UNIT-1
INTRODUCTION

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***.

1. **Transit time** is the amount of time required for a message to travel from one device to another.

2. **Response time** is the elapsed time between an inquiry and a response.

The performance of a network depends on a number of factors, including the number of users, the type of transmission medium, the capabilities of the connected hardware, and the efficiency of the software.

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

Before discussing networks, we need to define some network attributes.

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 (see Figure 1.1a). 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 (see Figure 1.1b).

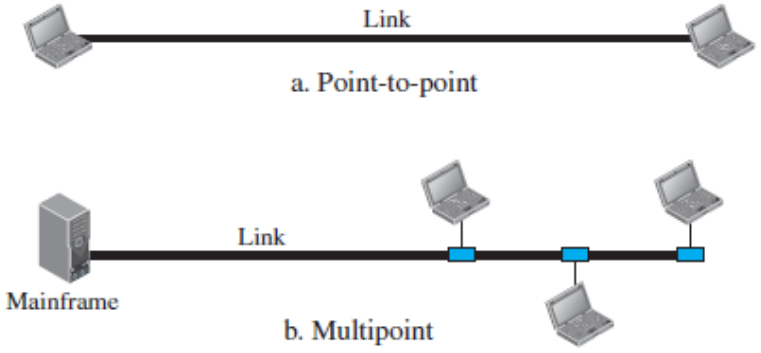


FIGURE 1.1 TYPES OF CONNECTIONS: POINT-TO-POINT AND MULTIPOINT

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 (see Figure 1.2) to be connected to the other *n* – 1 stations.

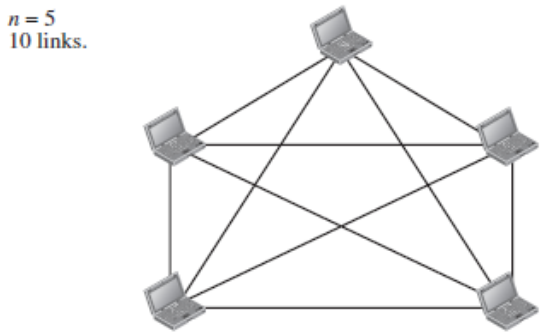


FIGURE 1.2: A FULLY CONNECTED MESH TOPOLOGY (FIVE DEVICES)

A mesh offers several *advantages* over other network topologies.

1. First, 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. Second, a mesh topology is robust. If one link becomes unusable, it does not incapacitate the entire system.

3. Third, 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. Finally, 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.

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

1. First, because every device must be connected to every other device, installation and reconnection are difficult.

2. Second, the sheer bulk of the wiring can be greater than the available space (in walls, ceilings, or floors) can accommodate.

3. Finally, 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.

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 (see Figure 1.3).

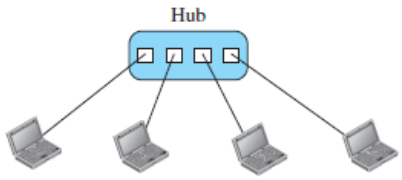


FIGURE 1.3: A STAR TOPOLOGY CONNECTING FOUR STATIONS

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:

The preceding examples all describe point-to-point connections. A **bus topology,** on the other hand, is multipoint. One long cable acts as a **backbone** to link all the devices in a network (see Figure 1.4).

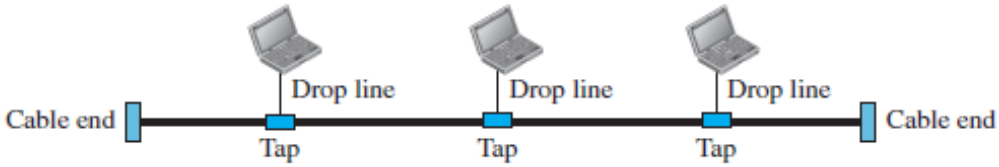


FIGURE 1.4: A BUS TOPOLOGY CONNECTING THREE STATIONS

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, and then connected to the nodes by drop lines of various lengths. In this way, a bus uses less cabling than mesh or star topologies.

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.

Bus topology was the *one of the first topologies* used in the design of early local area networks. Traditional Ethernet LANs can use a bus topology, but they are less popular.

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 (see Figure 1.5).

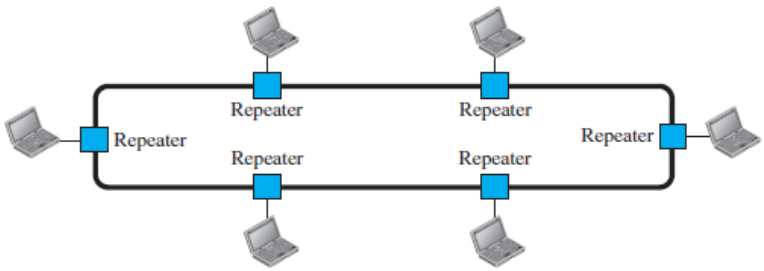


FIGURE 1.5: A RING TOPOLOGY CONNECTING SIX STATIONS

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.

Ring topology was prevalent when IBM introduced its local-area network, Token Ring. Today, the need for higher-speed LANs has made this topology less popular.

NETWORK TYPES:

The criterion of distinguishing one type of network from another is difficult and sometimes confusing.

We use a few criteria such as size, geographical coverage, and ownership to make this distinction. After discussing two types of networks, LANs and WANs, we define switching, which is used to connect networks to form an internetwork (a network of networks).

LOCAL AREA NETWORK:

A **local area network** (**LAN**) is usually privately owned and connects some hosts in a single office, building, or campus. Depending on the needs of an organization, a LAN can be as simple as two PCs and a printer in someone's home office, or it can extend throughout a company and include audio and video devices.

Each host in a LAN has an identifier, an address that uniquely defines the host in the LAN. A packet sent by a host to another host carries both the source host's and the destination host's addresses.

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 1.6 shows a LAN using either a common cable or a switch.

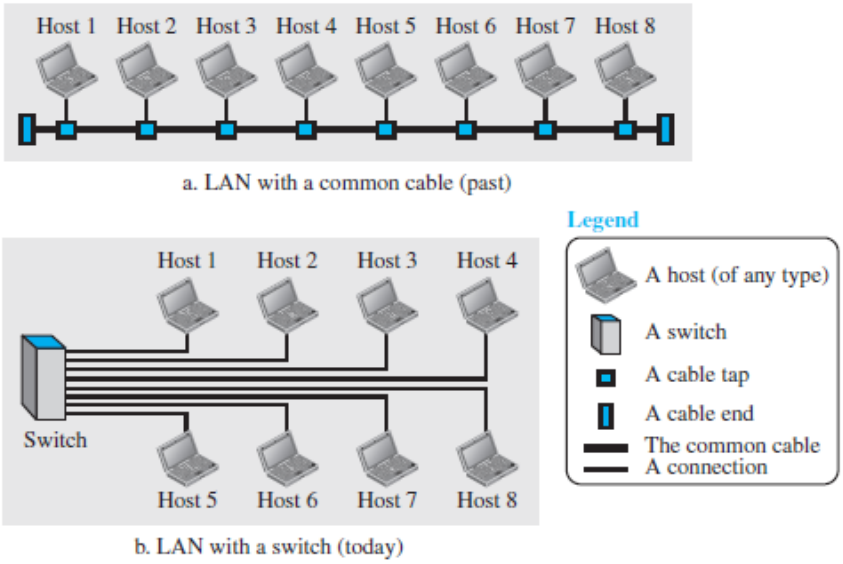


FIGURE 1.6: AN ISOLATED LAN IN THE PAST AND TODAY

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. We see two distinct examples of WANs today: point-to-point WANs and switched WANs.

Point-to-Point WAN:

A point-to-point WAN is a network that connects two communicating devices through a transmission media (cable or air). Figure 1.7 shows an example of a point-to-point WAN.

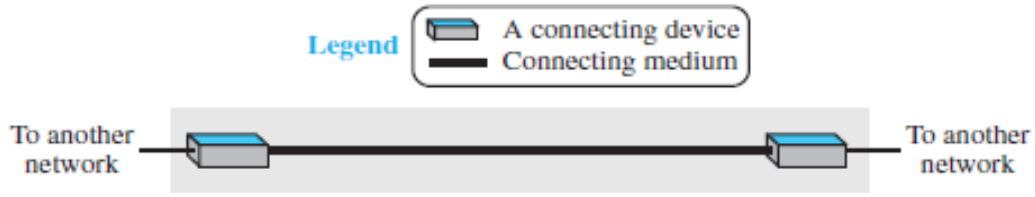


FIGURE 1.7: A POINT-TO-POINT WAN

Switched WAN:

A switched WAN is a network with more than two ends. We can say that a switched WAN is a combination of several point-to-point WANs that are connected by switches. Figure 1.8 shows an example of a switched WAN.

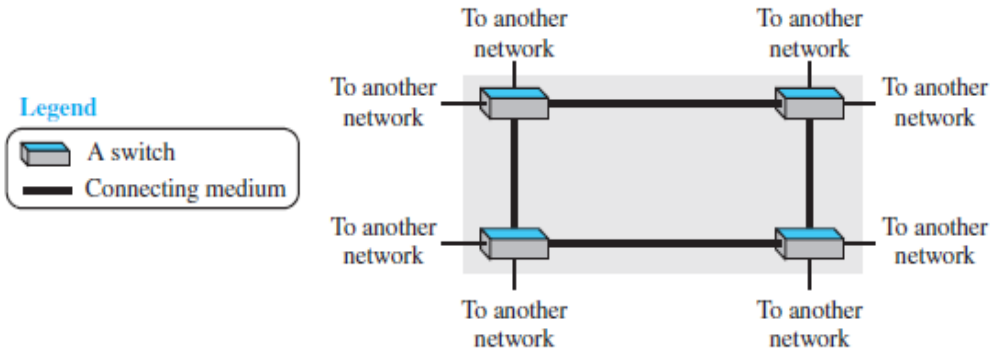


FIGURE 1.8: A SWITCHED WAN

INTERNETWORK:

Today, it is very rare to see a LAN or a WAN in isolation; they are connected to one another. When two or more networks are connected, they make an **internetwork,** or **internet.**

As an example, assume that an organization has two offices, one on the east coast and the other on the west coast. Each office has a LAN that allows all employees in the office to communicate with each other. To make the communication between employees at different offices possible, the management leases a point-to-point dedicated WAN from a service provider, such as a telephone company, and connects the two LANs.

Now the company has an internetwork, or a private internet (with lowercase *i*). Communication between offices is now possible. Figure 1.9 shows this internet.

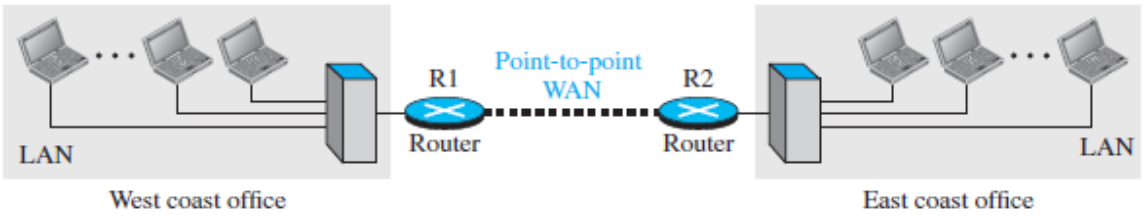


FIGURE 1.9: AN INTERNETWORK MADE OF TWO LANS AND ONE POINT-TO-POINT WAN

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 1.10 shows another internet with several LANs and WANs connected. One of the WANs is a switched WAN with four switches.

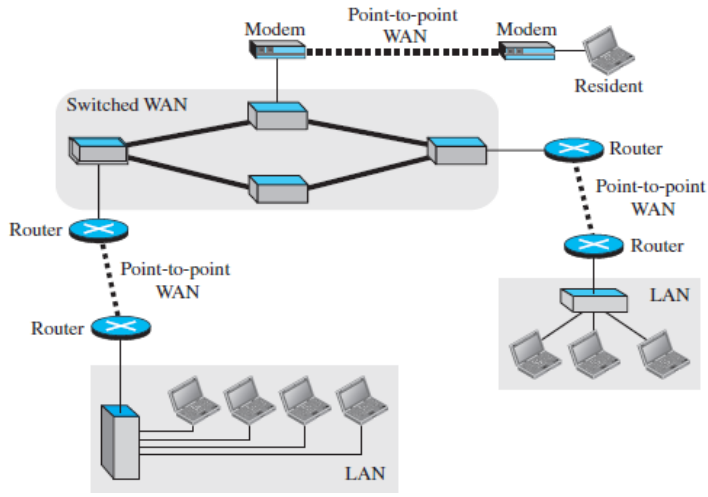


FIGURE 1.10: A HETEROGENEOUS NETWORK MADE OF FOUR WANS AND THREE LANS

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 1.11 shows a very simple switched network that connects four telephones to each end. We have used telephone sets instead of computers as an end system because circuit switching was very common in telephone networks in the past, although part of the telephone network today is a packet-switched network.

In Figure 1.11, the four telephones at each side are connected to a switch. The switch connects a telephone set at one side to a telephone set at the other side. The thick line connecting two switches is a high-capacity communication line that can handle four voice communications at the same time; the capacity can be shared between all pairs of telephone sets. The switches used in this example have forwarding tasks but no storing capability.

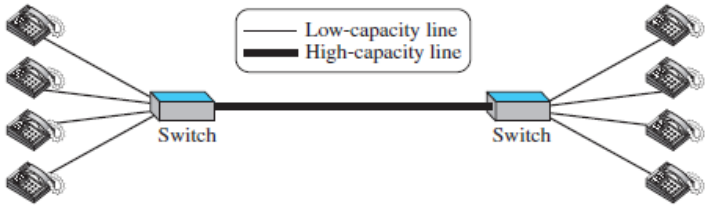


FIGURE 1.11: A CIRCUIT-SWITCHED NETWORK

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 1.12 shows a small packet-switched network that connects four computers at one site to four computers at the other site.

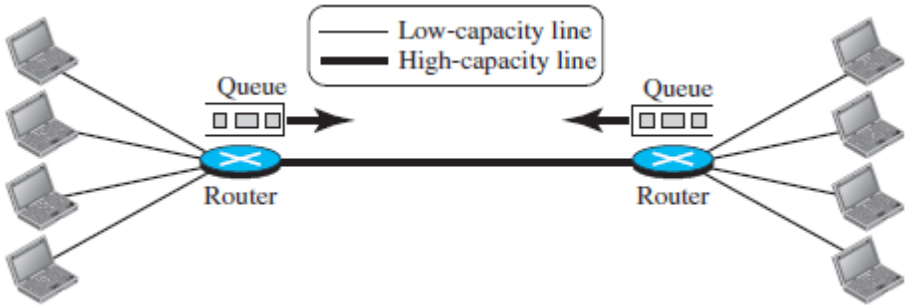


FIGURE 1.12: A PACKET-SWITCHED NETWORK

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: An internet (*note the lowercase i*) is two or more networks that can communicate with each other. The most notable internet is called the **Internet** (*uppercase I*), and is composed of thousands of interconnected networks. Figure 1.13 shows a conceptual (not geographical) view of the Internet.

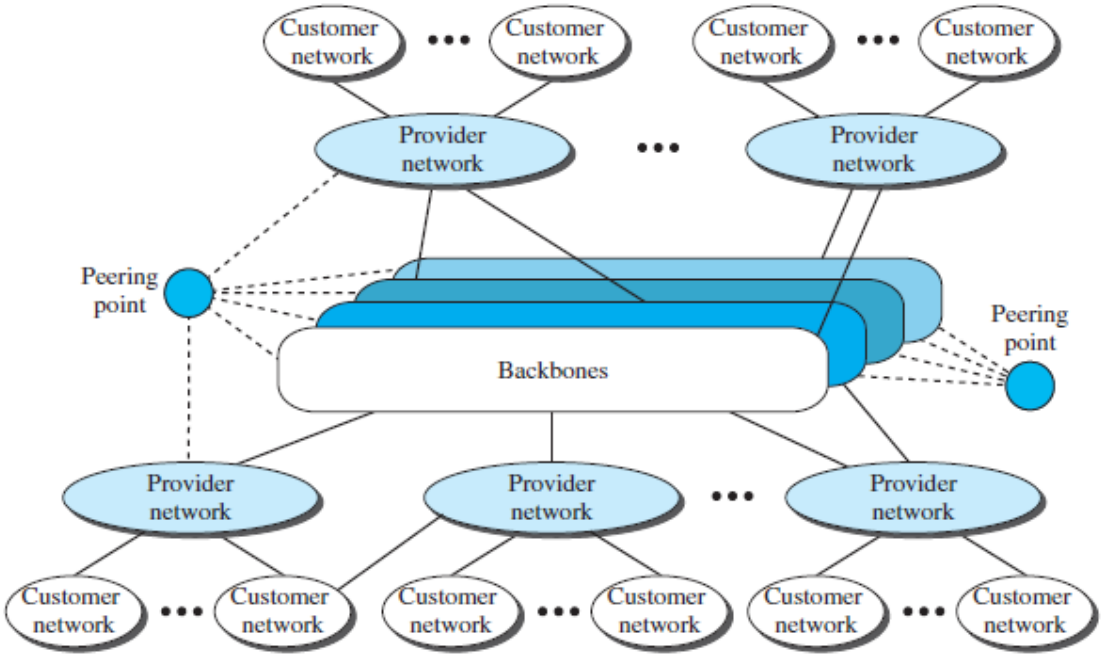


FIGURE 1.13: THE INTERNET TODAY

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*.

ACCESSING THE INTERNET:

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

USING TELEPHONE NETWORKS:

Today most residences and small businesses have telephone service, which means they are connected to a telephone network.

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

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

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

3. ***Using Cable Networks:*** More and more residents over the last two decades have begun using cable TV services instead of antennas to receive TV broadcasting. The cable companies have been upgrading their cable networks and connecting to the Internet. A residence or a small business can be connected to the Internet by using this service. It provides a higher speed connection, but the speed varies depending on the number of neighbors that use the same cable.

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

5. ***Direct Connection to the Internet:*** A large organization or a large corporation can itself become a local ISP and be connected to the Internet. This can be done if the organization or the corporation leases a high-speed WAN from a carrier provider and connects itself to a regional ISP.

a. For example, a large university with several campuses can create an internetwork and then connect the internetwork to the Internet.

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.

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

ARPANET: In the mid-1960s, mainframe computers in research organizations were stand-alone devices. Computers from different manufacturers were unable to communicate with one another.

The **Advanced Research Projects Agency (ARPA)** in the Department of Defense (DOD) was interested in finding a way to connect computers so that the researchers they funded could share their findings, thereby reducing costs and eliminating duplication of effort.

In 1967, at an Association for Computing Machinery (ACM) meeting, ARPA presented its ideas for the **Advanced Research Projects Agency Network (ARPANET),** a small network of connected computers.

The idea was that each host computer (not necessarily from the same manufacturer) would be attached to a specialized computer, called an *interface message processor* (IMP). The IMPs, in turn, would be connected to each other. Each IMP had to be able to communicate with other IMPs as well as with its own attached host.

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

BIRTH OF THE INTERNET:

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

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

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

Shortly thereafter, authorities made a decision to split TCP into two protocols: **Transmission Control Protocol (TCP)** and **Internet Protocol (IP).**

IP would handle datagram routing while TCP would be responsible for higher level functions such as segmentation, reassembly, and error detection. The new combination became known as TCP/IP.

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.

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.

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.

STANDARDS AND ADMINISTRATION:

INTERNET STANDARDS:

An **Internet standard** is a thoroughly tested specification that is useful to and adhered to by those who work with the Internet. It is a formalized regulation that must be followed. There is a strict procedure by which a specification attains Internet standard status.

A specification begins as an Internet draft. An **Internet draft** is a working document (a work in progress) with no official status and a six-month lifetime. Upon recommendation from the Internet authorities, a draft may be published as a **Request for Comment (RFC).**

Each RFC is edited, assigned a number, and made available to all interested parties. RFCs go through maturity levels and are categorized according to their requirement level.

Maturity Levels:

An RFC, during its lifetime, falls into one of six *maturity levels:* ***proposed standard***, ***draft standard***, ***Internet standard***, ***historic***, ***experimental***, and ***informational*** (see Figure 1.14)***.***

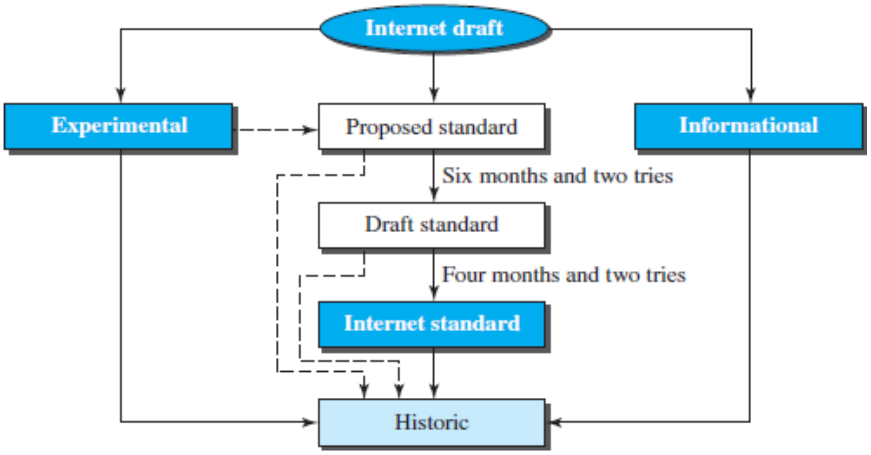


FIGURE 1.14: MATURITY LEVELS OF AN RFC

- ***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 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 1.15 shows the general organization of Internet administration.

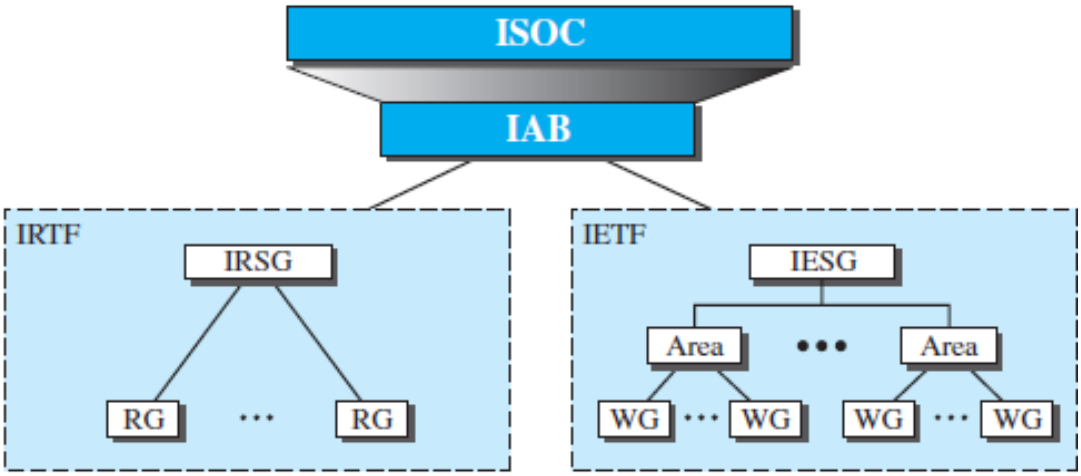


FIGURE 1.15: INTERNET ADMINISTRATION

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. IAB is also the external liaison (*meaning link / association*) 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.

NETWORK MODELS

PROTOCOL LAYERING:

In data communication and networking, a protocol defines the rules that both the sender and receiver and all intermediate devices need to follow to be able to communicate effectively.

- When communication is simple, we may need only one simple protocol.
- When the communication is complex, we may need to divide the task between different layers, in which case we need a protocol at each layer, or **protocol layering.**

Scenarios:

Let us develop two simple scenarios to better understand the need for protocol layering.

First Scenario

In the first scenario, communication is so simple that it can occur in only one layer. Assume Maria and Ann are neighbors with a lot of common ideas. Communication between Maria and Ann takes place in one layer, face to face, in the same language, as shown in Figure 1.16.

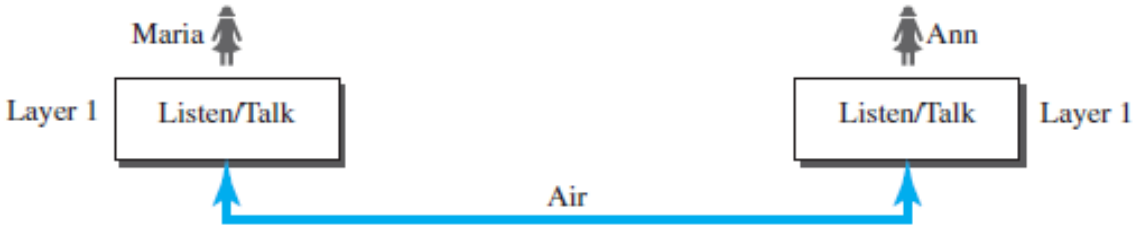


FIGURE 1.16: A SINGLE-LAYER PROTOCOL

Even in this simple scenario, we can see that a set of rules needs to be followed.

1. First, Maria and Ann know that they should greet each other when they meet.
2. Second, they know that they should confine their vocabulary to the level of their friendship.
3. Third, each party knows that she should refrain from speaking when the other party is speaking.
4. Fourth, each party knows that the conversation should be a dialog, not a monolog: both should have the opportunity to talk about the issue.
5. Fifth, they should exchange some nice words when they leave.

Second Scenario:

In the second scenario, we assume that Ann is offered a higher-level position in her company, but needs to move to another branch located in a city very far from Maria.

The two friends still want to continue their communication and exchange ideas because they have come up with an innovative project to start a new business when they both retire. They decide to continue their conversation using regular mail through the post office.

However, they do not want their ideas to be revealed by other people if the letters are intercepted. They agree on an encryption/decryption technique. The sender of the letter encrypts it to make it unreadable by an intruder; the receiver of the letter decrypts it to get the original letter.

For the moment we assume that Maria and Ann use one technique that makes it hard to decrypt the letter if one does not have the key for doing so. Now

we can say that the communication between Maria and Ann takes place in three layers, as shown in Figure 1.17.

We assume that Ann and Maria each have three machines (or robots) that can perform the task at each layer.

Let us assume that Maria sends the first letter to Ann. Maria talks to the machine at the third layer as though the machine is Ann and is listening to her.

The third layer machine listens to what Maria says and creates the plaintext (a letter in English), which is passed to the second layer machine. The second layer machine takes the plaintext, encrypts it, and creates the cipher text, which is passed to the first layer machine.

