subnet 1: 10000010 00110010 000001|00 00000001
Subnet 2: 10000010 00110010 000010|00 00000001
Subnet 3: 10000010 00110010 000011|00 00000001

- Here the vertical bar (|) shows the boundary between the subnet number and the host number. To its left is the 6-bit subnet number; to its right is the 10-bit host number.

CIDR—Classless Inter-Domain Routing

Classless Inter-Domain Routing (CIDR) is a method of IP address allocation and IP routing that allows for more efficient use of IP addresses. CIDR is based on the idea that IP addresses can be allocated and routed based on their network prefix rather than their class, which was the traditional way of IP address allocation.

CIDR addresses are represented using a slash notation, which specifies the number of bits in the network prefix. For example, an IP address of 192.168.1.0 with a prefix length of 24 would be represented as 192.168.1.0/24. This notation indicates that the first 24 bits of the IP address are the network prefix and the remaining 8 bits are the host identifier.

Classless Inter-Domain Routing is also known as Classless addressing.

In Classful addressing, the number of Hosts within a network always remains the same depending upon the class of the Network.

Class A network contains 224(IP addresses) or 224 - 2 Hosts,
Class B network contains 216(IP addresses) or 216 - 2 Hosts,
Class C network contains 28(IP addresses) or 28 - 2 Hosts
Now, let's suppose an Organization requires 214 hosts, then it must have to purchase a Class B network.

In this case, 49150 Hosts will be wasted. This is the major drawback of Classful Addressing.

In order to reduce the wastage of IP addresses a new concept of **Classless Inter-Domain Routing** is introduced.

Nowadays Internet Assigned Numbers Authority *(IANA)* is using this technique to provide IP addresses. Whenever any user asks for IP addresses,

Representation**:** It is as also a 32-bit address, which includes a special number that represents the number of bits that are present in the Block Id.

a . b . c . d / n

Where n is the number of bits that are present in Block Id / Network Id. .

Example**:** 20.10.50.100/20

Rules for forming CIDR Blocks:

1. All IP addresses must be contiguous.
2. Block size must be the power of 2 (2n). If the size of the block is the power of 2, then it will be easy to divide the Network. Finding out the Block Id is very easy if the block size is of the power of Example:
   If the Block size is 25 then, Host Id will contain 5 bits and Network will contain 32 – 5 = 27 bits.

3. First IP address of the Block must be evenly divisible by the size of the block. in simple words, the least significant part should always start with zeroes in Host Id. Since all the least significant bits of Host Id is zero, then we can use it as Block Id part.

Example: Check whether 100.1.2.32 to 100.1.2.47 is a valid IP address block or not?

1. All the IP addresses are contiguous.
2. Total number of IP addresses in the Block = 16 = 24.
3. 1st IP address: 100.1.2.00100000 Since, Host Id will contains last 4 bits and all the least significant 4 bits are zero. Hence, first IP address is evenly divisible by the size of the block.

All three rules are followed by this Block. Hence, it is a valid IP address block.

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), which is explained below.

Overview of NAT:

- 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.
- Within the company, every computer gets a unique IP address, which is used for routing intramural traffic.
- However, when a packet exits the company and goes to the ISP, an address translation takes place.
- To make this scheme possible, three ranges of IP addresses have been declared as private.
- Companies may use them internally as they wish.
- The only rule is that no packets containing these addresses may appear on the Internet itself. 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 first range provides for 16,777,216 addresses (except for 0 and -1, as usual)
- The operation of NAT is shown in Fig. 3.8. 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, 10.0.0.1 in the figure, to the company's true IP address, 198.60.42.12 in this example.
- The NAT box is often combined in a single device with a firewall, which provides security by carefully controlling what goes into the company and what comes out.

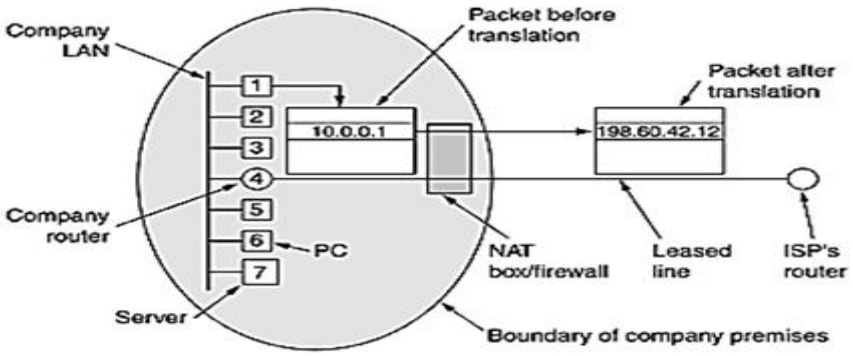


Figure: Placement and operation of a NAT box

How does the NAT box know which address to replace it with?

- The NAT designers observed that most IP packets carry either TCP or UDP payloads.
- TCP and UDP both of them have headers containing a **source port** and a **destination port**.
- The TCP ports are 16-bit integers that indicate where the TCP connection begins and ends.
- These ports provide the field needed to make NAT work.
- When a process wants to establish a TCP connection with a remote process, it attaches itself to an unused TCP port on its own machine.
- This is called the **source port** and tells the TCP code where to send incoming packets belonging to this connection.
- The process also supplies a destination port to tell who to give the packets to on the remote side.
- Ports **0-1023** are reserved for well-known services
- For example, port **80** is the port used by Web servers, so remote clients can locate them.
- Each outgoing TCP message contains both a source port and a destination port.
- Together, these ports serve to identify the processes using the connection on both ends.

Solution to mapping problem:

- Using the Source port field, mapping problem can be solved.
- Whenever an outgoing packet enters the NAT box, the 10.x.y.z source address is replaced by the company's true IP address.
- In addition, the TCP Source port field is replaced by an index into the NAT box's 65,536-entry translation table.
- This table entry contains the original IP address and the original source port.
- Finally, both the IP and TCP header checksums are recomputed and inserted into the packet.

Internet Control Protocols

The Internet Control Message Protocol:

- The operation of the Internet is monitored closely by the routers.
- When something unexpected occurs, the event is reported by the ICMP (Internet Control Message Protocol), which is also used to test the Internet.
- About a dozen types of ICMP messages are defined.
- The most important ones are listed in Fig. 3.9. Each ICMP message type is encapsulated in an IP packet.

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo	Ask a machine if it is alive
Echo reply	Yes, I am alive
Timestamp request	Same as Echo request, but with timestamp
Timestamp reply	Same as Echo reply, but with timestamp

Figure: The principal ICMP message types.

- 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'sIP 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. It is rarely used anymore because when congestion occurs, these packets tend to add more fuel to the fire.
- 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—The Address Resolution Protocol

Address Resolution Protocol (ARP) is a protocol or procedure that connects an ever-changing Internet Protocol (IP) address to a fixed physical machine address, also known as a media access control (MAC) address, in a local-area network (LAN).

This mapping procedure is important because the lengths of the IP and MAC addresses differ, and a translation is needed so that the systems can recognize one another. The most used IP today is IP version 4 (IPv4). An IP address is 32 bits long. However, MAC addresses are 48 bits long. ARP translates the 32-bit address to 48 and vice versa.

Imagine a device that wants to communicate with others over the internet. What does ARP do? It broadcast a packet to all the devices of the source network. The devices of the network peel the header of the data link layer from the Protocol Data Unit (PDU) called frame and transfer the packet to the network layer (layer 3 of OSI) where the network ID of the packet is validated with the destination IP's network ID of the packet and if it's equal then it responds to the source with the MAC address of the destination, else the packet reaches the gateway of the network and broadcasts packet to the devices it is connected with and validates their network ID. The above process continues till the second last network device in the path reaches the destination where it gets validated and ARP, in turn, responds with the destination MAC address.

1. ARP Cache**:** After resolving the MAC address, the ARP sends it to the source where it is stored in a table for future reference. The subsequent communications can use the MAC address from the table.
2. ARP Cache Timeout**:** It indicates the time for which the MAC address in the ARP cache can reside.
3. ARP request: This is nothing but broadcasting a packet over the network to validate whether we came across the destination MAC address or not.
    1. The physical address of the sender.
    2. The IP address of the sender.
    3. The physical address of the receiver is FF:FF:FF:FF:FF: FF or 1's.
    4. The IP address of the receiver.
4. ARP response/reply: It is the MAC address response that the source receives from the destination which aids in further communication of the data.

Although every machine on the Internet has one (or more) IP addresses, these cannot actually be used for sending packets because the data link layer hardware does not understand Internet addresses.

Nowadays, most hosts at companies and universities are attached to a LAN by an interface board that only understands LAN addresses

How do IP addresses get mapped onto data link layer addresses, such as Ethernet?

- Look Fig., in which a small university with several class C (now called /24) networks is illustrated.
- Here we have two Ethernets, one in the Computer Science Dept., with IP address 192.31.65.0 and one in Electrical Engineering, with IP address 192.31.63.0.
- These are connected by a campus backbone ring (e.g., FDDI) with IP address 192.31.60.0.
- Each machine on an Ethernet has a unique Ethernet address, labeled E1 through E6, and each machine on the FDDI ring has an FDDI address, labeled F1 through F3.

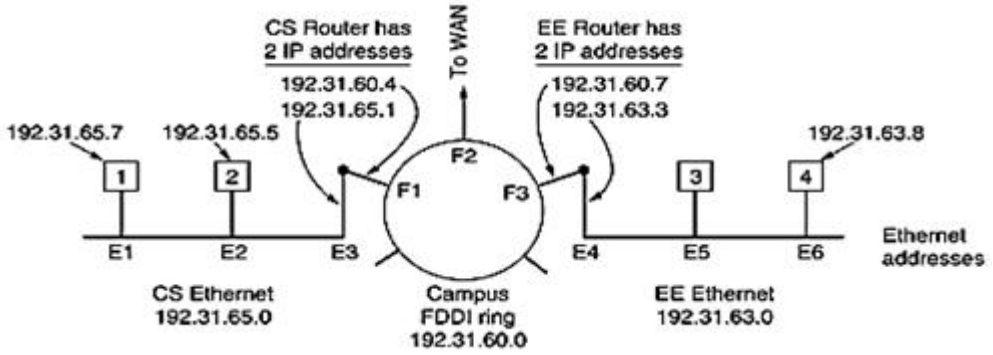


Figure: Three interconnected /24 networks: two Ethernets and an FDDI ring

How a user on host 1 sends a packet to a user on host 2?

- Let us assume the sender knows the name of the intended receiver, possibly something like mary@eagle.cs.uni.edu.
- The first step is to find the IP address for host 2, known as eagle.cs.uni.edu.
- This lookup is performed by the Domain Name System.
- For the moment, we will just assume that DNS returns the IP address for host 2 (192.31.65.5).
- The upper layer software on host 1 now builds a packet with 192.31.65.5 in the Destination address field and gives it to the IP software to transmit.
- The IP software can look at the address and see that the destination is on its own network, but it needs some way to find the destination's Ethernet address.
- One solution is to have a configuration file somewhere in the system that maps IP addresses onto Ethernet addresses.

Reverse Address Resolution Protocol (RARP)

Reverse ARP is a networking protocol used by a client machine in a local area network to request its Internet Protocol address (IPv4) from the gateway-router's ARP table. The network administrator creates a table in gateway-router, which is used to map the MAC address to corresponding IP address. When a new machine is setup or any machine which don't have memory to store IP address, needs an IP address for its own use. So the machine sends a RARP broadcast packet which contains its own MAC address in both sender and receiver hardware address field.

A special host configured inside the local area network, called as RARP-server is responsible to reply for these kind of broadcast packets. Now the RARP server attempt to find out the entry in IP to MAC address mapping table. If any entry matches in table, RARP server send the response packet to the requesting device along with IP address.

- LAN technologies like Ethernet, Ethernet II, Token Ring and Fiber Distributed Data Interface (FDDI) support the Address Resolution Protocol.
- RARP is not being used in today's networks. Because we have much great featured protocols like BOOTP (Bootstrap Protocol) and DHCP(Dynamic Host Configuration Protocol).

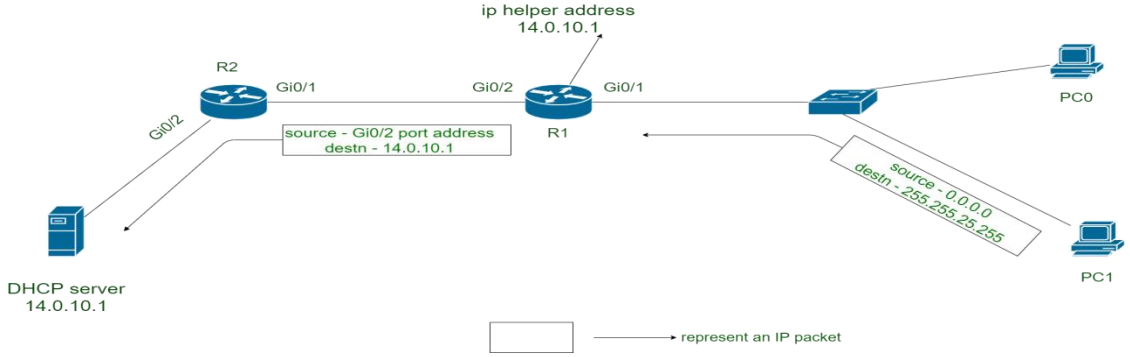
Bootstrap Protocol (BOOTP)

- At the very beginning, each network participant does not have an IP address. The network administrator then provides each host on the network with a unique IP address using the IPv4 protocol.
- The client installs the BOOTP network protocol using TCP / IP Intervention on its computer system to ensure compatibility with all network protocols when connected to this network.

- The BOOTP network administrator then sends a message that contains a valid unicast address. This unicast address is then forwarded to the BOOTP client by the master server.

Dynamic Host Configuration Protocol

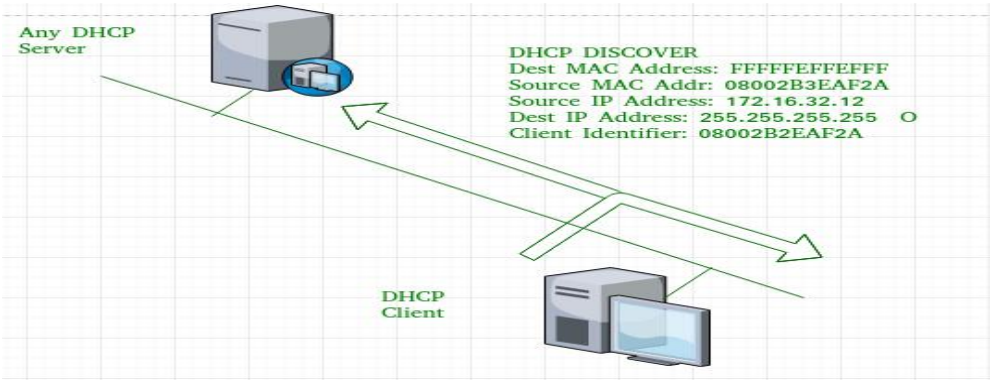
The DHCP **port number** for the server is 67 and for the client is 68. It is a client-server protocol that uses UDP services. An IP address is assigned from a pool of addresses. In DHCP, the client and the server exchange mainly 4 DHCP messages in order to make a connection, also called the DORA process, but there are 8 DHCP messages in the process.



Working of DHCP

The 8 DHCP Messages:

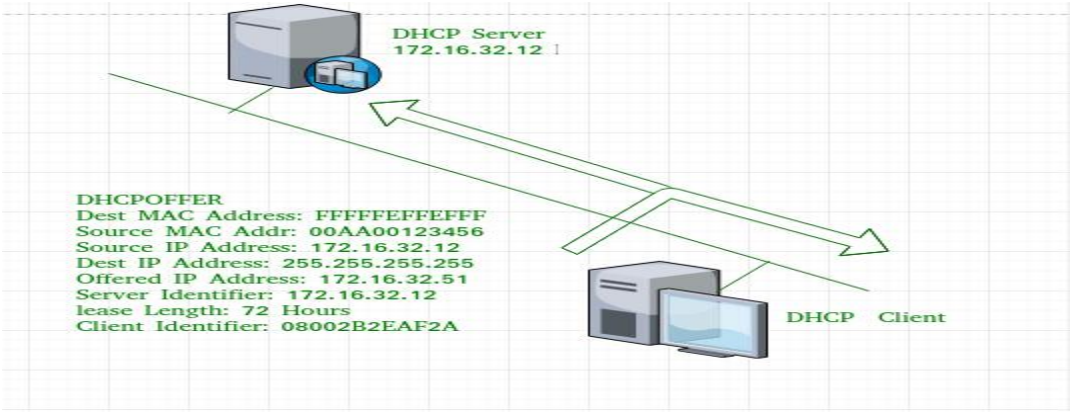
1. DHCP discover message**:** This is the first message generated in the communication process between the server and the client. This message is generated by the Client host in order to discover if there is any DHCP server/servers are present in a network or not. This message is broadcasted to all devices present in a network to find the DHCP server. This message is 342 or 576 bytes long



DHCP discover message

As shown in the figure, the source MAC address (client PC) is 08002B2EAF2A, the destination MAC address(server) is FFFFFFFFFFFF, the source IP address is 0.0.0.0(because the PC has had no IP address till now) and the destination IP address is 255.255.255.255 (IP address used for broadcasting). As they discover message is broadcast to find out the DHCP server or servers in the network therefore broadcast IP address and MAC address is used.

2. DHCP offers a message**:** The server will respond to the host in this message specifying the unleased IP address and other TCP configuration information. This message is broadcasted by the server. The size of the message is 342 bytes. If there is more than one DHCP server present in the network then the client host will accept the first DHCP OFFER message it receives. Also, a server ID is specified in the packet in order to identify the server.

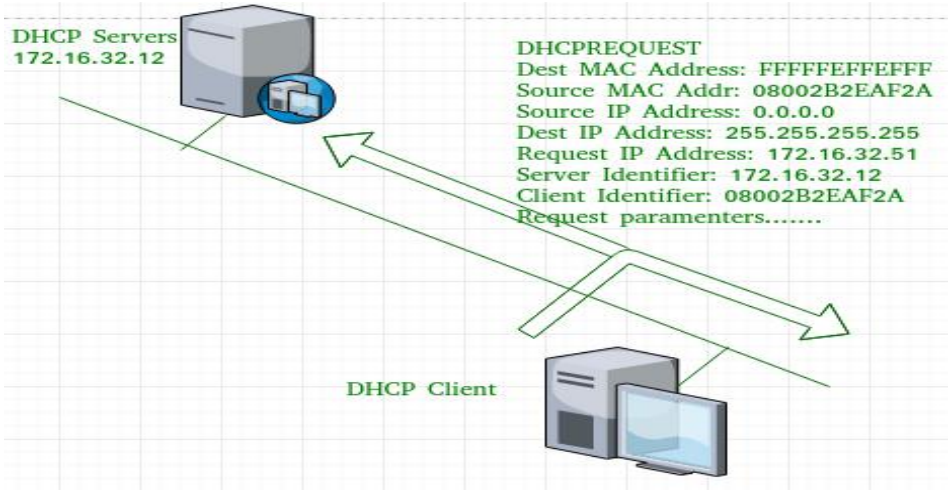


DHCP offer message

Now, for the offer message, the source IP address is 172.16.32.12 (server's IP address in the example), the destination IP address is 255.255.255.255 (broadcast IP address), the source MAC address is 00AA00123456, the destination MAC address is FFFFFFFFFFFF. Here, the offer message is broadcast by the DHCP server therefore destination IP address is the broadcast IP address and destination MAC address is FFFFFFFFFFFF and the source IP address is the server IP address and the MAC address is the server MAC address.

Also, the server has provided the offered IP address 192.16.32.51 and a lease time of 72 hours(after this time the entry of the host will be erased from the server automatically). Also, the client identifier is the PC MAC address (08002B2EAF2A) for all the messages.

3. DHCP request message**:** When a client receives an offer message, it responds by broadcasting a DHCP request message. The client will produce a gratuitous ARP in order to find if there is any other host present in the network with the same IP address. If there is no reply from another host, then there is no host with the same TCP configuration in the network and the message is broadcasted to the server showing the acceptance of the IP address. A Client ID is also added to this message.

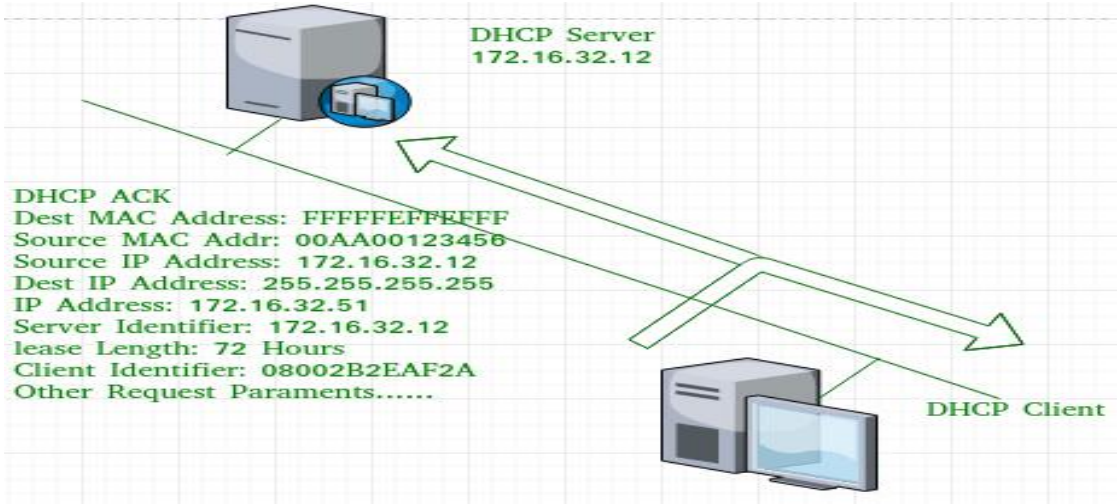


DHCP request message

Now, the request message is broadcast by the client PC therefore source IP address is 0.0.0.0(as the client has no IP right now) and destination IP address is 255.255.255.255 (the broadcast IP address) and the source MAC address is 08002B2EAF2A (PC MAC address) and destination MAC address is FFFFFFFFFFFF.

Note **–** This message is broadcast after the ARP request broadcast by the PC to find out whether any other host is not using that offered IP. If there is no reply, then the client host broadcast the DHCP request message for the server showing the acceptance of the IP address and Other TCP/IP Configuration.

4. DHCP acknowledgment message**:** In response to the request message received, the server will make an entry with a specified client ID and bind the IP address offered with lease time. Now, the client will have the IP address provided by the server.



DHCP acknowledgment message

Now the server will make an entry of the client host with the offered IP address and lease time. This IP address will not be provided by the server to any other host. The destination MAC address is FFFFFFFFFFFF and the destination IP address is 255.255.255.255 and the source IP address is 172.16.32.12 and the source MAC address is 00AA00123456 (server MAC address).

5. DHCP negative acknowledgment message: Whenever a DHCP server receives a request for an IP address that is invalid according to the scopes that are configured, it sends a DHCP Nak message to the client. Eg- when the server has no IP address unused or the pool is empty, then this message is sent by the server to the client.

6. DHCP decline: If the DHCP client determines the offered configuration parameters are different or invalid, it sends a DHCP decline message to the server. When there is a reply to the gratuitous ARP by any host to the client, the client sends a DHCP decline message to the server showing the offered IP address is already in use.

7. DHCP release: A DHCP client sends a DHCP release packet to the server to release the IP address and cancel any remaining lease time.

8. DHCP inform: If a client address has obtained an IP address manually then the client uses DHCP information to obtain other local configuration parameters, such as domain name. In reply to the DHCP inform message, the DHCP server generates a DHCP ack message with a local configuration suitable for the client without allocating a new IP address. This DHCP ack message is unicast to the client.

OSPF—The Interior Gateway Routing Protocol

Internet is made up of a large number of autonomous systems (AS). Each AS is operated by a different organization and can use its own routing algorithm inside.

Routing in the Internet:

- A routing algorithm within an AS is called an interior gateway protocol; an algorithm for routing between ASes is called an exterior gateway protocol.
- The original Internet interior gateway protocol was a distance vector protocol.
- It worked well in small systems, but less well as ASes got larger.
- It also suffered from the count-to-infinity problem and generally slows convergence, so it was replaced by a link state protocol.
- In 1988, the Internet Engineering Task Force began work on a successor.
- That successor, called OSPF (Open Shortest Path First), became a standard in 1990.

Connections and Network:

OSPF supports three kinds of connections and networks:

1. Point-to-point lines between exactly two routers.
2. Multiaccess networks with broadcasting (e.g., most LANs).
3. Multiaccess networks without broadcasting (e.g., most packet-switched WANs).