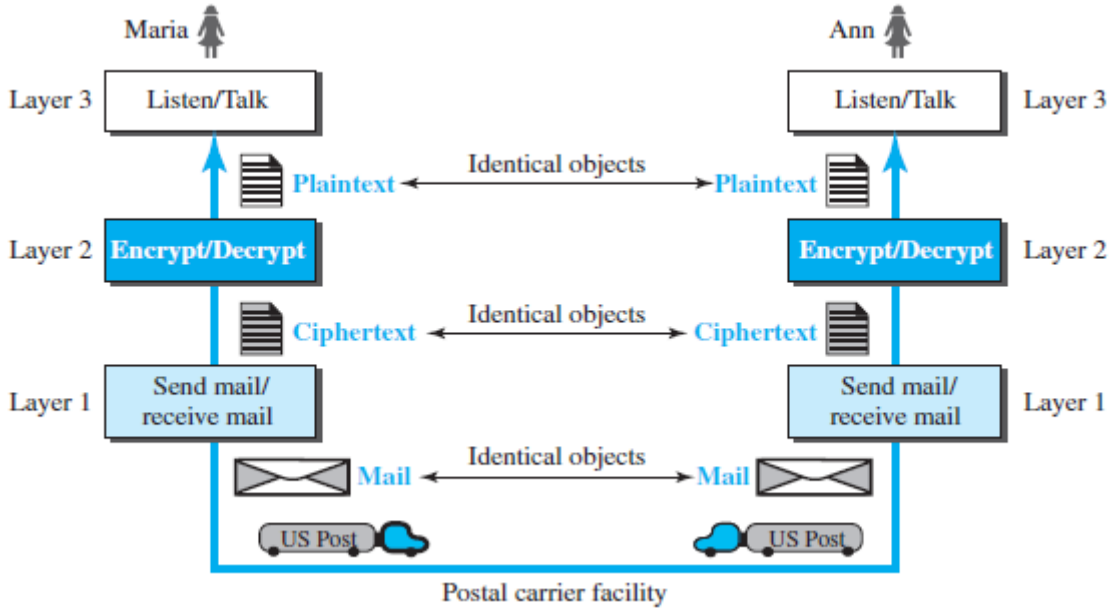


FIGURE 1.17: A THREE-LAYER PROTOCOL

The first layer machine, presumably a robot, takes the cipher text, puts it in an envelope, adds the sender and receiver addresses, and mails it.

At Ann's side, the first layer machine picks up the letter from Ann's mail box, recognizing the letter from Maria by the sender address. The machine takes out the cipher text from the envelope and delivers it to the second layer machine. The second layer machine decrypts the message, creates the plaintext, and passes the plaintext to the third-layer machine. The third layer machine takes the plaintext and reads it as though Maria is speaking.

Protocol layering enables us to divide a complex task into several smaller and simpler tasks. One of the advantages of protocol layering is that it allows us to separate the services from the implementation. A layer needs to be able to receive a set of services from the lower layer and to give the services to the upper layer; we don't care about how the layer is implemented.

Another advantage of protocol layering, which cannot be seen in our simple examples but reveals itself when we discuss protocol layering in the Internet, is that communication does not always use only two end systems; there are intermediate systems that need only some layers, but not all layers.

If we did not use protocol layering, we would have to make each intermediate system as complex as the end systems, which makes the whole system more expensive.

PRINCIPLES OF PROTOCOL LAYERING:

Let us discuss two principles of protocol layering.

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

For example, the third layer task is to listen (in one direction) and *talk* (in the other direction). The second layer needs to be able to encrypt and decrypt. The first layer needs to send and receive mail.

Second Principle: The second principle that we need to follow in protocol layering is that the two objects under each layer at both sites should be identical.

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 cipher text letter. The object under layer 1 at both sites should be a piece of mail.

Logical Connections:

After following the above two principles, we can think about logical connection between each layer as shown in Figure 1.18. This means that we have layer-to-layer communication. Maria and Ann can think that there is a logical (imaginary) connection at each layer through which they can send the object created from that layer.

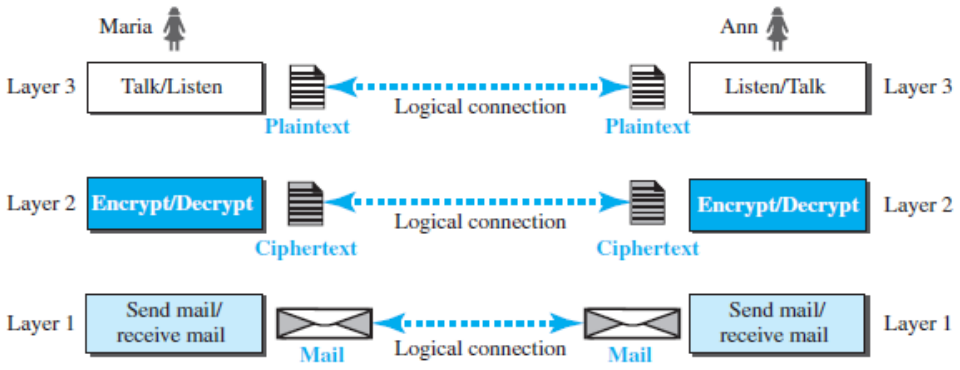


FIGURE 1.18: LOGICAL CONNECTION BETWEEN PEER LAYERS

TCP/IP PROTOCOL SUITE:

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. Figure 1.19 shows both configurations.

LAYERED ARCHITECTURE:

To show how the layers in the TCP/IP protocol suite are involved in communication between two hosts, we assume that we want to use the suite in a small internet made up of three LANs (links), each with a link-layer switch. We also assume that the links are connected by one router, as shown in Figure 1.20.

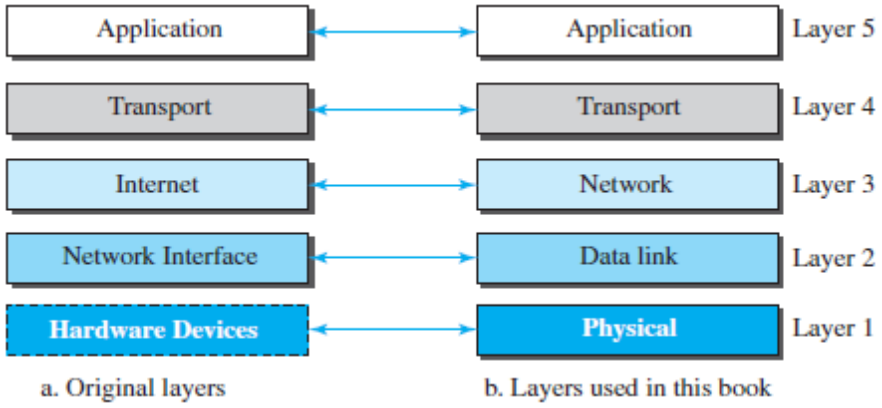


FIGURE 1.19: LAYERS IN THE TCP/IP PROTOCOL SUITE

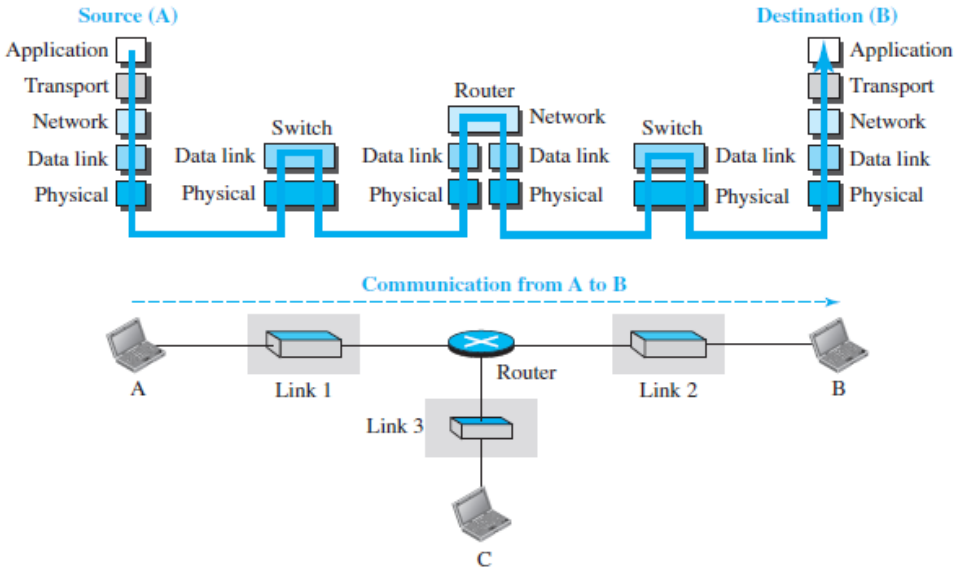


FIGURE 1.20: COMMUNICATION THROUGH AN INTERNET

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.

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:

To better understand the duties of each layer, we need to think about the logical connections between layers. Figure 1.21 shows logical connections in our simple internet.

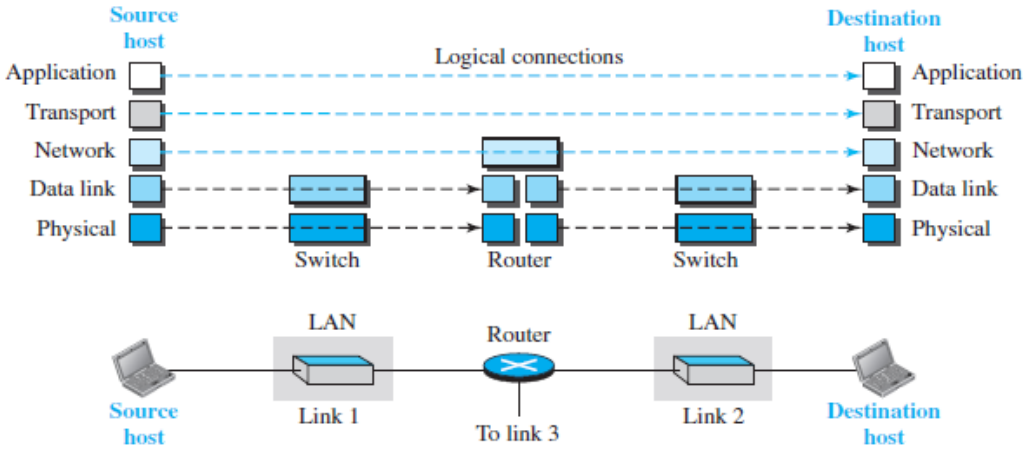


FIGURE 1.21: LOGICAL CONNECTIONS BETWEEN LAYERS OF THE TCP/IP PROTOCOL SUITE

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

Another way of thinking of the logical connections is to think about the data unit created from each layer. In the top three layers, the data unit (packets) should not be changed by any router or link-layer switch.

In the bottom two layers, the packet created by the host is changed only by the routers, not by the link-layer switches. Figure 1.22 shows the second principle discussed previously for protocol layering.

Note that, although the logical connection at the network layer is between the two hosts, we can only say that identical objects exist between two hops in this case because a router may fragment the packet at the network layer and send more packets than received. Note that the link between two hops does not change the object.

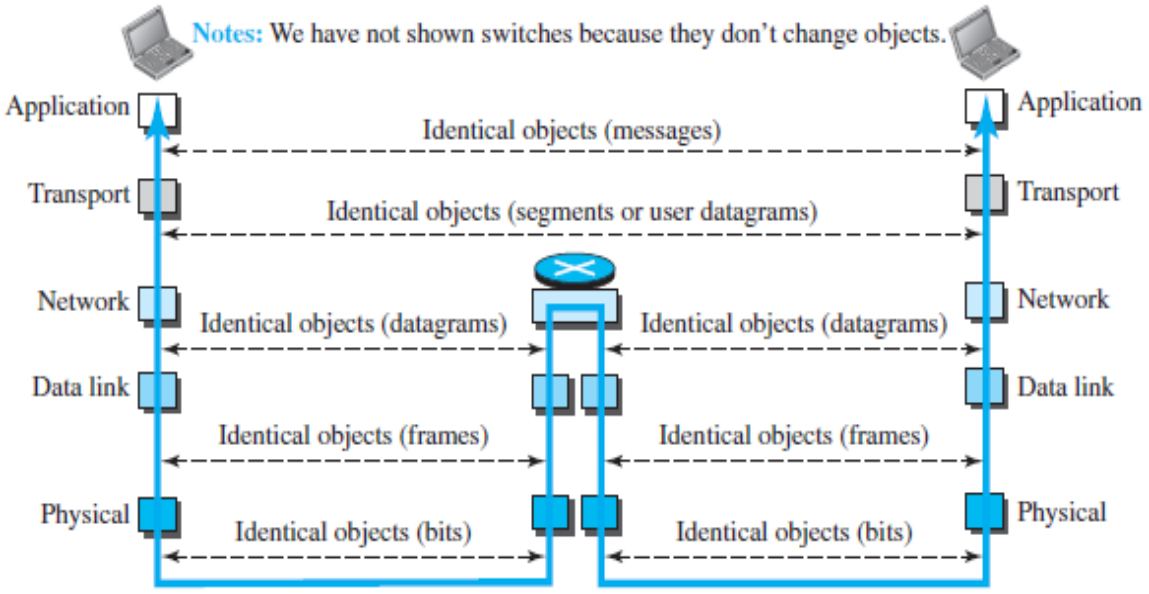


FIGURE 1.22: IDENTICAL OBJECTS IN THE TCP/IP PROTOCOL SUITE

DESCRIPTION OF EACH LAYER:

After understanding the concept of logical communication, we are ready to briefly discuss the duty of each layer.

Physical Layer:

We can say that 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*.

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.

We can say that the network layer is responsible for host-to-host communication and routing the packet through possible routes.

The network layer in the Internet includes the main protocol, Internet Protocol (IP) that defines the format of the packet, called a datagram at the network layer. IP also defines the format and the structure of addresses used in this layer.

IP is also responsible for routing a packet from its source to its destination, which is achieved by each router forwarding the datagram to the next router in its path. IP is a connectionless protocol that provides no flow control, no error control, and no congestion control services.

This means that if any of these services is required for an application, the application should rely only on the transport-layer protocol. The network layer also includes unicast (one-to-one) and multicast (one-to-many) routing protocols.

A routing protocol does not take part in routing (it is the responsibility of IP), but it creates forwarding tables for routers to help them in the routing process. The network layer also has some auxiliary protocols that help IP in its delivery and routing tasks.

The Internet Control Message Protocol (**ICMP**) helps IP to report some problems when routing a packet. The Internet Group Management Protocol (**IGMP**) is another protocol that helps IP in multitasking.

The Dynamic Host Configuration Protocol (**DHCP**) helps IP to get the network-layer address for a host. The Address Resolution Protocol (**ARP**) is a protocol that helps IP to find the link-layer address of a host or a router when its network-layer address is given.

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 transport layer packet (called a *segment* or a *user datagram* in different protocols) and sends it, through the logical 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.

There are a few transport-layer protocols in the Internet, each designed for some specific task. The main protocol, Transmission Control Protocol (TCP), is a connection-oriented protocol that first establishes a logical connection between transport layers at two hosts before transferring data. It creates a logical pipe between two TCPs for transferring a stream of bytes.

TCP provides flow control (matching the sending data rate of the source host with the receiving data rate of the destination host to prevent overwhelming the destination), error control (to guarantee that the segments arrive at the destination without error and resending the corrupted ones), and congestion control to reduce the loss of segments due to congestion in the network.

The other common protocol, User Datagram Protocol (UDP), is a connectionless protocol that transmits user datagram's without first creating a logical connection. In UDP, each user datagram is an independent entity without being related to the previous or the next one (the meaning of the term *connectionless*).

UDP is a simple protocol that does not provide flow, error, or congestion control. Its simplicity, which means small overhead, is attractive to an application program that needs to send short messages and cannot afford the retransmission of the packets involved in TCP, when a packet is corrupted or lost.

A new protocol, Stream Control Transmission Protocol (SCTP) is designed to respond to new applications that are emerging in the multimedia.

Application Layer:

As Figure 1.21 shows, the logical connection between the two application layers is end-to-end. The two application layers exchange *messages* between each other as though there were a bridge between the two layers. However, we should know that the communication is done through all the layers.

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.

The Hypertext Transfer Protocol (HTTP) is a vehicle for accessing the World Wide Web (WWW). The Simple Mail Transfer Protocol (SMTP) is the main protocol used in electronic mail (e-mail) service. The File Transfer Protocol (FTP) is used for transferring files from one host to another.

The Terminal Network (TELNET) and Secure Shell (SSH) are used for accessing a site remotely. The Simple Network Management Protocol (SNMP) is used by an administrator to manage the Internet at global and local levels.

The Domain Name System (DNS) is used by other protocols to find the network-layer address of a computer. The Internet Group Management Protocol (IGMP) is used to collect membership in a group.

ENCAPSULATION AND DECAPSULATION:

One of the important concepts in protocol layering in the Internet is encapsulation/ decapsulation. Figure 1.23 shows this concept for the small internet in Figure 1.20.

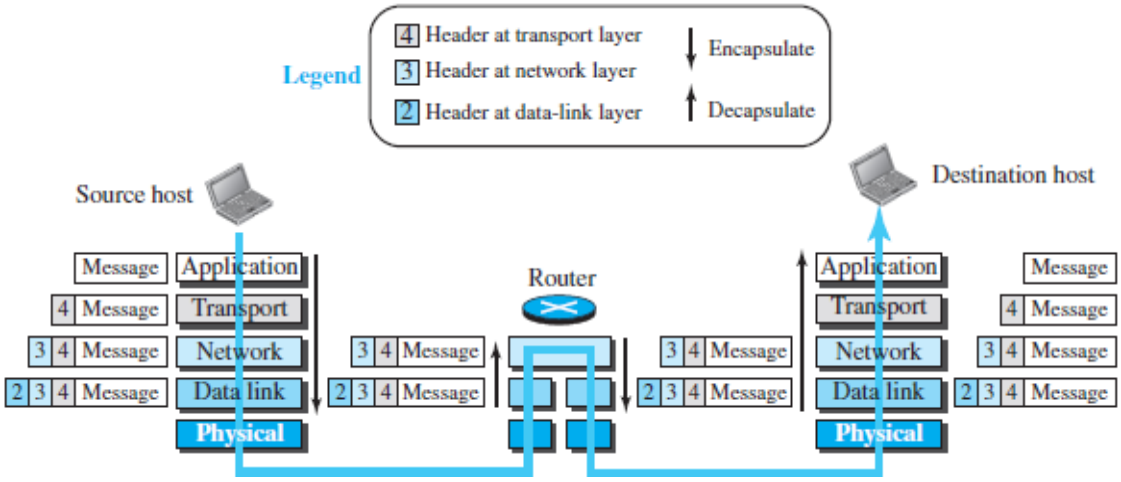


FIGURE 1.23: ENCAPSULATION/DECAPSULATION

We have not shown the layers for the link-layer switches because no encapsulation/decapsulation occurs in this device. In Figure 1.23, we show the encapsulation in the source host, decapsulation in the destination host, and encapsulation and decapsulation in the router.

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-to-end delivery of the message, such as information needed for flow, error control, or congestion control.

   The result is the transport-layer packet, which is called the *segment* (in TCP) and the *user datagram* (in UDP). The transport layer then passes the packet to the network layer.

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

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

Decapsulation and Encapsulation at the Router: At the router, we have both decapsulation and encapsulation because the router is connected to two or more links.

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

2. The network layer only inspects the source and destination addresses in the datagram header and consults its forwarding table to find the next hop to which the datagram is to be delivered.

   The contents of the datagram should not be changed by the network layer in the router unless there is a need to fragment the datagram if it is too big to be passed through the next link. The datagram is then passed to the data-link layer of the next link.

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

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: It is worth mentioning another concept related to protocol layering in the Internet, *addressing*. As we discussed before, we have logical communication between pairs of layers in this model.

Any communication that involves two parties needs two addresses: source address and destination address. Although it looks as if we need five pairs of addresses, one pair per layer, we normally have only four because the physical layer does not need addresses; the unit of data exchange at the physical layer is a bit, which definitely cannot have an address. Figure 1.24 shows the addressing at each layer.

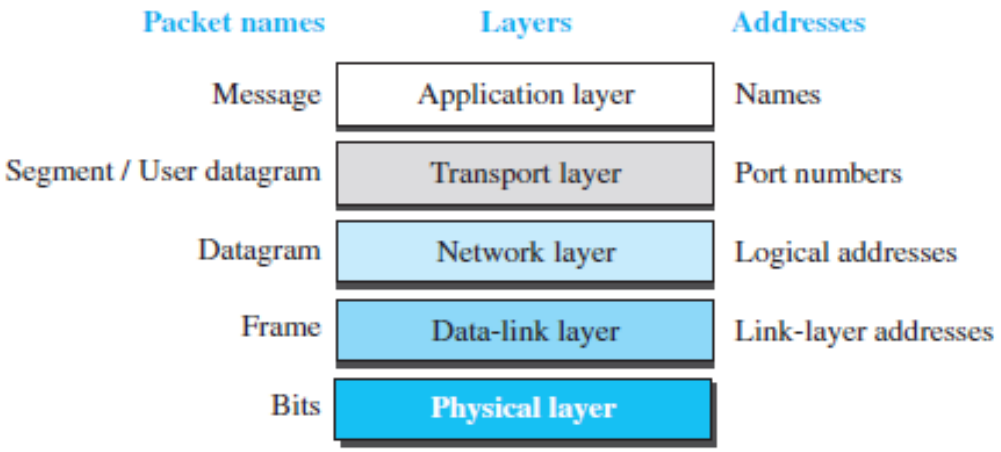


FIGURE 1.24: ADDRESSING IN THE TCP/IP PROTOCOL SUITE

As the figure shows, there is a *relationship between the layer*, *the address used in that layer*, and *the packet name at that layer*.

- ☑ At the application layer, we normally use names to define the site that provides services, such as *someorg.com*, or the e-mail address, such as *somebody@coldmail.com.*
- ☑ At the transport layer, addresses are called port numbers, and these define the application-layer programs at the source and destination. Port numbers are local addresses that distinguish between several programs running at the same time.
- ☑ At the network-layer, the addresses are global, with the whole Internet as the scope. A network-layer address uniquely defines the connection of a device to the Internet.
- ☑ 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).

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 1.25 shows the concept of multiplexing and demultiplexing at the three upper layers.

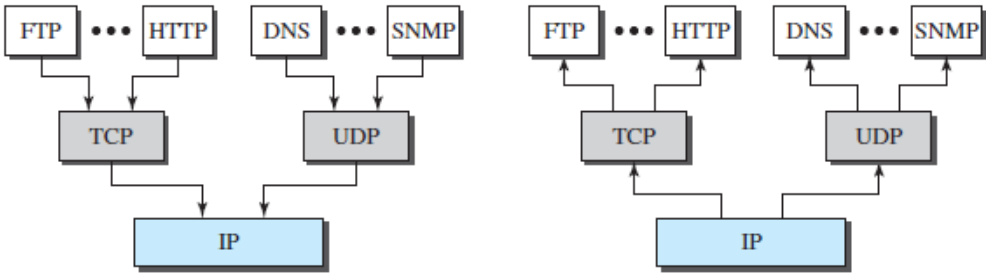


FIGURE 1.25: MULTIPLEXING AND DEMULTIPLEXING

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.

- ☑ At the transport layer, either UDP or TCP can accept a message from several application-layer protocols.
- ☑ At the network layer, IP can accept a segment from TCP or a user datagram from UDP. IP can also accept a packet from other protocols such as ICMP, IGMP, and so on.
- ☑ At the data-link layer, a frame may carry the payload coming from IP or other protocols such as ARP.

THE OSI MODEL:

*Established in 1947, the **International Organization for Standardization (ISO)** is a multinational body dedicated to worldwide agreement on international standards. Almost three-fourths of the countries in the world are represented in the ISO. 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 (see Figure 1.26).

OSI versus TCP/IP:

When we compare the two models, we find that two layers, session and presentation, are missing from the TCP/IP protocol suite. These two layers were not added to the TCP/IP protocol suite after the publication of the OSI model. The application layer in the suite is usually considered to be the combination of three layers in the OSI model, as shown in Figure 1.27.

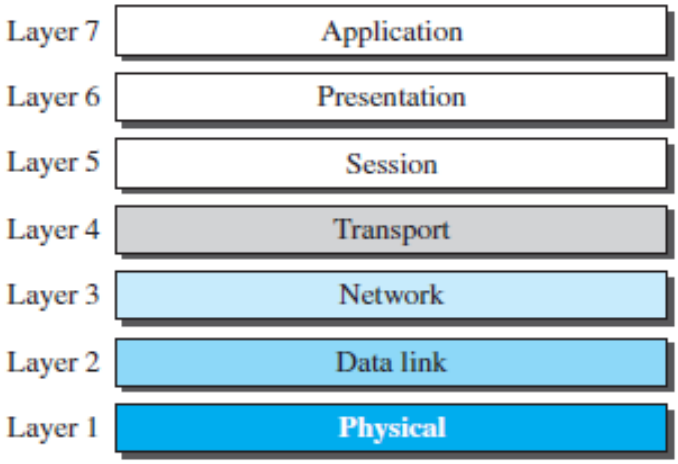


FIGURE 1.26: THE OSI MODEL

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.

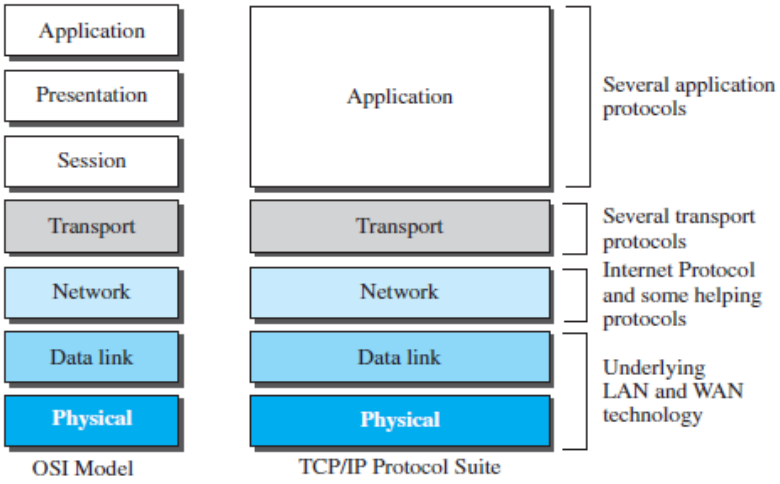


FIGURE 1.27: TCP/IP AND OSI MODEL

LACK OF OSI MODEL'S SUCCESS:

The OSI model appeared after the TCP/IP protocol suite. *Most experts were at first excited and thought that the TCP/IP protocol would be fully replaced by the OSI model. This did not happen for several reasons, but we describe only three, which are agreed upon by all experts in the field.*

First, OSI was completed when TCP/IP was fully in place and a lot of time and money had been spent on the suite; changing it would cost a lot.

Second, some layers in the OSI model were never fully defined. For example, although the services provided by the presentation and the session layers were listed in the document, actual protocols for these two layers were not fully defined, nor were they fully described, and the corresponding software was not fully developed.

Third, when OSI was implemented by an organization in a different application, it did not show a high enough level of performance to entice (*meaning attract*) the Internet authority to switch from the TCP/IP protocol suite to the OSI model.

INTRODUCTION TO PHYSICAL LAYER

DATA AND SIGNALS:

Figure 1.28 shows a scenario in which a scientist working in a research company, Sky Research, needs to order a book related to her research from an online bookseller, Scientific Books.

We can think of five different levels of communication between Alice, the computer on which our scientist is working, and Bob, the computer that provides online service. Communication at application, transport, network, or data-link is logical; communication at the physical layer is physical.

For simplicity, we have shown only host-to-router, router-to-router, and router-to-host, but the switches are also involved in the physical communication.

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

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.

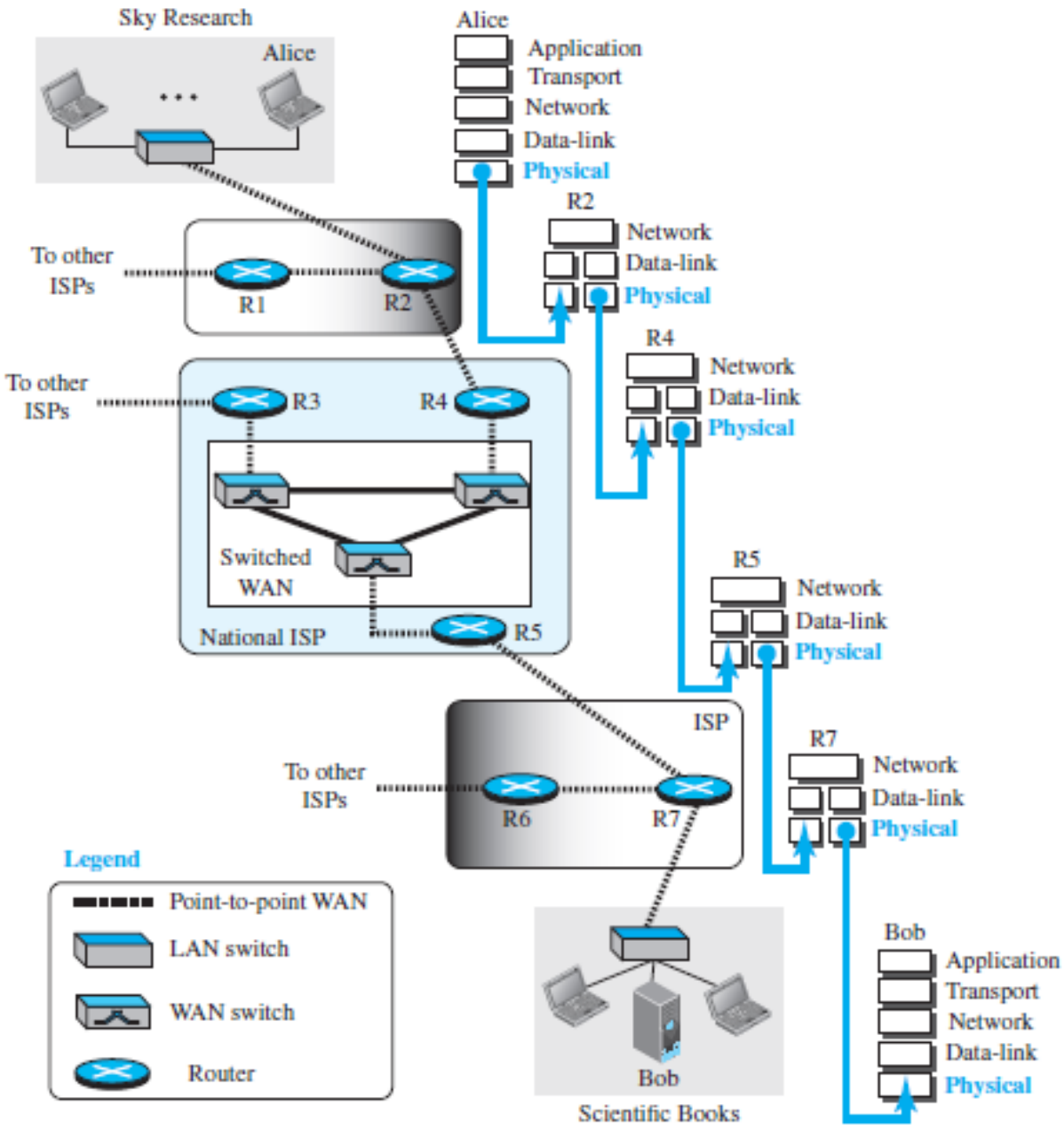


FIGURE 1.28: COMMUNICATION AT THE PHYSICAL LAYER

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 (*meaning strength/power*) 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 1.29 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.

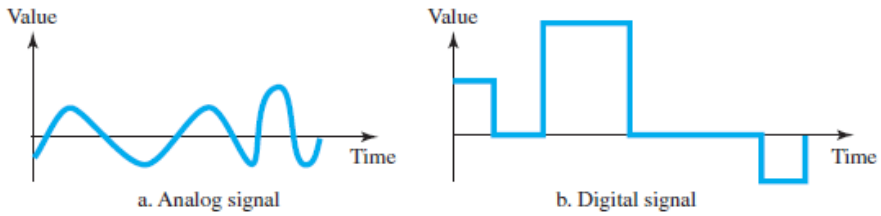


FIGURE 1.29: COMPARISON OF ANALOG AND DIGITAL SIGNALS

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.

TRANSMISSION IMPAIRMENT:

Signals travel through transmission media, which are not perfect. The imperfection causes signal impairment. This means that the signal at the beginning of the medium is not the same as the signal at the end of the medium. What is sent is not what is received. Three causes of impairment are ***attenuation***, ***distortion***, and ***noise*** (see Figure 1.32).

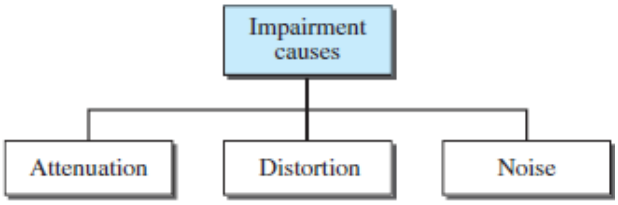


FIGURE 1.32: CAUSES OF IMPAIRMENT

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 (*meaning enlarge on/go into detail/develop/expand/clarify/add details to*) the signal.

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.

Noise:

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

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

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

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

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

DATA RATE LIMITS:

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

1. The bandwidth available
2. The level of the signals we use
3. The quality of the channel (the level of noise)

Two theoretical formulas were developed to calculate the data rate: one by ***Nyquist for a noiseless channel***, another by ***Shannon for a noisy channel***.

Noiseless Channel: Nyquist Bit Rate:

For a noiseless channel, the **Nyquist bit rate** formula defines the theoretical maximum bit rate **BitRate = 2 x bandwidth x $\log_2 L$**

In this formula, bandwidth is the bandwidth of the channel, *L* is the number of signal levels used to represent data, and BitRate is the bit rate in bits per second. According to the formula, we might think that, given a specific bandwidth, we can have any bit rate we want by increasing the number of signal levels.

Although the idea is theoretically correct, practically there is a limit. When we increase the number of signal levels, we impose a burden on the receiver.

Increasing the levels of a signal may reduce the reliability of the system.

Noisy Channel: Shannon Capacity:

In reality, we cannot have a noiseless channel; the channel is always noisy. In 1944, Claude Shannon introduced a formula, called the **Shannon capacity,** to determine the theoretical highest data rate for a noisy channel: **Capacity = bandwidth x $\log_2$ (1 + SNR)**

In this formula, bandwidth is the bandwidth of the channel, SNR is the signal-to-noise ratio, and capacity is the capacity of the channel in bits per second.

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? There are certain characteristics that measure the network performance which are given as follows:

BANDWIDTH:

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

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

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

Relationship: There is an explicit relationship between the bandwidth in hertz and bandwidth in bits per second. Basically, an increase in bandwidth in hertz means an increase in bandwidth in bits per second.

THROUGHPUT:

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

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

Imagine a highway designed to transmit 1000 cars per minute from one point to another. However, if there is congestion on the road, this figure may be reduced to 100 cars per minute. The bandwidth is 1000 cars per minute; the throughput is 100 cars per minute.

LATENCY (DELAY):

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

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

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

Propagation time = Distance / (Propagation Speed)

Transmission Time: In data communications we don't send just 1 bit, we send a message. The first bit may take a time equal to the propagation time to

reach its destination; the last bit also may take the same amount of time. However, there is a time between the first bit leaving the sender and the last bit arriving at the receiver.

The first bit leaves earlier and arrives earlier; the last bit leaves later and arrives later. The **transmission time** of a message depends on the size of the message and the bandwidth of the channel.

Transmission time = (Message size) / Bandwidth

Queuing Time: The third component in latency is the **queuing time**, the time needed for each intermediate or end device to hold the message before it can be processed. The queuing time is not a fixed factor; it changes with the load imposed on the network.

When there is heavy traffic on the network, the queuing time increases. An intermediate device, such as a router, queues they arrived messages and processes them one by one. If there are many messages, each message will have to wait.

Bandwidth-Delay Product

Bandwidth and delay are two performance metrics of a link. ***The bandwidth-delay product defines the number of bits that can fill the link.***

JITTER:

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

TRANSMISSION MEDIA

INTRODUCTION:

Transmission media are actually located below the physical layer and are directly controlled by the physical layer. We could say that transmission media belong to layer zero. Figure 1.33 shows the position of transmission media in relation to the physical layer.

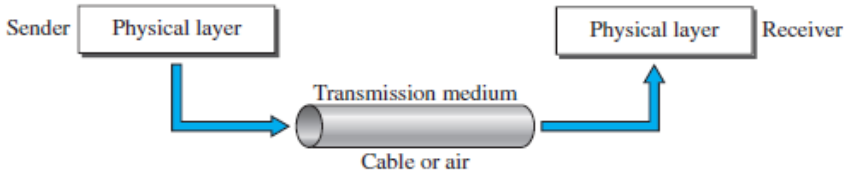


FIGURE 1.33: TRANSMISSION MEDIUM AND PHYSICAL LAYER

- ☑ 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.
- ☑ In data communications the definition of the information and the transmission medium is more specific. The transmission medium is usually free space, metallic cable, or fiber-optic cable.
- ☑ The information is usually a signal that is the result of a conversion of data from another form.
- ☑ The use of long-distance communication using electric signals started with the invention of the telegraph by Morse in the 19th century.
- ☑ Communication by telegraph was slow and dependent on a metallic medium. Extending the range of the human voice became possible when the *telephone was invented in 1869*.
- ☑ Telephone communication at that time also needed a metallic medium to carry the electric signals that were the result of a conversion from the human voice.
- ☑ The communication was, however, unreliable due to the poor quality of the wires. The lines were often noisy and the technology was unsophisticated.
- ☑ *Wireless communication started in 1895* when Hertz was able to send high frequency signals. Later, Marconi devised a method to send telegraph-type messages over the Atlantic Ocean.
- ☑ We have come a long way. Better metallic media have been invented (twisted-pair and coaxial cables, for example).

The use of optical fibers has increased the data rate incredibly. Free space (air, vacuum, and water) is used more efficiently, in part due to the technologies (such as modulation and multiplexing).

Electromagnetic energy, a combination of electric and magnetic fields vibrating in relation to each other, includes power, radio waves, infrared light, visible light, and ultraviolet light, and X, gamma, and cosmic rays. Each of these constitutes a portion of the **electromagnetic spectrum.**

In telecommunications, transmission media can be divided into two broad categories: ***guided*** and ***unguided***.

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

Unguided medium is ***free space***. Figure 1.34 shows this taxonomy.

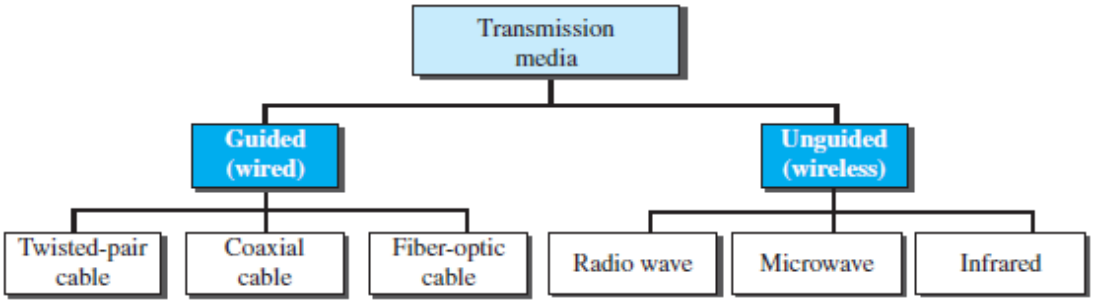


FIGURE 1.34: CLASSES OF TRANSMISSION MEDIA

GUIDED MEDIA:

Guided media, which are those that provide a conduit (*meaning medium*) 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, as shown in Figure 1.35.

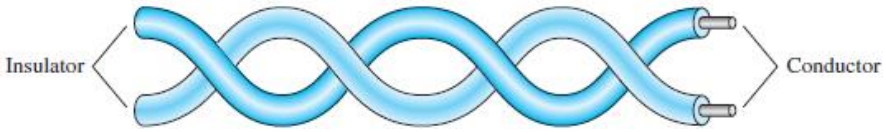


FIGURE 1.35: TWISTED-PAIR CABLE

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.

By twisting the pairs, a balance is maintained. For example, suppose in one twist, one wire is closer to the noise source and the other is farther; in the next twist, the reverse is true. Twisting makes it probable that both wires are equally affected by external influences (noise or crosstalk). This means that the receiver, which calculates the difference between the two, receives no unwanted signals.

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 1.36 shows the difference between UTP and STP.

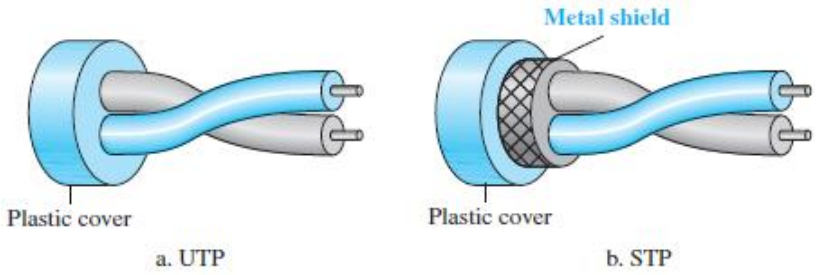


FIGURE 1.36: UTP AND STP CABLES

Categories: The Electronic Industries Association (EIA) has developed standards to classify unshielded twisted-pair cable into seven categories. Categories are determined by cable quality, with 1 as the lowest and 7 as the highest. Each EIA category is suitable for specific uses. Table 1.1 shows these categories.

CATEGORY	SPECIFICATION	DATA RATE (Mbps)	USE
1	Unshielded twisted-pair used in telephone	<0.1	Telephone
2	Unshielded twisted-pair originally used in T lines	2	T-1 lines
3	Improved CAT 2 used in LANs	10	LANs
4	Improved CAT 3 used in Token Ring networks	20	LANs
5	Cable wire is normally 24 AWG with a jacket and outside sheath (*meaning case/cover*)	100	LANs
5E	An extension to category 5 that includes extra features to minimize the crosstalk and	125	LANs

	electromagnetic interference		
6	A new category with matched components coming from the same manufacturer. The cable must be tested at a 200-Mbps data rate.	200	LANs
7	Sometimes called SSTP (shielded screen twisted-pair). Each pair is individually wrapped in a helical metallic foil followed by a metallic foil shield in addition to the outside sheath. The shield decreases the effect of crosstalk and increases the data rate.	600	LANs

TABLE 1.1: CATEGORIES OF UNSHIELDED TWISTED-PAIR CABLES

Connectors: The most common UTP connector is **RJ45** (RJ stands for registered jack), as shown in Figure 1.37. The RJ45 is a keyed connector, meaning the connector can be inserted in only one way.

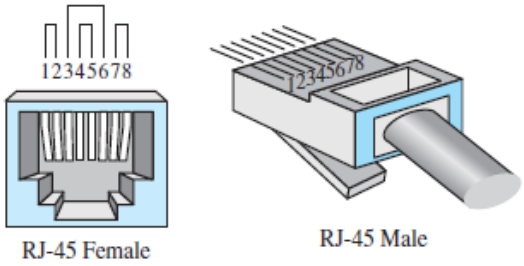


FIGURE 1.37: UTP CONNECTOR

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.

Applications: Twisted-pair cables are used in telephone lines to provide voice and data channels. The local loop—the line that connects subscribers to the central telephone office— commonly consists of unshielded twisted-pair cables.

The DSL lines that are used by the telephone companies to provide high-data-rate connections also use the high-bandwidth capability of unshielded twisted-pair cables.

COAXIAL CABLE:

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

insulating sheath, and the whole cable is protected by a plastic cover (see Figure 1.38).

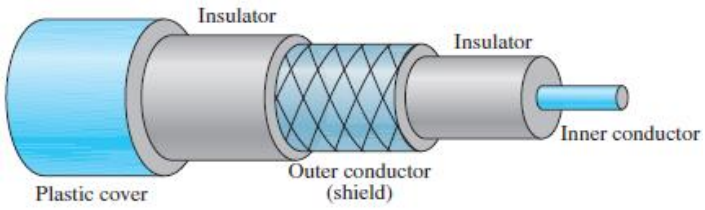


FIGURE 1.38: COAXIAL CABLE

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. Each cable defined by an RG rating is adapted for a specialized function, as shown in Table 1.2.

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

TABLE 1.2: CATEGORIES OF COAXIAL CABLES

Coaxial Cable Connectors:

To connect coaxial cable to devices, we need coaxial connectors. The most common type of connector used today is the **Bayonet Neill-Concelman (BNC)** connector. Figure 1.39 shows three popular types of these connectors: the BNC connector, the BNC T connector, and the BNC terminator.

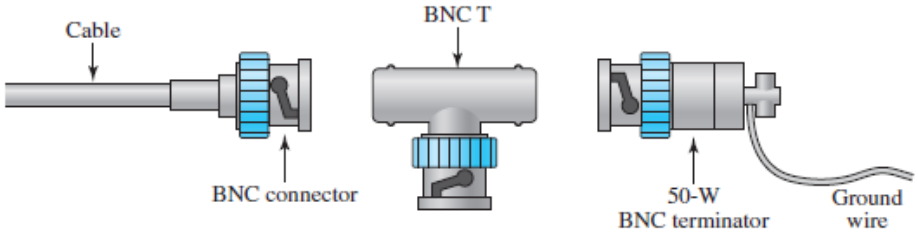


FIGURE 1.39: BNC CONNECTORS

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 is traveling through one substance suddenly enters another substance (of a different density), the ray changes direction.

Figure 1.40 shows how a ray of light changes direction when going from a more dense to a less dense substance.

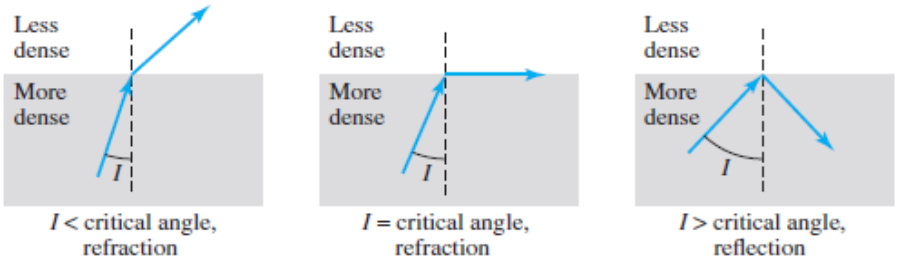


FIGURE 1.40: BENDING OF LIGHT RAY

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 substance.

Optical fibers use reflection to guide light through a channel. A glass or plastic **core** is surrounded by a **cladding** of less dense glass or plastic. The difference in density of the two materials must be such that a beam of light moving through the core is reflected off the cladding instead of being refracted into it. See Figure 1.41.

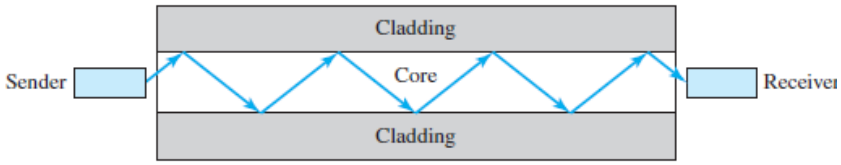


FIGURE 1.41: OPTICAL FIBER

Propagation Modes: Current technology supports two modes (multimode and single mode) for propagating light along optical channels, each requiring fiber with different physical characteristics. Multimode can be implemented in two forms: step-index or graded-index (see Figure 1.42).

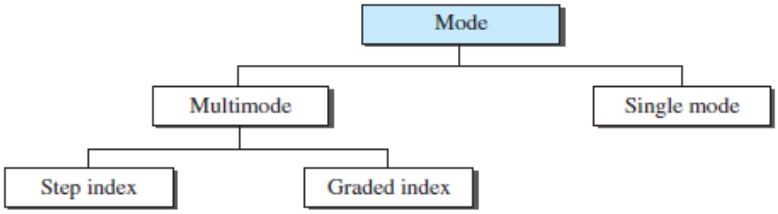


FIGURE 1.42: PROPAGATION MODES

Multimode: Multimode is so named because multiple beams from a light source move through the core in different paths. How these beams move within the cable depends on the structure of the core, as shown in Figure 1.43.

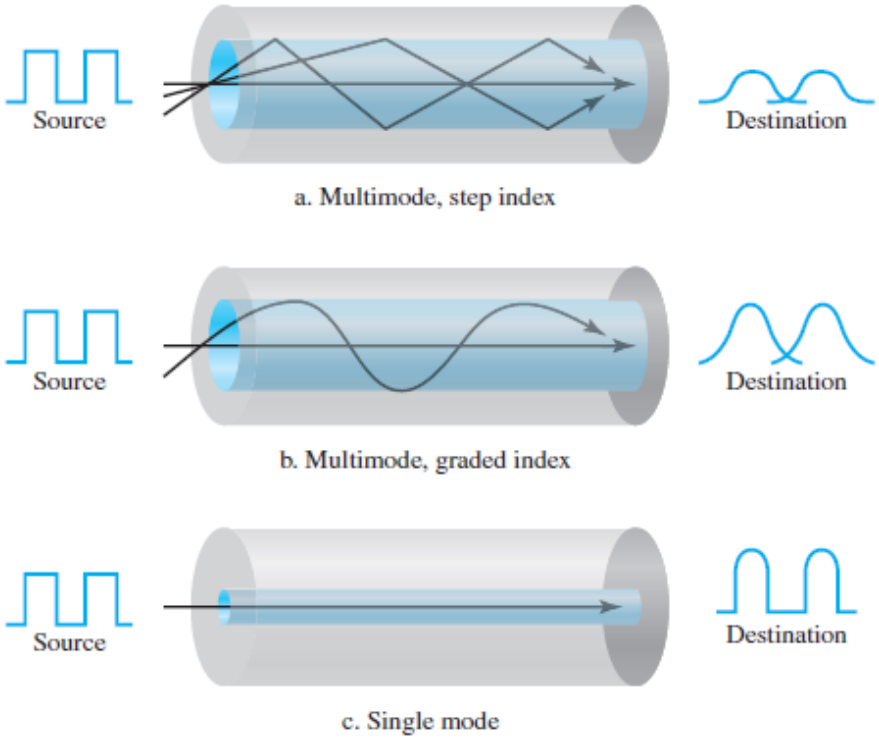


FIGURE 1.43: MODES

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

A second type of fiber, called **multimode graded-index fiber,** decreases this distortion of the signal through the cable. The word *index* here refers to the index of refraction. As we saw above, the index of refraction is related to density. A graded-index fiber, therefore, is one with varying densities. Density is highest at the center of the core and decreases gradually to its lowest at the edge. Figure 1.43 shows the impact of this variable density on the propagation of light beams.

Single-Mode: Single-mode uses step-index fiber and a highly focused source of light that limits beams to a small range of angles, all close to the horizontal.

The **single-mode fibers** itself is manufactured with a much smaller diameter than that of multimode fiber and with substantially lower density (index of refraction). The decrease in density results in a critical angle that is close enough to 90° to make the propagation of beams almost horizontal. In this case, propagation of different beams is almost identical, and delays are negligible. All the beams arrive at the destination "together" and can be recombined with little distortion to the signal (see Figure 1.43).

Fiber Sizes:

Optical fibers are defined by the ratio of the diameter of their core to the diameter of their cladding, both expressed in micrometers. The common sizes are shown in Table 1.3. Note that the last size listed is for single-mode only.

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

TABLE 1.3: FIBER TYPES

Cable Composition

Figure 1.44 shows the composition of a typical fiber-optic cable. The outer jacket is made of either PVC or Teflon. Inside the jacket are Kevlar strands to strengthen the cable.

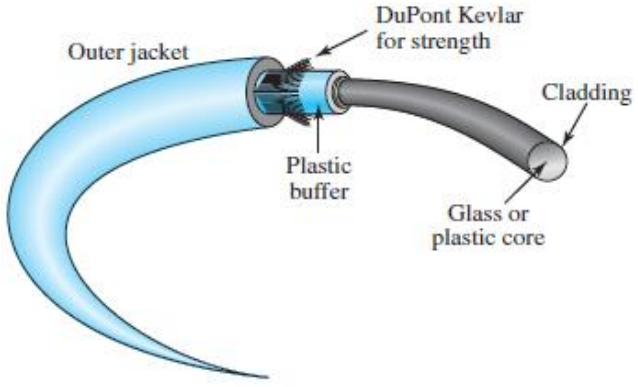


FIGURE 1.44: FIBER CONSTRUCTION

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.

Fiber-Optic Cable Connectors:

There are three types of connectors for fiber-optic cables, as shown in Figure 1.45. 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.

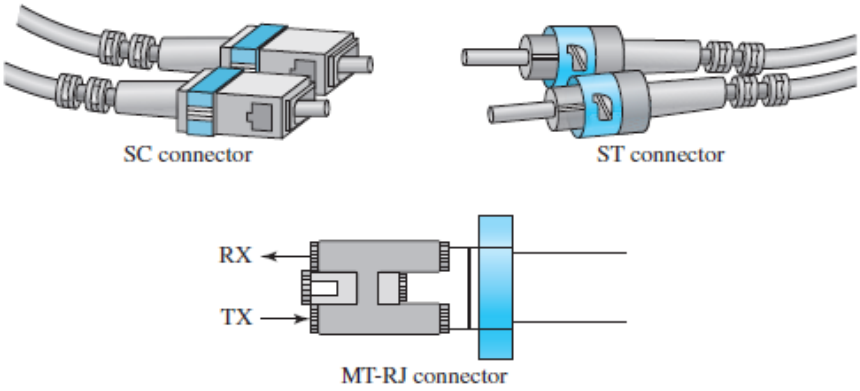


FIGURE 1.45: FIBER-OPTIC CABLE CONNECTORS

Performance

The plot of attenuation versus wavelength in Figure 1.46 shows a very interesting phenomenon in fiber-optic cable. Attenuation is flatter than in the case of twisted-pair cable and coaxial cable. The performance is such that we need fewer (actually one tenth as many) repeaters when we use fiber-optic cable.