- A multiaccess network is one that can have multiple routers on it, each of which can directly communicate with all the others.
- All LANs and WANs have this property.
- Figure 3.12(a) shows an AS containing all three kinds of networks.

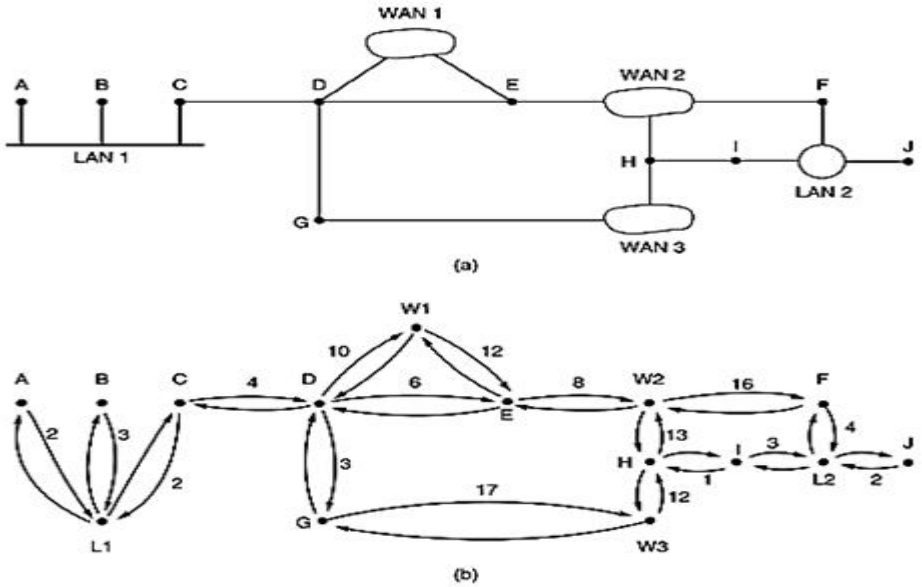


Figure: (a) An autonomous system. (b) A graph representation of (a).

v. Operation and its graph representation:

- OSPF operates 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.
- Figure shows the graph representation of the network of Fig. (a).
- What OSPF fundamentally does is represent the actual network as a graph like this and then compute the shortest path from every router to every other router.
- OSPF allows them to be divided into numbered areas, where an area is a network or a set of contiguous networks.

Figure 5-66. The five types of OSPF messages.

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

BGP-The Exterior Gateway Routing Protocol

Within a single AS, the recommended routing protocol is OSPF (although it is certainly not the only one in use). Between ASes, a different protocol, BGP (Border Gateway Protocol), is used. The following section elaborates the BGP in details.

BGP:

- Pairs of BGP routers communicate with each other by establishing TCP connections.
- Operating this way provides reliable communication and hides all the details of the network being passed through.
- BGP is fundamentally a distance vector protocol, but quite different from most others such as RIP.
- Instead of maintaining just the cost to each destination, each BGP router keeps track of the path used.
- Similarly, instead of periodically giving each neighbor its estimated cost to each possible destination, each BGP router tells its neighbors the exact path it is using.

  As an example, consider the BGP routers shown in Fig. In particular, 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 shown in Fig. (b) (for simplicity, only destination D is shown here).

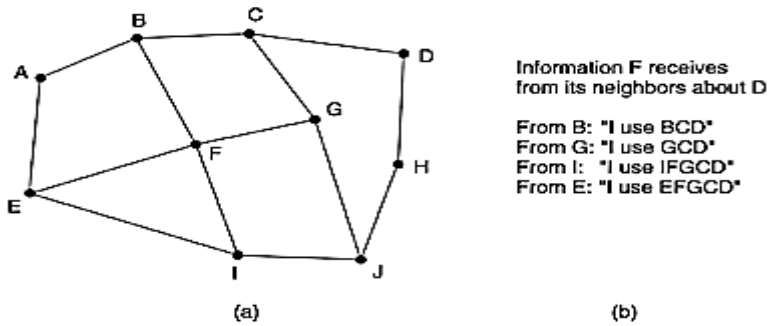


Figure (a) A set of BGP routers. (b) Information sent to F.

After all the paths come in from the neighbors, F examines them to see which is the best.
It quickly discards the paths from I and E, since these paths pass through F itself.
The choice is then between using B and G.

- Every BGP router contains a module that examines routes to a given destination and scores them, returning a number for the "distance" to that destination for each route.
- Any route violating a policy constraint automatically gets a score of infinity.
- The router then adopts the route with the shortest distance.
- The scoring function is not part of the BGP protocol and can be any function the system managers want.
- BGP easily solves the count-to-infinity problem that plagues other distance vector routing algorithms.

Internet Multicasting

For some applications it is useful for a process to be able to send to a large number of receivers simultaneously. Examples are updating replicated, distributed databases, transmitting stock quotes to multiple brokers, and handling digital conference (i.e., multiparty) telephone calls.

Multicasting:

- IP supports multicasting, using class D addresses.
- Each class D address identifies a group of hosts.
- Twenty-eight 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, a best-efforts attempt is made to deliver it to all the members of the group addressed, but no guarantees are given.
- Some members may not get the packet.
- Two kinds of group addresses are supported: permanent addresses and temporary ones.
- A permanent group is always there and does not have to be set up.
- Each permanent group has a permanent group address.
- Some examples of permanent group addresses are:

  224.0.0.1 All systems on a LAN
  224.0.0.2 All routers on a LAN
  224.0.0.5 All OSPF routers on a LAN
  224.0.0.6 All designated OSPF routers on a LAN
- Temporary groups must be created before they can be used.
- A process can ask its host to join a specific group. It can also ask its host to leave the group.
- When the last process on a host leaves a group, that group is no longer present on the host. Each host keeps track of which groups its processes currently belong to.

Implementing Multicasting:

- Multicasting is implemented by special multicast routers, which may or may not be collocated with the standard routers.

- About once a minute, each multicast router sends a hardware (i.e., data link layer) multicast to the hosts on its LAN (address 224.0.0.1) 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), which is vaguely analogous to ICMP.
- It has only two kinds of packets: query and response, each with a simple, fixed format containing some control information in the first word of the payload field and a class D address in the second word.

Multicast Routing**:**

- 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 the 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.

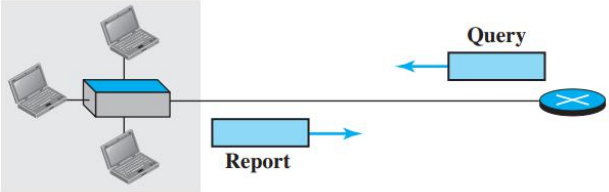
IGMP

The protocol that is used today for collecting information about group membership is the Internet Group Management Protocol (IGMP). IGMP is a protocol defined at the network layer; it is one of the auxiliary protocols, like ICMP, which is considered part of the IP. IGMP messages, like ICMP messages, are encapsulated in an IP datagram.

There are only two types of messages in IGMP version 3, query and report messages.

A query message is periodically sent by a router to all hosts attached to it to ask them to report their interests about membership in groups. A report message is sent by a host as a response to a query message.

Figure 21.16 *IGMP operation*



The query message is sent by a router to all hosts in each interface to collect information about their membership.

A report message is sent by a host as a response to a query message. The message contains a list of records in which each record gives the identifier of the corresponding group (multicast address) and the addresses of all sources that the host is interested in receiving messages from (inclusion). The record can also mention the source addresses from which the host does not desire to receive a group message (exclusion). The message is encapsulated in a datagram with the multicast address 224.0.0.22 (multicast address assigned to IGMPv3).

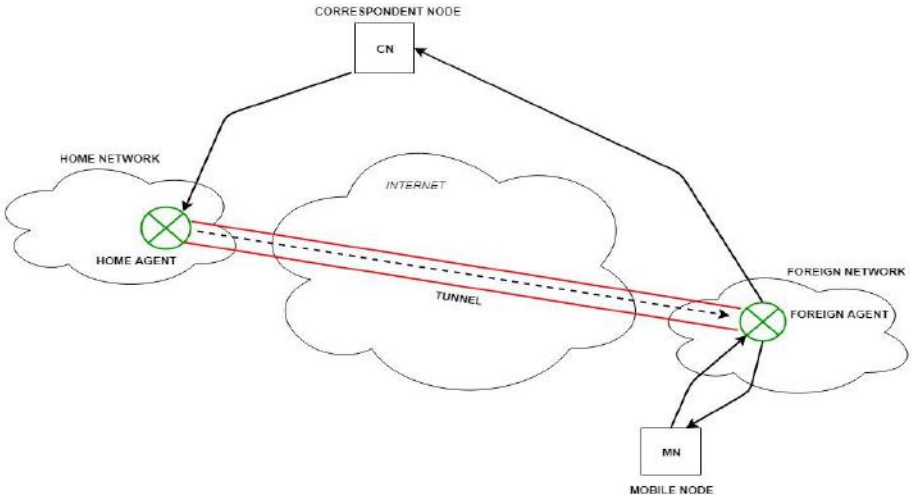
Mobile IP

Mobile IP is a communication protocol (created by extending Internet Protocol, IP) that allows the users to move from one network to another with the same IP address. It ensures that the communication will continue without the user's sessions or connections being dropped.

Terminologies:

1. Mobile Node (MN) is the hand-held communication device that the user carries e.g. Cell phone.
2. Home Network is a network to which the mobile node originally belongs as per its assigned IP address (home address).
3. Home Agent (HA) is a router in-home network to which the mobile node was originally connected
4. Home Address is the permanent IP address assigned to the mobile node (within its home network).
5. Foreign Network is the current network to which the mobile node is visiting (away from its home network).

6. Foreign Agent (FA) is a router in a foreign network to which the mobile node is currently connected. The packets from the home agent are sent to the foreign agent which delivers them to the mobile node.
7. Correspondent Node (CN) is a device on the internet communicating to the mobile node.
8. Care-of Address (COA) is the temporary address used by a mobile node while it is moving away from its home network.
9. Foreign agent COA, the COA could be located at the FA, i.e., the COA is an IP address of the FA. The FA is the tunnel end-point and forwards packets to the MN. Many MN using the FA can share this COA as a common COA.
10. Co-located COA, the COA is co-located if the MN temporarily acquired an additional IP address which acts as COA. This address is now topologically correct, and the tunnel endpoint is at the MN. Co-located addresses can be acquired using services such as DHCP.



Mobile IP

Working:

The correspondent node sends the data to the mobile node. Data packets contain the correspondent node's address (Source) and home address (Destination). Packets reach the home agent. But now mobile node is not in the home network, it has moved into the foreign network. The foreign agent sends the care-of-address to the home agent to which all the packets should be sent. Now, a tunnel will be established between the home agent and the foreign agent by the process of tunneling.

Tunneling establishes a virtual pipe for the packets available between a tunnel entry and an endpoint. It is the process of sending a packet via a tunnel and it is achieved by a mechanism called encapsulation.

Now, the home agent encapsulates the data packets into new packets in which the source address is the home address and destination is the care-of-address and sends it through the tunnel to the foreign agent. Foreign agent, on another side of the tunnel, receives the data packets, decapsulates them, and sends them to the mobile node. The mobile node in response to the data packets received sends a reply in response to the foreign agent. The foreign agent directly sends the reply to the correspondent node.

IPv6

With the impending convergence of the computer, communication, and entertainment industries, it may not be that long before every telephone and television set in the world is an Internet node, producing a billion machines being used audio and video on demand. Under these circumstances, it became apparent that IP had to evolve and become more flexible.

Goals of IPV6:

- Seeing these problems on the horizon, in 1990, IETF started work on a new version of IP, one which would never run out of addresses, would solve a variety of other problems, and be more flexible and efficient as well. 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

IPv6 meets the goals fairly well.

- It maintains the good features of IP, discards or deemphasizes the bad ones, and adds new ones where needed.
- In general, IPv6 is not compatible with IPv4, but it is compatible with the other auxiliary Internet protocols, including TCP, UDP, ICMP, IGMP, OSPF, BGP, and DNS, sometimes with small modifications being required (mostly to deal with longer addresses).

It offers the following features:

- Larger Address Space

In contrast to IPv4, IPv6 uses 4 times more bits to address a device on the Internet. This much of extra bits can provide approximately 3.4×10^{38} different combinations of addresses. This address can accumulate the aggressive requirement of address allotment for almost everything in this world. According to an estimate, 1564 addresses can be allocated to every square meter of this earth.

- Simplified Header

IPv6's header has been simplified by moving all unnecessary information and options (which are present in IPv4 header) to the end of the IPv6 header. IPv6 header is only twice as bigger than IPv4 provided the fact that IPv6 address is four times longer.

- End-to-end Connectivity

Every system now has unique IP address and can traverse through the Internet without using NAT or other translating components. After IPv6 is fully implemented, every host can directly reach other hosts on the Internet, with some limitations involved like Firewall, organization policies, etc.

- Auto-configuration

IPv6 supports both stateful and stateless auto configuration mode of its host devices. This way, absence of a DHCP server does not put a halt on inter segment communication.

- Faster Forwarding/Routing

Simplified header puts all unnecessary information at the end of the header. The information contained in the first part of the header is adequate for a Router to take routing decisions, thus making routing decision as quickly as looking at the mandatory header.

- IPSec

Initially it was decided that IPv6 must have IPSec security, making it more secure than IPv4. This feature has now been made optional.

- No Broadcast

Though Ethernet/Token Ring are considered as broadcast network because they support Broadcasting, IPv6 does not have any broadcast support any more. It uses multicast to communicate with multiple hosts.

- Anycast Support

This is another characteristic of IPv6. IPv6 has introduced Anycast mode of packet routing. In this mode, multiple interfaces over the Internet are assigned same Anycast IP address. Routers, while routing, send the packet to the nearest destination.

- Mobility

IPv6 was designed keeping mobility in mind. This feature enables hosts (such as mobile phone) to roam around in different geographical area and remain connected with the same IP address. The mobility feature of IPv6 takes advantage of auto IP configuration and Extension headers.

- Enhanced Priority Support

IPv4 used 6 bits DSCP (Differential Service Code Point) and 2 bits ECN (Explicit Congestion Notification) to provide Quality of Service but it could only be used if the end-to-end devices support it, that is, the source and destination device and underlying network must support it.

In IPv6, Traffic class and Flow label are used to tell the underlying routers how to efficiently process the packet and route it.

- Smooth Transition

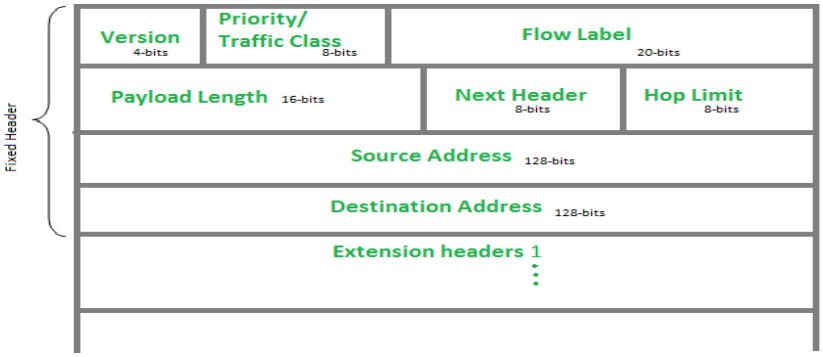
Large IP address scheme in IPv6 enables to allocate devices with globally unique IP addresses. This mechanism saves IP addresses and NAT is not required. So devices can send/receive data among each other, for example, VoIP and/or any streaming media can be used much efficiently.

Other fact is, the header is less loaded, so routers can take forwarding decisions and forward them as quickly as they arrive.

- Extensibility

One of the major advantages of IPv6 header is that it is extensible to add more information in the option part. IPv4 provides only 40-bytes for options, whereas options in IPv6 can be as much as the size of IPv6 packet itself.

Header Format :

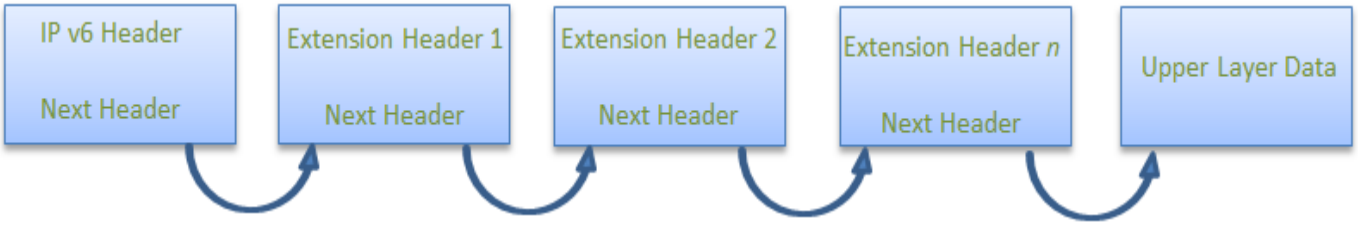


Version (4-bits): Indicates version of Internet Protocol which contains bit sequence 0110.
Traffic Class (8-bits): The Traffic Class field indicates class or priority of IPv6 packet which is similar to *Service Field* in IPv4 packet. It helps routers to handle the traffic based on the priority of the packet. If congestion occurs on the router then packets with the least priority will be discarded. As of now, only 4-bits are being used (and the remaining bits are under research), in which 0 to 7 are assigned to Congestion controlled traffic and 8 to 15 are assigned to Uncontrolled traffic.
Priority assignment of Congestion controlled traffic :

Priority	Meaning
0	No Specific traffic
1	Background data
2	Unattended data traffic
3	Reserved
4	Attended bulk data traffic
5	Reserved
6	Interactive traffic
7	Control traffic

Uncontrolled data traffic is mainly used for Audio/Video data. So we give higher priority to Uncontrolled data traffic.
The source node is allowed to set the priorities but on the way, routers can change it. Therefore, the destination should not expect the same priority which was set by the source node.
Flow Label (20-bits): Flow Label field is used by a source to label the packets belonging to the same flow in order to request special handling by intermediate IPv6 routers, such as non-default quality of service or real-time service. In order to distinguish the flow, an intermediate router can use the source address, a destination address, and flow label of the packets. Between a source and destination, multiple flows may exist because many processes might be running at the same time. Routers or Host that does not support the functionality of flow label field and for default router handling, flow label field is set to 0. While setting up the flow label, the source is also supposed to specify the lifetime of the flow.
Payload Length (16-bits): It is a 16-bit (unsigned integer) field, indicates the total size of the payload which tells routers about the amount of information a particular packet contains in its payload. The payload Length field includes extension headers(if any) and an upper-layer packet. In case the length of the payload is greater than 65,535 bytes (payload up to 65,535 bytes can be indicated with 16-bits), then the payload length field will be set to 0 and the jumbo payload option is used in the Hop-by-Hop options extension header.

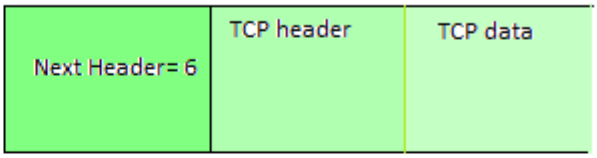
Next Header (8-bits): Next Header indicates the type of extension header(if present) immediately following the IPv6 header. Whereas In some cases it indicates the protocols contained within upper-layer packets, such as TCP, UDP.
Hop Limit (8-bits): Hop Limit field is the same as TTL in IPv4 packets. It indicates the maximum number of intermediate nodes IPv6 packet is allowed to travel. Its value gets decremented by one, by each node that forwards the packet and the packet is discarded if the value decrements to 0. This is used to discard the packets that are stuck in an infinite loop because of some routing error.
Source Address (128-bits): Source Address is the 128-bit IPv6 address of the original source of the packet.
Destination Address (128-bits): The destination Address field indicates the IPv6 address of the final destination(in most cases). All the intermediate nodes can use this information in order to correctly route the packet.
Extension Headers: In order to rectify the limitations of the *IPv4 Option Field*, Extension Headers are introduced in IP version 6. The extension header mechanism is a very important part of the IPv6 architecture. The next Header field of IPv6 fixed header points to the first Extension Header and this first extension header points to the second extension header and so on.



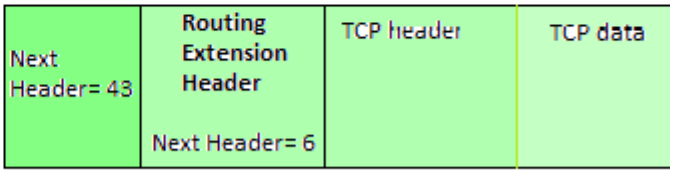
IPv6 packet may contain zero, one or more extension headers but these should be present in their recommended order:

Order	Header Type	Next Header Code
1	Basic IPv6 Header	-
2	Hop-by-Hop Options	0
3	Destination Options (with Routing Options)	60
4	Routing Header	43
5	Fragment Header	44
6	Authentication Header	51
7	Encapsulation Security Payload Header	50
8	Destination Options	60
9	Mobility Header	135
	No next header	59
Upper Layer	TCP	6
Upper Layer	UDP	17
Upper Layer	ICMPv6	58

Example: *TCP is used in IPv6 packet*



Example2:



Rule: Hop-by-Hop options header(if present) should always be placed after the IPv6 base header.

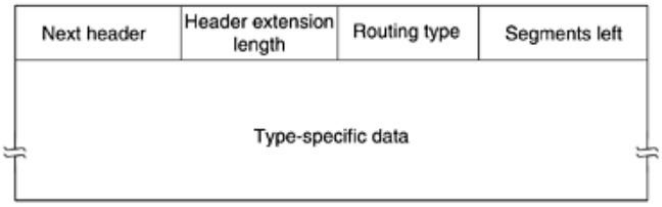
Conventions :

1. Any extension header can appear at most once except Destination Header because Destination Header is present two times in the above list itself.
2. If Destination Header is present before Routing Header then it will be examined by all intermediate nodes specified in the routing header.
3. If Destination Header is present just above the Upper layer then it will be examined only by the Destination node.

Given order in which all extension header should be chained in IPv6 packet and working of each extension header **:**

Ext. Header	Description
Hop-by-Hop Options	Examined by all devices on the path
Destination Options (with routing options)	Examined by destination of the packet
Routing Header	Methods to take routing decision
Fragment Header	Contains parameters of fragmented datagram done by source
Authentication Header	verify authenticity
Encapsulating Security Payload	Carries Encrypted data

Figure 5-71. The extension header for routing.



Unit IV

The Transport Layer: The Transport Service, Elements of Transport Protocols, The internet transport protocols: UDP, TCP, Performance problems in computer networks, Network performance measurement.

The transport layer in the TCP/IP suite is located between the application layer and the network layer. It provides services to the application layer and receives services from the network layer. The transport layer acts as a liaison between a client program and a server program, a process-to-process connection. The transport layer is the heart of the TCP/IP protocol suite; it is the end-to-end logical vehicle for transferring data from one point to another in the Internet.

Communication is provided using a logical connection, which means that the two application layers, which can be located in different parts of the globe, assume that there is an imaginary direct connection through which they can send and receive messages.

It is an end-to-end layer used to deliver messages to a host. It is termed an end-to-end layer because it provides a point-to-point connection rather than hop-to-hop, between the source host and destination host to deliver the services reliably. The unit of data encapsulation in the Transport Layer is a segment.

The Transport Layer Service

Services Provided to the Upper Layers

1. Process-to-Process Communication

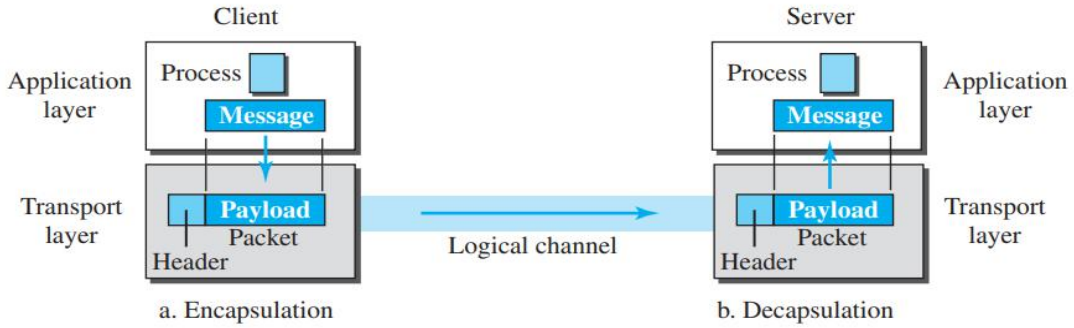
The first duty of a transport-layer protocol is to provide process-to-process communication. A process is an application-layer entity (running program) that uses the services of the transport layer.