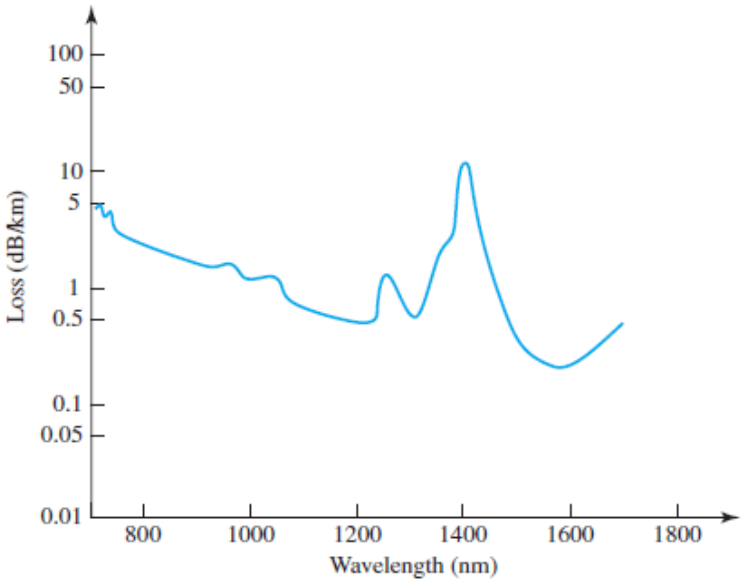


FIGURE 1.46: OPTICAL FIBER PERFORMANCE

Applications:

Fiber-optic cable is often found in backbone networks because its wide bandwidth is cost-effective. Today, with wavelength-division multiplexing (WDM), we can transfer data at a rate of 1600 Gbps.

Some cable TV companies use a combination of optical fiber and coaxial cable, thus creating a hybrid network. Optical fiber provides the backbone structure while coaxial cable provides the connection to the user premises. This is a cost-effective configuration since the narrow bandwidth requirement at the user end does not justify the use of optical fiber.

Local-area networks such as 100Base-FX network (Fast Ethernet) and 1000Base-X also use fiber-optic cable.

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.

Unguided signals can travel from the source to the destination in several ways: ground propagation, sky propagation, and line-of-sight propagation, as shown in Figure 1.47.

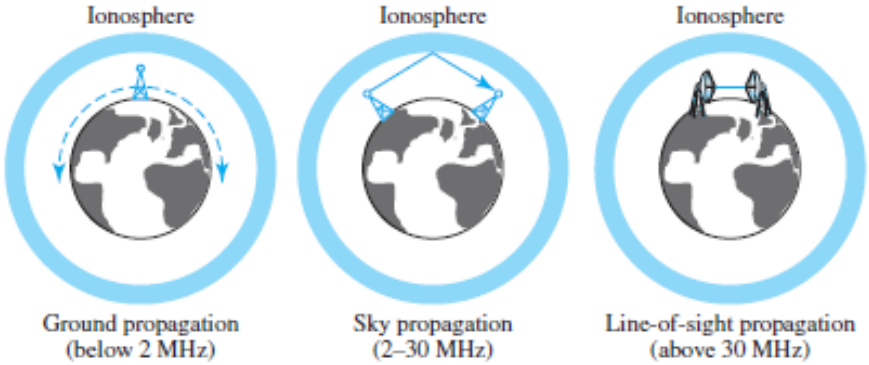


FIGURE 1.47: PROPAGATION METHODS

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). Table 1.4 lists these bands, their ranges, propagation methods, and some applications.

BAND	RANGE	PROPAGATION	APPLICATION
Very low frequency (VLF)	3-30kHz	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

TABLE 1.4: BANDS (CONTINUED)

We can divide wireless transmission into three broad groups: radio waves, microwaves, and infrared waves.

RADIO WAVES:

Although there is no clear-cut demarcation between radio waves and microwaves, electromagnetic waves ranging in frequencies between 3 kHz and 1 GHz are normally called **radio waves;** waves ranging in frequencies between 1 and 300 GHz are called **microwaves.**

However, the behavior of the waves, rather than the frequencies, is a better criterion for classification. Radio waves, for the most part, are omnidirectional. When an antenna transmits radio waves, they are propagated in all directions.

This means that the sending and receiving antennas do not have to be aligned. A sending antenna sends waves that can be received by any receiving antenna. The omnidirectional property has a disadvantage, too. The radio waves

transmitted by one antenna are susceptible to interference by another antenna that may send signals using the same frequency or band.

Radio waves, particularly those waves that propagate in the sky mode, can travel long distances. This makes radio waves a good candidate for long-distance broadcasting such as AM radio.

Radio waves, particularly those of low and medium frequencies, can penetrate walls. This characteristic can be both an advantage and a disadvantage.

- It is an advantage because, for example, an AM radio can receive signals inside a building.
- It is a disadvantage because we cannot isolate a communication to just inside or outside a building.

The radio wave band is relatively narrow, just under 1 GHz, compared to the microwave band. When this band is divided into subbands, the subbands are also narrow, leading to a low data rate for digital communications.

Omnidirectional Antenna: Radio waves use **omnidirectional antennas** that send out signals in all directions. Based on the wavelength, strength, and the purpose of transmission, we can have several types of antennas. Figure 1.48 shows an omnidirectional antenna.

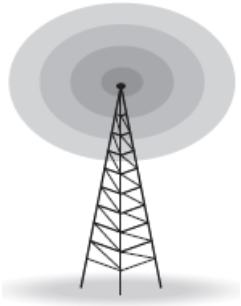


FIGURE 1.48: OMNIDIRECTIONAL ANTENNA

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: Electromagnetic waves having frequencies between 1 and 300 GHz are called microwaves. Microwaves are unidirectional. When an antenna transmits microwaves, they can be narrowly focused. This means that the sending and receiving antennas need to be aligned. The unidirectional property has an obvious advantage. A pair of antennas can be aligned without interfering with another pair of aligned antennas.

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 (see Figure 1.49).

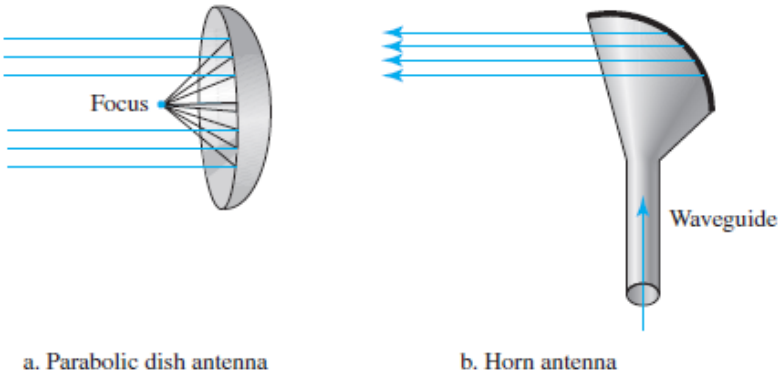


FIGURE 1.49: UNIDIRECTIONAL ANTENNAS

Applications:

Microwaves, due to their unidirectional properties, are very useful when unicast (one-to- one) communication is needed between the sender and the receiver. They are used in cellular phones, satellite networks and wireless LANs.

Infrared:

Infrared waves, with frequencies from 300 GHz to 400 THz (wavelengths from 1 mm to 770 nm), can be used for short-range communication. Infrared waves, having high frequencies, cannot penetrate walls. This advantageous characteristic prevents interference between one system and another; a short-range communication system in one room cannot be affected by another system in the next room.

When we use our infrared remote control, we do not interfere with the use of the remote by our neighbors. However, this same characteristic makes infrared signals useless for long-range communication. In addition, we cannot use infrared waves outside a building because the sun's rays contain infrared waves that can interfere with the communication.

Applications: The infrared band, almost 400 THz, has an excellent potential for data transmission. Such a wide bandwidth can be used to transmit digital data with a very high data rate.

SWITCHING:

We have switching at the physical layer, at the data-link layer, at the network layer, and even logically at the application layer (message switching).

INTRODUCTION:

A network is a set of connected devices. Whenever we have multiple devices, we have the problem of how to connect them to make one-to-one communication possible. One solution is to make a point-to-point connection between each pair of devices (a mesh topology) or between a central device and every other device (a star topology). ***These methods, however, are impractical and wasteful when applied to very large networks.***

The number and length of the links require too much infrastructure to be cost-efficient, and the majority of those links would be idle most of the time. Other topologies employing multipoint connections, such as a bus, are ruled out because the distances between devices and the total number of devices increase beyond the capacities of the media and equipment.

A better solution is **switching.** A switched network consists of a series of interlinked nodes, called ***switches*.** Switches are devices capable of creating temporary connections between two or more devices linked to the switch. In a switched network, some of these nodes are connected to the end systems (computers or telephones, for example). Others are used only for routing. Figure 1.50 shows a switched network.

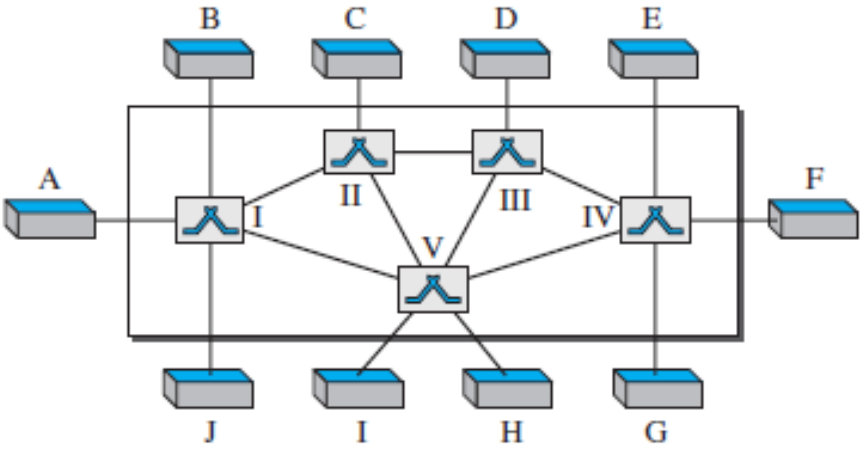


FIGURE 1.50: SWITCHED NETWORK

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.

THREE METHODS OF SWITCHING:

Traditionally, three methods of switching have been discussed: ***circuit switching***, ***packet switching***, and message switching. The first two are commonly used today. The third has been phased out in general communications but still has networking applications. Packet switching can further be divided into two subcategories—virtual circuit approach and datagram approach—as shown in Figure 1.51.

Note: ***we discuss only circuit switching and packet switching; message switching is more conceptual than practical.***

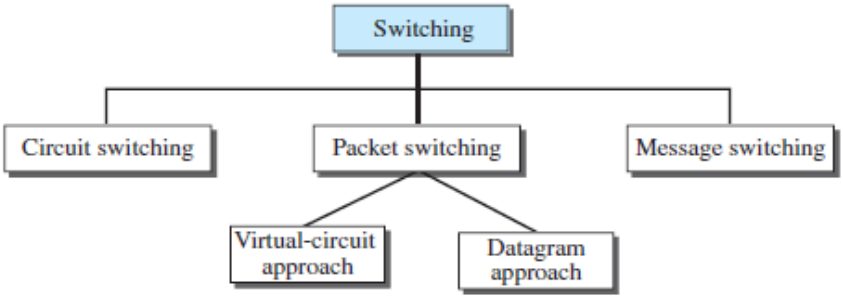


FIGURE 1.51: TAXONOMY OF SWITCHED NETWORKS

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

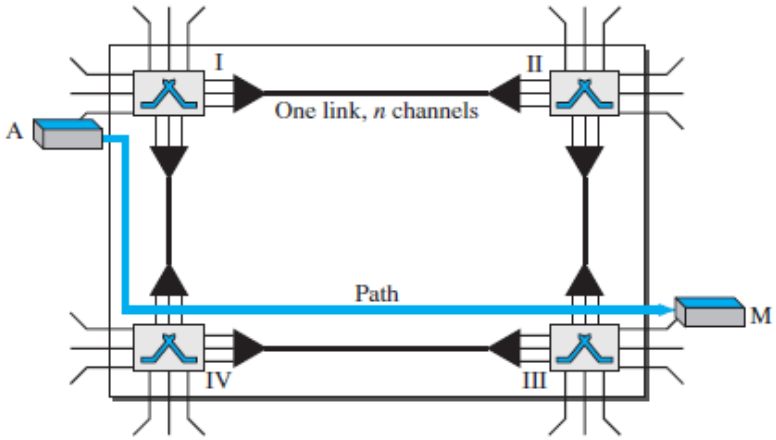


FIGURE 1.52: A TRIVIAL CIRCUIT-SWITCHED NETWORK

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

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

We need to emphasize several points here:

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

THREE PHASES:

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

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

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

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

Efficiency:

It can be argued that circuit-switched networks are not as efficient as the other two types of networks because resources are allocated during the entire duration of the connection. These resources are unavailable to other connections. In a telephone network, people normally terminate the communication when they have finished their conversation.

However, in computer networks, a computer can be connected to another computer even if there is no activity for a long time. In this case, allowing resources to be dedicated means that other connections are deprived.

Delay:

Although a circuit-switched network normally has low efficiency, the delay in this type of network is minimal. During data transfer the data are not delayed at each switch; the resources are allocated for the duration of the connection. Figure 1.53 shows the idea of delay in a circuit-switched network when only two switches are involved.

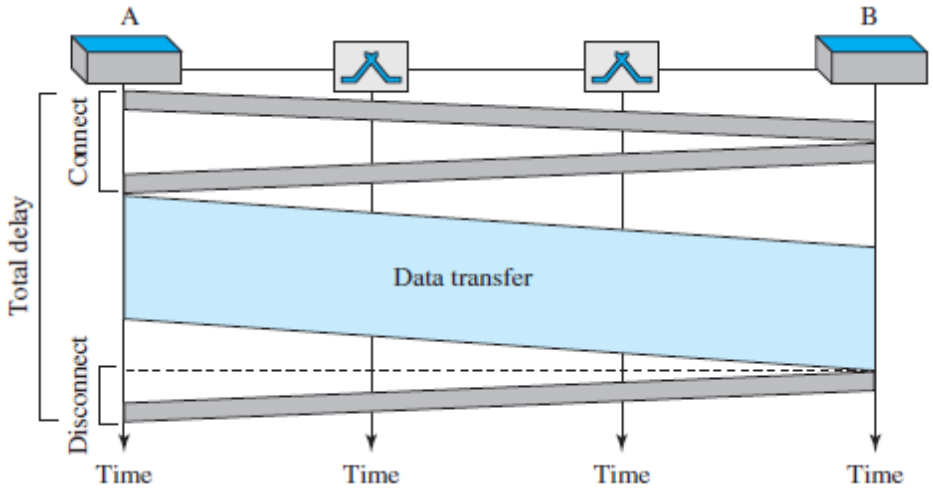


FIGURE 1.53: DELAY IN A CIRCUIT-SWITCHED NETWORK

As Figure 1.53 shows, there is no waiting time at each switch. The total delay is due to the time needed to create the connection, transfer data, and disconnect the circuit. The delay caused by the setup is the sum of four parts: the propagation time of the source computer request (slope of the first gray box), the request signal transfer time (height of the first gray box), the propagation time of the acknowledgment from the destination computer (slope of the second gray box), and the signal transfer time of the acknowledgment (height of the second gray box).

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

PACKET SWITCHING:

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

In packet switching, there is no resource allocation for a packet. This means that there is no reserved bandwidth on the links, and there is no scheduled processing time for each packet. Resources are allocated on demand. The allocation is done on a first come, first-served basis.

When a switch receives a packet, no matter what the source or destination is, the packet must wait if there are other packets being processed. As with other systems in our daily life, this lack of reservation may create delay. For example, if we do not have a reservation at a restaurant, we might have to wait.

In a packet-switched network, there is no resource reservation; resources are allocated on demand.

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

DATAGRAM NETWORKS:

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

Figure 1.54 shows how the datagram approach is used to deliver four packets from station A to station X. The switches in a datagram network are traditionally referred to as routers. That is why we use a different symbol for the switches in the figure.

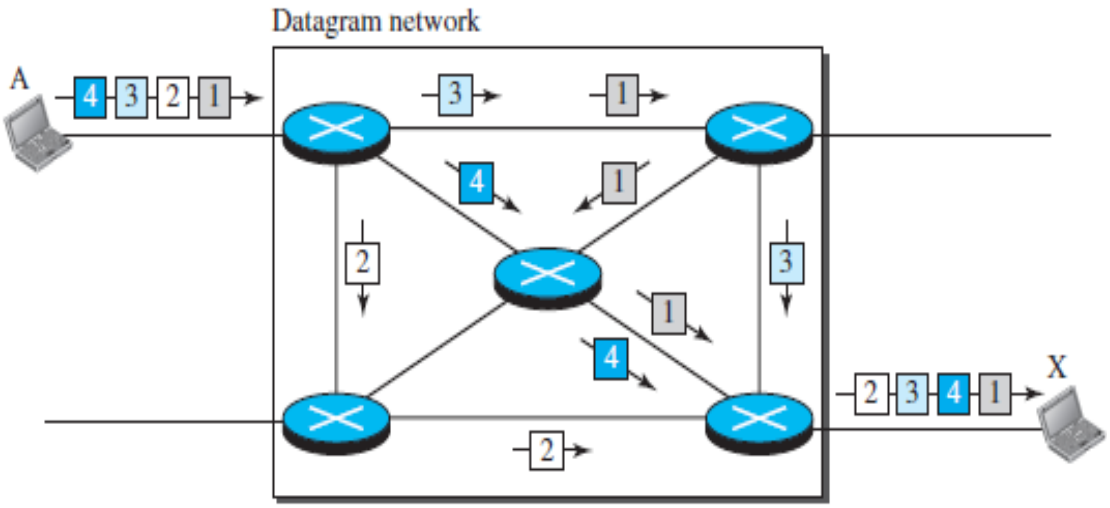


FIGURE 1.54: A DATAGRAM NETWORK WITH FOUR SWITCHES (ROUTERS)

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

This approach can cause the datagrams of a transmission to arrive at their destination out of order with different delays between the packets.

Packets may also be lost or dropped because of a lack of resources. In most protocols, it is the responsibility of an upper-layer protocol to reorder the datagrams or ask for lost datagrams before passing them on to the application.

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

ROUTING TABLE:

If there are no setup or teardown phases, how are the packets routed to their destinations in a datagram network? In this type of network, each switch (or packet switch) has a routing table which is based on the destination address. The routing tables are dynamic and are updated periodically. The destination addresses and the corresponding forwarding output ports are recorded in the tables. Figure 1.55 shows the routing table for a switch.

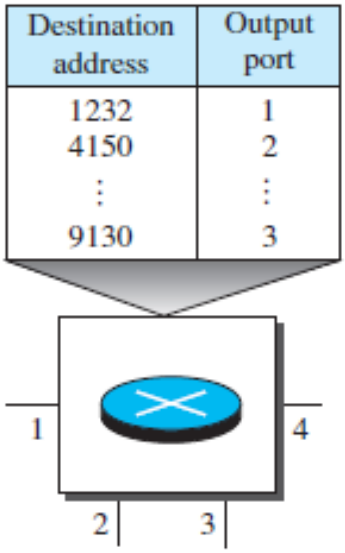


FIGURE 1.55: ROUTING TABLE IN A DATAGRAM NETWORK

A switch in a datagram network uses a routing table that is based on the destination address.

Destination Address: Every packet in a datagram network carries a header that contains, among other information, the destination address of the packet. When the switch receives the packet, this destination address is examined; the routing table is consulted to find the corresponding port through which the packet should be forwarded. This address, unlike the address in a virtual-circuit network, remains the same during the entire journey of the packet.

The destination address in the header of a packet in a datagram network remains the same during the entire journey of the packet.

Efficiency:

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

Delay:

There may be greater delay in a datagram network than in a virtual-circuit network. Although there are no setup and teardown phases, each packet may experience a wait at a switch before it is forwarded. Figure 1.56 gives an example of delay in a datagram network for one packet.

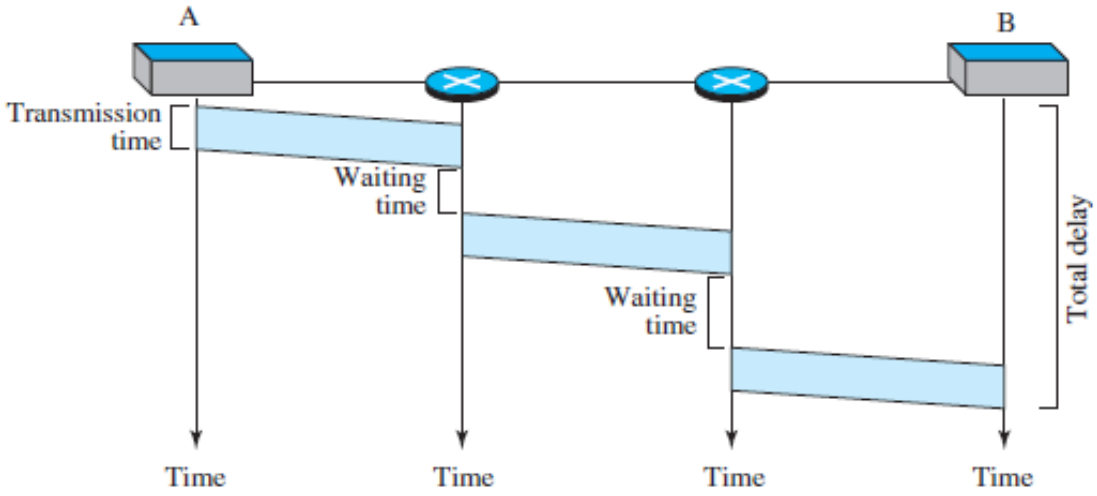


FIGURE 1.56: DELAY IN A DATAGRAM NETWORK

The packet travels through two switches. There are three transmission times ($3T$), three propagation delays (slopes 3τ of the lines), and two waiting times ($w1$ + $w2$). We ignore the processing time in each switch. The total delay is

$$\textbf{Total delay} = 3T + 3\tau + w_1 + w_2$$

VIRTUAL-CIRCUIT NETWORKS:

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

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

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

As in a datagram network, data are packetized and each packet carries an address in the header. However, the address in the header has local jurisdiction (it

defines what the next switch should be and the channel on which the packet is being carried), not end-to-end jurisdiction.

As in a circuit-switched network, all packets follow the same path established during the connection. A virtual-circuit network is normally implemented in the data-link layer, while a circuit-switched network is implemented in the physical layer and a datagram network in the network layer. But this may change in the future.

Figure 1.57 is an example of a virtual-circuit network. The network has switches that allow traffic from sources to destinations. A source or destination can be a computer, packet switch, bridge, or any other device that connects other networks.

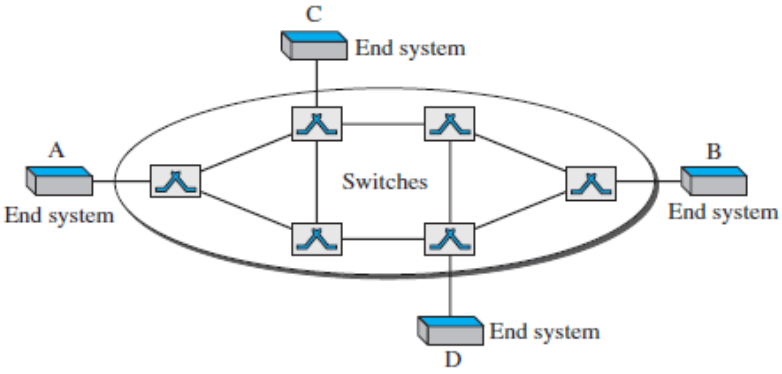


FIGURE 1.57: VIRTUAL-CIRCUIT NETWORK

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

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

However, we will see that 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.

Figure 1.58 shows how the VCI in a data frame changes from one switch to another. Note that a VCI does not need to be a large number since each switch can use its own unique set of VCIs.

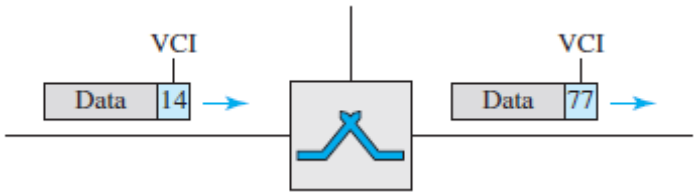


FIGURE 1.58: VIRTUAL-CIRCUIT IDENTIFIER

Three Phases:

As in a circuit-switched network, a source and destination need to go through three phases in a virtual-circuit network: setup, data transfer, and teardown. In the setup phase, the source and destination use their global addresses to help switches make table entries for the connection. In the teardown phase, the source and destination inform the switches to delete the corresponding entry. Data transfer occurs between these two phases.

Data-Transfer Phase: To transfer a frame from a source to its destination, all switches need to have a table entry for this virtual circuit. The table, in its simplest form, has four columns. This means that the switch holds four pieces of information for each virtual circuit that is already set up. Figure 1.59 shows such a switch and its corresponding table.

Figure 1.59 shows a frame arriving at port 1 with a VCI of 14. When the frame arrives, the switch looks in its table to find port 1 and a VCI of 14. When it is found, the switch knows to change the VCI to 22 and send out the frame from port 3.

Setup Phase:

In the setup phase, a switch creates an entry for a virtual circuit. For example, suppose source A needs to create a virtual circuit to B. Two steps are required: the ***setup request*** and the ***acknowledgment***.

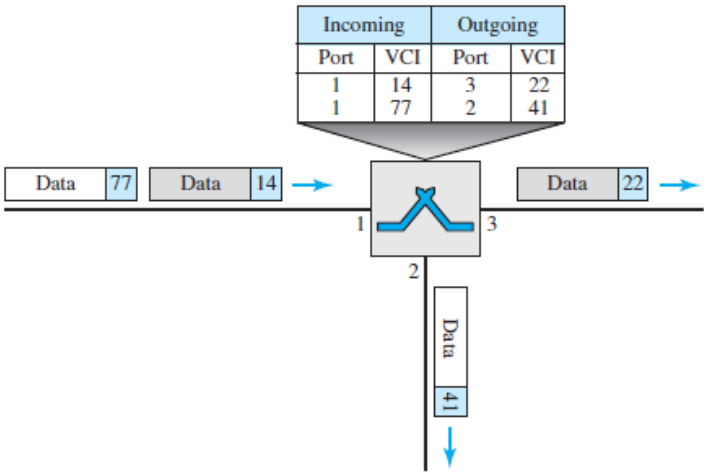


FIGURE 1.59: SWITCH AND TABLES IN A VIRTUAL-CIRCUIT NETWORK

Setup Request: A setup request frame is sent from the source to the destination. Figure 1.60 shows the process.

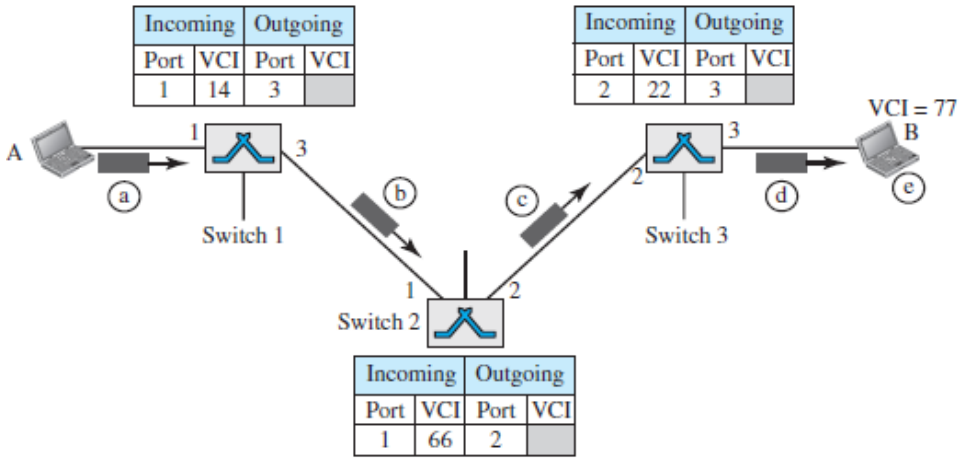


FIGURE 1.60: SETUP REQUEST IN A VIRTUAL-CIRCUIT NETWORK

Acknowledgment:

A special frame, called the *acknowledgment frame,* completes the entries in the switching tables. Figure 1.61 shows the process.