2. Encapsulation and Decapsulation

To send a message from one process to another, the transport-layer protocol encapsulates and decapsulates messages. Encapsulation happens at the sender site. When a process has a message to send, it passes the message to the transport layer along with a pair of socket addresses and some other pieces of information, which depend on the transport-layer protocol. The transport layer receives the data and adds the transport-layer header. The packets at the transport layer in the Internet are called user datagrams, segments, or packets, depending on what transport-layer protocol we use. In general discussion, we refer to transport-layer payloads as packets.

Decapsulation happens at the receiver site. When the message arrives at the destination transport layer, the header is dropped and the transport layer delivers the message to the process running at the application layer. The sender socket address is passed to the process in case it needs to respond to the message received.

Figure 23.7 *Encapsulation and decapsulation*



3. Error Control

Error control at the transport layer is responsible for

i. Detecting and discarding corrupted packets.

ii. Keeping track of lost and discarded packets and resending them.

iii. Recognizing duplicate packets and discarding them.

iv. Buffering out-of-order packets until the missing packets arrive.

a. Sequence Numbers

Error control requires that the sending transport layer knows which packet is to be resent and the receiving transport layer knows which packet is a duplicate, or which packet has arrived out of order. This can be done if the packets are numbered. We can add a field to the transport-layer packet to hold the sequence number of the packet.

When a packet is corrupted or lost, the receiving transport layer can somehow inform the sending transport layer to resend that packet using the sequence number. The receiving transport layer can also detect duplicate packets if two received packets have the same sequence number. The out-of-order packets can be recognized by observing gaps in the sequence numbers.

Packets are numbered sequentially. However, because we need to include the sequence number of each packet in the header, we need to set a limit. If the header of the packet allows m bits for the sequence number, the sequence numbers range from 0 to $2^m - 1$. The sequence numbers are modulo 2^m.

b. Acknowledgment

We can use both positive and negative signals as error control, but we discuss only positive signals, which are more common at the transport layer. The receiver side can send an acknowledgment (ACK) for each of a collection of packets that have arrived safe and sound. The receiver can simply discard the corrupted packets. The sender can detect lost packets if it uses a timer. When a packet is sent, the sender starts a timer. If an ACK does not arrive before the timer expires, the sender resends the packet. Duplicate packets can be silently discarded by the receiver. Out-of-order packets can be either discarded (to be treated as lost packets by the sender), or stored until the missing one arrives.

4. Combination of Flow and Error Control

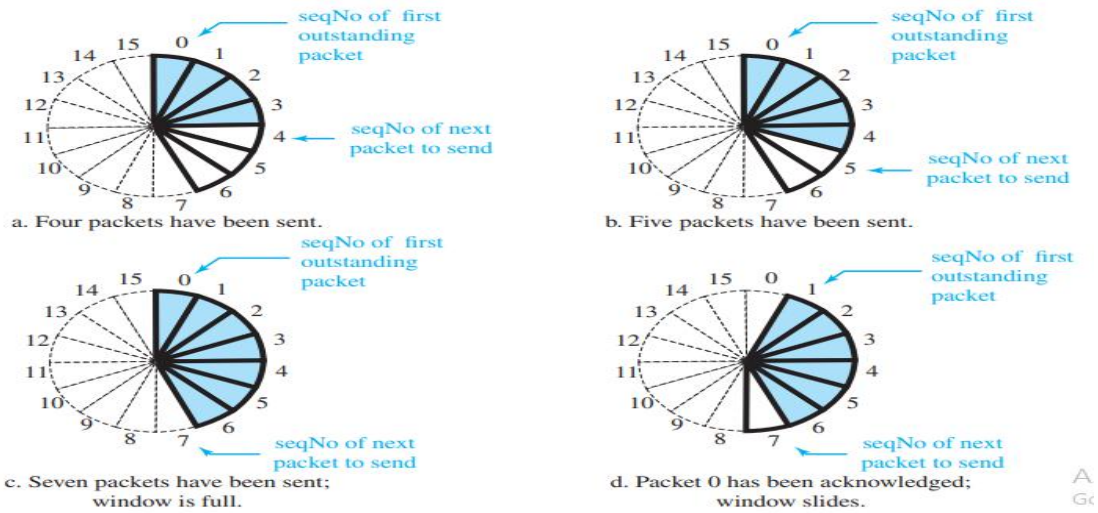
At the sender, when a packet is prepared to be sent, we use the number of the next free location, x, in the buffer as the sequence number of the packet. When the packet is sent, a copy is stored at memory location x, awaiting the acknowledgment from the other end. When an acknowledgment related to a sent packet arrives, the packet is purged and the memory location becomes free.

At the receiver, when a packet with sequence number y arrives, it is stored at the memory location y until the application layer is ready to receive it. An acknowledgment can be sent to announce the arrival of packet y.

a. Sliding Window

Since the sequence numbers use modulo 2^m, a circle can represent the sequence numbers from 0 to $2^m - 1$ (Figure 23.12). The buffer is represented as a set of slices, called the sliding window, that occupies part of the circle at any time. At the sender site, when a packet is sent, the corresponding slice is marked. When all the slices are marked, it means that the buffer is full and no further messages can be accepted from the application layer. When an acknowledgment arrives, the corresponding slice is unmarked. If some consecutive slices from the beginning of the window are unmarked, the window slides over the range of the corresponding sequence numbers to allow more free slices at the end of the window. Figure 23.12 shows the sliding window at the sender. The sequence numbers are in modulo 16 (m = 4) and the size of the window is 7. Note that the sliding window is just an abstraction: the actual situation uses computer variables to hold the sequence numbers of the next packet to be sent and the last packet sent.

Figure 23.12 *Sliding window in circular format*



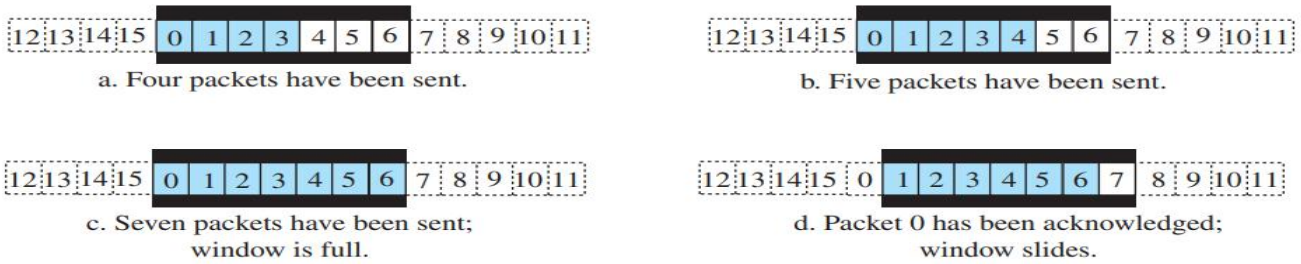
a. Four packets have been sent.

b. Five packets have been sent.

c. Seven packets have been sent; window is full.

d. Packet 0 has been acknowledged; window slides.

Figure 23.13 *Sliding window in linear format*



a. Four packets have been sent.

b. Five packets have been sent.

c. Seven packets have been sent; window is full.

d. Packet 0 has been acknowledged; window slides.

5. Congestion Control

An important issue in a packet-switched network, such as the Internet, is congestion. Congestion in a network may occur if the load on the network—the number of packets sent to the network—is greater than the capacity of the network—the number of packets a network can handle. Congestion control refers to the mechanisms and techniques that control the congestion and keep the load below the capacity.

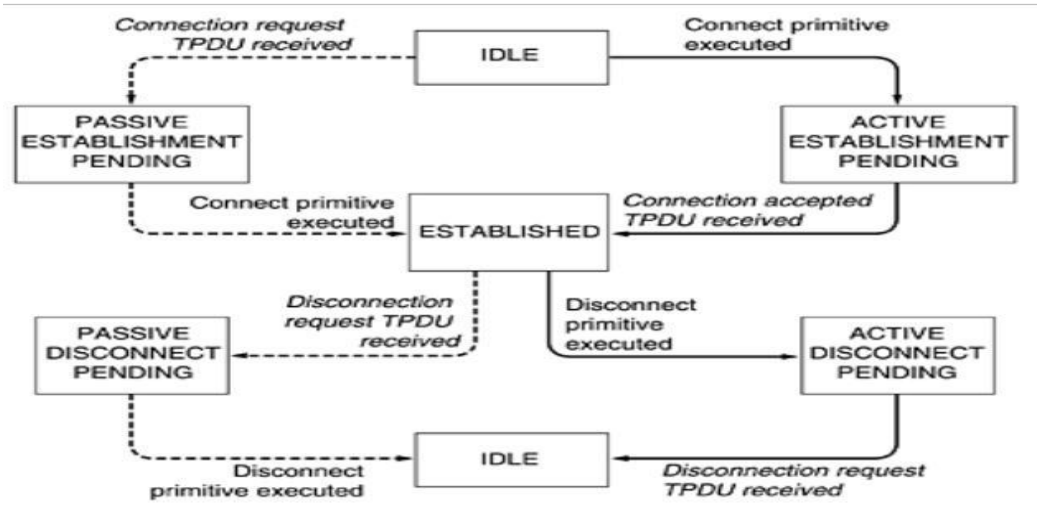
6. The primitives for a simple transport service.

A service is a set of primitives or we call it as operations where a user can invoke to access the service. The selective primitives tell the service to perform an action taken by peer nodes.

Figure 6-2. The primitives for a simple transport service.

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

A state diagram for a simple connection management scheme. Transitions labeled in italics are caused by packet arrivals. The solid lines show the client's state sequence. The dashed lines show the server's state sequence.



7. Berkeley Sockets

Figure 6-5. The socket primitives 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

On the SERVER side:

- 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.
- Newly-created sockets do not have network addresses. These are assigned using the BIND primitive.
- Once a server has bound an address to a socket, remote clients can connect to it.
- LISTEN call, allocates space to queue incoming calls for the case that several clients try to connect at the same time.
- To block waiting for an incoming connection, the server executes an ACCEPT primitive.
- When a TPDU asking for a connection arrives, the transport entity creates a new socket with the same properties as the original one and returns a file descriptor for it.
- The server can then fork off a process or thread to handle the connection on the new socket and go back to waiting for the next connection on the original socket.
- ACCEPT returns a normal file descriptor, which can be used for reading and writing in the standard way, the same as for files.

On the CLIENT side:

- Here a socket must first be created using the SOCKET primitive, but BIND is not required since the address used does not matter to the server.
- The CONNECT primitive blocks the caller and actively starts the connection process.
- When it completes (i.e., when the appropriate TPDU is received from the server), the client process is unblocked and the connection is established.

- Both sides can now use SEND and RECV to transmit and receive data over the full-duplex connection.
- The standard UNIX READ and WRITE system calls can also be used if none of the special options of SEND and RECV are required.
- Connection release with sockets is symmetric. When both sides have executed a CLOSE primitive, the connection is released.

Elements of Transport Protocols

1. Addressing

Port Address

Although there are a few ways to achieve process-to-process communication, the most common is through the client-server paradigm. A process on the local host, called a client, needs services from a process usually on the remote host, called a server.

For communication, we must define the local host, local process, remote host, and remote process. The local host and the remote host are defined using IP addresses. To define the processes, we need second identifiers, called port numbers. In the TCP/IP protocol suite, the port numbers are integers between 0 and 65,535 (16 bits).

The client program defines itself with a port number, called the ephemeral port number. The word ephemeral means "short-lived" and is used because the life of a client is normally short.

The server process must also define itself with a port number. This port number, however, cannot be chosen randomly. If the computer at the server site runs a server process and assigns a random number as the port number, the process at the client site that wants to access that server and use its services will not know the port number. Of course, one solution would be to send a special packet and request the port number of a specific server, but this creates more overhead. TCP/IP has decided to use universal port numbers for servers; these are called well-known port numbers.

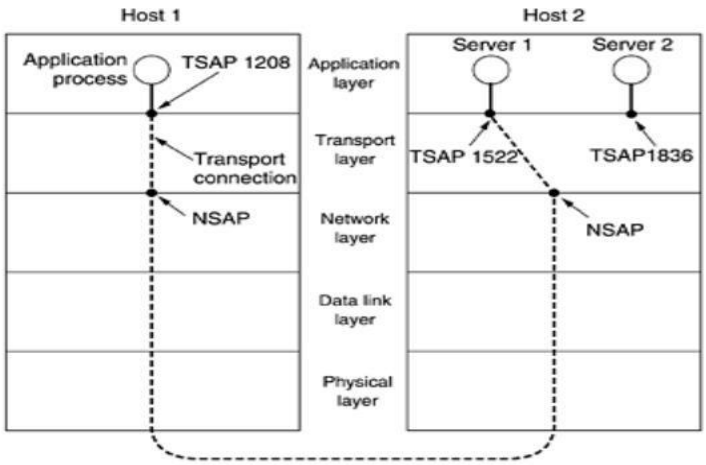
Socket Addresses

A transport-layer protocol in the TCP suite needs both the IP address and the port number, at each end, to make a connection. The combination of an IP address and a port number is called a socket address. The client socket address defines the client process uniquely just as the server socket address defines the server process uniquely.

We will use the generic term TSAP, (Transport Service Access Point). The analogous end points in the network layer (i.e., network layer addresses) are then called NSAPs. IP addresses are examples of NSAPs.

Figure 6-8 illustrates the relationship between the NSAP, TSAP and transport connection. 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.

Figure 6-8. TSAPs, NSAPs, and transport connections.

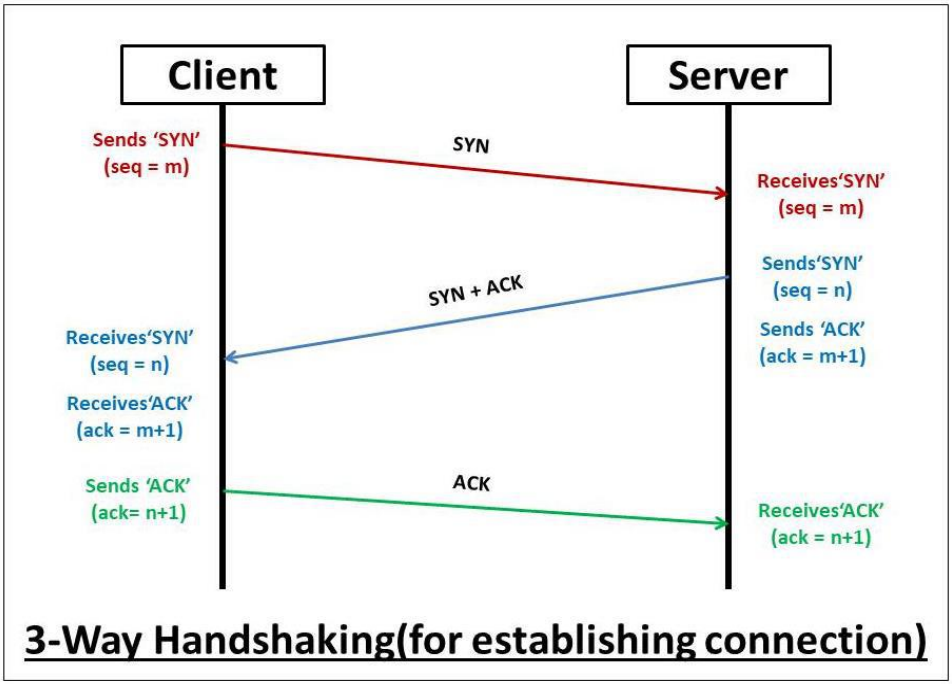


A possible scenario for a transport connection is as follows.
1. A time of day server process on host 2 attaches itself to TSAP 1522 to wait for an incoming call. How a process attaches itself to a TSAP is outside the networking model and depends entirely on the local operating system. A call such as 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.

2. Connection Establishment
Sender transport entity sends a CONNECTION REQUEST TPDU to the destination and waits for a CONNECTION ACCEPTED reply from the receiver.
The problem occurs when the network can lose, store, and duplicate packets.
Possible Solutions:
1. One way is to use throw-away transport addresses. In this approach, each time a transport address is needed, a new one is generated. When a connection is released, the address is discarded and never used again.
2. To give each connection a connection identifier (i.e., a sequence number incremented for each connection established) chosen by the initiating party and put in each TPDU, including the one requesting the connection. After each connection is released, each transport entity could update a table listing obsolete connections as (peer transport entity, connection identifier) pairs. Whenever a connection request comes in, it could be checked against the table, to see if it belonged to a previously-released connection.
Unfortunately, this scheme has a basic flaw: it requires each transport entity to maintain a certain amount of history information indefinitely. If a machine crashes and loses its memory, it will no longer know which connection identifiers have already been used.
Devise a mechanism to kill off aged packets that are still in the network.
Packet lifetime can be restricted to a known maximum using one (or more) of the following techniques:
a. Restricted subnet design.
b. Putting a hop counter in each packet.
c. Timestamping each packet.
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.
The second method consists of having the hop count initialized to some appropriate value and decremented each time the packet is forwarded. The network protocol simply discards any packet whose hop counter becomes zero.
The third method requires each packet to bear the time it was created, with the routers agreeing to discard any packet older than some agreedupon time. This latter method requires the router clocks to be synchronized, which itself is a nontrivial task unless synchronization is achieved external to the network, for example by using GPS or some radio station that broadcasts the precise time periodically.

Three-way handshake Protocol
The 3-Way Handshake process is the defined set of steps that takes place in the TCP for creating a secure and reliable communication link and also closing it.
Actually, TCP uses the 3-way handshake process to establish a connection between two devices before transmitting the data. After the establishment of the connection, the data transfer takes place between the devices. After which the connection needs to be terminated, which is also done by using the 3-way handshake process. The secure and reliable connection is established to reserve the CPU, buffer, and bandwidth of the devices to communicate properly. Thus, it is a must to free these resources by terminating the connection after data transmission. Hence, the TCP 3-way handshake process can be used to establish and terminate connections in the network in a secure way.

Connection establishment:
Below is the pictorial representation of the connection establishment using the 3-way handshake process.



Following are the three steps involved in establishing the connection using the 3-way handshake process in TCP:

1. The client sends the SYN to the server: When the client wants to connect to the server. It sets the 'SYN' flag as 1 and sends the message to the server. The message has also some additional information like the sequence number(any random 32 bits number), the ACK is set here to 0, the window size, and the maximum segment size. For Example, if the window size is 2000 bits, and the maximum segment size is 200 bits then a maximum of 10 data segments (2000/200 = 10) can be transmitted in the connection.
2. The server replies with the SYN and the ACK to the client: After receiving the client's synchronization request, the server sends an acknowledge to the client by setting the ACK flag to '1'. The acknowledgement number of the ACK is one more than the received sequence number. For Example, if the client has sent the SYN with sequence number = 1000, then the server will send the ACK with acknowledgement number = 10001. Also, the server sets the SYN flag to '1' and sends it to the client, if the server also wants to establish the connection. The sequence number used here for the SYN will be different from the client's SYN. The server also advertises its window size and maximum segment size to the client. After completion of this step, the connection is established from the client to the server-side.
3. The client sends the ACK to the server: After receiving the SYN from the server, the client sets the ACK flag to '1' and sends it with an acknowledgement number 1 greater than the server's SYN sequence number to the client. Here, the SYN flag is kept '0'. After completion of this step, the connection is now established from the server to the client-side also. After the connection is being established, the minimum of the sender's and receiver's maximum segment size is taken under consideration for data transmission.

3. Connection Release:

Releasing a connection is easier than establishing one. Nevertheless, there are more pitfalls than one might expect.

Style of terminating a connection:

- There are two styles of terminating a connection: asymmetric release and symmetric release.
- Asymmetric release is the way the telephone system works: when one party hangs up, the connection is broken.
- Symmetric release treats the connection as two separate unidirectional connections and requires each one to be released separately.

Abrupt Disconnections with Loss of data:

- Asymmetric release is abrupt and may result in data loss.
- Consider the scenario of Fig. 4.14.
- 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.

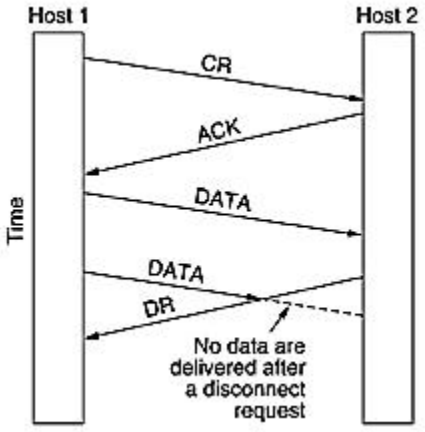
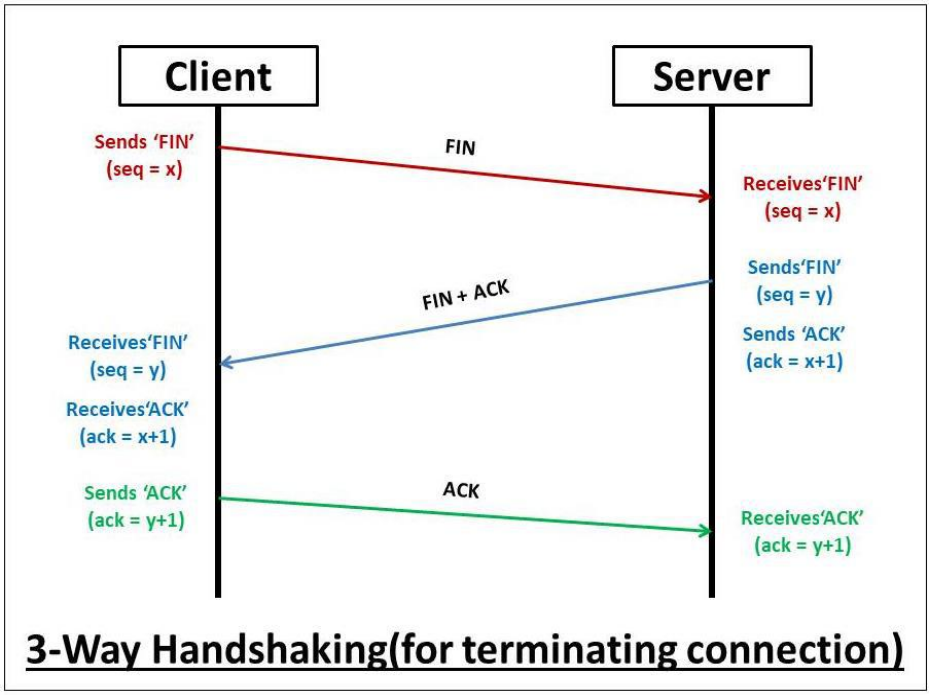


Figure 4.14 Abrupt disconnections with loss of data.

Symmetric Release:

- One way is to use symmetric release, in which each direction is released independently of the other one.
- Here, a host can continue to receive data even after it has sent a DISCONNECT TPDU.
- Symmetric release does the job when each process has a fixed amount of data to send and clearly knows when it has sent it.
- In other situations, determining that all the work has been done and the connection should be terminated is not so obvious.

Below is the pictorial representation of the connection termination using the 3-way handshake process.



Following are the three steps involved in terminating the connection using the 3-way handshake process in TCP:

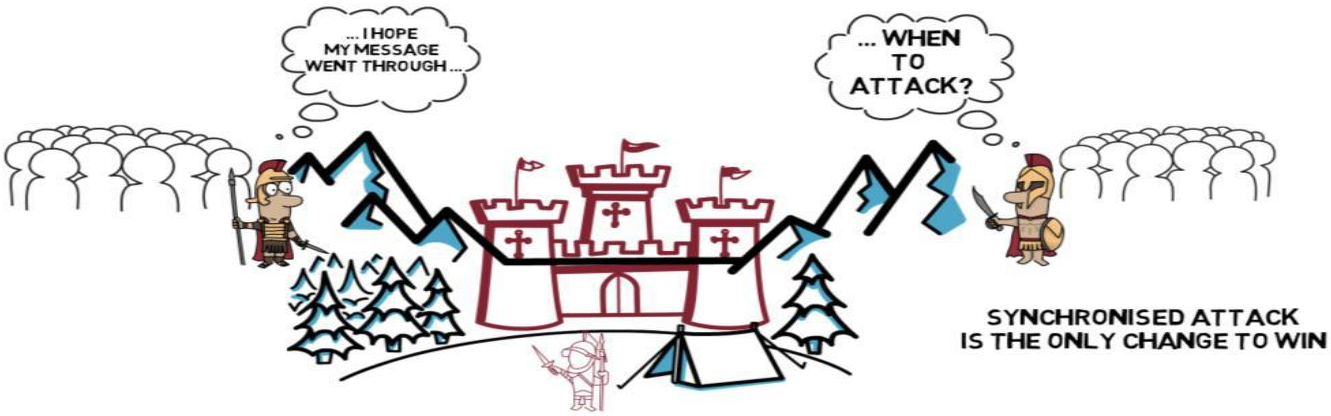
1. The client sends the FIN to the server: When the client wants to terminate the connection. It sets the FIN flag as '1' and sends the message to the server with a random sequence number. Here, the ACK is set to 0.
2. The server replies with the FIN and the ACK to the client: After receiving the client's termination request, the server sends an acknowledge to the client by setting the ACK flag to '1'. The acknowledgement number of the ACK is one more than the received sequence number. For Example, if the client has sent the FIN with sequence number = 1000, then the server will send the ACK with acknowledgement number = 10001. Also, the server sets the FIN flag to '1' and sends it to the client, if

the server also wants to terminate the connection. The sequence number used here for the FIN will be different from the client's FIN. After completion of this step, the connection is terminated from the client to the server-side.