Teardown Phase:

In this phase, source A, after sending all frames to B, sends a special frame called a *teardown request*. Destination B responds with a teardown confirmation frame. All switches delete the corresponding entry from their tables.

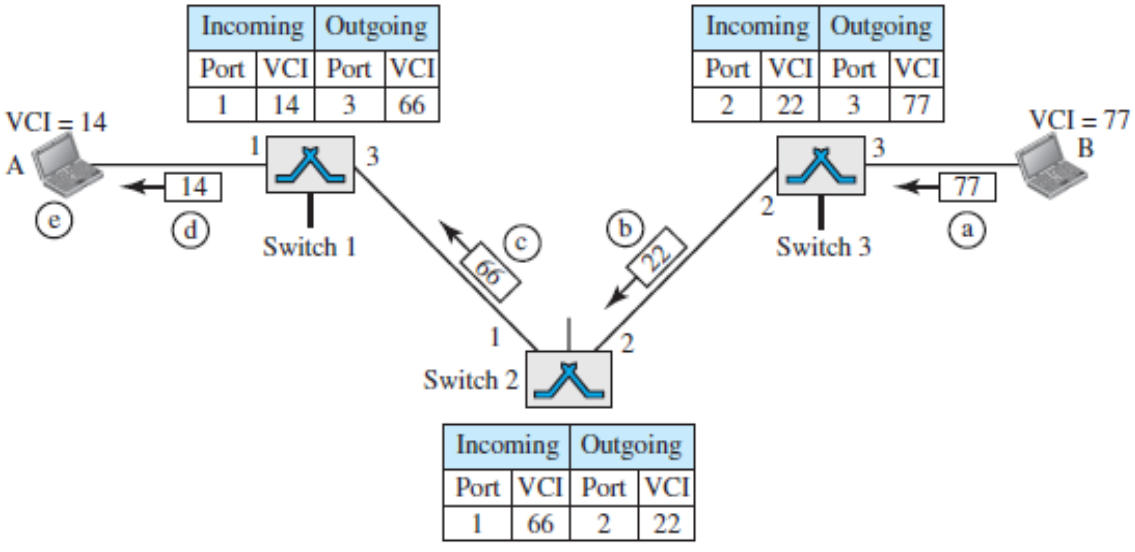


FIGURE 1.61: SETUP ACKNOWLEDGMENT IN A VIRTUAL-CIRCUIT NETWORK

Efficiency:

As we said before, resource reservation in a virtual-circuit network can be made during the setup or can be on demand during the data-transfer phase.

In the first case, the delay for each packet is the same; in the second case, each packet may encounter different delays.

There is one big advantage in a virtual-circuit network even if resource allocation is on demand. The source can check the availability of the resources, without actually reserving it. Consider a family that wants to dine at a restaurant.

Although the restaurant may not accept reservations (allocation of the tables is on demand), the family can call and find out the waiting time. This can save the family time and effort.

Delay in Virtual-Circuit Networks:

In a virtual-circuit network, there is a one-time delay for setup and a one-time delay for teardown. If resources are allocated during the setup phase, there is no wait time for individual packets. Figure 1.62 shows the delay for a packet traveling through two switches in a virtual-circuit network.

The packet is traveling through two switches (routers). There are three transmission times (3*T*), three propagation times (3τ), data transfer depicted by the sloping lines, a setup delay (which includes transmission and propagation in two directions), and a teardown delay (which includes transmission and propagation in one direction).

We ignore the processing time in each switch. The total delay time is **Total delay + 3*T* + 3**τ + **setup delay + teardown delay**

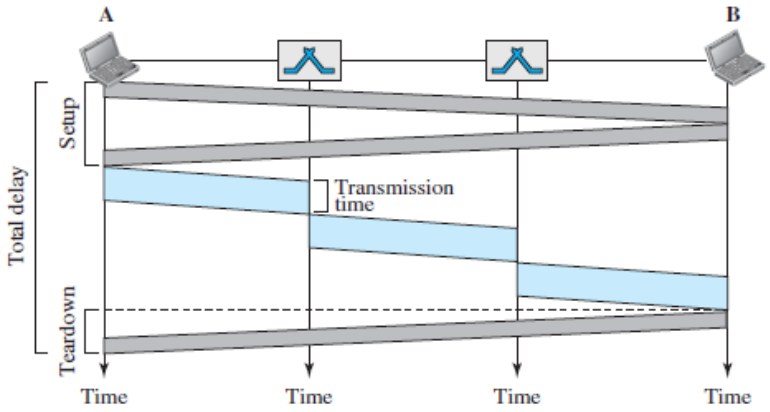


FIGURE 1.62: DELAY IN A VIRTUAL-CIRCUIT NETWORK

Circuit-Switched Technology in WANs: virtual-circuit networks are used in switched WANs such as ATM networks. The data-link layer of these technologies is well suited to the virtual circuit technology.

Switching at the data-link layer in a switched WAN is normally implemented by using virtual-circuit techniques.

***** **USEFUL QUESTIONS with ANSWERS** *****

Q: Define the term protocol.

Ans: Set of rules established for users to exchange information.

Q: Define the term topology.

Ans: Architecture of a network.

Q: Define the term deterministic.

Ans: Access to the network is provided at fixed time intervals

Q: What is the difference between a hub and a switch? **Ans:** *Hub* – Broadcasts data it receives to all devices connected to its ports. *Switch* – Establishes a direct connection from the sender to the destination without passing the data traffic to other networking devices.

Q: Cite the three advantages of a wired network. **Ans:** 1) Faster network data transfer speeds (within the LAN), 2) Relatively inexpensive to setup & 3) the network is not susceptible to outside interference.

Q: Cite three advantages of a wireless network. **Ans:** 1) User mobility 2) Simple installations & 3) No cables

Q: What does it mean for a wireless networking device to be Wi-Fi compliant? **Ans:** That the device has been tested by the Wi-Fi Alliance (Wireless Fidelity) and is certified for compliance with 802.11x wireless standards.

Q: List five steps that can be used to protect the home network. **Ans:** 1) Change the default factory passwords. 2) Change the default SSID. "Service Set Identifier" 3) turn on encryption. 4) Turn off the SSID broadcast. & 5) Enable MAC address filtering.

Q: What is Stateful Packet Inspection "SPI"? **Ans:** A type of firewall protection that inspects incoming data packets to make sure they correspond to an outgoing request.

Above questions link: http://pcr-tech-sc.2302233.n4.nabble.com/Chapter-1-Introduction-to-Computer-Networks-Questions-Answers-td3891566.html

UNIT –I
Objective questions

1. The _____ is the physical path over which a message travels.
A) Protocol **B) Medium**
C) Signal D) All the above

2. The information to be communicated in a data communications system is the _____.
A) Medium B) Protocol
C) Message D) Transmission

3. Frequency of failure and network recovery time after a failure is measures of the _____ of network.
A) Performance **B) Reliability**
C) Security D) Feasibility

4. TDM Stands for _____.
A) Time discrete measures **B) Time Division Multiplexing**

C) Time division measures D) All the above

5. Which topology requires a central controller or hub?
A) Mesh **B) Star**
C) Bus D) Ring

6. Which topology requires a multipoint connection?
A) Mesh B) Star
C) Bus D) Ring

7. Communication between a computer and a keyboard involves _____ transmission.
A) Simplex B) half-duplex
C) full-duplex D) automatic

8. A television broadcast is an example of _____ transmission.
A) Simplex B) half-duplex
C) full-duplex D) automatic

9. A _____ connection provides a dedicated link between two devices.
A) Point-to-point B) multipoint
C) Primary D) secondary

10. In a _____ connection, more than two devices can share a single link.
A) Point-to-point **B) multipoint**
C) Primary D) secondary

11. In _____ transmission, the channel capacity is shared by both communicating devices at all times.

A) Simplex B) half-duplex
C) full-duplex D) half-simplex

12. In the original ARPANET, _____ were directly connected together.
A) IMPs B) host computers
C) Networks D) routers

13. This was the first network.
A) CSNET B) NSFNET
C) ANSNET **D) ARPANET**

14. Which organization has authority over interstate and international commerce in the
Communications field?
A) ITU-T B) IEEE
C) FCC D) ISOC

15. _____ is special-interest groups that quickly test, evaluate, and standardize new technologies.
A) **Forums** B) Regulatory agencies
B) C) Standards organizations D) All of the above

16. Which agency developed standards for physical connection interfaces and electronic signalling a specification?
A) EIA B) ITU-T
C) ANSI D) ISO

17. _____ is the protocol suite for the current Internet.
A) TCP/IP B) NCP
C) UNIX D) ACM

18. _____ refers to the structure or format of the data, meaning the order in which they are
Presented.
A) Semantics **B) Syntax**
C) Timing D) All of the above

19. _____ defines how a particular pattern to be interpreted, and what action is to be taken
based on that interpretation.
A) **Semantics** B) Syntax
C) Timing D) None of the above

20. _____ refers to two characteristics: when data should be sent and how fast it can be sent.
A) Semantics B) Syntax
C) Timing D) none of the above

21. The Physical layer of GSM handles of _____.
A) Radio-specific B) Satellite
C) FDM D) DSS

22. Fiber optic cable consists of _____
A) Plastic core B) plastic cladding
C) Glass cladding **D) ALL**

23) Telephone system consists of _____
A) Local loops B) Trunks &Multiplexing
C) Switching **D) ALL**

24) In the OSI model, as a data packet moves from the lower to the upper layers, headers are_____
A) Added **B) removed**
C) Rearranged D) modified

25. Which transmission media has the highest transmission speed in a network?
A) Coaxial cable B) twisted pair cable
C) Optical fiber D) electrical cable

26. Physical layer provides
A) mechanical specifications of electrical connectors and cables B) electrical specification of transmission line signal level
C) Specification for IR over optical fiber **D) all of the mentioned**

27. A single channel is shared by multiple signals by
A) analogy modulation B) digital modulation
C) Multiplexing D) none of the mentioned

28. Wireless transmission can be done via
A) radio waves B) microwaves
C) Infrared **D) all of the mentioned**

29. The physical layer translates logical communication requests from the _____ into hardware specific operations.
A) Data link layer B) network layer
C) Transport layer D) application layer

30. In asynchronous serial communication the physical layer provides
A) start and stop signalling B) flow control
C) Both (a) and (b) D) none of the mentioned

UNIT –I

DESCRIPTIVE QUESTIONS

1. List out the advantages and drawbacks of bus topology.

2 Explain the Difference between LAN, MAN, WAN.

3. Explain the OSI Reference model

4. Explain about the TCP/IP model

5. Explain and draw the ARPANET

6 Write any four reasons for using Layer Protocol.

7. List out the advantages and disadvantages of OSI Reference model compare with TCP/IP model.

8. Difference between the datagram packet switching and virtual circuit switching.

9. Write the advantages of optical fiber over twisted pair and coaxial cable.

10. Explain about Public switching telephone network.

UNIT 2
INTRODUCTION TO DATALINK LAYER

INTRODUCTION:

The Internet is a combination of networks glued together by connecting devices (routers or switches). If a packet is to travel from a host to another host, it needs to pass through these networks. Figure 2.1 shows the same scenario.

Communication at the data-link layer is made up of five separate logical connections between the data-link layers in the path.

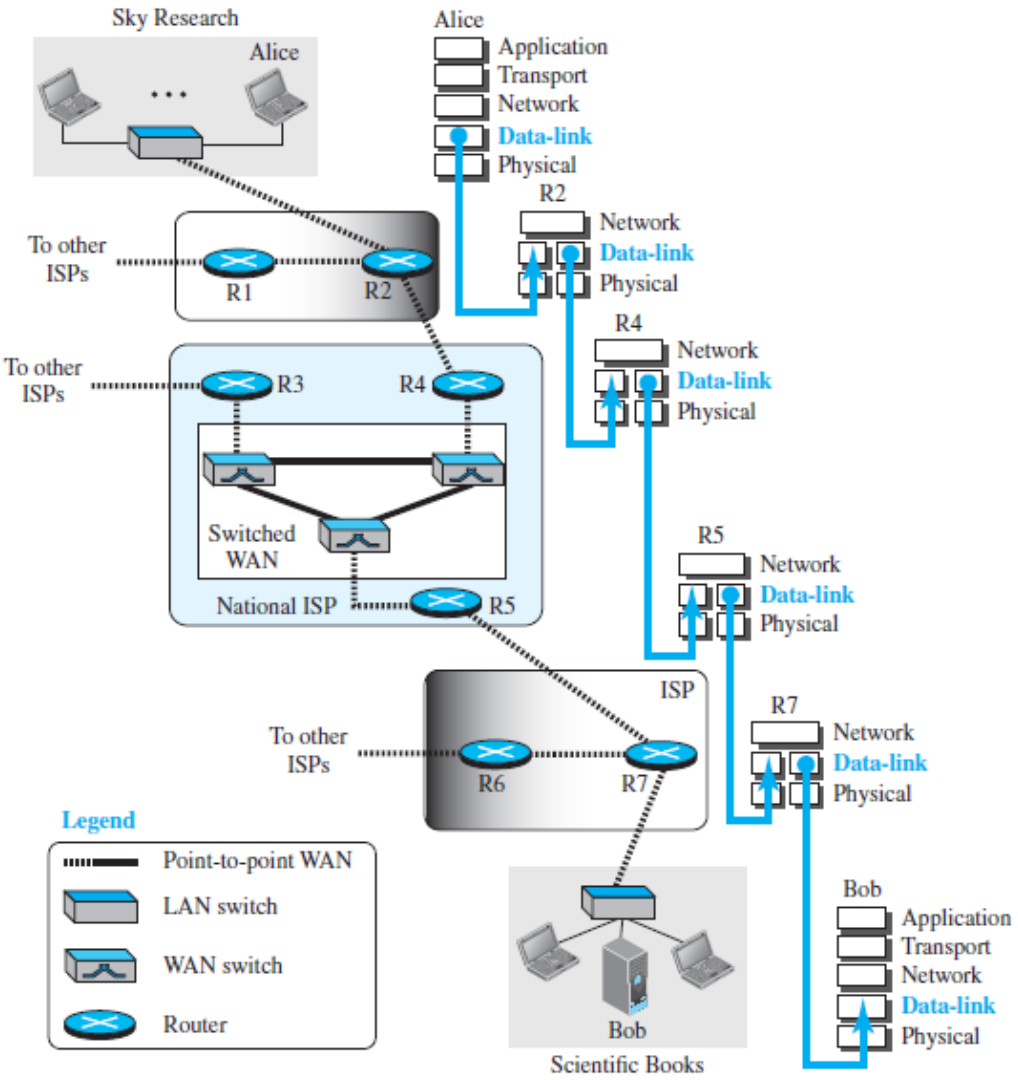


FIGURE 2.1: COMMUNICATION AT THE DATA-LINK LAYER

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

The reason is that Alice's and Bob's computers are each connected to a single network, but each router takes input from one network and sends output to another network. Note that although switches are also involved in the data-link-layer communication, for simplicity we have not shown them in the figure.

NODES AND LINKS:

Communication at the data-link layer is node-to-node. A data unit from one point in the Internet needs to pass through many networks (LANs and WANs) to reach another point. Theses LANs and WANs are connected by routers. It is customary to refer to the two end hosts and the routers as ***nodes*** and the networks in between as ***links*.** Figure 2.2 is a simple representation of links and nodes when the path of the data unit is only six nodes.

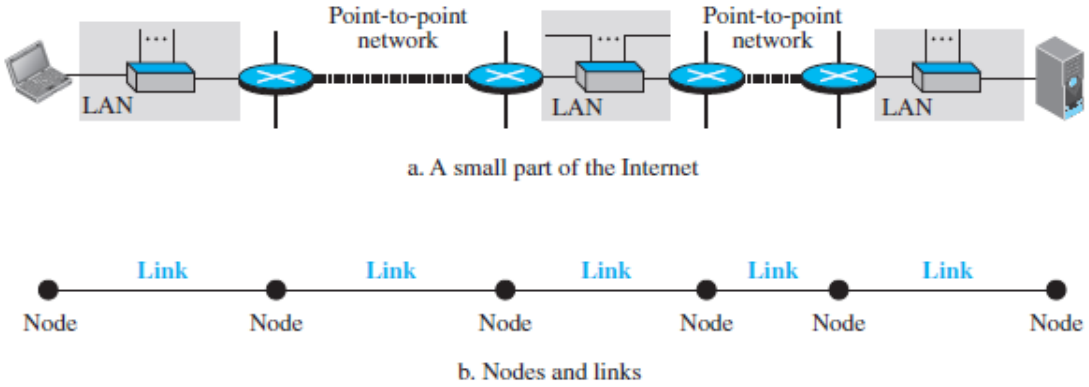


FIGURE 2.2: NODES AND LINKS

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

SERVICES:

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

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

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

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

Figure 2.3 shows the encapsulation and decapsulation at the data-link layer. For simplicity, we have assumed that we have only one router between the source and destination. The datagram received by the data-link layer of the source host is encapsulated in a frame. The frame is logically transported from the source host to the router.

The frame is decapsulated at the data-link layer of the router and encapsulated at another frame. The new frame is logically transported from the router to the destination host. Note that, although we have shown only two data-link layers at the router, the router actually has three data-link layers because it is connected to three physical links.

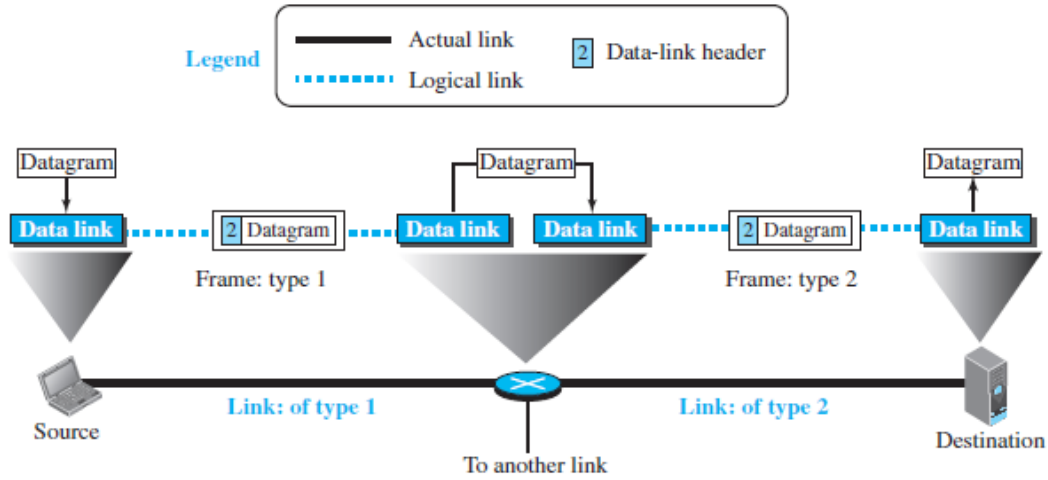


FIGURE 2.3: A COMMUNICATION WITH ONLY THREE NODES

With the contents of the above figure in mind, we can list the services provided by a data-link layer as shown below.

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

The node also needs to decapsulate the datagram from the frame received on the logical channel. Although we have shown only a header for a frame, we will see in future chapters that a frame may have both a header and a trailer. Different data-link layers have different formats for framing.

FLOW CONTROL: Whenever we have a producer and a consumer, we need to think about flow control. If the producer produces items that cannot be consumed, accumulation of items occurs. The sending data-link layer at the end of

a link is a producer of frames; the receiving data-link layer at the other end of a link is a consumer.

If the rate of produced frames is higher than the rate of consumed frames, frames at the receiving end need to be buffered while waiting to be consumed (processed). Definitely, we cannot have an unlimited buffer size at the receiving side. We have two choices. The first choice is to let the receiving data-link layer drop the frames if its buffer is full.

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

ERROR CONTROL: At the sending node, a frame in a data-link layer needs to be changed to bits, transformed to electromagnetic signals, and transmitted through the transmission media. At the receiving node, electromagnetic signals are received, transformed to bits, and put together to create a frame.

Since electromagnetic signals are susceptible to error, a frame is susceptible to error. The error needs first to be detected. After detection, it needs to be either corrected at the receiver node or discarded and retransmitted by the sending node.

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

TWO CATEGORIES OF LINKS: Although two nodes are physically connected by a transmission medium such as cable or air, we need to remember that the data-link layer controls how the medium is used. We can have a data-link layer that uses the whole capacity of the medium; we can also have a data-link layer that uses only part of the capacity of the link.

In other words, we can have a *point-to-point link* or a *broadcast link*. 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.

Two Sub layers:

To better understand the functionality of and the services provided by the link layer, we can divide the data-link layer into two sub layers: **data link control (DLC)** and **media access control (MAC).**

The data link control sub layer deals with all issues common to both point-to-point and broadcast links; the media access control sub layer deals only with issues specific to broadcast links.

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 2.4 demonstrates the concept in a small internet.

In the internet in Figure 2.4, 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.

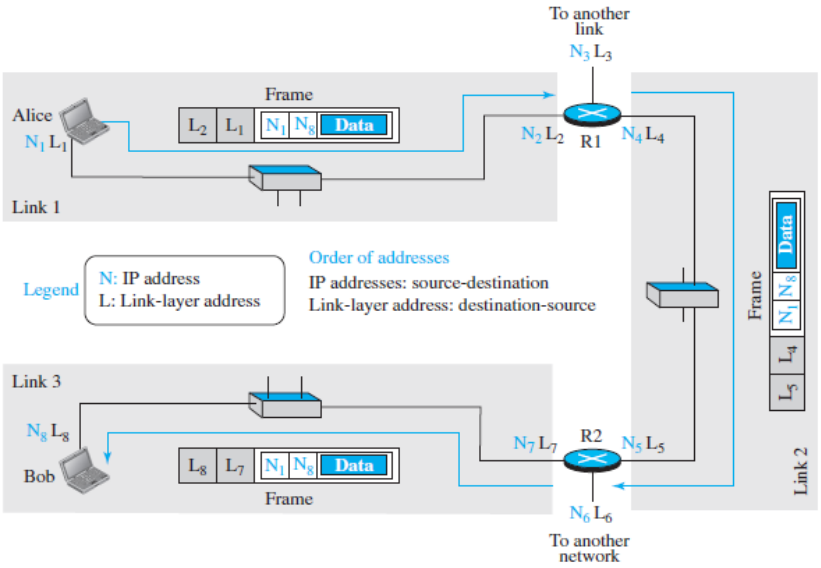


FIGURE 2.4: IP ADDRESSES AND LINK-LAYER ADDRESSES IN A SMALL INTERNET

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:

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

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

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

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.

Address Resolution Protocol (ARP):

Anytime a node has an IP datagram to send to another node in a link, it has the IP address of the receiving node. The source host knows the IP address of the default router.

Each router except the last one in the path gets the IP address of the next router by using its forwarding table. The last router knows the IP address of the destination host. However, the IP address of the next node is not helpful in moving a frame through a link; we need the link-layer address of the next node. This is the time when the **Address Resolution Protocol (ARP)** becomes helpful.

The ARP protocol is one of the auxiliary protocols defined in the network layer, as shown in Figure 2.5. It belongs to the network layer, but we discuss it here because it maps an IP address to a logical-link address. ARP accepts an IP address from the IP protocol, maps the address to the corresponding link-layer address, and passes it to the data-link layer.

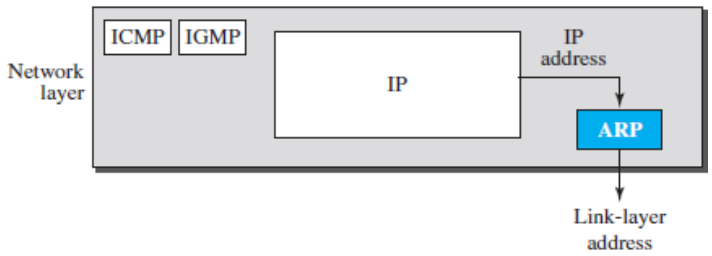


FIGURE 2.5: POSITION OF ARP IN TCP/IP PROTOCOL SUITE

Anytime a host or a router needs to find the link-layer address of another host or router in its network, it sends an ARP request packet. The packet includes the link-layer and IP addresses of the sender and the IP address of the receiver.

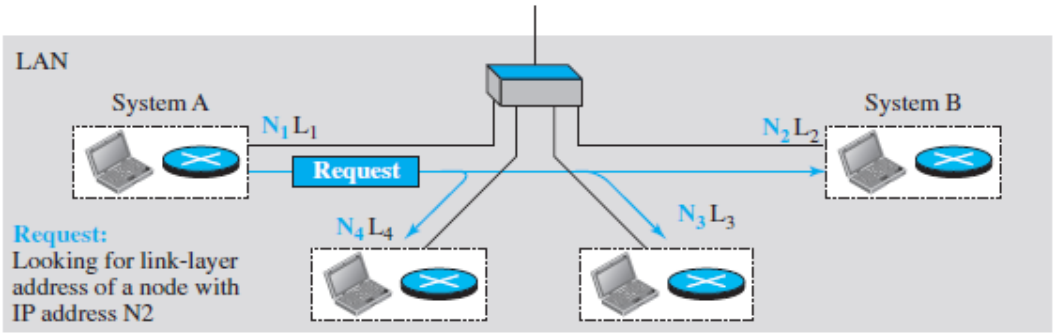
Because the sender does not know the link-layer address of the receiver, the query is broadcast over the link using the link-layer broadcast address (see Figure 2.6).

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

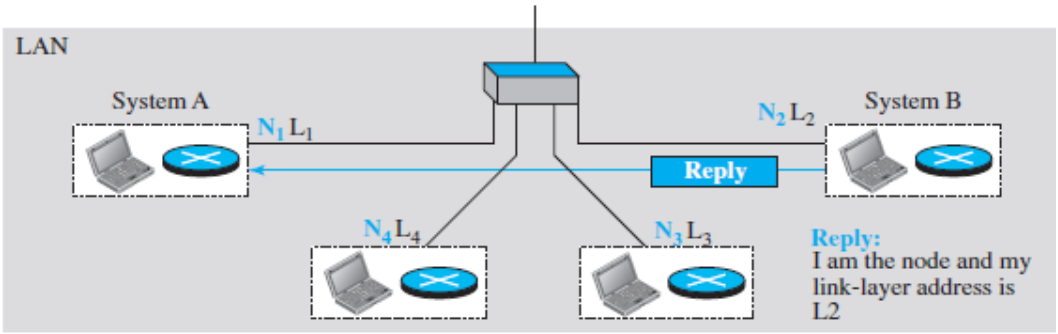
In Figure 2.6a, the system on the left (A) has a packet that needs to be delivered to another system (B) with IP address **N2**. System A needs to pass the packet to its data-link layer for the actual delivery, but it does not know the physical address of the recipient.

It uses the services of ARP by asking the ARP protocol to send a broadcast ARP request packet to ask for the physical address of a system with an IP address of **N2**. This packet is received by every system on the physical network, but only system B will answer it, as shown in Figure 2.6b.