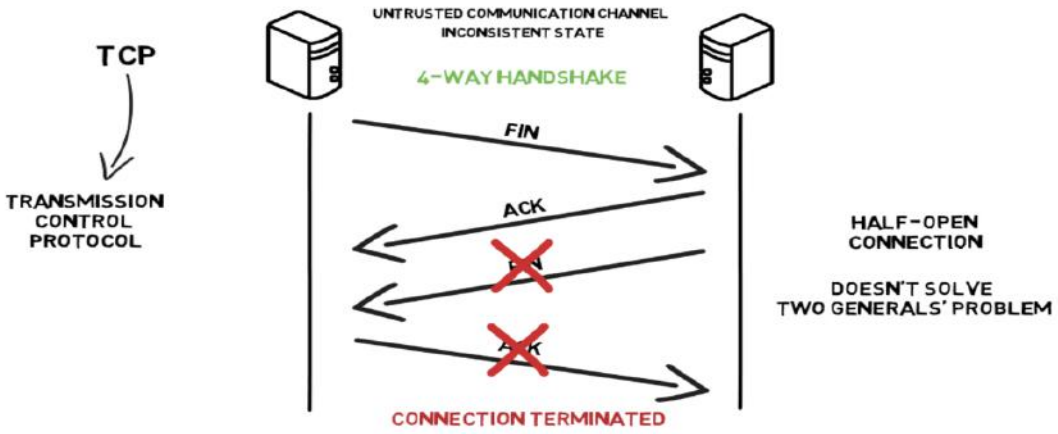
3. The client sends the ACK to the server: After receiving the FIN from the server, the client sets the ACK flag to '1' and sends it with an acknowledgement number 1 greater than the server's FIN sequence number to the client. Here, the FIN flag is kept '0'. After completion of this step, the connection is now terminated from the server to the client-side also.

https://afteracademy.com/blog/what-is-a-tcp-3-way-handshake-process/

Two General's Problem or Two armies Problem:
https://finematics.com/two-generals-problem/
The Two Generals' Problem, also known as the Two Generals' Paradox or the Two Armies Problem, is a classic computer science and computer communication thought experiment.
The story of the Two Generals



Let's imagine two armies, led by two generals, planning an attack on a common enemy. The enemy's city is in a valley and has a strong defence that can easily fight off a single army. The two generals have to communicate with each other to plan a synchronised attack as this is their only chance to win. The only problem is that to communicate with each other they have to send a messenger across the enemy's territory. If a messenger is captured the message he's carrying is lost. Also, each general wants to know that the other general knows when to attack. Otherwise, a general wouldn't be sure if he's attacking alone and as we know attacking alone is rather pointless.
Now, let's go through a simple scenario. Let's call our generals A and B and let's assume everything goes perfectly fine. General A, who is the leader, sends a message – "Attack tomorrow at dawn". General B receives a message and sends back an acknowledgement – "I confirm, attack tomorrow at dawn". A receives B's confirmation. Is this enough to form a consensus between the generals? Unfortunately not, as General B still doesn't know if his confirmation was received by General A. Ok, so what if General A confirms General's B confirmation? Then, of course, that confirmation has to be also confirmed and we end up with an infinite exchange of confirmations.
In the second scenario, let's also assume that General A sends a message to General B. Some time has passed and General A starts wondering what happened to his message as there is no confirmation coming back from General B. There are two possibilities here. Either the messenger sent by General A has been captured and hasn't delivered a message or maybe B's messenger carrying B's confirmation has been captured. In both scenarios, they cannot come to a consensus again as A is not able to tell if his message was lost or if it was B's confirmation that didn't get through. Again, we ended up in an inconsistent state which would result in either General A or B attacking by himself.
We can quickly realize that no matter how many different scenarios we try and how many messages we send we cannot guarantee that consensus is reached and each general is certain that his ally will attack at the same time. To make it even worse, there is no solution to the Two Generals' Problem, so the problem remains unsolvable.
How is this related to computer science and TCP?
Instead of two generals, let's imagine two computer systems talking to each other. The main problem here is again the untrusted communication channel and inconsistent state between two machines. A very common example that always comes up when talking about the Two Generals' Problem is the TCP protocol.



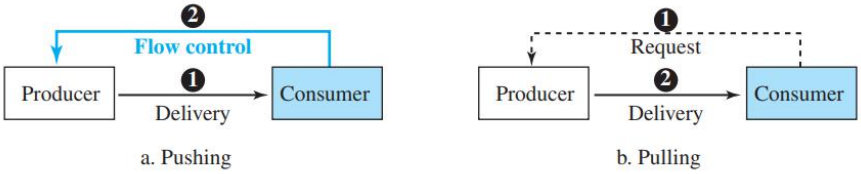
As we probably know, TCP uses a mechanism called 4-way handshake to terminate the connection. In this mechanism, a system that wants to terminate a connection sends a FIN message. The system on the other side of the communication channel replies with an ACK and sends its own FIN message which is followed by another ACK from the system which initialised termination. When all of those messages are received correctly, both sides know that the connection is terminated. So far it looks ok, but the problem here is again the shared knowledge between the two systems. When, for example, the second FIN is lost we end up with a half-open connection where one side is not aware that the connection has been closed. That's why even though TCP is very reliable protocol it doesn't solve the Two Generals' Problem.

So maybe a pragmatic approach?
Unsurprisingly, there was a number of people trying to solve unsolvable Two General's Problem and they came up with a few practical approaches. The main assumption here is to accept the uncertainty of the communication channel and mitigate it to a sufficient degree.
Let's go back to our generals. What if General A instead of sending only 1 messenger sends 100 of them assuming that General B will receive at least 1 message. How about marking each message with a serial number starting from 1 up to 100. General B, based on the missing numbers in the sequence, would be able to gauge how reliable the communication channel is and reply with an appropriate number of confirmations. These approaches, even though, quite expensive are helping the generals to build up their confidence and come to a consensus.
If sacrificing messengers is a problem, we can come up with yet another approach where the absence of the messengers would build up generals' confidence. Let's assume that it takes 20 minutes to cross the valley, deliver a message and come back. General A starts sending messengers every 20 minutes until he gets a confirmation from General B. Whenever confirmation arrives General A stops sending messengers. In the meantime, General B after sending his messenger with his confirmation awaits for the other messengers coming from General A, but this time an absence of a messenger builds up General's B confidence as this is what the Generals agreed on.
In this case, we have a clear speed vs cost tradeoff and it's up to us which approach is more suitable to our problem.
That's the end of the story of the Two Generals. Time for a quick summary.
Summary
Two Generals' Problem is a classic computer science problem that remains unsolvable.
It comes up whenever we talk about communication over an unreliable channel.
The main problem is an inconsistent state caused by lack of common knowledge.

4. Flow Control and Buffering
Flow Control
If the items are produced faster than they can be consumed, the consumer can be overwhelmed and may need to discard some items. If the items are produced more slowly than they can be consumed, the consumer must wait, and the system becomes less efficient. Flow control is related to the first issue. We need to prevent losing the data items at the consumer site.
a. Pushing or Pulling
Delivery of items from a producer to a consumer can occur in one of two ways: pushing or pulling. If the sender delivers items whenever they are produced – without a prior request from the consumer – the

delivery is referred to as pushing. If the producer delivers the items after the consumer has requested them, the delivery is referred to as pulling.

Figure 23.9 *Pushing or pulling*



When the producer pushes the items, the consumer may be overwhelmed and there is a need for flow control, in the opposite direction, to prevent discarding of the items. In other words, the consumer needs to warn the producer to stop the delivery and to inform the producer when it is again ready to receive the items. When the consumer pulls the items, it requests them when it is ready. In this case, there is no need for flow control.

b. Buffers

Although flow control can be implemented in several ways, one of the solutions is normally to use two buffers: one at the sending transport layer and the other at the receiving transport layer.

A buffer is a set of memory locations that can hold packets at the sender and receiver.

he flow control communication can occur by sending signals from the consumer to the producer.

When the buffer of the sending transport layer is full, it informs the application layer to stop passing chunks of messages; when there are some vacancies, it informs the application layer that it can pass message chunks again. When the buffer of the receiving transport layer is full, it informs the sending transport layer to stop sending packets. When there are some vacancies, it informs the sending transport layer that it can send packets again.

5. Multiplexing and Demultiplexing

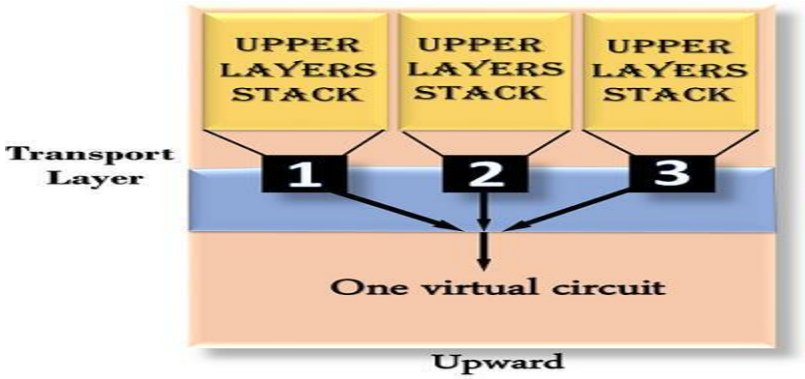
Multiplexing (many to one) is when data is acquired from several processes from the sender and merged into one packet along with headers and sent as a single packet. Multiplexing allows the simultaneous use of different processes over a network that is running on a host. The processes are differentiated by their port numbers.

Demultiplexing (one to many) is required at the receiver side when the message is distributed into different processes. Transport receives the segments of data from the network layer distributes and delivers it to the appropriate process running on the receiver's machine.

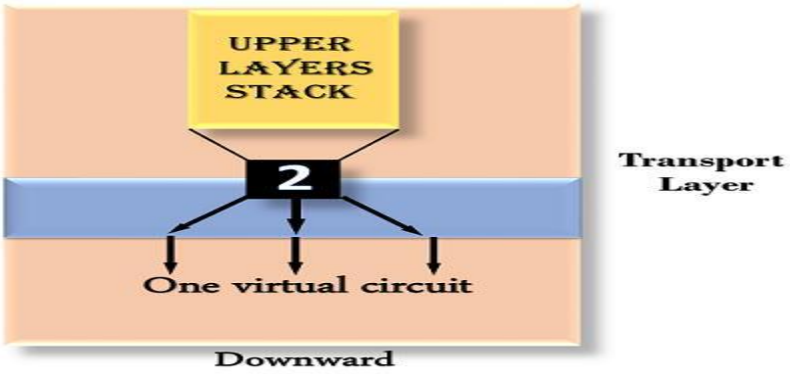
The transport layer uses the multiplexing to improve transmission efficiency.

Multiplexing can occur in two ways:

- Upward multiplexing**:** Upward multiplexing means multiple transport layer connections use the same network connection. To make more cost-effective, the transport layer sends several transmissions bound for the same destination along the same path; this is achieved through upward multiplexing.



- Downward multiplexing**:** Downward multiplexing means one transport layer connection uses the multiple network connections. Downward multiplexing allows the transport layer to split a connection among several paths to improve the throughput. This type of multiplexing is used when networks have a low or slow capacity.



6. Crash Recovery

If hosts and routers are subject to crashes, recovery from these crashes becomes an issue. If the transport entity is entirely within the hosts, recovery from network and router crashes is straightforward.

Problem from host crashes?

- In particular, it may be desirable for clients to be able to continue working when servers crash and then quickly reboot.
- let us assume that one host, the client, is sending a long file to another host, the file server, using a simple stop-and-wait protocol.
- The transport layer on the server simply passes the incoming TPDUs to the transport user, one by one. Partway through the transmission, the server crashes.
- When it comes back up, its tables are reinitialized, so it no longer knows precisely where it was.

How to recover from host crashes?

- In an attempt to recover its previous status, the server might send a broadcast TPDU to all other hosts, announcing that it had just crashed and requesting that its clients inform it of the status of all open connections.
- Each client can be in one of two states: one TPDU outstanding, S1, or no TPDUs outstanding,
- So, Based on only this state information, the client must decide whether to retransmit the most recent TPDU.
- At first glance it would seem obvious: the client should retransmit only if and only if it has an unacknowledged TPDU outstanding (i.e., is in state S1) when it learns of the crash.

Problem with this approach:

- Consider, for example, the situation in which the server's transport entity first sends an acknowledgement, and then, when the acknowledgement has been sent, writes to the application process.
- Writing a TPDU onto the output stream and sending an acknowledgement are two distinct events that cannot be done simultaneously.
- If a crash occurs after the acknowledgement has been sent but before the write has been done, the client will receive the acknowledgement and thus be in state S0 when the crash recovery announcement arrives.
- The client will therefore not retransmit, (incorrectly) thinking that the TPDU has arrived.
- This decision by the client leads to a missing TPDU.
- At this point you may be thinking: "That problem can be solved easily.
- All you have to do is reprogram the transport entity to first do the write and then send the acknowledgement." Try again.
- Imagine that the write has been done but the crash occurs before the acknowledgement can be sent.
- The client will be in state S1 and thus retransmit, leading to an undetected duplicate TPDU in the output stream to the server application process.

Different combinations of client and server strategy:

- The server can be programmed in one of two ways: acknowledge first or write first.
- The client can be programmed in one of four ways: always retransmit the last TPDU, never retransmit the last TPDU, retransmit only in state S0, or retransmit only in state S1.
- This gives eight combinations,

- Three events are possible at the server: sending an acknowledgement (A), writing to the output process (W), and crashing (C).
- The three events can occur in six different orderings: AC(W), AWC, C(AW), C(WA), WAC, and WC(A), where the parentheses are used to indicate that neither A nor W can follow C (i.e., once it has crashed, it has crashed).
- Figure 4.20 shows all eight combinations of client and server strategy and the valid event sequences for each one.
- Notice that for each strategy there is some sequence of events that causes the protocol to fail. For example, if the client always retransmits, the AWC event will generate an undetected duplicate, even though the other two events work properly.

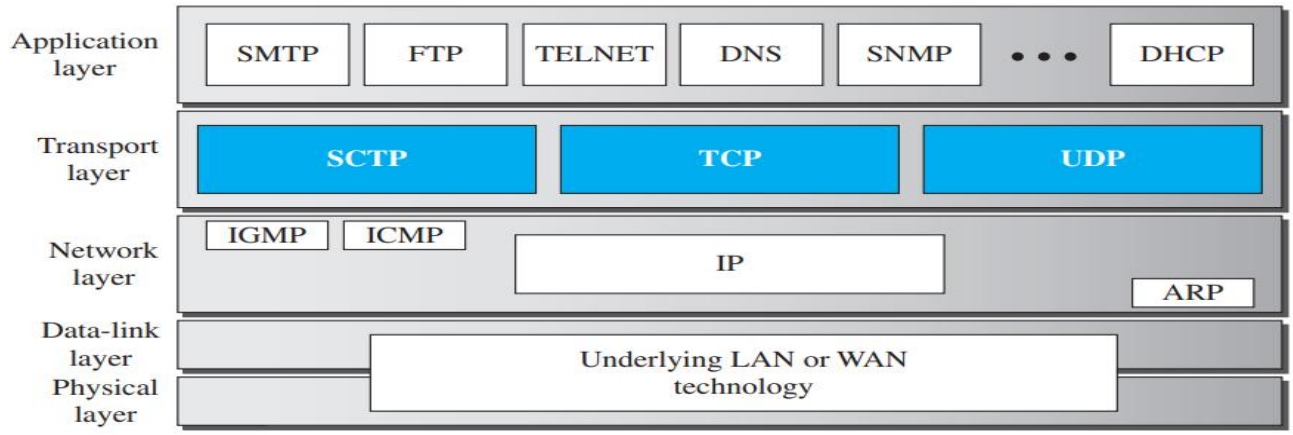
Strategy used by receiving host

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

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

Figure 4.20 Different combinations of client and server strategy.

Figure 24.1 *Position of transport-layer protocols in the TCP/IP protocol suite*



User datagram protocol(UDP)

The Internet has two main protocols in the transport layer, a connectionless protocol and a connection-oriented one. The User Datagram Protocol (UDP) is called a connectionless, unreliable transport protocol. It does not add anything to the services of IP except to provide process-to process communication instead of host-to-host communication.

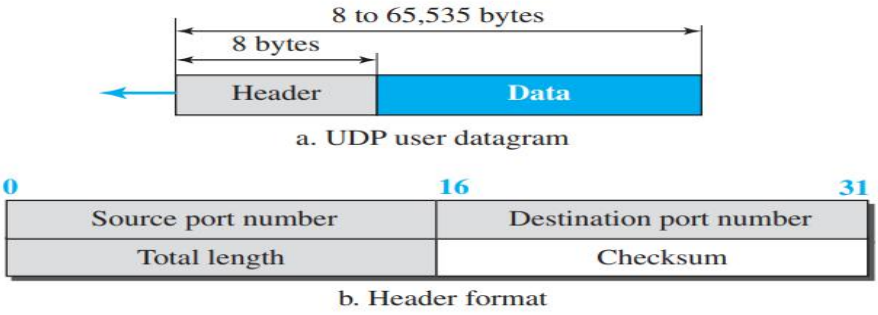
UDP is a very simple protocol using a minimum of overhead. If a process wants to send a small message and does not care much about reliability, it can use UDP. Sending a small message using UDP takes much less interaction between the sender and receiver than using TCP.

User Datagram

UDP packets, called user datagrams, have a fixed-size header of 8 bytes made of four fields, each of 2 bytes (16 bits). Figure 24.2 shows the format of a user datagram. The first two fields define the source and destination port numbers. The third field defines the total length of the user datagram, header plus data.

The 16 bits can define a total length of 0 to 65,535 bytes. However, the total length needs to be less because a UDP user datagram is stored in an IP datagram with the total length of 65,535 bytes. The last field can carry the optional checksum.

Figure 24.2 *User datagram packet format*



Example: The following is the content of a UDP header in hexadecimal format. CB84000D001C001C
a. What is the source port number?
b. What is the destination port number?
c. What is the total length of the user datagram?
d. What is the length of the data?
e. Is the packet directed from a client to a server or vice versa?
f. What is the client process?

a. The source port number is the first four hexadecimal digits (CB84)16, which means that the source port number is 52100.
b. The destination port number is the second four hexadecimal digits (000D)16, which means that the destination port number is 13.
c. The third four hexadecimal digits (001C)16 define the length of the whole UDP packet as 28 bytes.
d. The length of the data is the length of the whole packet minus the length of the header, or 28 − 8 = 20 bytes.
e. Since the destination port number is 13 (well-known port), the packet is from the client to the server.
f. The client process is the Daytime

UDP Features
1. Connectionless Service
UDP is a connectionless protocol.
Each UDP packet is independent from other packets sent by the same application program. This feature can be considered as an advantage or disadvantage depending on the application requirements. It is an advantage if, for example, a client application needs to send a short request to a server and to receive a short response. If the request and response can each fit in a single user datagram, a connectionless service may be preferable. The overhead to establish and close a connection may be significant in this case.
The connectionless service provides less delay; the connection-oriented service creates more delay. If delay is an important issue for the application, the connectionless service is preferred.
2. Lack of Error Control
UDP does not provide error control; it provides an unreliable service. Most applications expect reliable service from a transport-layer protocol. Although a reliable service is desirable, it may have some side effects that are not acceptable to some applications. When a transport layer provides reliable services, if a part of the message is lost or corrupted, it needs to be resent. This means that the receiving transport layer cannot deliver that part to the application immediately; there is an uneven delay between different parts of the message delivered to the application layer.
3. Lack of Congestion Control
UDP does not provide congestion control. However, UDP does not create additional traffic in an error-prone network. TCP may resend a packet several times and thus contribute to the creation of congestion or worsen a congested situation.

TRANSMISSION CONTROL PROTOCOL

Transmission Control Protocol (TCP) is a connection-oriented, reliable protocol. TCP explicitly defines connection establishment, data transfer, and connection teardown phases to provide a connection-oriented service. TCP uses a combination of GBN and SR protocols to provide reliability. To achieve this goal, TCP uses checksum (for error detection), retransmission of lost or corrupted packets, cumulative and selective acknowledgments, and timers.

TCP Services

1. Process-to-Process Communication: As with UDP, TCP provides process-to-process communication using port numbers.

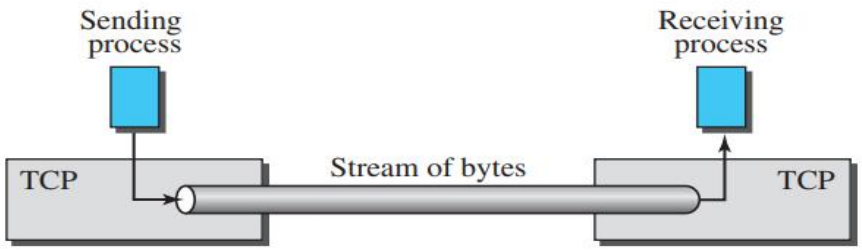
2. Stream Delivery Service

TCP, unlike UDP, is a stream-oriented protocol.

In UDP, a process sends messages with predefined boundaries to UDP for delivery. UDP adds its own header to each of these messages and delivers it to IP for transmission. Each message from the process is called a user datagram, and becomes, eventually, one IP datagram. Neither IP nor UDP recognizes any relationship between the datagrams.

TCP, on the other hand, allows the sending process to deliver data as a stream of bytes and allows the receiving process to obtain data as a stream of bytes. TCP creates an environment in which the two processes seem to be connected by an imaginary "tube" that carries their bytes across the Internet.

Figure 24.4 *Stream delivery*



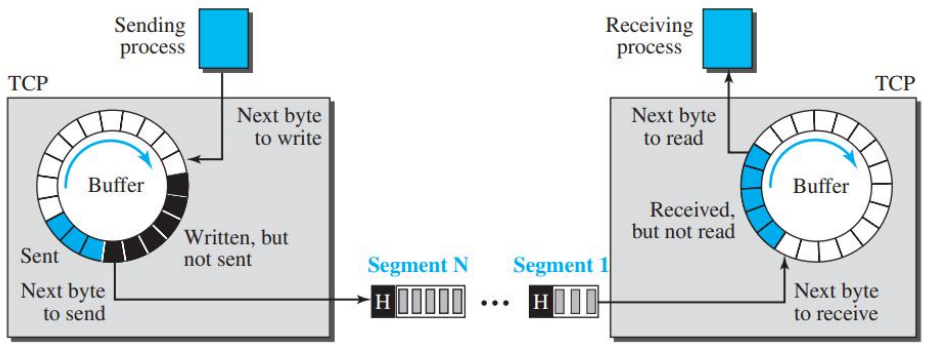
3. Sending and Receiving Buffers

Because the sending and the receiving processes may not necessarily write or read data at the same rate, TCP needs buffers for storage. There are two buffers, the sending buffer and the receiving buffer, one for each direction.

4. Segments

The network layer, as a service provider for TCP, needs to send data in packets, not as a stream of bytes. At the transport layer, TCP groups a number of bytes together into a packet called a segment. TCP adds a header to each segment (for control purposes) and delivers the segment to the network layer for transmission. The segments are encapsulated in an IP datagram and transmitted. This entire operation is transparent to the receiving process.

Figure 24.6 *TCP segments*



Note that segments are not necessarily all the same size.

5. Full-Duplex Communication
TCP offers full-duplex service, where data can flow in both directions at the same time. Each TCP endpoint then has its own sending and receiving buffer, and segments move in both directions.

6. Multiplexing and Demultiplexing
Like UDP, TCP performs multiplexing at the sender and demultiplexing at the receiver. However, since TCP is a connection-oriented protocol, a connection needs to be established for each pair of processes.

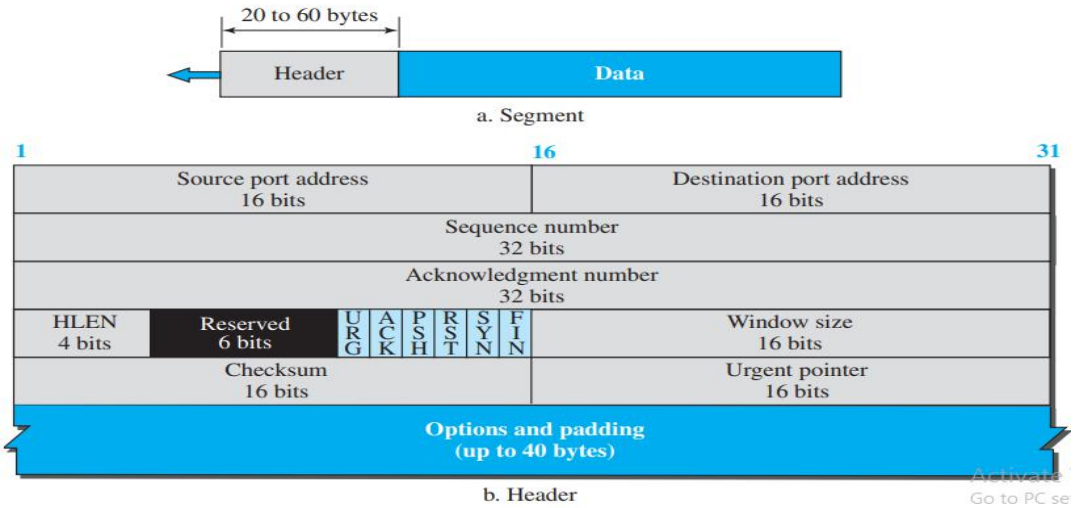
7. Connection-Oriented Service
TCP, unlike UDP, is a connection-oriented protocol. When a process at site A wants to send to and receive data from another process at site B, the following three phases occur:
1. The two TCP's establish a logical connection between them.
2. Data are exchanged in both directions.
3. The connection is terminated.
Note that this is a logical connection, not a physical connection. The TCP segment is encapsulated in an IP datagram and can be sent out of order, or lost or corrupted, and then resent. Each may be routed over a different path to reach the destination. There is no physical connection. TCP creates a stream-oriented environment in which it accepts the responsibility of delivering the bytes in order to the other site.

8. Reliable Service
TCP is a reliable transport protocol. It uses an acknowledgment mechanism to check the safe and sound arrival of data.

Segment

A packet in TCP is called a segment.
The format of a segment is shown in Figure 24.7. The segment consists of a header of 20 to 60 bytes, followed by data from the application program. The header is 20 bytes if there are no options and up to 60 bytes if it contains options

Figure 24.7 *TCP segment format*



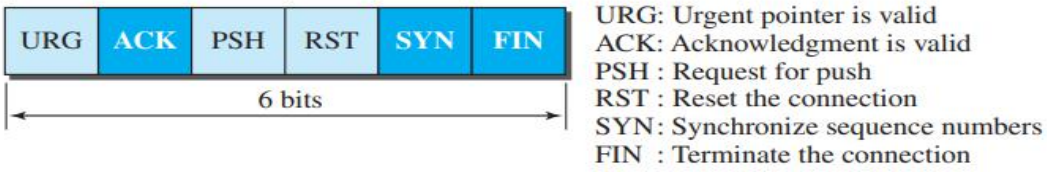
Source port address. This is a 16-bit field that defines the port number of the application program in the host that is sending the segment.
Destination port address. This is a 16-bit field that defines the port number of the application program in the host that is receiving the segment.
Sequence number. This 32-bit field defines the number assigned to the first byte of data contained in this segment. As we said before, TCP is a stream transport protocol. To ensure connectivity, each byte to be transmitted is numbered. The sequence number tells the destination which byte in this sequence is the first byte in the segment. During connection establishment, each party uses a random number generator to create an initial sequence number (ISN), which is usually different in each direction.
Acknowledgment number. This 32-bit field defines the byte number that the receiver of the segment is expecting to receive from the other party. If the receiver of the segment has successfully received byte

number x from the other party, it returns x + 1 as the acknowledgment number. Acknowledgment and data can be piggybacked together.

Header length. This 4-bit field indicates the number of 4-byte words in the TCP header. The length of the header can be between 20 and 60 bytes. Therefore, the value of this field is always between 5 (5 × 4 = 20) and 15 (15 × 4 = 60).

Control. This field defines 6 different control bits or flags, as shown in Figure 24.8. One or more of these bits can be set at a time. These bits enable flow control, connection establishment and termination, connection abortion, and the mode of data transfer in TCP. A brief description of each bit is shown in the figure.

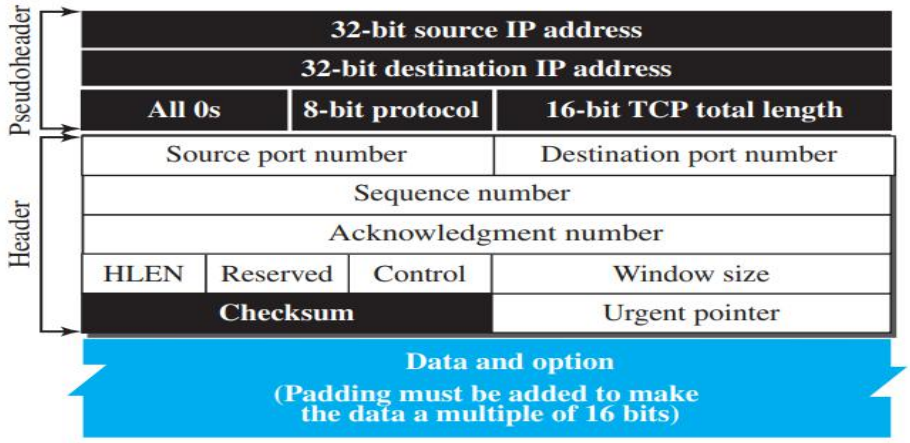
Figure 24.8 *Control field*



Window size. This field defines the window size of the sending TCP in bytes. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65,535 bytes. This value is normally referred to as the receiving window (rwnd) and is determined by the receiver. The sender must obey the dictation of the receiver in this case.

Checksum. This 16-bit field contains the checksum. The calculation of the checksum for TCP follows the same procedure as the one described for UDP. However, the use of the checksum in the UDP datagram is optional, whereas the use of the checksum for TCP is mandatory. The same pseudoheader, serving the same purpose, is added to the segment. For the TCP pseudoheader, the value for the protocol field is 6. The use of the checksum in TCP is mandatory.

Figure 24.9 *Pseudoheader added to the TCP datagram*



Urgent pointer. This 16-bit field, which is valid only if the urgent flag is set, is used when the segment contains urgent data. It defines a value that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.

Options. There can be up to 40 bytes of optional information in the TCP header

Encapsulation

A TCP segment encapsulates the data received from the application layer. The TCP segment is encapsulated in an IP datagram, which in turn is encapsulated in a frame at the data-link layer.

A TCP Connection

TCP is connection-oriented. A connection-oriented transport protocol establishes a virtual path between the source and destination. In TCP, connection-oriented transmission requires three phases: connection establishment, data transfer, and connection termination.

Connection Establishment:

Three-Way Handshaking: The connection establishment in TCP is called three-way handshaking.

- The process starts with the server.
- The server program tells its TCP that it is ready to accept a connection. This is called a request for a passive open.
- The client program issues a request for an active open.
- A client that wishes to connect to an open server tells its TCP that it needs to be connected to that particular server.
- TCP can now start the three-way handshaking process as shown in Figure 4.29.
- To show the process, we use two time lines: one at each site.
- Each segment has values for all its header fields and perhaps for some of its option fields, too.

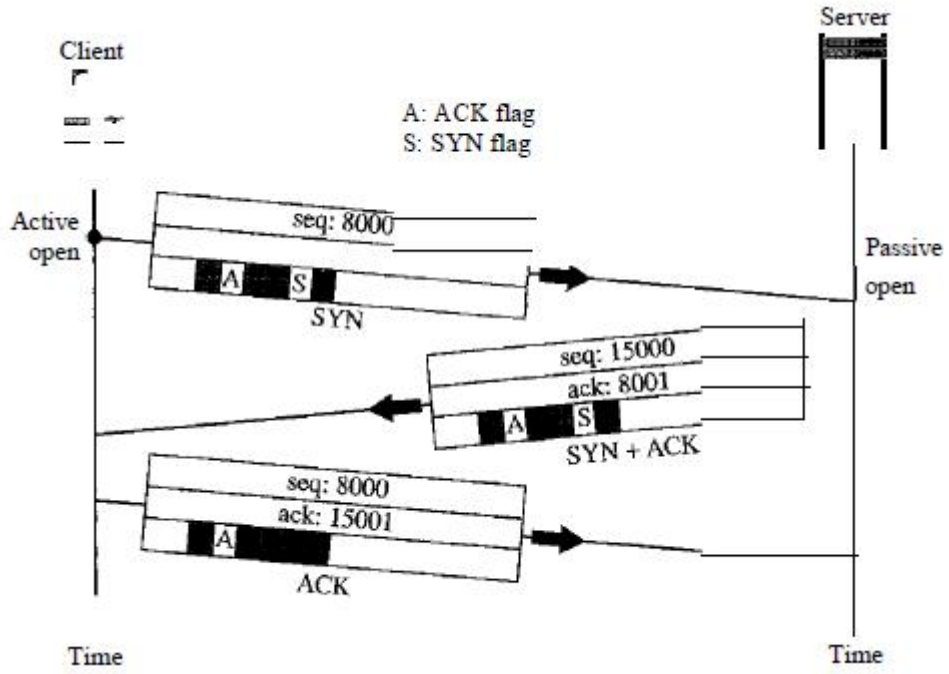


Figure 4.29 Connection establishment using three-way handshaking

The three steps in this phase are as follows.

1. The client sends the first segment, a SYN segment, in which only the SYN flag is set.
- This segment is for synchronization of sequence numbers. It consumes one sequence number.
- When the data transfer starts, the sequence number is incremented by 1.
- We can say that the SYN segment carries no real data, but we can think of it as containing 1 imaginary byte.

2. The server sends the second segment, a SYN ACK segment, with 2 flag bits set: SYN and ACK.
- This segment has a dual purpose. It is a SYN segment for communication in the other direction and serves as the acknowledgment for the SYN segment.
- It consumes one sequence number.

3. The client sends the third segment.
- This is just an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field.
- Note that the sequence number in this segment is the same as the one in the SYN segment; the ACK segment does not consume any sequence numbers.

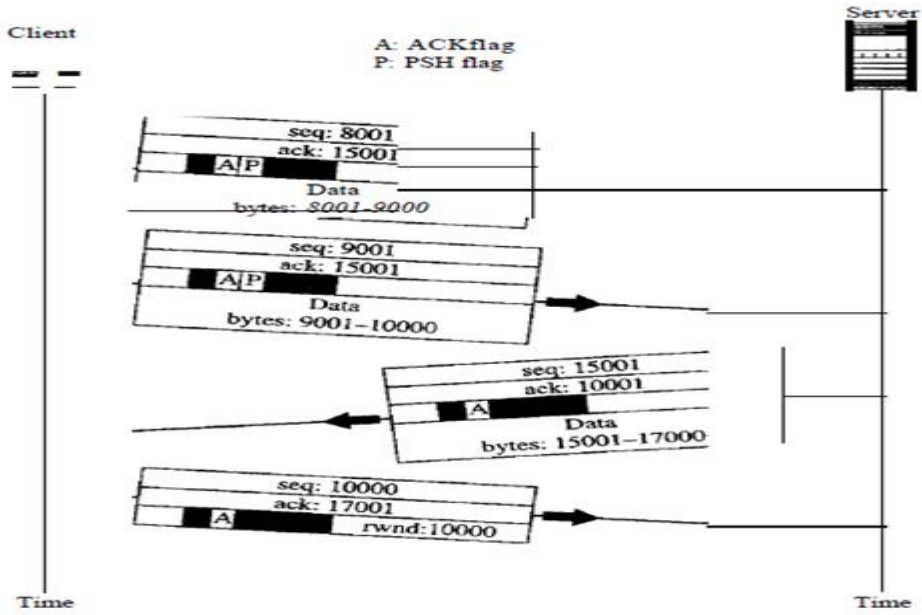
Data Transfer in TCP

Data transfer and Acknowledgement:

- After connection is established, bidirectional data transfer can take place.
- The client and server can both send data and acknowledgments.
- The acknowledgment is piggybacked with the data.

  Figure 4.30 shows an example. In this example, after connection is established (not shown in the figure), the client sends 2000 bytes of data in two segments. The server then sends 2000 bytes in one segment. The client sends one more segment. The first three segments carry both data and

acknowledgment, but the last segment carries only an acknowledgment because there are no more data to be sent. The data segments sent by the client have the PSH (push) flag set so that the server TCP knows to deliver data to the server process as soon as they are received. The segment from the server, on the other hand, does not set the push flag.



Connection Termination

Option for connection termination:

- Most implementations today allow two options for connection termination: three-way handshaking and four-way handshaking with a half-close option.

Three way Handshaking:

Three-way handshaking for connection termination as shown in Figure 4.31.

1. In a normal situation, the client TCP, after receiving a close command from the client process, sends the first segment, a FIN segment in which the FIN flag is set.
   - A FIN segment can include the last chunk of data sent by the client, or it can be just a control segment as shown in Figure 4.31.
   - If it is only a control segment, it consumes only one sequence number.

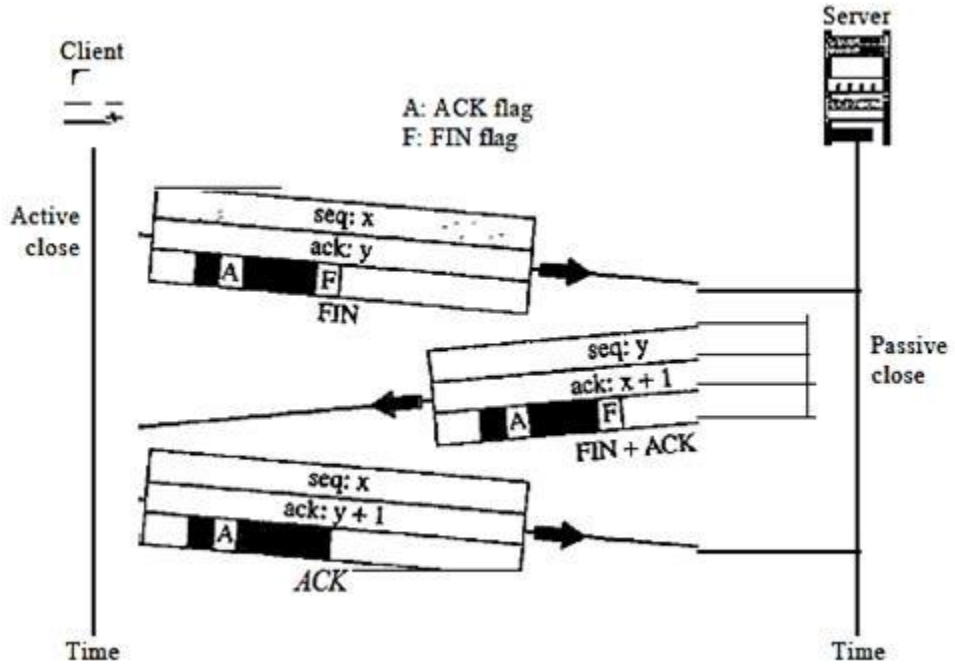


Figure 4.31 Connection termination using three-way handshaking

2. The server TCP, after receiving the FIN segment, informs its process of the situation and sends the second segment, a FIN ACK segment, to confirm the receipt of the FIN segment from the client and at the same time to announce the closing of the connection in the other direction.

- This segment can also contain the last chunk of data from the server. If it does not carry data, it consumes only one sequence number.

3. The client TCP sends the last segment, an ACK segment, to confirm the receipt of the FIN segment from the TCP server.
   - This segment contains the acknowledgment number, which is 1 plus the sequence number received in the FIN segment from the server.
   - This segment cannot carry data and consumes no sequence numbers.

Error Control for TCP

TCP provides reliability using error control. Error control includes mechanisms for detecting corrupted segments, lost segments, out-of-order segments, and duplicated segments. Error detection and correction in TCP is achieved through the use of three simple tools: checksum, acknowledgment, and time-out.

Checksum

- Each segment includes a checksum field which is used to check for a corrupted segment.
- If the segment is corrupted, it is discarded by the destination TCP and is considered as lost. TCP uses a 16-bit checksum that is mandatory in every segment.

Acknowledgment

- TCP uses acknowledgments to confirm the receipt of data segments.
- Control segments that carry no data but consume a sequence number are also acknowledged.
- ACK segments are never acknowledged.

Retransmission

- The heart of the error control mechanism is the retransmission of segments.
- When a segment is corrupted, lost, or delayed, it is retransmitted.
- In modern implementations, a segment is retransmitted on two occasions: when a retransmission timer expires or when the sender receives three duplicate ACKs.

Retransmission after RTO

- A recent implementation of TCP maintains one retransmission time-out (RTO) timer for all outstanding (sent, but not acknowledged) segments.
- When the timer matures, the earliest outstanding segment is retransmitted even though lack of a received ACK can be due to a delayed segment, a delayed ACK, or a lost acknowledgment.
- Note that no time-out timer is set for a segment that carries only an acknowledgment, which means that no such segment is resent.
- The value of RTO is dynamic in TCP and is updated based on the round-trip time (RTT) of segments. An RTI is the time needed for a segment to reach a destination and for an acknowledgment to be received.

TCP Congestion Control

To control the number of segments to transmit, TCP uses another variable called a congestion window, cwnd, whose size is controlled by the congestion situation in the network. The cwnd variable and the rwnd variable together define the size of the send window in TCP. The first is related to the congestion in the middle (network); the second is related to the congestion at the end. The actual size of the window is the minimum of these two. Lost acknowledgments may create deadlock if they are not properly handled.

The receive window size determines the number of bytes that the receive window can accept from the sender before being overwhelmed (flow control). The receive window size, normally called rwnd, can be determined as:

rwnd = buffer size - number of waiting bytes to be pulled

Actual window size = minimum (rwnd, cwnd)

Congestion Detection

The TCP sender uses the occurrence of two events as signs of congestion in the network: time-out and receiving three duplicate ACKs.

Congestion Policies

TCP's general policy for handling congestion is based on three algorithms: slow start, congestion avoidance, and fast recovery.

i. Slow Start: Exponential Increase The slow-start algorithm is based on the idea that the size of the congestion window (cwnd) starts with one maximum segment size (MSS), but it increases one MSS each time an acknowledgment arrives.
If an ACK arrives, cwnd = cwnd + 1.
If we look at the size of the cwnd in terms of round-trip times (RTTs), we find that the growth rate is exponential in terms of each round trip time, which is a very aggressive approach:

Start	→	$cwnd = 1 \rightarrow 2^0$
After 1 RTT	→	$cwnd = cwnd + 1 = 1 + 1 = 2 \rightarrow 2^1$
After 2 RTT	→	$cwnd = cwnd + 2 = 2 + 2 = 4 \rightarrow 2^2$
After 3 RTT	→	$cwnd = cwnd + 4 = 4 + 4 = 8 \rightarrow 2^3$

A slow start cannot continue indefinitely. There must be a threshold to stop this phase. The sender keeps track of a variable named ssthresh (slow-start threshold). When the size of the window in bytes reaches this threshold, slow start stops and the next phase starts.

ii. Congestion Avoidance: Additive Increase
When the size of the congestion window reaches the slow-start threshold in the case where cwnd = i, the slow-start phase stops and the additive phase begins. In this algorithm, each time the whole "window" of segments is acknowledged, the size of the congestion window is increased by one. A window is the number of segments transmitted during RTT
If an ACK arrives, cwnd = cwnd + (1/cwnd).

iii. Fast Recovery
It starts when three duplicate ACKs arrive, which is interpreted as light congestion in the network. Like congestion avoidance, this algorithm is also an additive increase, but it increases the size of the congestion window when a duplicate ACK arrives (after the three duplicate ACKs that trigger the use of this algorithm).
If a duplicate ACK arrives, cwnd = cwnd + (1 / cwnd).

What it does to try to prevent congestion from occurring in the first place?

- When a connection is established, a suitable window size has to be chosen.
- The receiver can specify a window based on its buffer size.
- If the sender sticks to this window size, problems will not occur due to buffer overflow at the receiving end, but they may still occur due to internal congestion within the network.
- In Fig. 4.34, we see this problem illustrated hydraulically.
- In Fig. 4.34(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. 4.34(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).

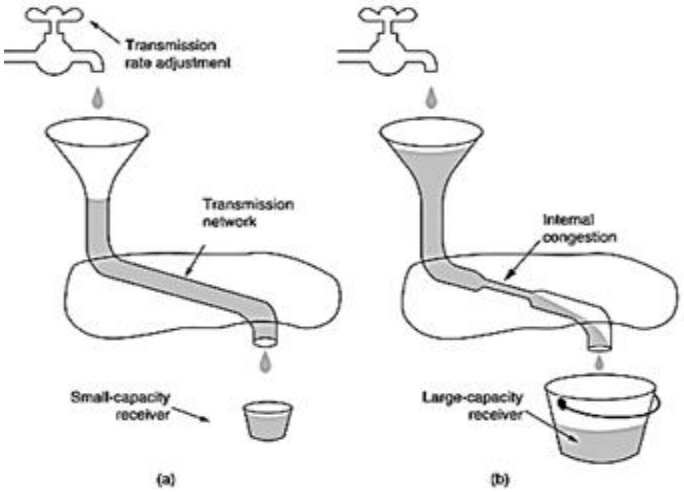


Figure 4.34 (a) A fast network feeding a low-capacity receiver. (b) A slow network feeding a high-capacity receiver

Internet Solution:

- The Internet solution is to realize that two potential problems exist-network capacity and receiver capacity-and to deal with each of them separately.
- To do so, each sender maintains two windows**:** the window the receiver has granted and a second window, the congestion window.
- Each reflects the number of bytes the sender may transmit.
- The number of bytes that may be sent is the minimum of the two windows.

Slow Start:

- When a connection is established, the sender initializes the congestion window to the size of the maximum segment in use on the connection.
- It then sends one maximum segment.
- If this 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 two segments.
- As each of these segments is acknowledged, the congestion window is increased by one maximum 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.
- In effect, each burst acknowledged doubles the congestion window.
- The congestion window keeps growing exponentially until either a timeout occurs or the receiver's window is reached.
- The idea is that if bursts of size, say, 1024, 2048, and 4096 bytes work fine but a burst of 8192 bytes gives a timeout, the congestion window should be set to 4096 to avoid congestion.
- As long as the congestion window remains at 4096, no bursts longer than that will be sent, no matter how much window space the receiver grants.
- This algorithm is called slow start, but it is not slow at all. It is exponential.

TCP Timers

To perform their operations smoothly, most TCP implementations use at least four timers: retransmission, persistence, keepalive, and TIME-WAIT.

i. Retransmission Timer

To retransmit lost segments, TCP employs one retransmission timer (for the whole connection period) that handles the retransmission time-out (RTO), the waiting time for an acknowledgment of a segment. We can define the following rules for the retransmission timer:

1. When TCP sends the segment in front of the sending queue, it starts the timer.
2. When the timer expires, TCP resends the first segment in front of the queue, and restarts the timer.
3. When a segment or segments are cumulatively acknowledged, the segment or segments are purged from the queue.

4. If the queue is empty, TCP stops the timer; otherwise, TCP restarts the timer.

ii. Persistence Timer
When the sending TCP receives an acknowledgment with a window size of zero, it starts a persistence timer. When the persistence timer goes off, the sending TCP sends a special segment called a probe. This segment contains only 1 byte of new data. It has a sequence number, but its sequence number is never acknowledged; it is even ignored in calculating the sequence number for the rest of the data. The probe causes the receiving TCP to resend the acknowledgment.
The value of the persistence timer is set to the value of the retransmission time.
However, if a response is not received from the receiver, another probe segment is sent and the value of the persistence timer is doubled and reset. The sender continues sending the probe segments and doubling and resetting the value of the persistence timer until the value reaches a threshold (usually 60 s). After that the sender sends one probe segment every 60 seconds until the window is reopened.

iii. Keepalive Timer
Each time the server hears from a client, it resets this timer. The time-out is usually 2 hours. If the server does not hear from the client after 2 hours, it sends a probe segment. If there is no response after 10 probes, each of which is 75 seconds apart, it assumes that the client is down and terminates the connection.