System B sends an ARP reply packet that includes its physical address. Now system A can send all the packets it has for this destination using the physical address it received.



a. ARP request is broadcast



b. ARP reply is unicast

FIGURE 2.6: ARP OPERATION

Packet Format:

Figure 2.7 shows the format of an ARP packet. The names of the fields are self-explanatory. The *hardware type* field defines the type of the link-layer protocol; Ethernet is given the type 1.

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

The destination hardware address and destination protocol address fields define the receiver link-layer and network-layer addresses. An ARP packet is encapsulated directly into a data-link frame. The frame needs to have a field to show that the payload belongs to the ARP and not to the network-layer datagram.

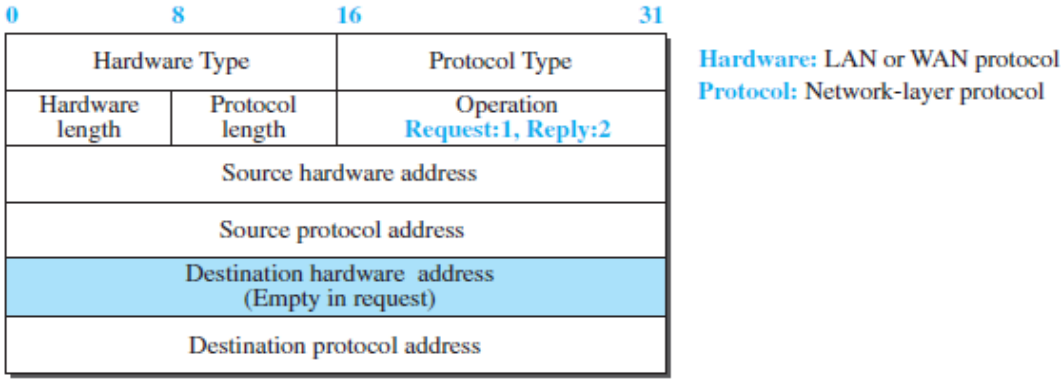


FIGURE 2.7: ARP PACKET

ERROR DETECTION AND 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.

INTRODUCTION:

Types of Errors:

Whenever bits flow from one point to another, they are subject to unpredictable changes because of **interference.** This interference can change the shape of the signal. The term ***single-bit error*** means that only 1 bit of a given data unit (such as a byte, character, or packet) is changed from 1 to 0 or from 0 to 1.

The term ***burst error*** means that 2 or more bits in the data unit have changed from 1 to 0 or from 0 to 1. Figure 2.8 shows the effect of a single-bit and a burst error on a data unit.

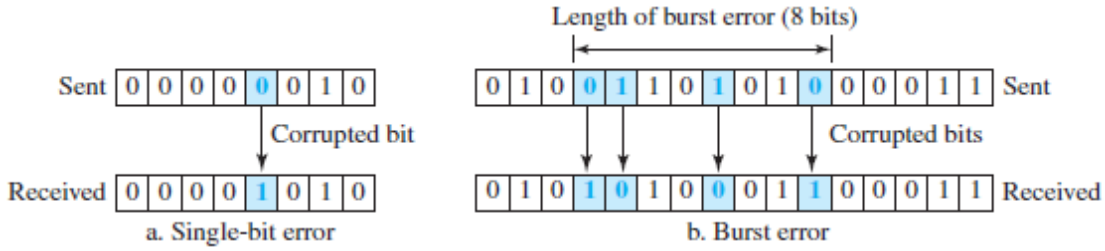


FIGURE 2.8: SINGLE-BIT AND BURST ERROR

A burst error is more likely to occur than a single-bit error because the duration of the noise signal is normally longer than the duration of 1 bit, which means that when noise affects data, it affects a set of bits.

The number of bits affected depends on the data rate and duration of noise. For example, if we are sending data at 1 kbps, a noise of 1/100 second can affect 10 bits; if we are sending data at 1 Mbps, the same noise can affect 10,000 bits.

Redundancy:

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

Detection versus Correction:

The correction of errors is more difficult than the detection. In **error detection**, we are only looking to see if any error has occurred. The answer is a simple yes or no. 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**.

***NOTE:* Convolution coding is more complex and beyond the scope of this book**

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*.**

Error Detection:

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 2.9 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.

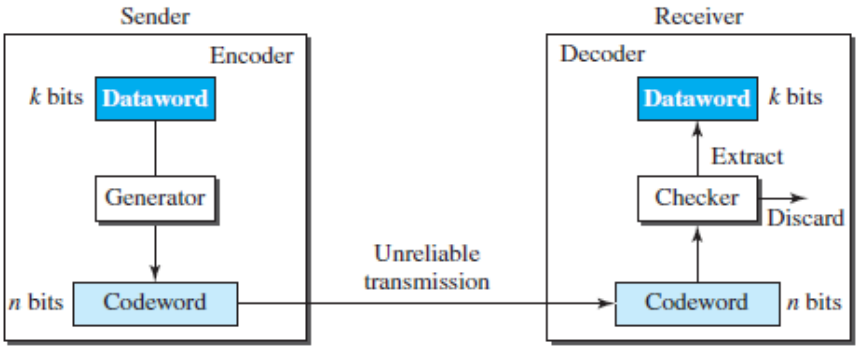


FIGURE 2.9: PROCESS OF ERROR DETECTION IN BLOCK CODING

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.

Example 2.a:

Let us assume that $k = 2$ and $n = 3$. Table 2.1 shows the list of datawords and codewords.

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

TABLE 2.1: A CODE FOR ERROR DETECTION IN EXAMPLE 2.a

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 $\boldsymbol{d}(x, y)$. We may wonder why Hamming distance is important for error detection.

The reason is that the Hamming distance between the received codeword and the sent codeword is the number of bits that are corrupted during transmission. For example, if the codeword 00000 is sent and 01101 is received, 3 bits are in error and the Hamming distance between the two is $\boldsymbol{d}(00000, 01101) = 3$.

In other words, if the Hamming distance between the sent and the received codeword is not zero, the codeword has been corrupted during transmission.

The Hamming distance can easily be found if we apply the XOR operation ($\oplus$) 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

Example 2.b:

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.

Parity-Check Code:

Perhaps 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, we discuss the even case.

The minimum Hamming distance for this category is *d*min = 2, which means that the code is a single-bit error-detecting code. Our first code (table 2.1) is a parity-check code (*k* = 2 and *n* = 3). The code in table 2.2 is also a parity-check code with *k* = 4 and *n* = 5.

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

TABLE 2.2: SIMPLE PARITY-CHECK CODE C(5, 4)

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 *a*0 to *a*6, and the bits in the second word *b*0 to *b*6, we can shift the bits by using the following:

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

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

Cyclic Redundancy Check:

We can create cyclic codes to correct errors. In this section, we simply discuss a subset of cyclic codes called the **cyclic redundancy check (CRC)**, which is used in networks such as LANs and WANs.

Table 2.3 shows an example of a CRC code. We can see both the linear and cyclic properties of this code.

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

TABLE 2.3: A CRC CODE WITH C(7, 4)

Figure 2.10 shows one possible design for the encoder and decoder.

In the encoder, the dataword has *k* bits (4 here); the codeword has *n* bits (7 here). The size of the dataword is augmented by adding *n* − *k* (3 here) 0s to the right-hand side of the word. The *n*-bit result is fed into the generator.

The generator uses a divisor of size *n* − *k* + 1 (4 here), predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder (*r*2*r*1*r*0) is appended to the dataword to create the codeword.

The decoder receives the codeword (possibly corrupted in transition). A copy of all *n* bits is fed to the checker, which is a replica of the generator.

The remainder produced by the checker is a syndrome of *n* − *k* (3 here) bits, which is fed to the decision logic analyzer. The analyzer has a simple function. If the syndrome bits are all 0s, the 4 leftmost bits of the codeword are accepted as the dataword (interpreted as no error); otherwise, the 4 bits are discarded (error).

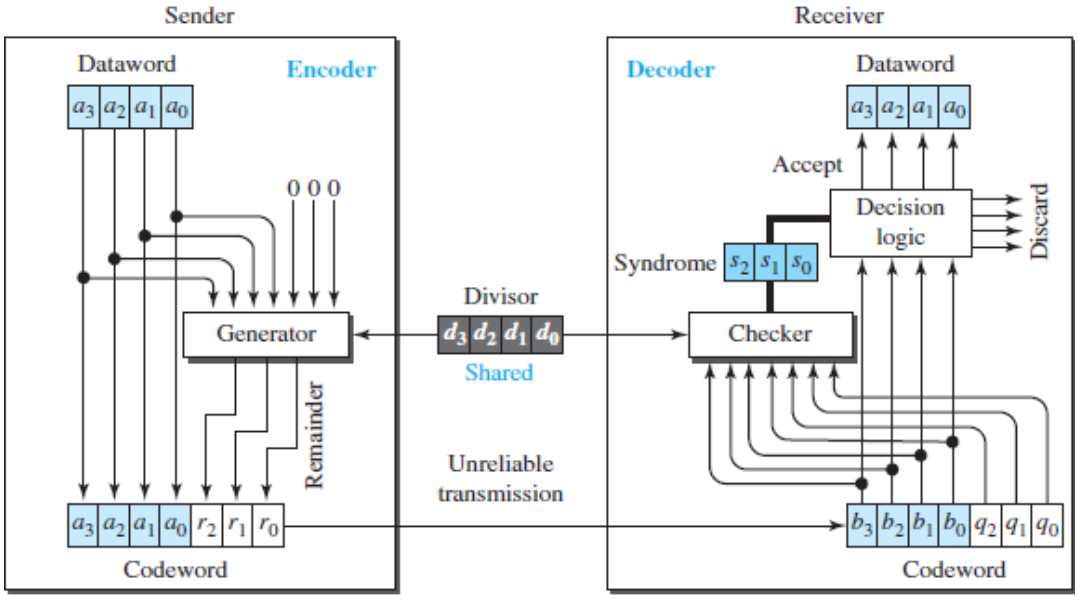


FIGURE 2.10: CRC ENCODER AND DECODER

Encoder:

Let us take a closer look at the encoder. The encoder takes a dataword and augments it with *n* − *k* number of 0s. It then divides the augmented dataword by the divisor, as shown in Figure 2.11.

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 2.12 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).

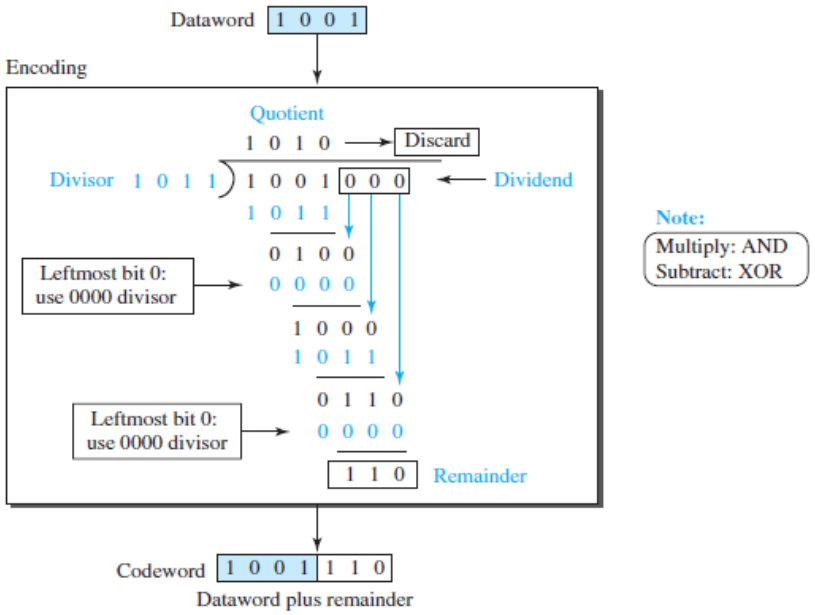


FIGURE 2.11: DIVISION IN CRC ENCODER

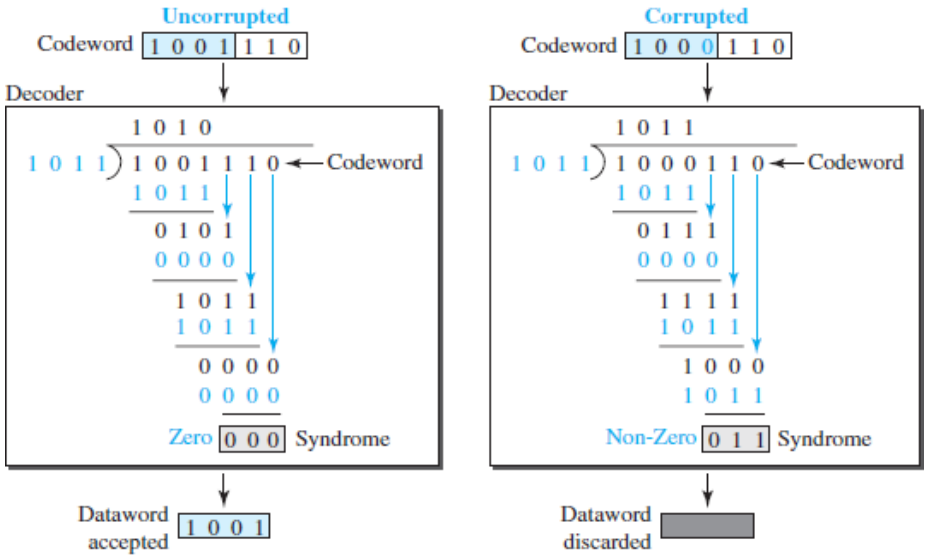


FIGURE 2.12: DIVISION IN THE CRC DECODER FOR TWO CASES

ADVANTAGES OF CYCLIC CODES:

We have seen that cyclic codes have a very good performance in detecting single-bit errors, double errors, an odd number of errors, and burst errors. They can easily be implemented in hardware and software. They are especially fast when

implemented in hardware. This has made cyclic codes a good candidate for many networks.

CHECKSUM:

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

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 2.13). Note that in the real implementation, the checksum unit is not necessarily added at the end of the message; it can be inserted in the middle of the message.

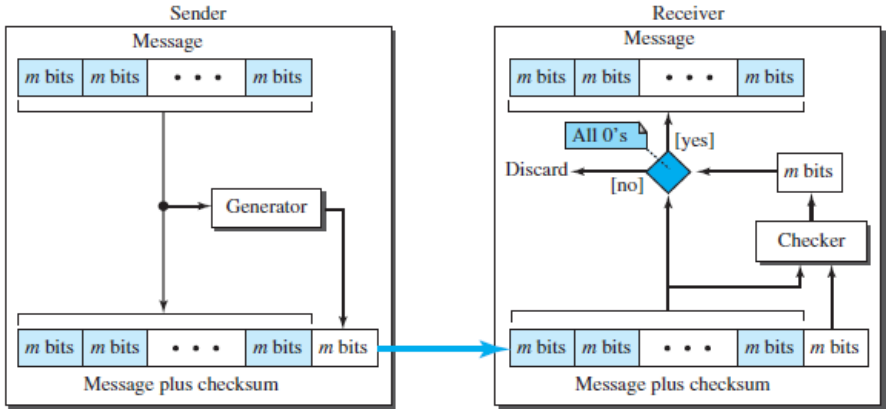


FIGURE 2.13: CHECKSUM

Concept

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

Example 2.c:

Suppose the message is a list of five 4-bit numbers that we want to send to a destination. In addition to sending these numbers, we send the sum of the numbers. For example, if the set of numbers is (7, 11, 12, 0, 6), we send (7, 11, 12, 0, 6, **36**), where 36 is the sum of the original numbers.

The receiver adds the five numbers and compares the result with the sum. If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum. Otherwise, there is an error somewhere and the message is not accepted.

One's Complement Addition:

The previous example has one major drawback. Each number can be written as a 4-bit word (each is less than 15) except for the sum. One solution is to use **one's complement** arithmetic.

In this arithmetic, we can represent unsigned numbers between 0 and $2^m - 1$ using only *m* bits. If the number has more than *m* bits, the extra leftmost bits need to be added to the *m* rightmost bits (wrapping).

Example 2.d:

In the previous example, the decimal number 36 in binary is (100100)2. To change it to a 4-bit number we add the extra leftmost bit to the right four bits as shown below.

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

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

Algorithm

We can use the flow diagram of Figure 2.14 to show the algorithm for calculation of the checksum. A program in any language can easily be written based on the algorithm. Note that the first loop just calculates the sum of the data units in two's complement; the second loop wraps the extra bits created from the two's complement calculation to simulate the calculations in one's complement. This is needed because almost all computers today do calculation in two's complement.

FORWARD ERROR CORRECTION:

Retransmission of corrupted and lost packets is not useful for real-time multimedia transmission because it creates an unacceptable delay in reproducing: we need to wait until the lost or corrupted packet is resent.

We need to correct the error or reproduce the packet immediately. Several schemes have been designed and used in this case that is collectively referred to as **forward error correction** (**FEC**) techniques.

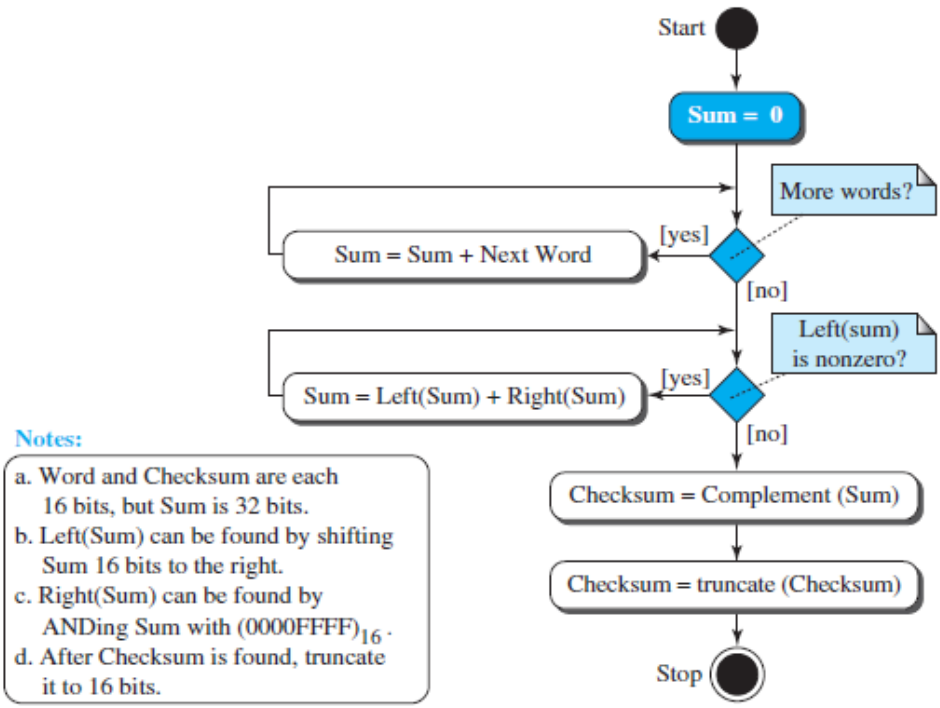


FIGURE 2.14: ALGORITHM TO CALCULATE A TRADITIONAL CHECKSUM

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 = 2*t* + 1. In other words, if we want to correct 10 bits in a packet, we need to make the minimum hamming distance 21 bits, which means a lot of redundant bits, need to be sent with the data.

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. We give some examples of how to calculate the required bits in the practice set.

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

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

In other words, if we apply the exclusive OR operation on *N* data items (P1 to P*N*), we can recreate any of the data items by exclusive-ORing all of the items, replacing the one to be created by the result of the previous operation (R).

This means that we can divide a packet into *N* chunks, create the exclusive OR of all the chunks and send *N* + 1 chunks. If any chunk is lost or corrupted, it can be created at the receiver site. Now the question is what the value of N should be. If *N* = 4, it means that we need to send 25 percent extra data and be able to correct the data if only one out of four chunks is lost.

CHUNK INTERLEAVING:

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

Figure 2.15 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.

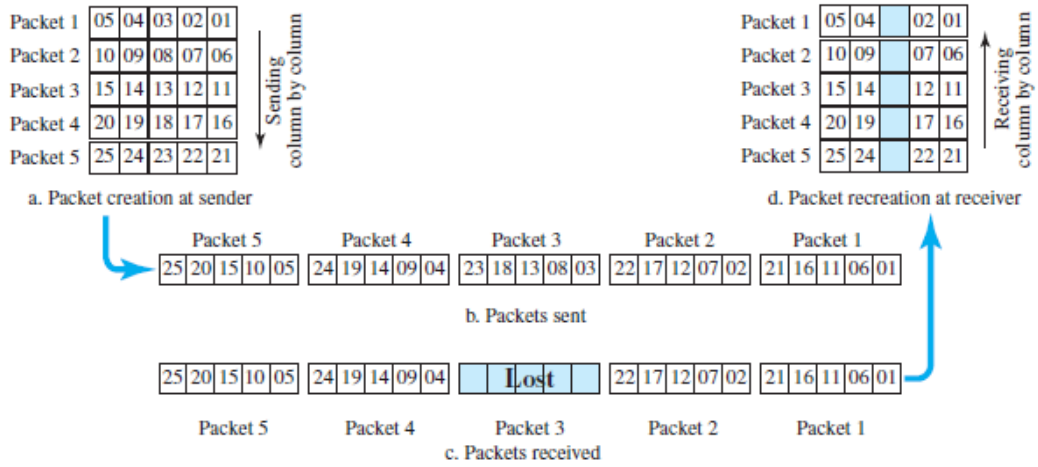


FIGURE 2.15: INTERLEAVING

COMBINING HAMMING DISTANCE AND INTERLEAVING:

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

COMPOUNDING HIGH- AND LOW-RESOLUTION PACKETS:

Still another solution is to create a duplicate of each packet with a low-resolution redundancy and combine the redundant version with the next packet. For example, we can create four low-resolution packets out of five high-resolution packets and send them as shown in Figure 2.16. If a packet is lost, we can use the

low-resolution version from the next packet. Note that the low-resolution section in the first packet is empty.

In this method, if the last packet is lost, it cannot be recovered, but we use the low-resolution version of a packet if the lost packet is not the last one. The audio and video reproduction does not have the same quality, but the lack of quality is not recognized most of the time.

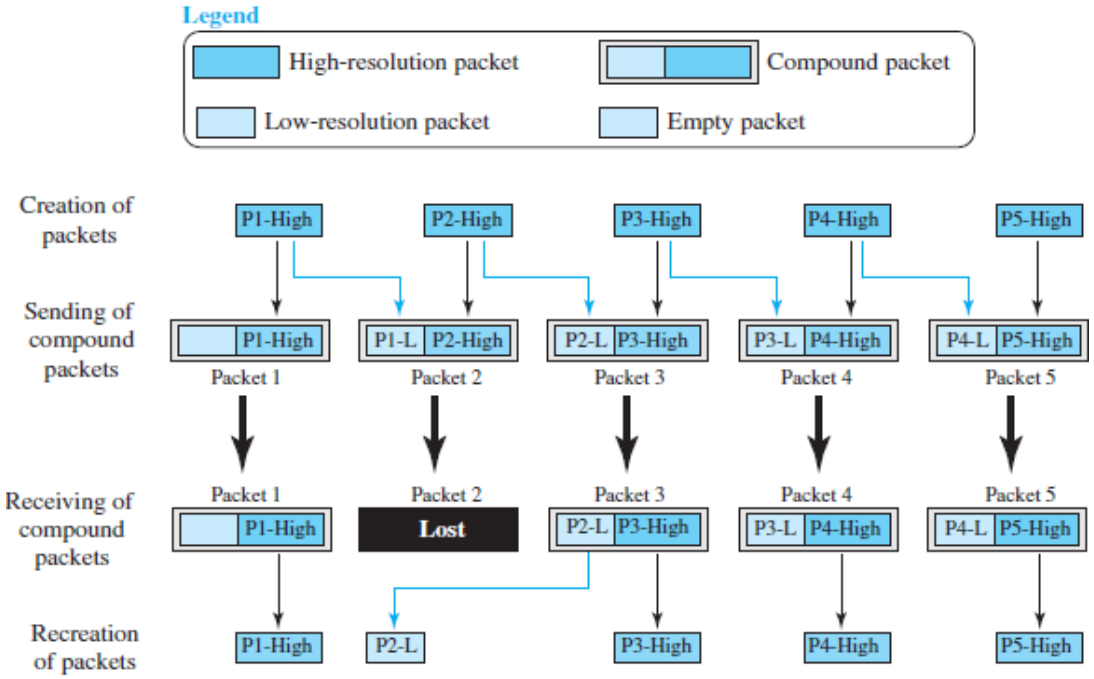


FIGURE 2.16: COMPOUNDING HIGH- AND LOW-RESOLUTION PACKETS

DATA LINK CONTROL

DLC SERVICES:

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*.

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, on the other hand, 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, which 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*.

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 2.17 shows the format of a frame in a character-oriented protocol.

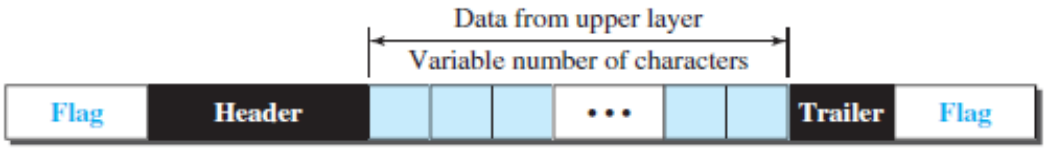


FIGURE 2.17: A FRAME IN A CHARACTER-ORIENTED PROTOCOL

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 2.18 shows the situation.

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?

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.

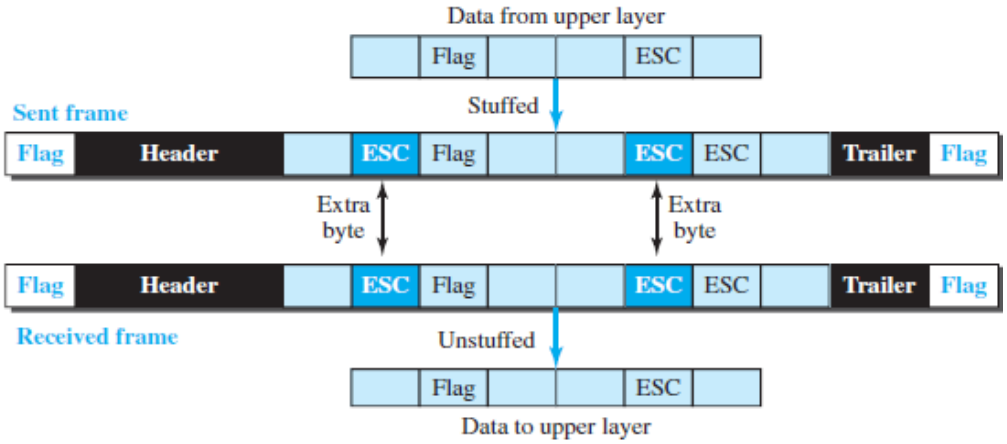


FIGURE 2.18: BYTE STUFFING AND UNSTUFFING

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, as shown in Figure 2.19.