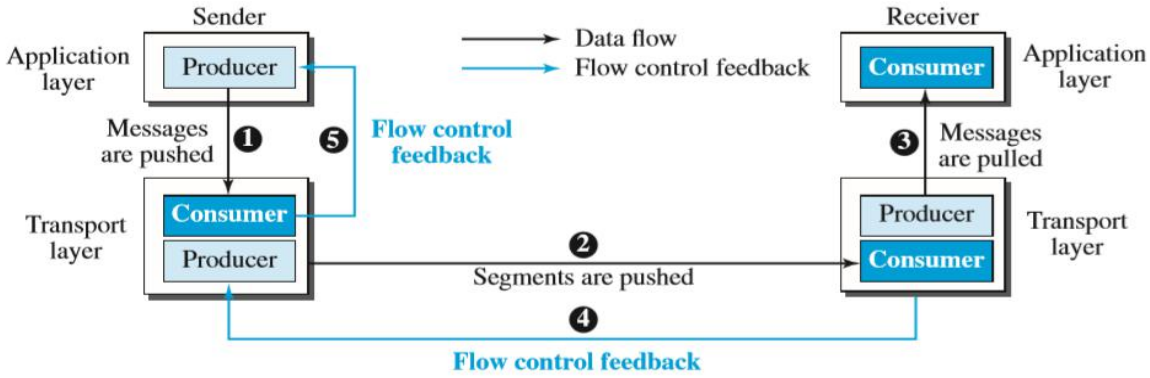
iv. TIME-WAIT Timer
The TIME-WAIT (2MSL) timer is used during connection termination. The maximum segment lifetime (MSL) is the amount of time any segment can exist in a network before being discarded. The implementation needs to choose a value for MSL. Common values are 30 seconds, 1 minute, or even 2 minutes. The 2MSL timer is used when TCP performs an active close and sends the final ACK. The connection must stay open for 2 MSL amount of time to allow TCP to resend the final ACK in case the ACK is lost. This requires that the RTO timer at the other end times out and new FIN and ACK segments are resent.

Options
The Options field provides a way to add extra facilities 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. Using large segments is more efficient than using small ones because the 20-byte header can then be amortized over more data, but small hosts may not be able to handle big segments. During connection setup, each side can announce its maximum and see its partner's. If a host does not use this option, it defaults to a 536-byte payload. All Internet hosts are required to accept TCP segments of 536 + 20 = 556 bytes. The maximum segment size in the two directions need not be the same.

Flow Control
Flow control balances the rate a producer creates data with the rate a consumer can use the data. TCP separates flow control from error control.

Figure 24.19 shows unidirectional data transfer between a sender and a receiver; bidirectional data transfer can be deduced from the unidirectional process.

Figure 24.19 *Data flow and flow control feedbacks in TCP*



The figure shows that data travel from the sending process down to the sending TCP, from the sending TCP to the receiving TCP, and from the receiving TCP up to the receiving process (paths 1, 2, and 3). Flow control feedbacks, however, are traveling from the receiving TCP to the sending TCP and from the sending TCP up to the sending process (paths 4 and 5). Most implementations of TCP do not provide flow control feedback from the receiving process to the receiving TCP; they let the receiving process pull data from the receiving TCP whenever it is ready to do so. In other words, the receiving TCP controls the sending TCP; the sending TCP controls the sending process.

Flow control feedback from the sending TCP to the sending process (path 5) is achieved through simple rejection of data by the sending TCP when its window is full. This means that our discussion of flow control concentrates on the feedback sent from the receiving TCP to the sending TCP (path 4).

Opening and Closing Windows

To achieve flow control, TCP forces the sender and the receiver to adjust their window sizes, although the size of the buffer for both parties is fixed when the connection is established. The receive window closes (moves its left wall to the right) when more bytes arrive from the sender; it opens (moves its right wall to the right) when more bytes are pulled by the process. We assume that it does not shrink (the right wall does not move to the left).

The opening, closing, and shrinking of the send window is controlled by the receiver. The send window closes (moves its left wall to the right) when a new acknowledgment allows it to do so. The send window opens (its right wall moves to the right) when the receive window size (rwnd) advertised by the receiver allows it to do so (new ackNo + new rwnd > last ackNo + last rwnd). The send window shrinks in the event this situation does not occur.

Shrinking of Windows

The receive window cannot shrink. The send window, can shrink if the receiver defines a value for rwnd that results in shrinking the window.

However, some implementations do not allow shrinking of the send window. The limitation does not allow the right wall of the send window to move to the left. In other words, the receiver needs to keep the following relationship between the last and new acknowledgment and the last and new rwnd values to prevent shrinking of the send window.

new ackNo + new rwnd ≥ last ackNo + last rwnd

The left side of the inequality represents the new position of the right wall with respect to the sequence number space; the right side shows the old position of the right wall. The relationship shows that the right wall should not move to the left. The inequality is a mandate for the receiver to check its advertisement. However, note that the inequality is valid only if $S_f < S_n$; we need to remember that all calculations are in modulo 2^{32}.

Window Shutdown

Shrinking the send window by moving its right wall to the left is strongly discouraged. However, there is one exception: the receiver can temporarily shut down the window by sending a rwnd of 0. This can happen if for some reason the receiver does not want to receive any data from the sender for a while.

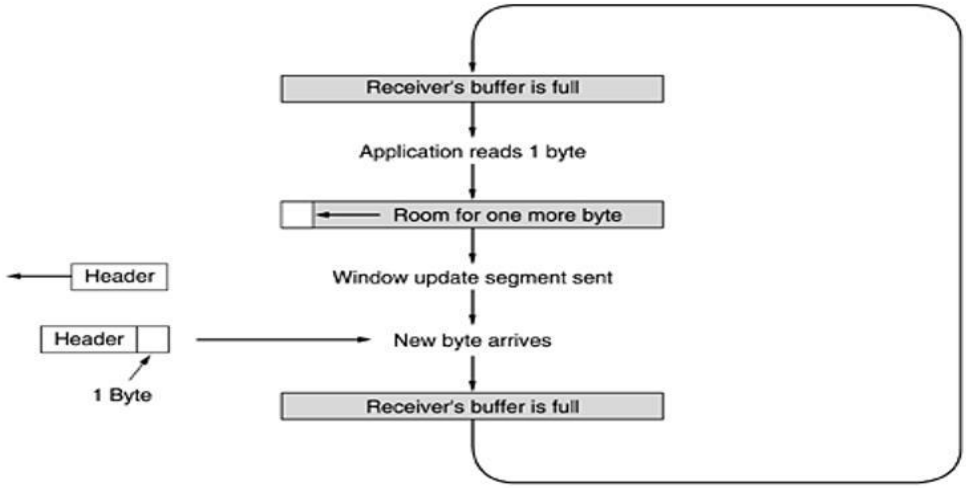
In this case, the sender does not actually shrink the size of the window, but stops sending data until a new advertisement has arrived.
Even when the window is shut down by an order from the receiver, the sender can always send a segment with 1 byte of data. This is called probing and is used to prevent a deadlock.

Silly Window Syndrome
A serious problem can arise in the sliding window operation when either the sending application program creates data slowly or the receiving application program consumes data slowly, or both. Any of these situations results in the sending of data in very small segments, which reduces the efficiency of the operation. For example, if TCP sends segments containing only 1 byte of data, it means that a 41-byte datagram (20 bytes of TCP header and 20 bytes of IP header) transfers only 1 byte of user data. Here the overhead is 41/1, which indicates that we are using the capacity of the network very inefficiently. The inefficiency is even worse after accounting for the data-link layer and physical-layer overhead. This problem is called the silly window syndrome.

Syndrome Created by the Sender
The solution is to prevent the sending TCP from sending the data byte by byte. The sending TCP must be forced to wait and collect data to send in a larger block. How long should the sending TCP wait? If it waits too long, it may delay the process. If it does not wait long enough, it may end up sending small segments. Nagle found an elegant solution. Nagle's algorithm is simple:
1. The sending TCP sends the first piece of data it receives from the sending application program even if it is only 1 byte.
2. After sending the first segment, the sending TCP accumulates data in the output buffer and waits until either the receiving TCP sends an acknowledgment or until enough data have accumulated to fill a maximum-size segment. At this time, the sending TCP can send the segment.
3. Step 2 is repeated for the rest of the transmission. Segment 3 is sent immediately if an acknowledgment is received for segment 2, or if enough data have accumulated to fill a maximum-size segment.
If the application program is faster than the network, the segments are larger (maximum-size segments). If the application program is slower than the network, the segments are smaller (less than the maximum segment size).

Syndrome Created by the Receiver
The receiving TCP may create a silly window syndrome if it is serving an application program that consumes data slowly, for example, 1 byte at a time. Suppose that the sending application program creates data in blocks of 1 kilobyte, but the receiving application program consumes data 1 byte at a time. Also suppose that the input buffer of the receiving TCP is 4 kilobytes. The sender sends the first 4 kilobytes of data. The receiver stores it in its buffer. Now its buffer is full. It advertises a window size of zero, which means the sender should stop sending data. The receiving application reads the first byte of data from the input buffer of the receiving TCP. Now there is 1 byte of space in the incoming buffer. The receiving TCP announces a window size of 1 byte, which means that the sending TCP, which is eagerly waiting to send data, takes this advertisement as good news and sends a segment carrying only 1 byte of data. The procedure will continue. One byte of data is consumed and a segment carrying 1 byte of data is sent. Again we have an efficiency problem and the silly window syndrome.
Two solutions have been proposed to prevent the silly window syndrome created by an application program that consumes data more slowly than they arrive.
Clark's solution is to send an acknowledgment as soon as the data arrive, but to announce a window size of zero until either there is enough space to accommodate a segment of maximum size or until at least half of the receive buffer is empty.
The second solution is to delay sending the acknowledgment. This means that when a segment arrives, it is not acknowledged immediately. The receiver waits until there is a decent amount of space in its incoming buffer before acknowledging the arrived segments. The delayed acknowledgment prevents the sending TCP from sliding its window. After the sending TCP has sent the data in the window, it stops. This kills the syndrome.
Delayed acknowledgment also has another advantage: it reduces traffic. The receiver does not have to acknowledge each segment. However, there also is a disadvantage in that the delayed acknowledgment may result in the sender unnecessarily retransmitting the unacknowledged segments.

The protocol balances the advantages and disadvantages. It now defines that the acknowledgment should not be delayed by more than 500 ms.



Transactional TCP

If both the request and reply are small enough to fit into single packets and the operation is idempotent, UDP can simply be used, However, if these conditions are not met, using UDP is less attractive. For example, if the reply can be quite large, then the pieces must be sequenced and a mechanism must be devised to retransmit lost pieces.

T-TCP:

- The normal sequence of packets for doing an RPC over TCP is shown in Fig. 4.36(a).
- Nine packets are required in the best case.

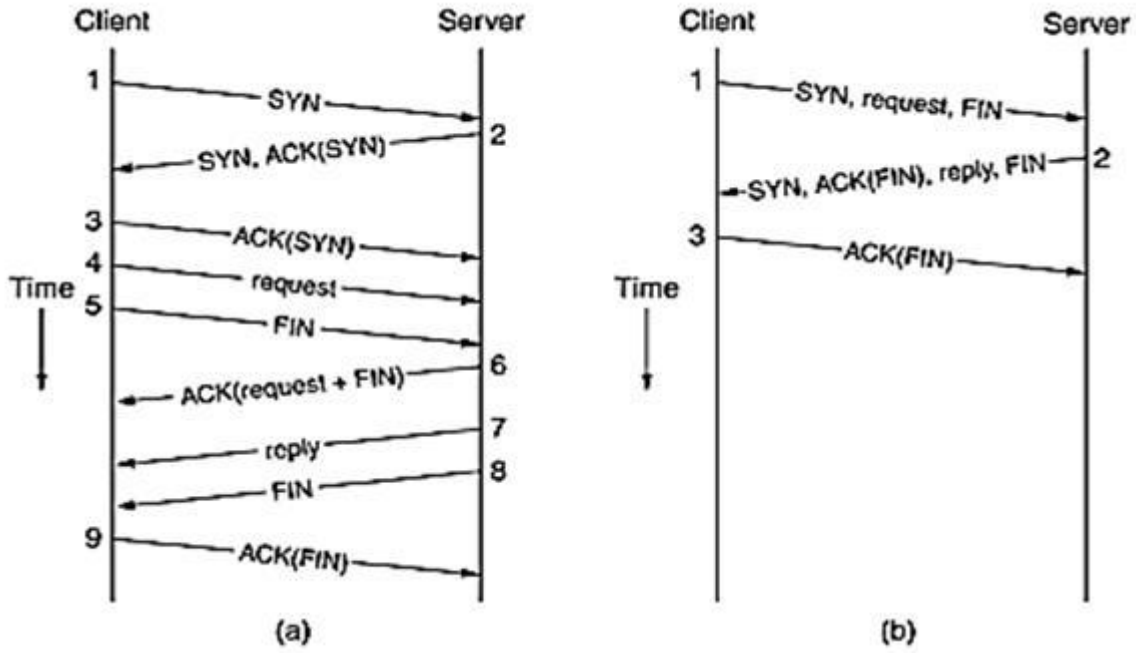


Figure 4.36 (a) RPC using normal TCP. (b) RPC using T/TCP

The nine packets are as follows:

1. The client sends a SYN packet to establish a connection.
2. The server sends an ACK packet to acknowledge the SYN packet.
3. The client completes the three-way handshake.
4. The client sends the actual request.
5. The client sends a FIN packet to indicate that it is done sending.
6. The server acknowledges the request and the FIN.
7. The server sends the reply back to the client.
8. The server sends a FIN packet to indicate that it is also done.
9. The client acknowledges the server's FIN.

The T/TCP protocol is illustrated in Fig. **4.36**(b).

- The client's first packet contains the SYN bit, the request itself, and the FIN. In effect it says: I want to establish a connection, here is the data, and I am done.
- When the server gets the request, it looks up or computes the reply, and chooses how to respond. If the reply fits in one packet, it gives the reply of Fig. 4.36(b), which says: I acknowledge your FIN, here is the answer, and I am done.
- The client then acknowledges the server's FIN and the protocol terminates in three messages.
- However, if the result is larger than 1 packet, the server also has the option of not turning on the FIN bit, in which case it can send multiple packets before closing its direction.
- It is probably worth mentioning that T/TCP is not the only proposed improvement to TCP. Another proposal is SCTP (Stream Control Transmission Protocol).
- Its features include message boundary preservation, multiple delivery modes (e.g., unordered delivery), multihoming (backup destinations), and selective acknowledgements.

Performance Problems in Computer Networks

1. 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.
2. 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.
3. Overloads can also be synchronously triggered. For example, if a TPDU contains a bad parameter (e.g., the port for which it is destined), in many cases the receiver will thoughtfully send back an error notification. Now consider what could happen if a bad TPDU is broadcast to 10,000 machines: each one might send back an error message. The resulting broadcast storm could cripple the network.
4. Poor performance can occur due to lack of system tuning. For example, if a machine has plenty of CPU power and memory but not enough of the memory has been allocated for buffer space, overruns will occur and TPDUs will be lost. Similarly, if the scheduling algorithm does not give a high enough priority to processing incoming TPDUs, some of them may be lost.
5. Another tuning issue is setting timeouts correctly. When a TPDU is sent, a timer is typically set to guard against loss of the TPDU. If the timeout is set too short, unnecessary retransmissions will occur, clogging the wires. If the timeout is set too long, unnecessary delays will occur after a TPDU is lost.
6. Other tunable parameters include how long to wait for data on which to piggyback before sending a separate acknowledgement, and how many retransmissions before giving up.
7. A useful quantity to keep in mind when analyzing network performance is the bandwidthdelay 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). 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.
8. 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

Network performance explains the level of satisfaction that network users derive from using their network. It entails the quality of service a network user receives from the web.

Metrics

1. Bandwidth

This is one of the network performance metrics that is measured in Hertz or bit per second. **Bandwidth** explains the amount of data transmitted through a network in a given amount of time. *Hertz* categorizes

signals transmitted in analog form, while *bit per second* is used to classify signals transmitted in digital format.

2. Throughput

Throughput refers to the particular amount of deliverable or service produced at the production end and successfully delivered at the receiving end. *In-network* relates to the amount of data transmitted successfully through a network bandwidth.

Throughput checks the exact number of messages delivered through a network. We can say that a throughput exists within a network bandwidth, so throughput *cannot exceed* a network bandwidth but *can be less* than network bandwidth.

3. Latency

Latency in-network is said to be the lag time for message delivery in a network. It explains how long it takes for a message that leaves the sender end to arrive at the receiver end along with a network. If it takes longer for a message to be delivered along with a network, we can say a network has *more latency*, the lesser time it takes we can say the network has *less latency*.

4. Jitter

A jitter is a form of latency. It is experienced when a delay variation is caused by the delay of specific data packets sent through a network.

For instance, when we send audio data, video data, and image data through a network, the video and audio data may encounter latency along their transmission path. In this case, the receiving end would experience jitter because the video data is time-sensitive and needs no latency or delay for the successful transmission of data. This is the same with the audio data. But, when downloading a file, jitter is hardly a concern, as the data here is not time-sensitive, and the receiving end can receive the data whenever its download is complete.

5. Packet loss

Packet loss is used to categorize the amount of data packets sent from the sender end that did not arrive at the receiver end. This metric can be measured by identifying the number of data packets that left the sender end and calculating the amount of data that arrived at the receiver end. Now, we can know the number of data packets lost along the transmission process.

6. Bandwidth-delay product

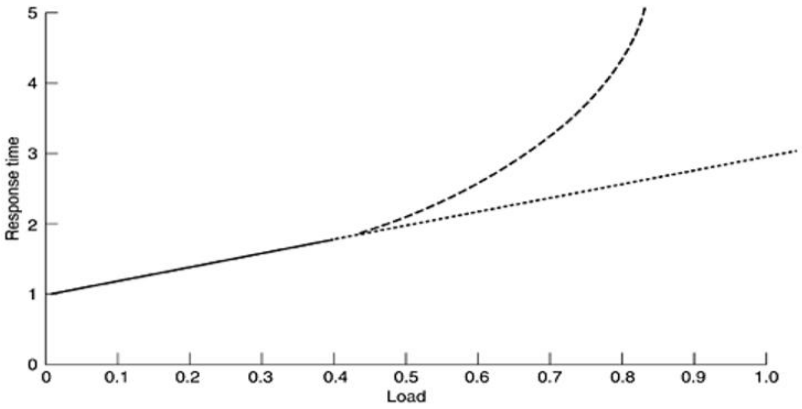
The bandwidth-delay product is a performance metric that helps to capture lost data packets during transmission. It is a performance metric that ascertains buffering needs of a network system. This would help in capturing the lost data packets during the transmission of data along a network path.

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.

1. Make Sure That the Sample Size Is Large Enough - Do not measure the time to send one TPDU, but repeat the measurement, say, one million times and take the average. Having a large sample will reduce the uncertainty in the measured mean and standard deviation. This uncertainty can be computed using standard statistical formulas.
2. Make Sure That the Samples Are Representative
3. Be Careful When Using a Coarse-Grained Clock - To measure the time to send a TPDU, for example, the system clock (say, in milliseconds) should be read out when the transport layer code is entered and again when it is exited.
4. Be Sure That Nothing Unexpected Is Going On during Your Tests
5. Caching Can Wreak Havoc with Measurements - The obvious way to measure file transfer times or buffering is to open a large file, read the whole thing, close it, and see how long it takes. Repeat with different files of different sizes. Get a good average.
6. Understand What You Are Measuring - When you measure the time to read a remote file, your measurements depend on the network, the operating systems on both the client and server, the particular hardware interface boards used, their drivers, and other factors. If the measurements are done carefully, you will ultimately discover the file transfer time for the configuration you are using. If your goal is to tune this particular configuration, these measurements are fine.
7. Be Careful about Extrapolating the Results - Suppose that you make measurements of something with simulated network loads running from 0 (idle) to 0.4 (40 percent of capacity), as shown by the data points and solid line through them in Fig. 6-42. It may be tempting to extrapolate linearly, as shown

by the dotted line. However, many queueing results involve a factor of 1/(1 - ρ), where ρ is the load, so the true values may look more like the dashed line, which rises much faster than linearly.

Figure 6-42. Response as a function of load.



Unit V: The Application Layer: Introduction, Client-Server Programming, WWW and HTTP, FTP, e-mail, TELNET, Secure Shell, Domain Name System, SNMP.

The whole Internet, hardware and software, was designed and developed to provide services at the application layer. The fifth layer of the TCP/IP protocol suite is where these services are provided for Internet users. The other four layers are there to make these services possible.

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.

- Application Layer provides a facility by which users can forward several emails and it also provides a storage facility.
- This layer allows users to access, retrieve and manage files in a remote computer.
- It allows users to log on as a remote host.
- This layer provides access to global information about various services.
- This layer provides services which include: e-mail, transferring files, distributing results to the user, directory services, network resources and so on.
- It provides protocols that allow software to send and receive information and present meaningful data to users.
- It handles issues such as network transparency, resource allocation and so on.
- This layer serves as a window for users and application processes to access network services.
- Application Layer is basically not a function, but it performs application layer functions.
- The application layer is actually an abstraction layer that specifies the shared protocols and interface methods used by hosts in a communication network.
- Application Layer helps us to identify communication partners, and synchronizing communication.
- This layer allows users to interact with other software applications.
- In this layer, data is in visual form, which makes users truly understand data rather than remembering or visualize the data in the binary format (0's or 1's).
- This application layer basically interacts with Operating System (OS) and thus further preserves the data in a suitable manner.
- This layer also receives and preserves data from it's previous layer, which is Presentation Layer in OSI Model and Transport Layer in TCP/IP Protocol Suite (which carries in itself the syntax and semantics of the information transmitted).
- The protocols which are used in this application layer depend upon what information users wish to send or receive.
- This application layer, in general, performs host initialization followed by remote login to hosts.

Standard and Nonstandard Protocols

To provide smooth operation of the Internet, the protocols used in the first four layers of the TCP/IP suite need to be standardized and documented. They normally become part of the package that is included in operating systems such as Windows or UNIX. To be flexible, however, 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. The study of these protocols enables a network manager to easily solve the problems that may occur when using these protocols. The deep understanding of how these protocols work will also give us some ideas about how to create new nonstandard protocols.

Nonstandard Application-Layer Protocols

A programmer can create a nonstandard application-layer program if she 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, which use the available services provided by the transport-layer protocols.

CLIENT-SERVER PROGRAMMING

In a client-server paradigm, communication at the application layer is between two running application programs called processes: a client and a server.

A client is a running program that initializes the communication by sending a request; 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. This definition of a server implies that a server must be running when a request from a client arrives, but the client needs to be run only when it is needed. This means that if we have two computers connected to each other somewhere, we can run a client process on one of them and the server on the other. However, we need to be careful that the server program is started before we start running the client program. In other words, the lifetime of a server is infinite: it should be started and run forever, waiting for the clients. The lifetime of a client is finite: it normally sends a finite number of requests to the corresponding server, receives the responses, and stops.

Application Programming Interface

How can a client process communicate with a server process?

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).

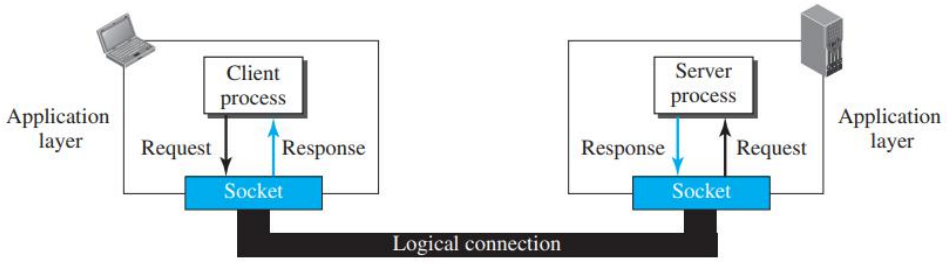
An interface in programming is a set of instructions between two entities.

In this case, 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. In other words, 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 socket interface is a set of instructions that provide communication between the application layer and the operating system.

Figure 25.6 *Use of sockets in process-to-process communication*



Communication between a client process and a server process is communication between two sockets, created at two ends.

The client thinks that the socket is the entity that receives the request and gives the response; the server thinks that the socket is the one that has a request and needs the response. If we create two sockets, one

at each end, and define the source and destination addresses correctly, we can use the available instructions to send and receive data. The rest is the responsibility of the operating system and the embedded TCP/IP protocol.

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 is combination of an IP address and a port number.

Figure 25.7 *A socket address*

How can a client or a server find a pair of socket addresses for communication?
Server Site
The server needs a local (server) and a remote (client) socket address for communication.

a. 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.

b. 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.
In other words, 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.
a. 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.
b. 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.

- Sometimes, 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. For example, at the end of this

chapter we write some simple client and server programs and we test them using this approach. In this situation, the programmer can provide these two pieces of information when he runs the client program.

- 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).

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 popularity and growth of the Web can be related to two terms in the above statement: distributed and linked.

i. 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.

ii. 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. Today, 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.