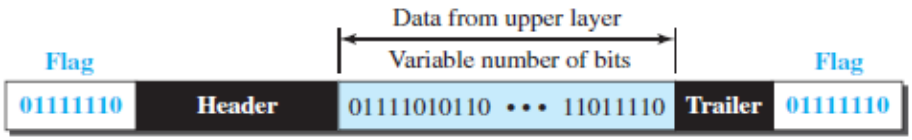


FIGURE 2.19: A FRAME IN A BIT-ORIENTED PROTOCOL

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 2.20 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 flag like 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.

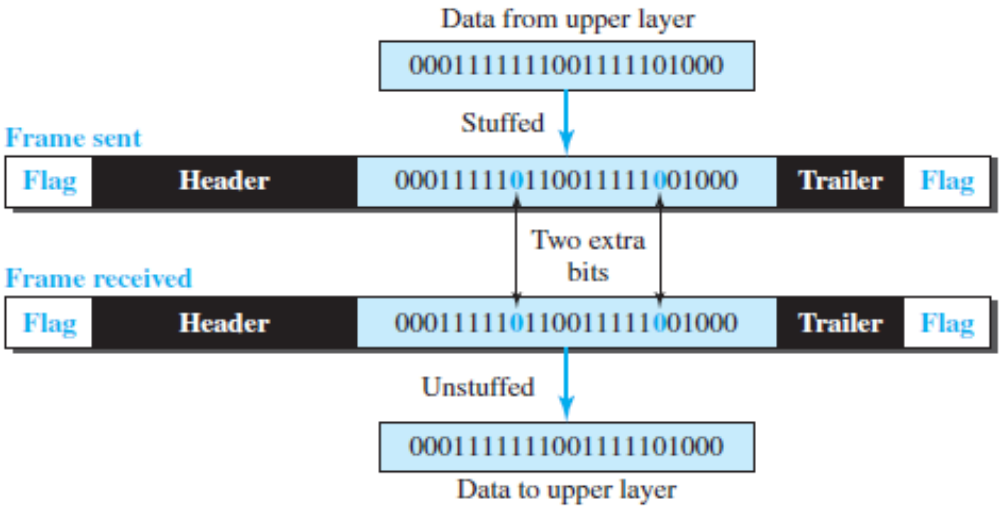


FIGURE 2.20: BIT STUFFING AND UNSTUFFING

FLOW AND ERROR 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, as shown in Figure 2.21.

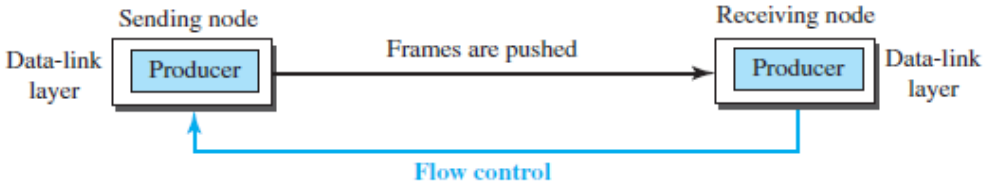


FIGURE 2.21: FLOW CONTROL AT THE DATA-LINK LAYER

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.

CONNECTIONLESS AND CONNECTION-ORIENTED:

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

Connectionless Protocol:

In a connectionless protocol, frames are sent from one node to the next without any relationship between the frames; each frame is independent. Note that the term *connectionless* here does not mean that there is no physical connection (transmission medium) between the nodes; it means that there is no *connection*

between frames. The frames are not numbered and there is no sense of ordering. Most of the data-link protocols for LANs are connectionless protocols.

Connection-Oriented Protocol:

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

If they are not received in order, the receiver needs to wait until all frames belonging to the same set are received and then deliver them in order to the network layer. Connection oriented protocols are rare in wired LANs, but we can see them in some point-to-point protocols, some wireless LANs, and some WANs.

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. Although the first two protocols still are used at the data-link layer, the last two have disappeared.

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 2.22, 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.

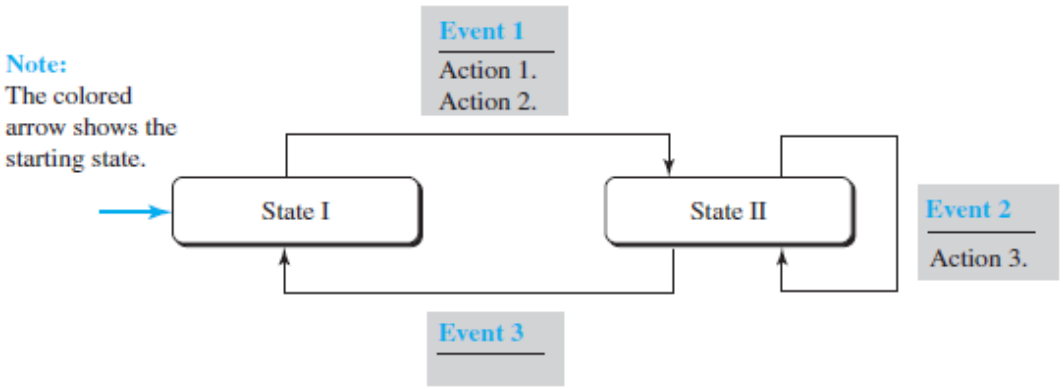


FIGURE 2.22: CONNECTIONLESS AND CONNECTION-ORIENTED SERVICE REPRESENTED AS FSMS

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.

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 2.23 shows the layout for this protocol.

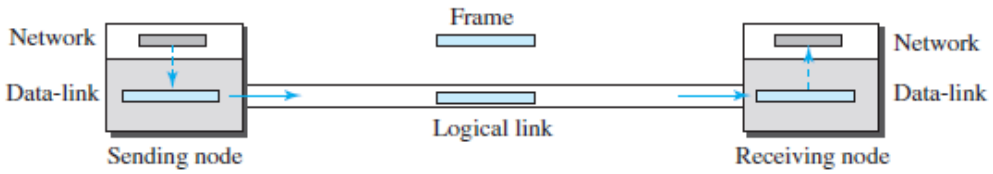


FIGURE 2.23: SIMPLE PROTOCOL

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.

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 2.24 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.

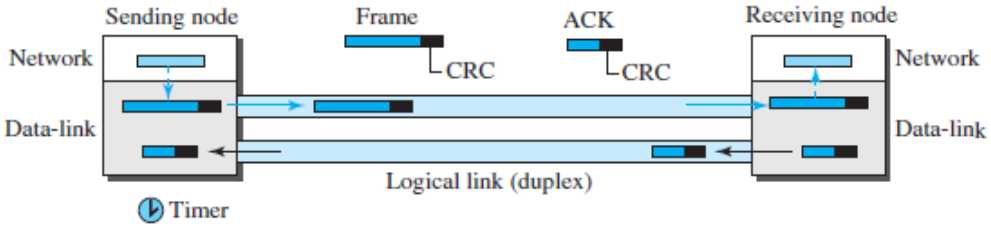


FIGURE 2.24: STOP-AND-WAIT PROTOCOL

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 data link 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 2.25.

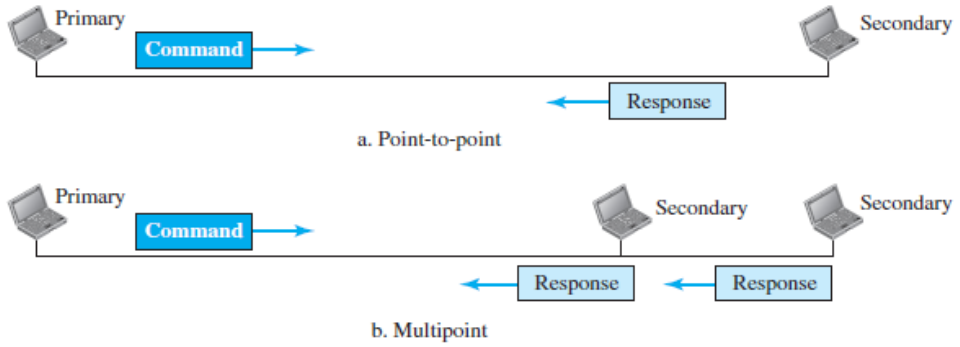


FIGURE 2.25: NORMAL RESPONSE MODE

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 2.26. This is the common mode today.

FIGURE 2.26: ASYNCHRONOUS BALANCED MODE

Framing:

To provide the flexibility necessary to support all the options possible in the modes and configurations just described, 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, as shown in Figure 2.27: 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.

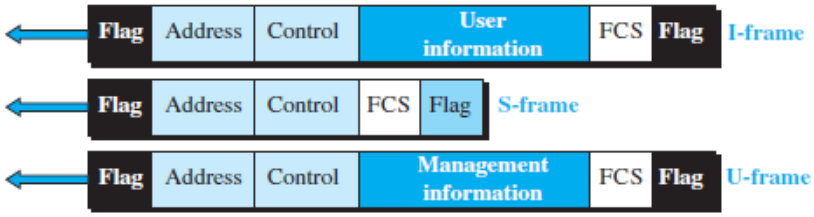


FIGURE 2.27: HDLC FRAMES

- ***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 *to* address. If a secondary station creates the frame, it contains *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.
- ***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. The format is specific for the type of frame, as shown in Figure 2.28.

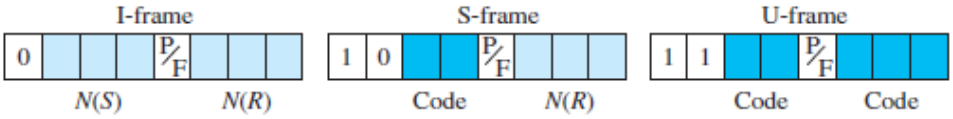


FIGURE 2.28: CONTROL FIELD FORMAT FOR THE DIFFERENT FRAME TYPES

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:

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

Services Provided by PPP:

PPP defines the format of the frame to be exchanged between devices. It also defines how two devices can negotiate the establishment of the link and the exchange of data. PPP is designed to accept payloads from several network layers (not only IP).

Authentication is also provided in the protocol, but it is optional. The new version of PPP, called *Multilink PPP,* provides connections over multiple links. One interesting feature of PPP is that it provides network address configuration. This is particularly useful when a home user needs a temporary network address to connect to the Internet.

Services Not Provided by PPP:

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

If the frame is corrupted, it is silently discarded; the upper-layer protocol needs to take care of the problem. Lack of error control and sequence numbering may cause a packet to be received out of order. PPP does not provide a sophisticated addressing mechanism to handle frames in a multipoint configuration.

Framing:

PPP uses a character-oriented (or byte-oriented) frame. Figure 2.29 shows the format of a PPP frame. The description of each field follows:

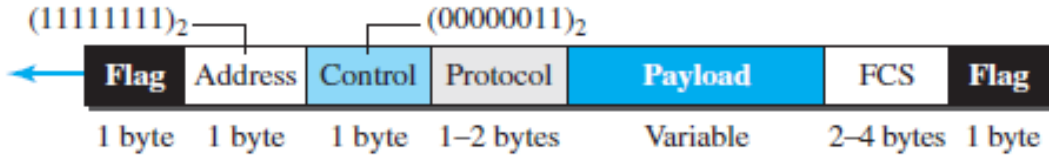


FIGURE 2.29: PPP FRAME FORMAT

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

- ***Address.*** The address field in this protocol is a constant value and set to 11111111 (broadcast address).
- ***Control.*** This field is set to the constant value 00000011 (imitating unnumbered frames in HDLC). As we will discuss later, PPP does not provide any flow control. Error control is also limited to error detection.
- ***Protocol.*** The protocol field defines what is being carried in the data field: either user data or other information. This field is by default 2 bytes long, but the two parties can agree to use only 1 byte.
- ***Payload field.*** 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 flag like pattern appears in the data, this extra byte is stuffed to tell the receiver that the next byte is not a flag. Obviously, the escape byte itself should be stuffed with another escape byte.

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.

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.

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 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.

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.

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.

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 2.30. IPCP defines seven packets, distinguished by their code values, as shown in Table 2.4.

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

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

TABLE 2.4: CODE VALUE FOR IPCP PACKETS

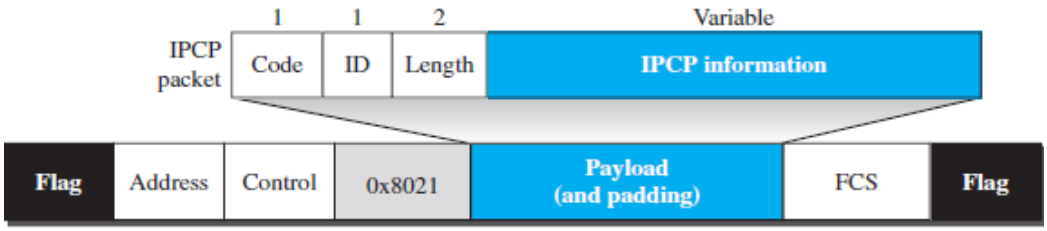


FIGURE 2.30: IPCP PACKET ENCAPSULATED IN PPP FRAME

Multilink PPP:

PPP was originally designed for a single-channel point-to-point physical link. The availability of multiple channels in a single point-to-point link motivated the development of Multilink PPP. In this case, a logical PPP frame is divided into several actual PPP frames. A segment of the logical frame is carried in the payload of an actual PPP frame, as shown in Figure 2.31.

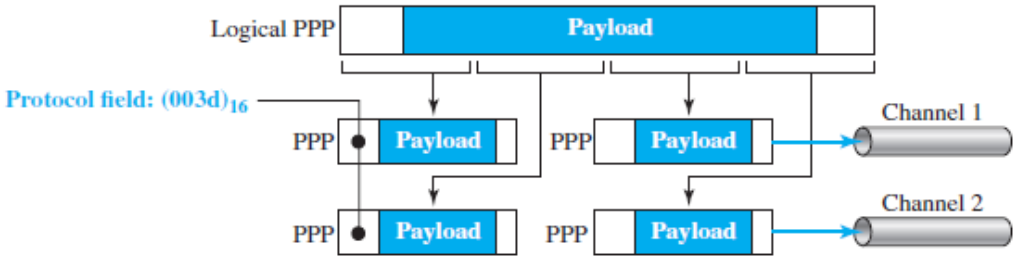


FIGURE 2.31: MULTILINK PPP

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).* We categorize them into three groups, as shown in Figure 2.32.

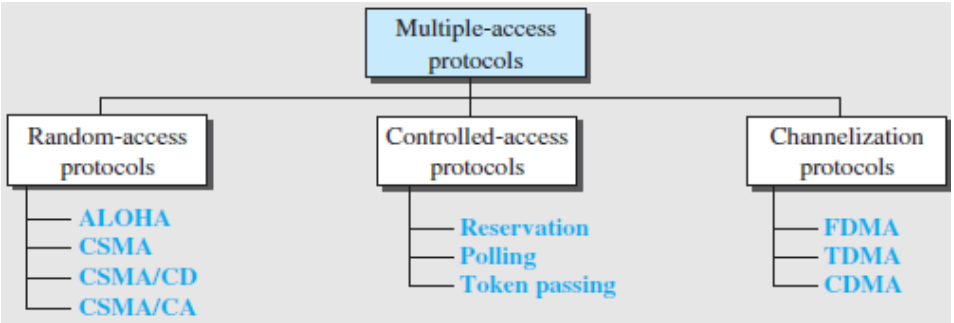


FIGURE 2.32: TAXONOMY OF MULTIPLE-ACCESS PROTOCOLS

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.

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.

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.

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.

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:

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 2.33 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.

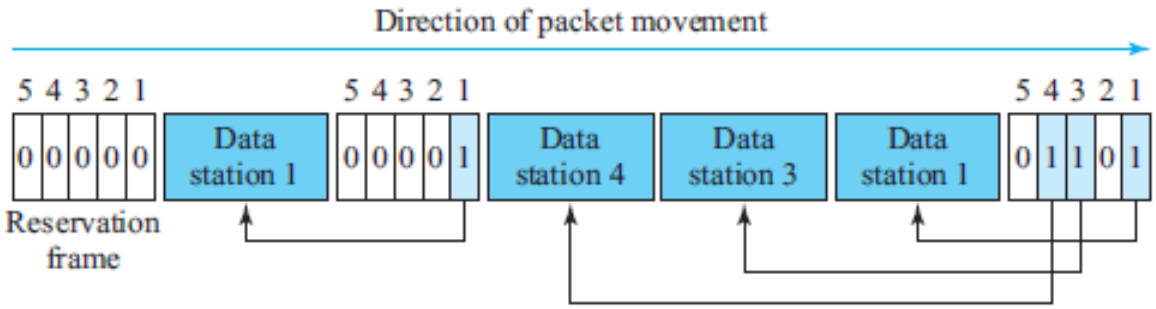


FIGURE 2.33: RESERVATION ACCESS METHOD

POLLING:

Polling works with topologies in which one device is designated as a ***primary station*** and the other devices are ***secondary stations*****.** All data exchanges must be made through the primary device even when the ultimate destination is a secondary device.

The primary device controls the link; the secondary devices follow its instructions. It is up to the primary device to determine which device is allowed to use the channel at a given time.

The primary device, therefore, is always the initiator of a session (see Figure 2.34). This method uses poll and select functions to prevent collisions. However, the drawback is if the primary station fails, the system goes down.

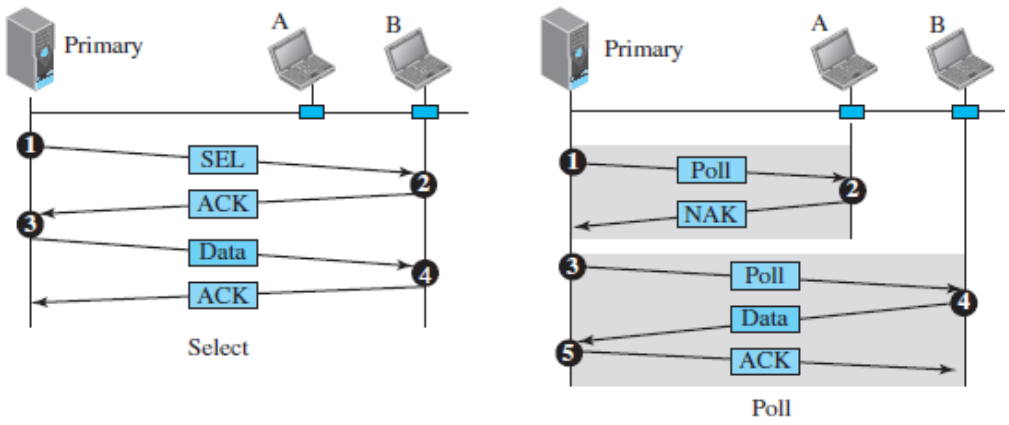


FIGURE 2.34: SELECT AND POLL FUNCTIONS IN POLLING-ACCESS METHOD

Select:

The *select* function is used whenever the primary device has something to send. Remember that the primary controls the link. If the primary is neither sending nor receiving data, it knows the link is available. If it has something to send, the primary device sends it.

What it does not know, however, is whether the target device is prepared to receive. So the primary must alert the secondary to the upcoming transmission and wait for an acknowledgment of the secondary's ready status.

Before sending data, the primary creates and transmits a select (SEL) frame, one field of which includes the address of the intended secondary.

Poll:

The *poll* function is used by the primary device to solicit transmissions from the secondary devices. When the primary is ready to receive data, it must ask (poll) each device in turn if it has anything to send.

When the first secondary is approached, it responds either with a NAK frame if it has nothing to send or with data (in the form of a data frame) if it does. If the response is negative (a NAK frame), then the primary polls the next secondary in the same manner until it finds one with data to send.

When the response is positive (a data frame), the primary reads the frame and returns an acknowledgment (ACK frame), verifying its receipt.

TOKEN PASSING:

In the **token-passing** method, the stations in a network are organized in a logical ring. 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 (*meaning control*) 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.

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 band pass filter to confine the transmitter frequencies. To prevent station interferences, the allocated bands are separated from one another by small *guard bands.*

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

We need to emphasize that although FDMA and frequency-division multiplexing (FDM) conceptually seem similar, there are differences between them.

FDM is a physical layer technique that combines the loads from low bandwidth channels and transmits them by using a high-bandwidth channel.

The channels that are combined are low-pass. The multiplexer modulates the signals, combines them, and creates a bandpass signal. The bandwidth of each channel is shifted by the multiplexer.

FDMA, on the other hand, is an access method in the data-link layer. The datalink layer in each station tells its physical layer to make a bandpass signal from the data passed to it. The signal must be created in the allocated band.

There is no physical multiplexer at the physical layer. The signals created at each station are automatically bandpass-filtered. They are mixed when they are sent to the common channel.

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.

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.

We also need to emphasize that although TDMA and time-division multiplexing (TDM) conceptually seem the same, there are differences between them.

TDM is a physical layer technique that combines the data from slower channels and transmits them by using a faster channel. The process uses a physical multiplexer that interleaves data units from each channel.

TDMA, on the other hand, is an access method in the data-link layer. The data-link layer in each station tells its physical layer to use the allocated time slot. There is no physical multiplexer at the physical layer.

CDMA:

Code-division multiple access (CDMA) was conceived (*meaning imagine/visualize*) 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 AND VIRTUAL LAN'S

Hosts or LANs do not normally operate in isolation. They are connected to one another or to the Internet. To connect hosts or LANs, we use connecting devices. Connecting devices can operate in different layers of the Internet model.

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.

We discuss 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. (See Figure 2.35)

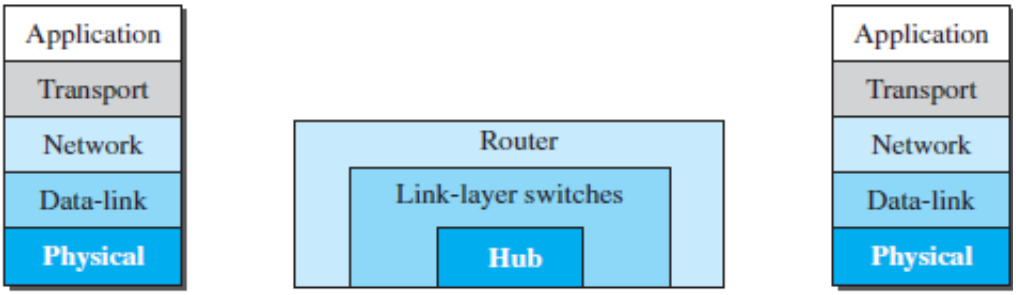


FIGURE 2.35: THREE CATEGORIES OF CONNECTING DEVICES

HUBS: A **hub** is a device that operates only in the physical layer. Signals that carry information within a network can travel a fixed distance before attenuation endangers the integrity of the data. A **repeater** receives a signal and, before it becomes too weak or corrupted, *regenerates* and *retimes* the original bit pattern.

The repeater then sends the refreshed signal. 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 2.36 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.

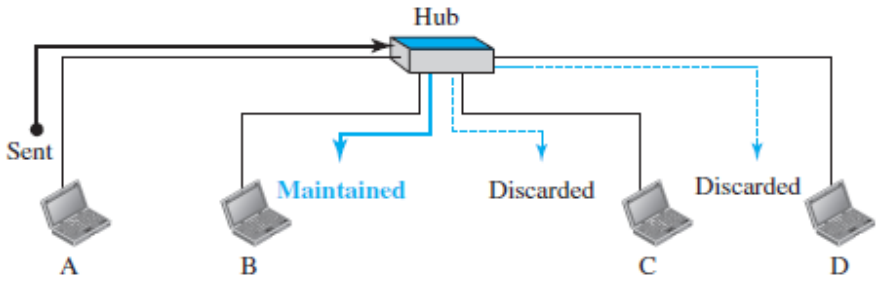


FIGURE 2.36: A HUB

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 2.36 shows the role of a repeater or a hub in a switched LAN.

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.

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 2.37, 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.

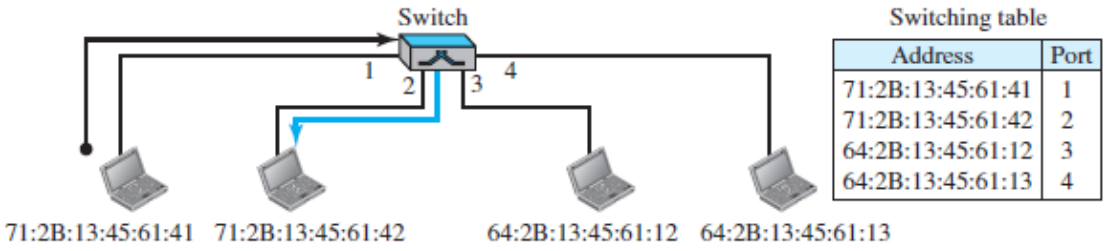


FIGURE 2.37: LINK-LAYER SWITCH

A Link-Layer Switch does not change the Link-Layer (MAC) addresses in a frame.

Transparent Switches:

A **transparent switch** is a switch in which the stations are completely unaware of the switch's existence. If a switch is added or deleted from the system, reconfiguration of the stations is unnecessary. According to the IEEE 802.1d specification, a system equipped with transparent switches must meet three criteria:

- Frames must be forwarded from one station to another.
- The forwarding table is automatically made by learning frame movements in the network.
- Loops in the system must be prevented.

Advantages of Switches:

A link-layer switch has several advantages over a hub. We discuss only two of them here.

Collision Elimination:

This means increasing the average bandwidth available to a host in the network. In a switched LAN, there is no need for carrier sensing and collision detection; each host can transmit at any time.

Connecting Heterogeneous Devices:

A link-layer switch can connect devices that use different protocols at the physical layer (data rates) and different transmission media.

As long as the format of the frame at the data-link layer does not change, a switch can receive a frame from a device that uses twisted-pair cable and sends data at 10 Mbps and deliver the frame to another device that uses fiber-optic cable and can receive data at 100 Mbps.

ROUTERS:

A **router** is a three-layer device; it operates in the physical, data-link, and network layers. As a physical-layer device, it regenerates the signal it receives. As a link-layer device, the router checks the physical addresses (source and destination) contained in the packet. As a network-layer device, a router checks the network-layer addresses.

A Router is a three-layer (Physical, Data-Link, and Network) device.

A router can connect networks. In other words, a router is an internetworking device; it connects independent networks to form an internetwork. According to this definition, two networks connected by a router become an internetwork or an internet.

There are three major differences between a router and a repeater or a switch.

1. A router has a physical and logical (IP) address for each of its interfaces.

2. A router acts only on those packets in which the link-layer destination address matches the address of the interface at which the packet arrives.

3. A router changes the link-layer address of the packet (both source and destination) when it forwards the packet.

Let us give an example. In Figure 2.38, assume an organization has two separate buildings with a Gigabit Ethernet LAN installed in each building. The organization uses switches in each LAN.

The two LANs can be connected to form a larger LAN using 10 Gigabit Ethernet technologies that speeds up the connection to the Ethernet and the connection to the organization server. A router then can connect the whole system to the Internet.

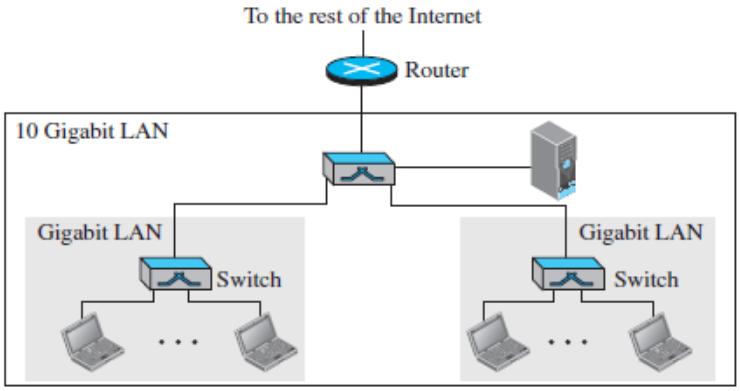


FIGURE 2.38: ROUTING EXAMPLE

A ROUTER CHANGES THE LINK-LAYER ADDRESSES IN A PACKET.