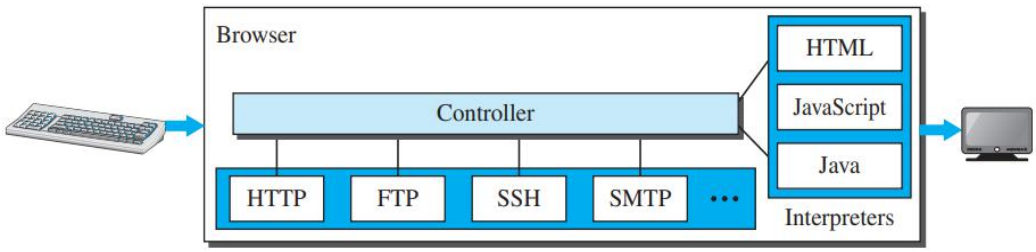
Architecture

The WWW today is a distributed client-server service, in which a client using a browser can access a service using a server. However, the service provided is distributed over many locations called sites. Each site holds one or more web pages. Each web page, however, can contain some links to other web pages in the same or other sites. In other words, a web page can be simple or composite. A simple web page has no links to other web pages; a 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.

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. In this case, 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 clientserver 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 (HyperText 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, which we discuss in Domain Name System (DNS) later in this chapter.
- 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, and active.

i. 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. In other words, the contents of the file are determined when the file is created, not when it is used. Of course, 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).

ii. 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. Although 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.

iii. 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 bytecode (binary) format. Another way is to use JavaScripts but download and run the script at the client site.

HyperText Transfer Protocol (HTTP)

The HyperText 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, however, need not worry about errors in messages exchanged or loss of any message, because the TCP is reliable and will take care of this matter.

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. However, if some of the objects are located on the same 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 nonpersistent connection, the second as a persistent connection. HTTP, prior to version 1.1, specified nonpersistent connections, while persistent connections are the default in version 1.1, but it can be changed by the user.

Nonpersistent Connections

In a nonpersistent 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.

In this strategy, if a file contains links to N different pictures in different files (all located on the same server), the connection must be opened and closed N + 1 times. The nonpersistent strategy imposes high overhead on the server because the server needs N + 1 different buffers each time a connection is opened.

Persistent Connections

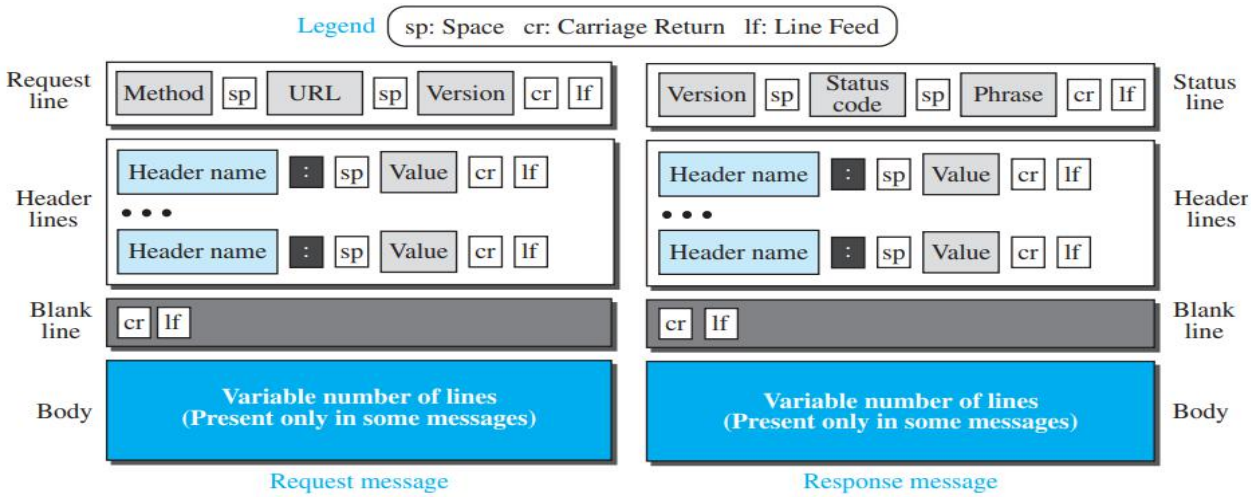
HTTP version 1.1 specifies a persistent connection by default. In a persistent connection, the server leaves the connection open for more requests after sending a response. The server can close the connection at the request of a client or if a time-out has been reached. The sender usually sends the length of the data with each response. However, 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.

Message Formats

The HTTP protocol defines the format of the request and response messages, as shown in Figure 26.5.

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.

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).

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. In this case, 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, 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.

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

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.

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

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.

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.

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.

Cookies

The World Wide Web was originally designed as a stateless entity.
A client sends a request; a server responds. Their relationship is over.
The original purpose of the Web, retrieving publicly available documents, exactly fits this design. Today the Web has other functions that need to remember some information about the clients; some are listed below:

- Websites are being used as electronic stores that allow users to browse through the store, select wanted items, put them in an electronic cart, and pay at the end with a credit card.
- Some websites need to allow access to registered clients only.
- Some websites are used as portals: the user selects the web pages he wants to see.
- Some websites are just advertising agencies. For these purposes, the cookie mechanism was devised.

Creating and Storing Cookies

The creation and storing of cookies depend on the implementation; however, the principle is the same.
1. When a server receives a request from a client, it stores information about the client in a file or a string. The information may include the domain name of the client, the contents of the cookie (information the server has gathered about the client such as name, registration number, and so on), a timestamp, and other information depending on the implementation.
2. The server includes the cookie in the response that it sends to the client.
3. When the client receives the response, the browser stores the cookie in the cookie directory, which is sorted by the server domain name.

Using Cookies

When a client sends a request to a server, the browser looks in the cookie directory to see if it can find a cookie sent by that server. If found then the cookie is included in the request.
When the server receives the request, it knows that this is an old client, not a new one.
Note that the contents of the cookie are never read by the browser or disclosed to the user.
It is a cookie made by the server and eaten by the server.
Example:
An electronic store (e-commerce) can use a cookie for its client shoppers. When a client selects an item and inserts it in a cart, a cookie that contains information about the item, such as its number and unit price, is sent to the browser. If the client selects a second item, the cookie is updated with the new selection information, and so on. When the client finishes shopping and wants to check out, the last cookie is retrieved and the total charge is calculated.
The site that restricts access to registered clients only sends a cookie to the client when the client registers for the first time. For any repeated access, only those clients that send the appropriate cookie are allowed.

Web Caching: Proxy Servers

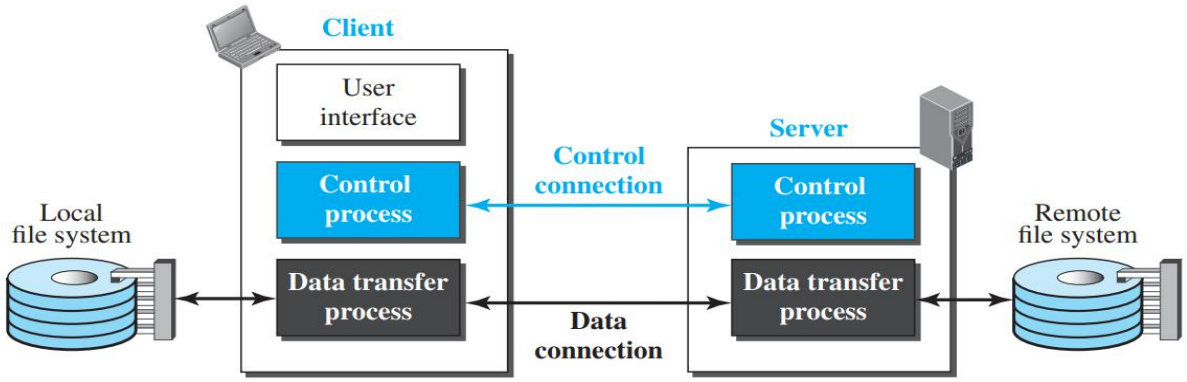
HTTP supports proxy servers. A proxy server is a computer that keeps copies of responses to recent requests. The HTTP client sends a request to the proxy server. The proxy server checks its cache. If the response is not stored in the cache, the proxy server sends the request to the corresponding server. Incoming responses are sent to the proxy server and stored for future requests from other clients.
The proxy server reduces the load on the original server, decreases traffic, and improves latency. However, to use the proxy server, the client must be configured to access the proxy instead of the target server.
Note that the proxy server acts as both server and client. When it receives a request from a client for which it has a response, it acts as a server and sends the response to the client. When it receives a request from a client for which it does not have a response, it first acts as a client and sends a request to the target server. When the response has been received, it acts again as a server and sends the response to the client.

Proxy Server Location
The proxy servers are normally located at the client site. This means that we can have a hierarchy of proxy servers, as shown below:
1. A client computer can also be used as a proxy server, in a small capacity, which stores responses to requests often invoked by the client.
2. In a company, a proxy server may be installed on the computer LAN to reduce the load going out of and coming into the LAN.
3. An ISP with many customers can install a proxy server to reduce the load going out of and coming into the ISP network.

FTP

File Transfer Protocol (FTP) is the standard protocol provided by TCP/IP for copying a file from one host to another. Although transferring files from one system to another seems simple and straightforward, some problems must be dealt with first. 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.

Figure 26.10 *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.

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.

In other words, 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.

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.

Table 26.4 *Some FTP commands*

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

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 *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

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, block mode, or compressed mode. The stream mode is the default mode; data are delivered from FTP to TCP as a continuous stream of bytes. 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).

ELECTRONIC MAIL

Electronic mail (or e-mail) allows users to exchange messages.

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.

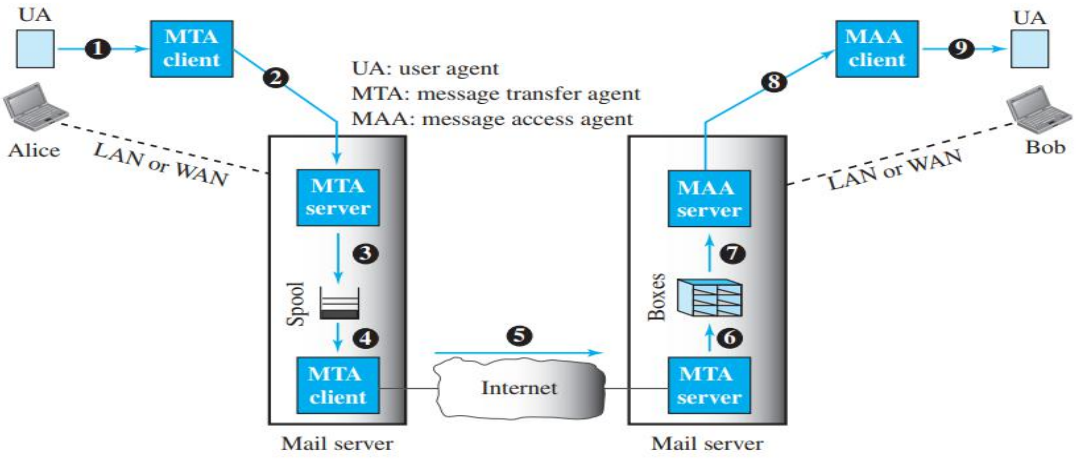
This means that the idea of client/server programming should be implemented in another way: using some intermediate computers (servers). The users run only client programs when they want and the intermediate servers apply the client/server paradigm.

Architecture

To explain the architecture of e-mail, we give a common scenario.

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.

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.

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: command-driven and GUI-based.

Commanddriven 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 onecharacter 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 commanddriven user agents are mail, pine, and elm.

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. Some 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. 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.

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 and a domain name, separated by @ sign

Figure 26.14 *E-mail address*

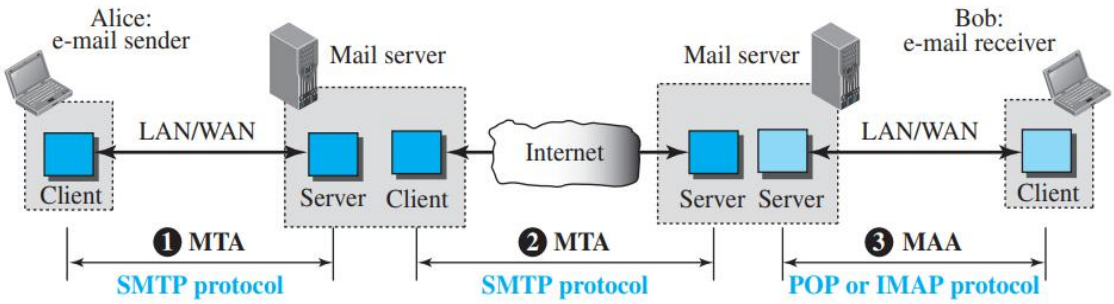
The local part defines the name of a special file, called the user mailbox, where all the mail received for a user is stored for retrieval by the message access agent. The second part of the address is the domain name. An organization usually selects one or more hosts to receive and send e-mail; they 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).

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.

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.

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 twocharacter (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.

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 threedigit 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.

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.

Message Transfer

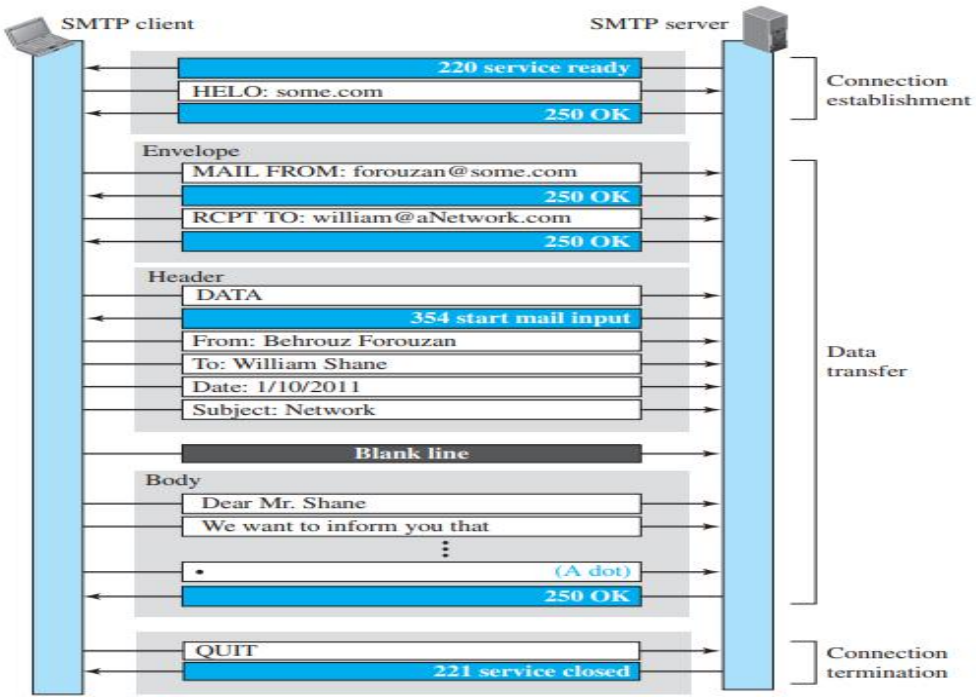
After connection has been established between the SMTP client and server, a single message between a sender and one or more recipients can be exchanged. This phase involves eight steps. Steps 3 and 4 are repeated if there is more than one recipient.

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.

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 is a push protocol; it pushes the message from the client to the server. In other words, the direction of the bulk data (messages) is from the client to the server. On the other hand, the third stage needs a 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

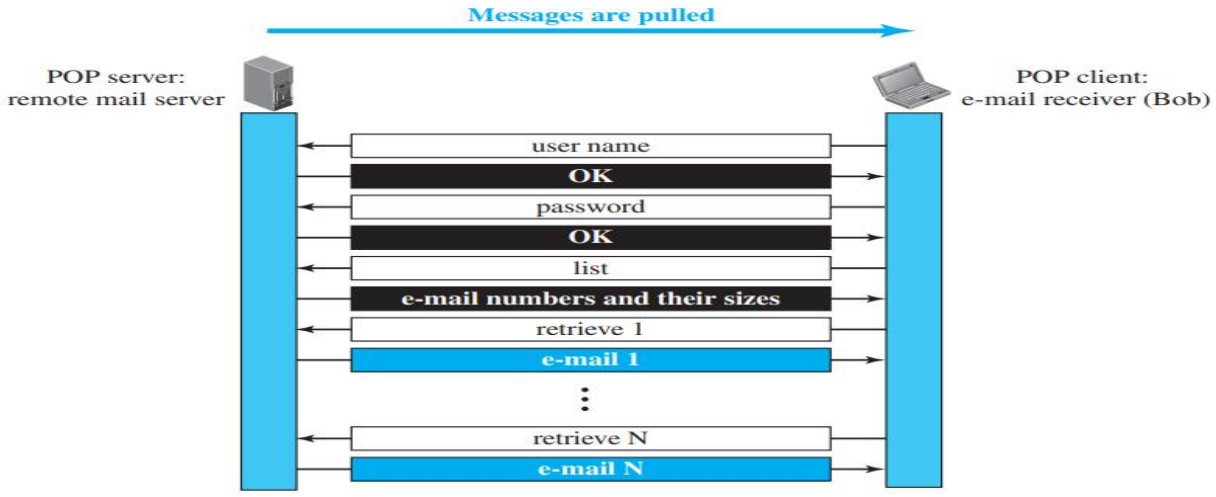
Post Office Protocol, version 3 (POP3) is simple but limited in functionality.

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 shows an example of downloading using POP3. Unlike other figures in this chapter, we have put the client on the right hand side because the e-mail receiver (Bob) is running the client process to pull messages from the remote mail server.

Figure 26.17 *POP3*



POP3 has two modes: the delete mode and the keep 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 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

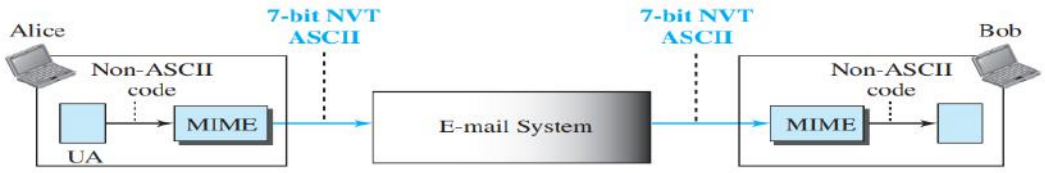
Another mail access protocol is Internet Mail Access Protocol, version 4 (IMAP4). IMAP4 is similar to POP3, but it has more features; IMAP4 is more powerful and more complex.
POP3 is deficient in several ways. It does not allow the user to organize her mail on the server; the user cannot have different 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

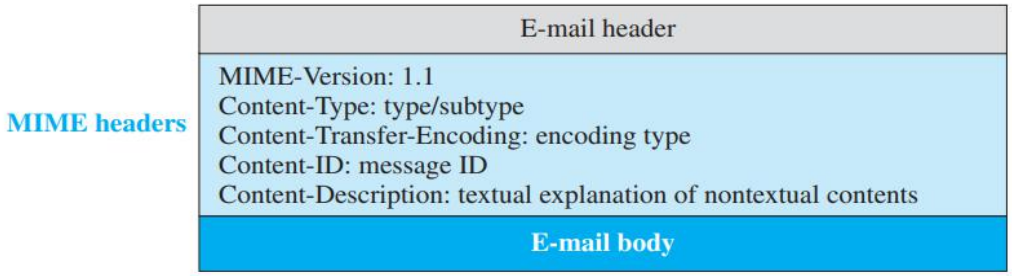
Electronic mail has a simple structure. Its simplicity, however, comes with a price. It can send messages only in NVT 7-bit ASCII format. In other words, it has some limitations. It cannot be used for languages other than English (such as French, German, Hebrew, Russian, Chinese, and Japanese). Also, 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.

Figure 26.18 *MIME*



MIME Headers
MIME defines five headers, 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,

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

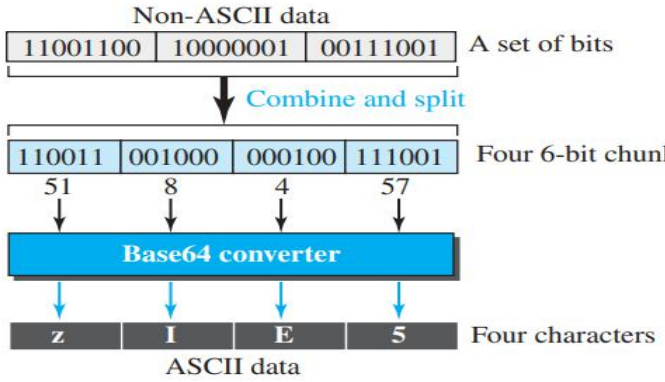
This header defines the method used to encode the messages into 0s and 1s for transport. The five types of encoding methods are listed.

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.

Figure 26.20 *Base64 conversion*



Each 6-bit section is then converted into an ASCII character.

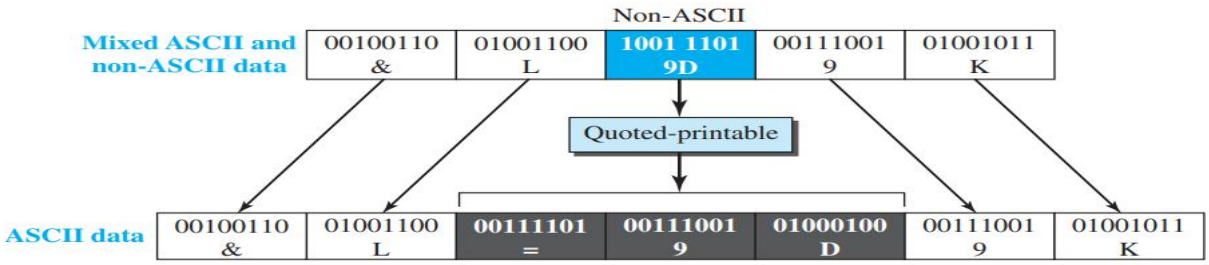
Table 26.10 *Base64 converting table*

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 (9D16), which is replaced by three ASCII characters (=, 9, and D).

Figure 26.21 *Quoted-printable*



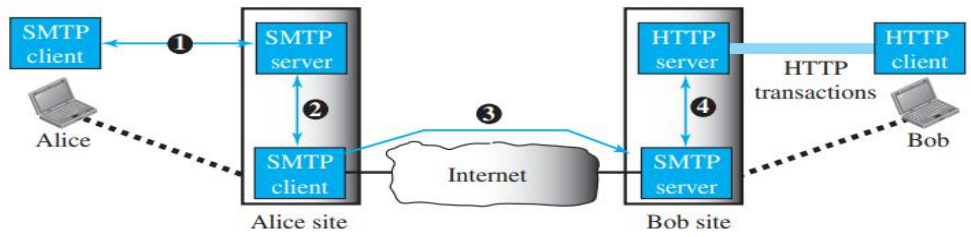
Content-ID
This header uniquely identifies the whole message in a multiple message environment.

Content-Description
This header defines whether the body is image, audio, or video.

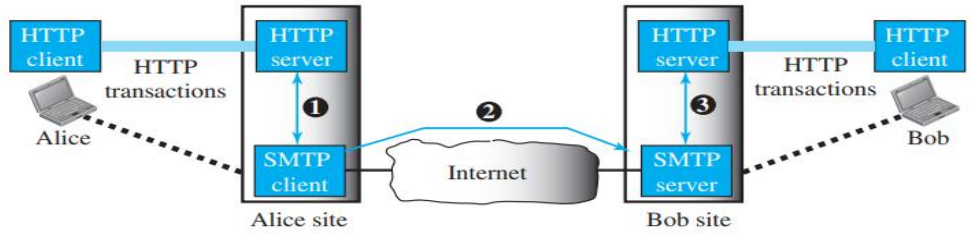
Web-Based Mail

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:

Figure 26.22 *Web-based e-mail, cases I and II*



Case 1: Only receiver uses HTTP



Case 2: Both sender and receiver use HTTP

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.

TELNET

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.

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 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 pseudoterminal 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)

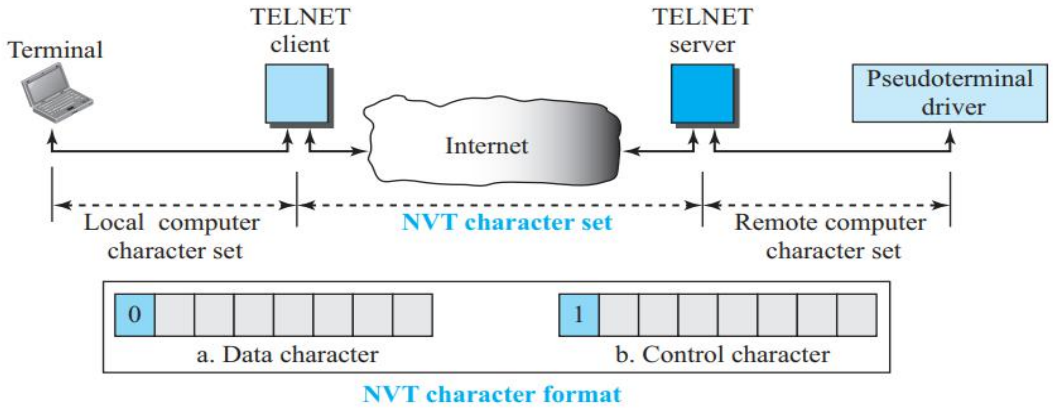
The mechanism to access a remote computer is complex. This is because every computer and its operating system accepts a special combination of characters as tokens.

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 *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.

Table 26.11 *Examples of interface commands*

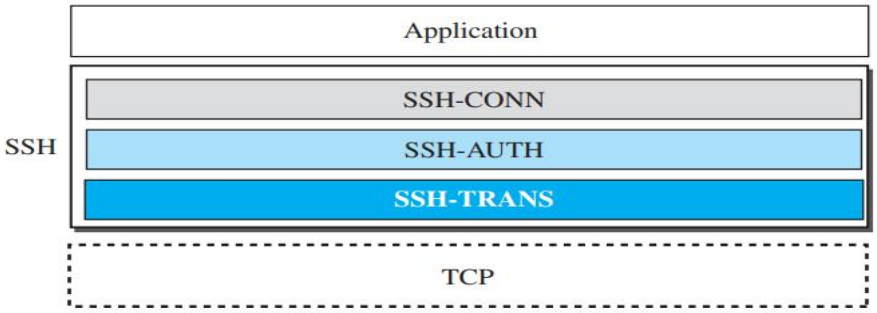
Command	*Meaning*	*Command*	*Meaning*
open	Connect to a remote computer	**set**	Set the operating parameters
close	Close the connection	**status**	Display the status information
display	Show the operating parameters	**send**	Send special characters
mode	Change to line or character mode	**quit**	Exit TELNET

SECURE SHELL (SSH)

Although 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, which are totally incompatible. The first version, SSH-1, is now deprecated because of security flaws in it. So, we discuss only SSH-2.

Components
SSH is an application-layer protocol with three components.

Figure 26.25 *Components of SSH*



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.
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. This layer defines a number of authentication tools.
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.

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.

1. SSH for Remote Logging

Several applications (free and commercial) 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.

2. SSH for File Transfer

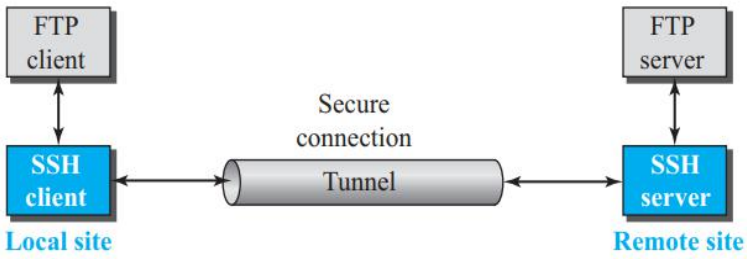
One of the application programs that is built on top of SSH for file transfer is the Secure File Transfer Program (sftp). The sftp application program uses one of the channels provided by the SSH to transfer files. Another common application is called Secure Copy (scp). This application uses the same format as the UNIX copy command, cp, to copy files.

3. 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*

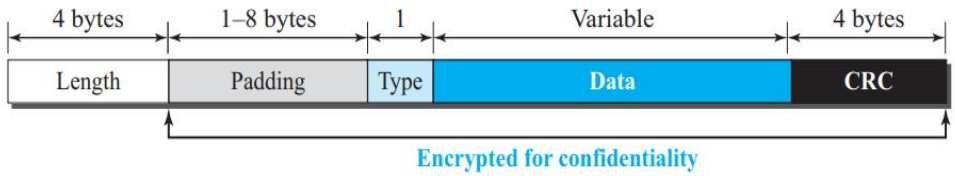


The FTP client can use the SSH client on the local site to make a secure connection with the SSH server on the remote site. Any request from the FTP client to the FTP server is carried through the tunnel provided by the SSH client and server. Any response from the FTP server to the FTP client is also carried through the tunnel provided by the SSH client and server.

4. Format of the SSH Packets

Figure 26.27 shows the format of packets used by the SSH protocols.

Figure 26.27 *SSH packet format*



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.

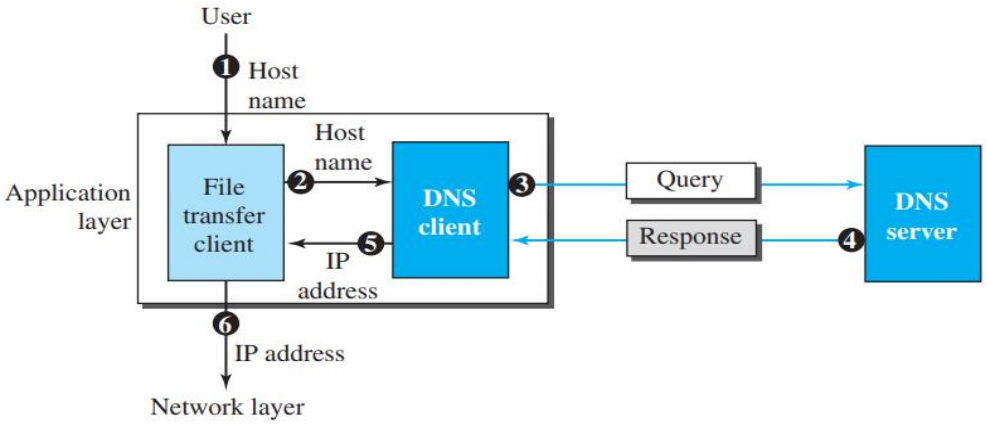
DOMAIN NAME SYSTEM (DNS)

To identify an entity, TCP/IP protocols use the IP address, which uniquely identifies the connection of a host to the Internet. However, people prefer to use names instead of numeric addresses.

The Domain Name System (DNS) turns domain names into IP addresses, which browsers use to load internet pages. Every device connected to the internet has its own IP address, which is used by other devices to locate the device. DNS servers make it possible for people to input normal words into their browsers, such as google.com, without having to keep track of the IP address for every website.

Figure 26.28 shows how TCP/IP uses a DNS client and a DNS server to map a name to an address.

Figure 26.28 *Purpose of DNS*



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. In other words, 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.

Name Space
To be unambiguous, 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.
In other words, 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: flat or 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. The first part can define the nature of the organization, the second part can define the name of an organization, the third part can define departments in the organization, and so on. In this case, the authority to assign and control the name

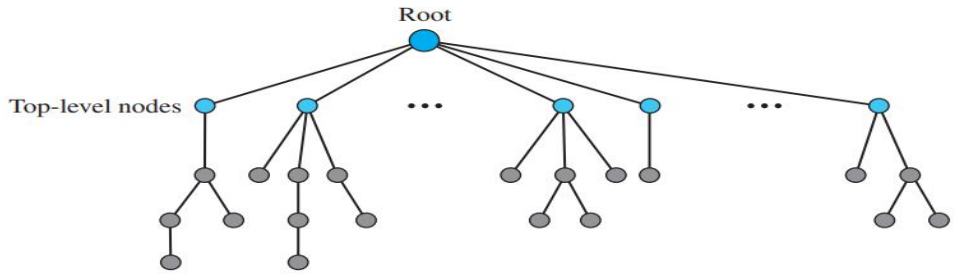
spaces can be decentralized. A central authority can assign the part of the name that defines the nature of the organization and the name of the organization. The responsibility for the rest of the name can be given to the organization itself. The organization can add suffixes (or prefixes) to the name 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.

Figure 26.29 *Domain name space*



Label

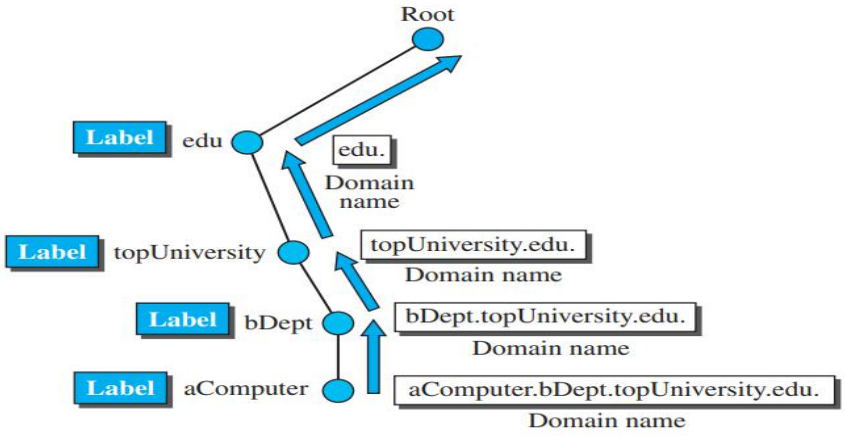
Each node in the tree has a label, which is a string with a maximum of 63 characters. The root label is a null string (empty string). DNS requires that children of a node (nodes that branch from the same node) have different labels, which guarantees the uniqueness of the domain names.

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.

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.

Figure 26.30 *Domain names and labels*

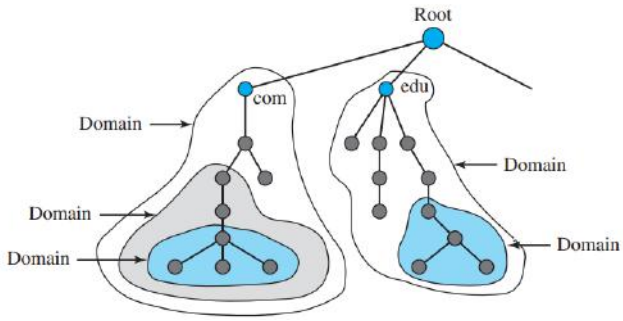


Domain

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. However, 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.

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: primary and secondary.

i. 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.

ii. 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.

Note also that a server can be a primary server for a specific zone and a secondary server for another zone. Therefore, when we refer to a server as a primary or secondary server, we should be careful about which zone we refer to.

A primary server loads all information from the disk file; the secondary server loads all information from the primary server.

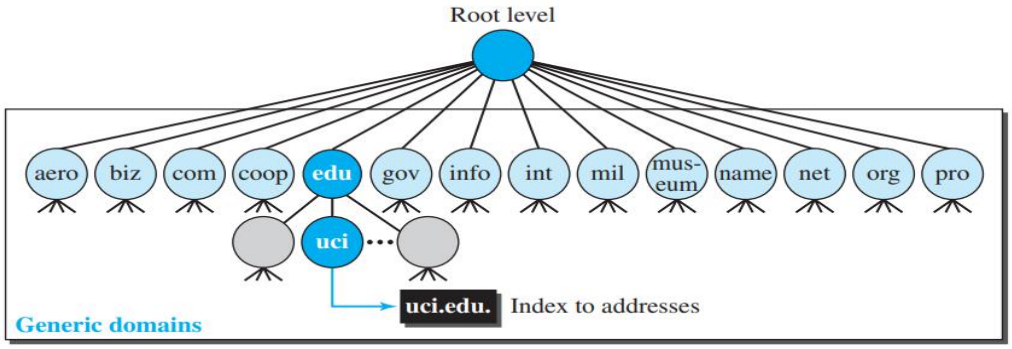
DNS in the Internet

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 the inverse domains. However, 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).

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.

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.

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.).

Resolution

Mapping a name to an address is called name-address resolution.

DNS is designed as a client-server application.

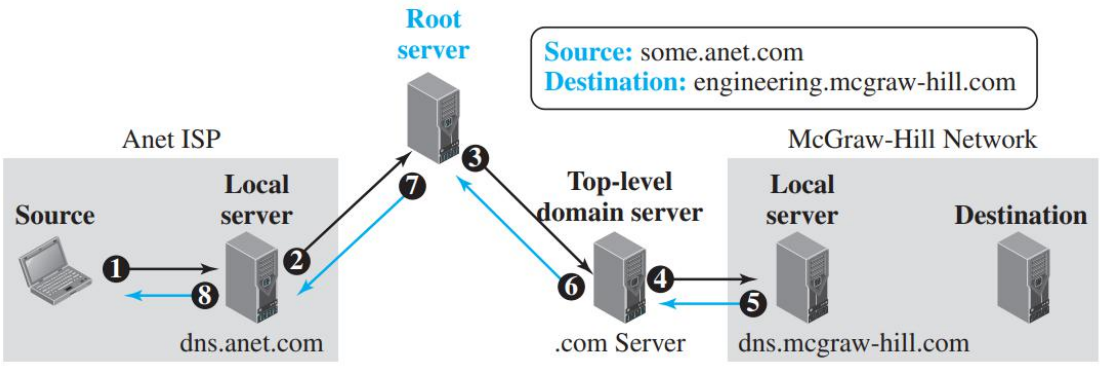
A host that needs to map an address to a name or a name to an address calls a DNS client called a resolver. The resolver accesses the closest DNS server with a mapping request. If the server has the information, it satisfies the resolver; otherwise, it either refers the resolver to other servers or asks 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.

Figure 26.36 *Recursive resolution*



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).

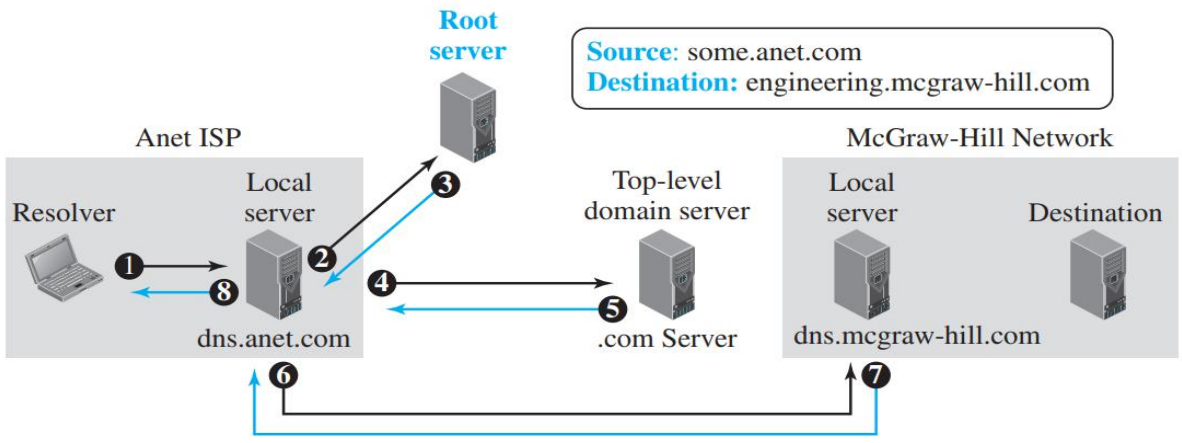
Iterative Resolution

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*



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.

i. 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.

ii. 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.

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.

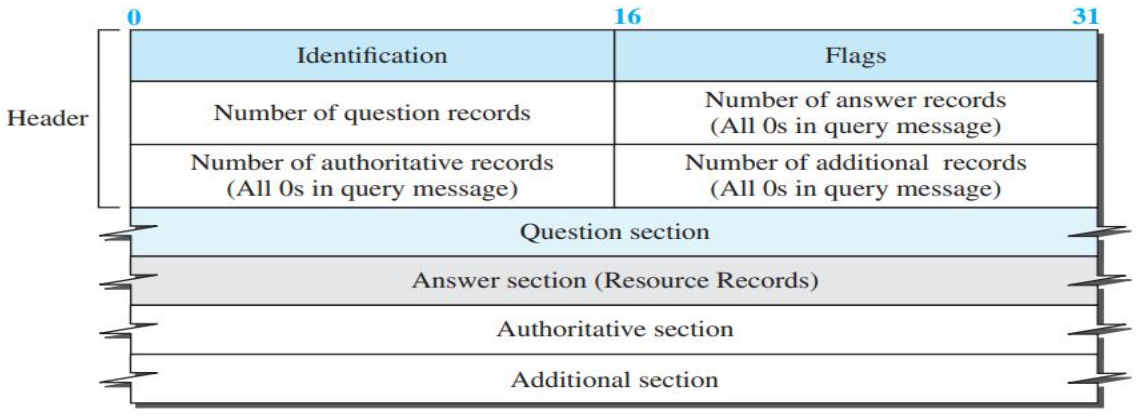
Table 26.13 *Types*

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)

DNS Messages

To retrieve information about hosts, DNS uses two types of messages: query and response.

Figure 26.38 *DNS message*



Note:
The query message contains only the question section.
The response message includes the question section,
the answer section, and possibly two other sections.

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

Registrars
How are new domains added to DNS?
This is done through a registrar, a commercial entity accredited by ICANN. A registrar first verifies that the requested domain name is unique and then enters it into the DNS database. A fee is charged. Today, there are many registrars; their names and addresses can be found at http://www.intenic.net

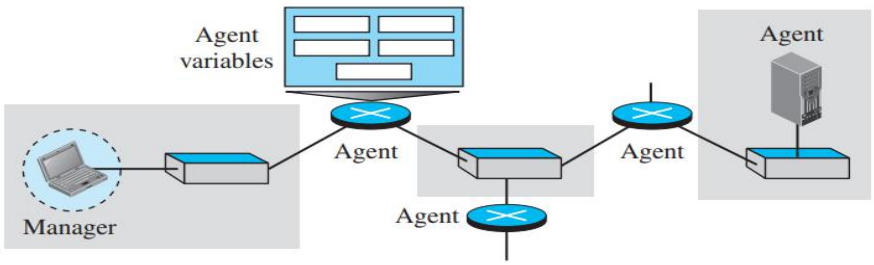
To register, the organization needs to give the name of its server and the IP address of the server. For example, a new commercial organization named wonderful with a server named ws and IP address 200.200.200.5 needs to give the following information to one of the registrars:
Domain name: ws.wonderful.com
IP address: 200.200.200.5

DDNS
When the DNS was designed, no one predicted that there would be so many address changes. In DNS, when there is a change, such as adding a new host, removing a host, or changing an IP address, the change must be made to the DNS master file. These types of changes involve a lot of manual updating. The size of today's Internet does not allow for this kind of manual operation.
The DNS master file must be updated dynamically. The Dynamic Domain Name System (DDNS) therefore was devised to respond to this need. In DDNS, when a binding between a name and an address is determined, the information is sent, usually by DHCP to a primary DNS server.
The primary server updates the zone. The secondary servers are notified either actively or passively. In active notification, the primary server sends a message to the secondary servers about the change in the zone, whereas in passive notification, the secondary servers periodically check for any changes. In either case, after being notified about the change, the secondary server requests information about the entire zone (called the zone transfer). To provide security and prevent unauthorized changes in the DNS records, DDNS can use an authentication mechanism.

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.

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. In other words, 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.

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. It reboots 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.

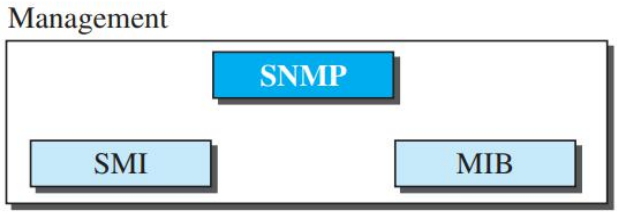
In other words, management with SNMP is based on three basic ideas:

1. A manager checks an agent by requesting information that reflects the behavior 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

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.

Figure 27.3 *Components of network management on the Internet*



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.

SNMP defines the format of packets exchanged between a manager and an agent. It reads and changes the status of objects (values of variables) in SNMP packets.

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.

SMI defines the general rules for naming objects, defining object types (including range and length), and showing how to encode objects and values.

Role of MIB

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).

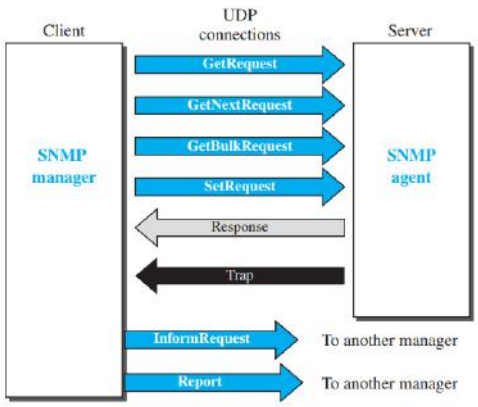
SNMP uses both SMI and MIB in Internet network management. It is an application program that allows:

- A manager to retrieve the value of an object defined in an agent.
- A manager to store a value in an object defined in an agent.
- An agent to send an alarm message about an abnormal situation to the manager.

PDUs

SNMPv3 defines eight types of protocol data units (or PDUs): GetRequest, GetNextRequest, GetBulkRequest, SetRequest, Response, Trap, InformRequest, and Report.

Figure 27.17 *SNMP PDUs*



GetRequest

The GetRequest PDU is sent from the manager (client) to the agent (server) to retrieve the value of a variable or a set of variables.

GetNextRequest

The GetNextRequest PDU is sent from the manager to the agent to retrieve the value of a variable. The retrieved value is the value of the object following the defined ObjectId in the PDU. It is mostly used to retrieve the values of the entries in a table. If the manager does not know the indexes of the entries, it cannot retrieve the values. However, it can use GetNextRequest and define the ObjectId of the table. Because the first entry has the ObjectId immediately after the ObjectId of the table, the value of the first entry is returned. The manager can use this ObjectId to get the value of the next one, and so on.

GetBulkRequest

The GetBulkRequest PDU is sent from the manager to the agent to retrieve a large amount of data. It can be used instead of multiple GetRequest and GetNextRequest PDUs.

SetRequest

The SetRequest PDU is sent from the manager to the agent to set (store) a value in a variable.

Response

The Response PDU is sent from an agent to a manager in response to GetRequest or GetNextRequest. It contains the value(s) of the variable(s) requested by the manager.

Trap

The Trap (also called SNMPv2 Trap to distinguish it from SNMPv1 Trap) PDU is sent from the agent to the manager to report an event. For example, if the agent is rebooted, it informs the manager and reports the time of rebooting.

nformRequest

The InformRequest PDU is sent from one manager to another remote manager to get the value of some variables from agents under the control of the remote manager. The remote manager responds with a Response PDU.

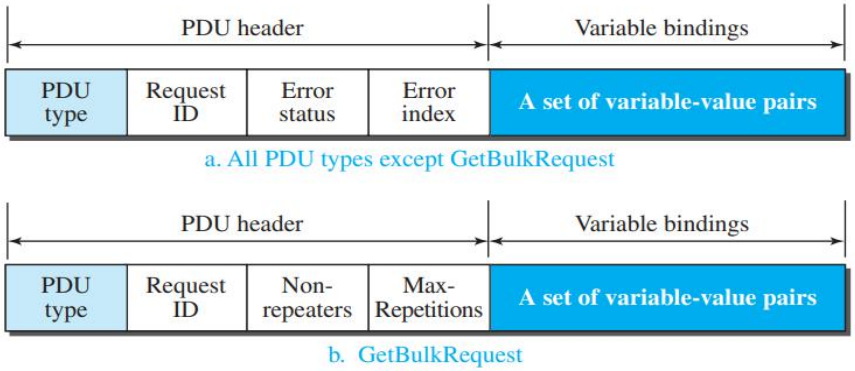
Report

The Report PDU is designed to report some types of errors between managers. It is not yet in use.

Format

The format for the eight SNMP PDUs is shown in Figure 27.18. The GetBulkRequest PDU differs from the others in two areas, as shown in the figure.

Figure 27.18 *SNMP PDU format*



Note:
The error status and error index values are set to 0 for all request messages.

The fields are listed below:
PDU type. This field defines the type of the PDU

Table 27.3 *PDU types*

Type	*Tag (Hex)*	*Type*	*Tag (Hex)*
GetRequest	**A0**	GetBulkRequest	**A5**
GetNextRequest	**A1**	InformRequest	**A6**
Response	**A2**	Trap (SNMPv2)	**A7**
SetRequest	**A3**	Report	**A8**

Request ID.
This field is a sequence number used by the manager in a request PDU and repeated by the agent in a response. It is used to match a request to a response.

Error status.
This is an integer that is used only in response PDUs to show the types of errors reported by the agent. Its value is 0 in request PDUs. Table 27.4 lists the types of errors that can occur.

Table 27.4 *Types of errors*

Status	*Name*	*Meaning*
0	noError	No error
1	tooBig	Response too big to fit in one message
2	noSuchName	Variable does not exist
3	badValue	The value to be stored is invalid
4	readOnly	The value cannot be modified
5	genErr	Other errors

Non-repeaters.
This field is used only in a GetBulkRequest PDU. The field defines the number of non-repeating (regular objects) at the start of the variable value list.

- Error index. The error index is an offset that tells the manager which variable caused the error.
- Max-repetitions. This field is also used only in a GetBulkRequest PDU. The field defines the maximum number of iterations in the table to read all repeating objects.
- Variable-value pair list. This is a set of variables with the corresponding values the manager wants to retrieve or set. The values are null in request PDUs.

SNMP uses the services of UDP on two well-known ports, 161 and 162. The well-known port 161 is used by the server (agent), and the well-known port 162 is used by the client (manager).