VIRTUAL LANS:

A station is considered part of a LAN if it physically belongs to that LAN. The criterion of membership is geographic. What happens if we need a virtual connection between two stations belonging to two different physical LANs? We can roughly define a **virtual local area network (VLAN)** as a local area network configured by software, not by physical wiring.

Let us use an example to elaborate on this definition. Figure 2.39 shows a switched LAN in an engineering firm in which nine stations are grouped into three LANs that are connected by a switch.

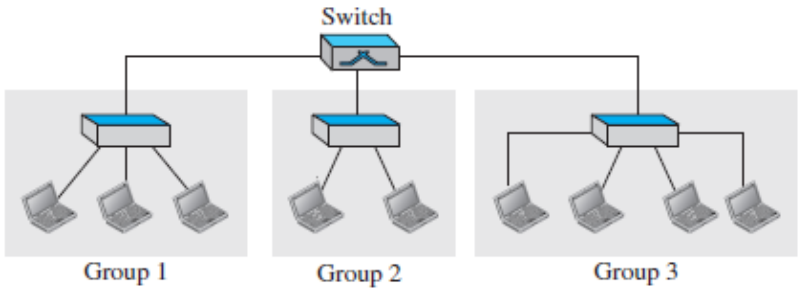


FIGURE 2.39: A SWITCH CONNECTING THREE LANS

The first three engineers work together as the first group, the next two engineers work together as the second group, and the last four engineers work together as the third group. The LAN is configured to allow this arrangement.

But what would happen if the administrators needed to move two engineers from the first group to the third group, to speed up the project being done by the third group?

The LAN configuration would need to be changed. The network technician must rewire. The problem is repeated if, in another week, the two engineers move back to their previous group. In a switched LAN, changes in the work group mean physical changes in the network configuration.

Figure 2.40 shows the same switched LAN divided into VLANs. The whole idea of VLAN technology is to divide a LAN into logical, instead of physical, segments. A LAN can be divided into several logical LANs, called *VLANs.*

Each VLAN is a work group in the organization. If a person moves from one group to another, there is no need to change the physical configuration. The group membership in VLANs is defined by software, not hardware. Any station can be logically moved to another VLAN.

All members belonging to a VLAN can receive broadcast messages sent to that particular VLAN. This means that if a station moves from VLAN 1 to VLAN 2, it receives broadcast messages sent to VLAN 2, but no longer receives broadcast messages sent to VLAN 1.

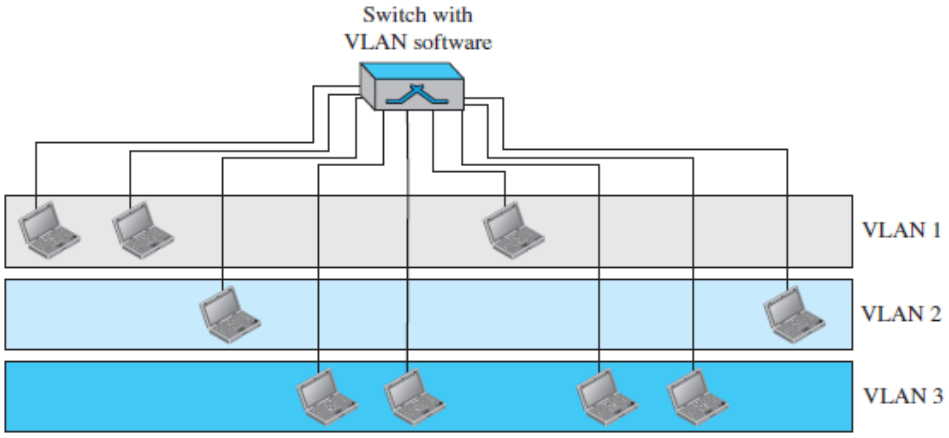


FIGURE 2.40: A SWITCH USING VLAN SOFTWARE

It is obvious that the problem in our previous example can easily be solved by using VLANs. Moving engineers from one group to another through software is easier than changing the configuration of the physical network.

VLAN technology even allows the grouping of stations connected to different switches in a VLAN. Figure 2.41 shows a backbone local area network with two switches and three VLANs. Stations from switches A and B belong to each VLAN.

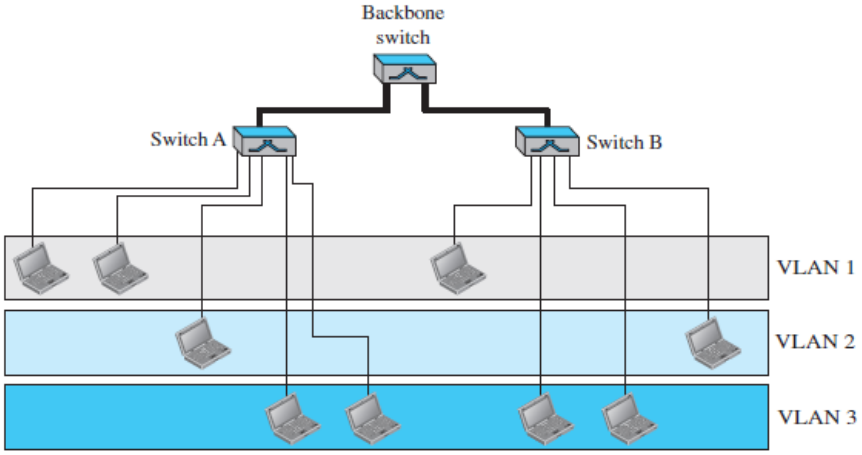


FIGURE 2.41: TWO SWITCHES IN A BACKBONE USING VLAN SOFTWARE

MEMBERSHIP:

Vendors use different characteristics such as ***interface numbers***, ***port numbers***, ***MAC addresses***, ***IP addresses***, ***IP multicast addresses***, or a combination of two or more of these to group stations in VLAN.

Interface Numbers:

Some VLAN vendor's uses switch interface numbers as a membership characteristic. For example, the administrator can define that stations connecting to ports 1, 2, 3, and 7 belong to VLAN 1, stations connecting to ports 4, 10, and 12 belong to VLAN 2, and so on.

MAC Addresses:

Some VLAN vendors use the 48-bit MAC address as a membership characteristic. For example, the administrator can stipulate that stations having MAC addresses E2:13:42:A1:23:34 and F2:A1:23:BC:D3:41 belong to VLAN 1.

IP Addresses:

Some VLAN vendors use the 32-bit IP address as a membership characteristic. For example, the administrator can stipulate that stations having IP addresses 181.34.23.67, 181.34.23.72, 181.34.23.98, and 181.34.23.112 belong to VLAN 1.

Multicast IP Addresses:

Some VLAN vendors use the multicast IP address as a membership characteristic. Multicasting at the IP layer is now translated to multicasting at the datalink layer.

Combination:

Recently, the software available from some vendors allows all these characteristics to be combined. The administrator can choose one or more characteristics when installing the software. In addition, the software can be reconfigured to change the settings.

ADVANTAGES:

There are several advantages to using VLANs.

Cost and Time Reduction:

VLANs can reduce the migration cost of stations going from one group to another. Physical reconfiguration takes time and is costly. Instead of physically moving one station to another segment or even to another switch, it is much easier and quicker to move it by using software.

Creating Virtual Work Groups:

VLANs can be used to create virtual work groups. For example, in a campus environment, professors working on the same project can send broadcast messages to one another without the necessity of belonging to the same department. This can reduce traffic if the multicasting capability of IP was previously used.

Security:

VLANs provide an extra measure of security. People belonging to the same group can send broadcast messages with the guaranteed assurance that users in other groups will not receive these messages.

UNIT-II
DATA LINK LAYER
OBJECTIVE QUESTIONS

1. Framing is done at_____.
a. **Data link layer** b. network layer
c. transport layer d. none

2. An example of error detecting code is _____.
a. VRC b. LRC
c.**CRC** d. none

3. An example of error correcting code is _____.
a. **Hamming code** b. hashing
c. a&b d. none

4. Data link layer also perform _____control.
a. Error b. flow
c. **both a&b** d. none

5. A frame contains_____.
a. header b. payload field
c. trailer **d. all the above**

6. Payload field contains actual **_____.**

a. **Data** b. address

c. both a&b d. none

7. Two consecutive **flag bytes** indicate the end of 1 frame & start of the other.

8. **"ESC" (escape)** byte is inserted before each flag byte.

9. Byte stuffing is also called as _____.

a. **Character stuffing** b. bit stuffing

c. a&b d. none

10. _____is stuffed when data link layer encounters 5 consecutive 1's.

a. 0 bit b. 1 bit

c. a&b d. none

11. Special bit pattern used in data link layer is **01111110**.

12. Error control can be done using_____.

a. Acknowledgments b. timers

c. a&b d. none

13. **Rate based flow control** contains protocol which controls the flow between sender & receiver.

14. Feedback based flow control moderates flow of sender & receiver through_____.

a. Feedback b. acknowledgments

c. a&b d. none

15. The number of bit positions in which 2 code words differ is called_____.

a. Hamming distance b. pipelining

c. a&b d. none

16. Codeword= **data unit + check bits.**

17. We perform_____operation for calculating hamming distance between 2 codewords.

a. AND **b. XOR**

c. NAND d. none

18. _____ Technique is considered in GoBack N protocol.

a. Hamming distance **b. pipelining**

c. a&b d. none

19. Connectionless service is provided using_____.

a. **UDP** b. TCP

c. a&b d. none

20. Connection oriented service is provided using **_____.**

a. UDP **b. TCP**

c. a&b d. none

21. Data in data link layer is called as **_____.**

a. **Frames** b. packets

c. bits d. none

22. In character count, frame starting field used to indicate the **number of character** in the frame.

23. In unrestricted simplex protocol data frame is transmitted in **_____-** direction only.

a. **One** b. both

c. a&b d. none

24. In sliding window protocol data frames are transmitted in _____directions.

a. One b. both

c. a&b d. none

25. The technique of temporarily delaying outgoing acknowledgments later hooked onto the next outgoing data frame is called_____.

a. Hamming distance **b. piggybacking**

c. a&b d. none

26. The _____is 2 layer in OSI model.

a. **Data link layer** b. network layer c. transport layer d. none

27. OSI – **Open System interconnection** model.

28.Sender's / receiver's window can _____as frames sent or received.

a. Grow b. shrink

c. a&b d. none

29. _____are used for packet order transmission.
a. Circuit number **b. sequence number**
c. a&b d. none

30.In _____only the damaged/ lost frame is transmitted.
a. **Selective repeat** b. goback –n
c. a&b d. none

31. The protocols used to determine who goes next on a multi-access channel belong to a sublayer of the data link layer called the _____.
a) Medium access control sublayer b) Logical link control sublayer
c) Multiple link control sublayer d) None

32. Medium access control sublayer is present in _____ layer of the OSI model.
a) Application layer **b) Data link layer**
c) Network layer d) Session layer

33. Stations are sometime called as _____.
a) Terminals b) Constant nodes
c) LAN d) None

34. _____ does not require global time synchronization.
a) Slotted ALOHA **b) Pure ALOHA**
c) Both a and b d) None

35. Systems in which multiple users share a common channel in a way that can lead to conflicts are known as _____ systems.
a) Contention b) Broadcast
c) Multicast d) None

36. CSMA stands for_____.
a**) Carrier sense multiple access** b) Carrier sends multiple accesses
c) Carry sense multiple access d) None

37. Protocols in which stations listen for a carrier and act accordingly are called _____
a) Multiple protocols **b) Carrier sense protocols**
c) Both a and b d) None

38. A Bit-Map Protocol and Binary countdown protocols are the protocols type of _____.
a) CSMA b) CSMA/CD
c) Collision-Free Protocols d) All the above

39. To allow multiple transmissions at the same time, the spectrum is divided into channels using _____.
a) WDMA b) CSMA/CD

c) CSMA d) DWDM

40. In Wavelength Division multiple access the station is divided in _____

a) Control channel b) Data channel
c) Both a and b d) None

41. Wavelength Division multiple access protocol supports ___ traffic classes.
a) 5 b) 6
c) 7 **d) 3**

42. Each station in WDMA has two _____ and two _____.
a) Transmitters –Receiver b) sender-Receiver
c) Server – Host d) None

43. When a large number of frequencies are being used, the system is sometimes called as _____.
a) WDMA b) CSMA/CD
c) CSMA **d) DWDM**

44. MACA stands for _____.
a) Medium access with collision avoidance b) Medium access with collision acceptance
c) Medium allow with collision avoidance d) None

45. What is RTS _____.
a) Ready to send **b) Request to send**
c) Both a and b d) None

46. What is CTS _____.
a) Clear to send b) Collect to send
c) Both a and b d) None

47. The IEEE has standardized a number of local area network and metropolitan area network under the name of _____.
a) IEEE 803 b) IEEE 805
c) IEEE 802 d) IEEE 804

48. The IEEE 802.3 standard is for _____.
a) Ethernet b) APRANET
c) Internet d) Subnet

49. The IEEE 802.16 standard is for _____.
a) Ethernet b) Bluetooth
c) Wireless MAN d) None

50. The IEEE 802.15 standard is for _____.

a) Ethernet **b) Bluetooth**
c) Wireless MAN d) None

51.10Base5 cabling, popularly called _____.
a) Thick Ethernet b) Thin Ethernet
c) Long Ethernet d) None

52. FDDI is_____.
a) First device division instruction b) Fiber distributed data interface
c) Fiber Device data interface d) none

53. The IEEE 802.15 standard is for _____.
a) Ethernet b) Bluetooth
c) Wireless MAN **d) Logical Link Control**

54. Which sub layer performs the channel allocation?
a) LLC **b) MAC**
c) Physical d) Network

55. The maximum through put of pure ALOHA is
a) 1/e **b) 1/2e**
c) e d) 2e

56. The average number of unsuccessful attempt per packet in pure ALOHA scheme is _____.
a) e^{2G} b) $e^{2G}+1$
c) $e^{G}-1$ **d) $e^{2G}-1$**

57. Generally the through put of a CSMA/CD over CSMA is _____.
a) Less **b) Greater**
c) Equal d) Not comparable

58. Which of the following state is not belongs to the CSMA/CD protocol
a) Transmission b) Idle
c) Contention **d) Waiting**

59. ALOHA _____.
a) Is used for channel allocation problem b) Is used of data transfer
c) Is buffering d) All the above

60. IEEE project 802 divides the data link layer into _____ upper sublayer and _____ lower sublayer?
a) MAC, LLC **b) LLC, MAC**
c) PDU, HDLC d) HDLC, PDU

UNIT-II
DESCRIPTIVE QUESTIONS

1. Discuss design issues of Data Link Layer.

2. Explain error detecting code using CRC with an example?

3. Discuss error correcting codes of Data link layer?

4. Briefly explain about simplex stop & wait protocol & unrestricted simplex protocol?

5. Explain about simplex protocol for noisy channel?

6. Mention different elementary data link protocols in brief?

7. Explain about sliding window protocol?

8. Explain about 1-bit sliding window protocol?

9. Discuss GoBack- N protocol & selective repeat protocol?

10. Explain hamming distance with an example?

11. Explain framing?

12. Explain how static channel allocation is done in LANs and MANs.

13. Briefly discuss about key assumptions in dynamic channel allocation in LANs and MANs.

14. State the advantages & disadvantages of FDDI.

15. Write short notes on the following:
    (a) Binary encoding.
    (b) Manchester encoding.
    (c) Differential Manchester encoding

16. Explain CAMA/CD operations.

17. Explain CSMA operations.

18. Draw & explain LLC frame format.

19. What is MAC sublayer? State its position in layered architecture.

20. What is Ethernet? Explain Ethernet Cabling.

21. Explain the 802.11 frame structure.

UNIT 3

NETWORK LAYER:

The network layer is concerned with getting packets from the source all the way to the destination. Getting to the destination may require making many hops at intermediate routers along the way. This function clearly contrasts with that of the data link layer, which has the more modest goal of just moving frames from one end of a wire to the other. Thus, the network layer is the lowest layer that deals with end-to-end transmission.

To achieve its goals, the network layer must know about the topology of the network (i.e., the set of all routers and links) and choose appropriate paths through it, even for large networks. It must also take care when choosing routes to avoid overloading some of the communication lines and routers while leaving others idle. Finally, when the source and destination are in different networks, new problems occur. It is up to the network layer to deal with them.

NETWORK LAYER DESIGN ISSUES:

STORE-AND-FORWARD PACKET SWITCHING:

Before starting to explain the details of the network layer, it is worth restating the context in which the network layer protocols operate. This context can be seen in Fig. 3.1. The major components of the network are the ISP's equipment (routers connected by transmission lines), shown inside the shaded oval, and the customers' equipment, shown outside the oval.

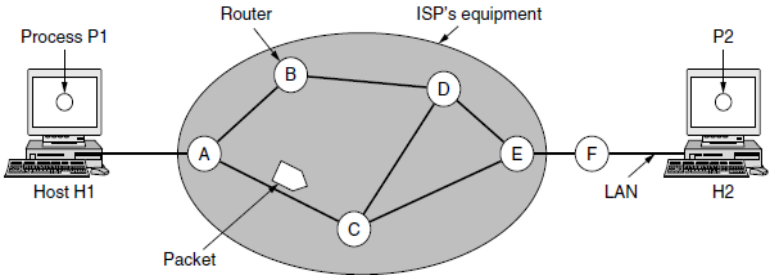


FIGURE 3.1: THE ENVIRONMENT OF THE NETWORK LAYER PROTOCOLS.

Host *H1* is directly connected to one of the ISP's routers, *A*, perhaps as a home computer that is plugged into a DSL modem. In contrast, *H2* is on a LAN, which might be an office Ethernet, with a router, *F*, owned and operated by the customer.

This router has a leased line to the ISP's equipment. We have shown *F* as being outside the oval because it does not belong to the ISP. For the purposes of this chapter, however, routers on customer premises are considered part of the ISP

network because they run the same algorithms as the ISP's routers (and our main concern here is algorithms).

This equipment is used as follows. A host with a packet to send transmits it to the nearest router, either on its own LAN or over a point-to-point link to the ISP. The packet is stored there until it has fully arrived and the link has finished its processing by verifying the checksum. Then it is forwarded to the next router along the path until it reaches the destination host, where it is delivered. This mechanism is called store-and-forward packet switching.

SERVICES PROVIDED TO THE TRANSPORT LAYER:

The network layer provides services to the transport layer at the network layer/transport layer interface. An important question is precisely what kind of services the network layer provides to the transport layer. The services need to be carefully designed with the following goals in mind:

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

IMPLEMENTATION OF CONNECTIONLESS SERVICE:

Having looked at the two classes of service the network layer can provide to its users, it is time to see how this layer works inside. Two different organizations are possible, depending on the type of service offered.

If connectionless service is offered, packets are injected into the network individually and routed independently of each other. No advance setup is needed. In this context, the packets are frequently called **datagrams** (in analogy with telegrams) and the network is called a **datagram network**.

If connection-oriented service is used, a path from the source router all the way to the destination router must be established before any data packets can be sent. This connection is called a **VC** (**virtual circuit**), in analogy with the physical circuits set up by the telephone system, and the network is called a **virtual-circuit network**.

Let us now see how a datagram network works. Suppose that the process *P1* in Fig. 3.2 has a long message for *P2*. It hands the message to the transport layer, with instructions to deliver it to process *P2* on host *H2*. The transport layer code runs on *H1*, typically within the operating system. It prepends a transport header to

the front of the message and hands the result to the network layer, probably just another procedure within the operating system.

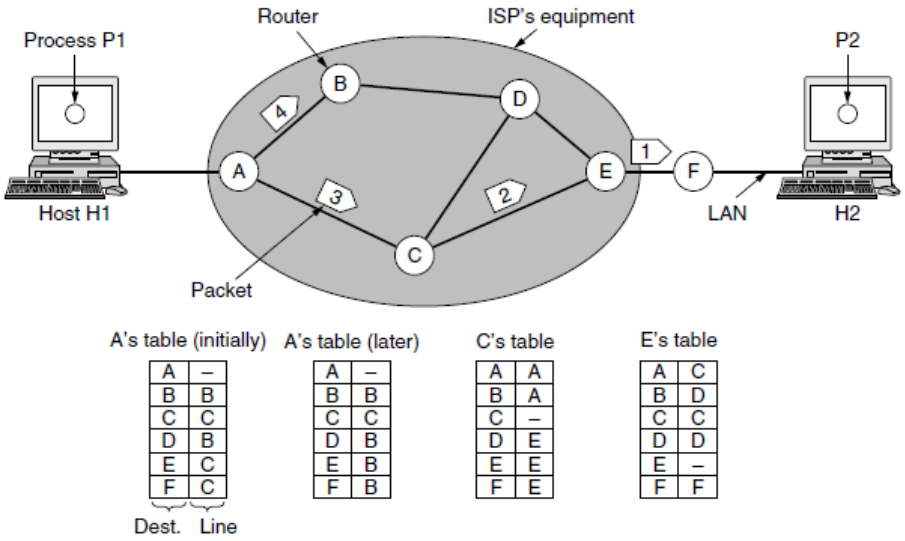


FIGURE 3.2: ROUTING WITHIN A DATAGRAM NETWORK

IMPLEMENTATION OF CONNECTION-ORIENTED SERVICE

For connection-oriented service, we need a virtual-circuit network. Let us see how that works. The idea behind virtual circuits is to avoid having to choose a new route for every packet sent, as in Fig. 3.2. Instead, when a connection is established, a route from the source machine to the destination machine is chosen as part of the connection setup and stored in tables inside the routers.

That route is used for all traffic flowing over the connection, exactly the same way that the telephone system works. When the connection is released, the virtual circuit is also terminated. With connection-oriented service, each packet carries an identifier telling which virtual circuit it belongs to.

As an example, consider the situation shown in Fig. 3.3. Here, host *H1* has established connection 1 with host *H2*. This connection is remembered as the first entry in each of the routing tables. The first line of *A*'s table says that if a packet bearing connection identifier 1 comes in from *H1*, it is to be sent to router *C* and given connection identifier 1. Similarly, the first entry at *C* routes the packet to *E*, also with connection identifier 1.

COMPARISON OF VIRTUAL-CIRCUIT AND DATAGRAM NETWORKS:

Both virtual circuits and datagrams have their supporters and their detractors. We will now attempt to summarize both sets of arguments. The major issues are listed in Fig. 3.4. Inside the network, several trade-offs exist between virtual circuits and datagrams.

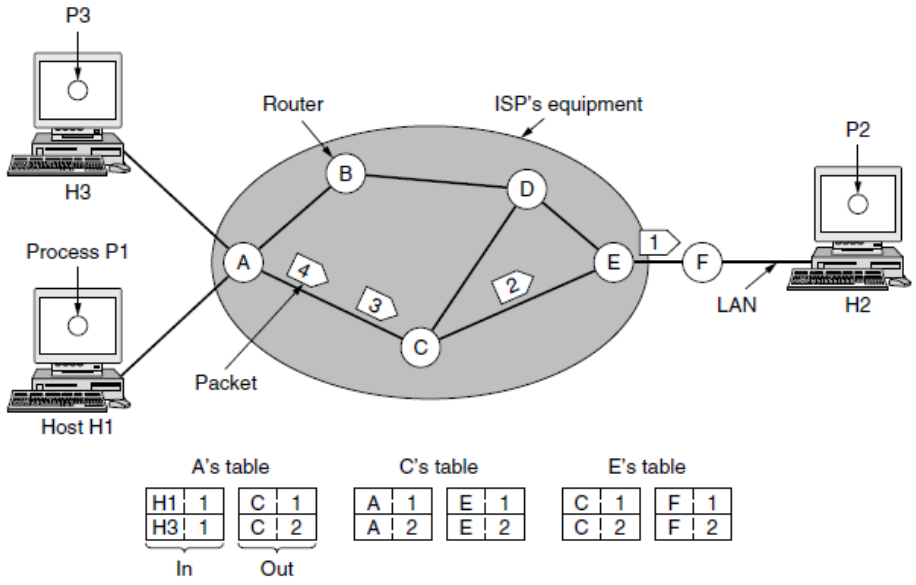


FIGURE 3.3: ROUTING WITHIN A VIRTUAL-CIRCUIT NETWORK

One trade-off is setup time versus address parsing time. Using virtual circuits requires a setup phase, which takes time and consumes resources. However, once this price is paid, figuring out what to do with a data packet in a virtual-circuit network is easy: the router just uses the circuit number to index into a table to find out where the packet goes. In a datagram network, no setup is needed but a more complicated lookup procedure is required to locate the entry for the destination.

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

FIGURE 3.4: COMPARISON OF DATAGRAM AND VIRTUAL-CIRCUIT NETWORKS

ROUTING ALGORITHMS:

The main function of the network layer is routing packets from the source machine to the destination machine. In most networks, packets will require multiple hops to make the journey.

The only notable exception is for broadcast networks, but even here routing is an issue if the source and destination are not on the same network segment. The algorithms that choose the routes and the data structures that they use are a major area of network layer design.

The **routing algorithm** is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on. If the network uses datagrams internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time.

If the network uses virtual circuits internally, routing decisions are made only when a new virtual circuit is being set up. Thereafter, data packets just follow the already established route. The latter case is sometimes called **session routing** because a route remains in force for an entire session (e.g., while logged in over a VPN).

It is sometimes useful to make a distinction between routing, which is making the decision which routes to use, and forwarding, which is what happens when a packet arrives. One can think of a router as having two processes inside it. One of them handles each packet as it arrives, looking up the outgoing line to use for it in the routing tables. This process is **forwarding**. The other process is responsible for filling in and updating the routing tables. That is where the routing algorithm comes into play.

Regardless of whether routes are chosen independently for each packet sent or only when new connections are established, certain properties are desirable in a routing algorithm: ***correctness***, ***simplicity***, ***robustness***, ***stability***, ***fairness***, and ***efficiency***.

Correctness and simplicity hardly require comment, but the need for robustness may be less obvious at first. Once a major network comes on the air, it may be expected to run continuously for years without system-wide failures.

During that period there will be hardware and software failures of all kinds. Hosts, routers, and lines will fail repeatedly, and the topology will change many times. The routing algorithm should be able to cope with changes in the topology and traffic without requiring all jobs in all hosts to be aborted. Imagine the havoc if the network needed to be rebooted every time some router crashed!

Stability is also an important goal for the routing algorithm. There exist routing algorithms that never converge to a fixed set of paths, no matter how long

they run. A stable algorithm reaches equilibrium and stays there. It should converge quickly too, since communication may be disrupted until the routing algorithm has reached equilibrium.

Fairness and efficiency may sound obvious—surely no reasonable person would oppose them—but as it turns out, they are often contradictory goals. As a simple example of this conflict, look at Fig. 3.5. Suppose that there is enough traffic between *A* and *A*′, between *B* and *B*′, and between *C* and *C*′ to saturate the horizontal links.

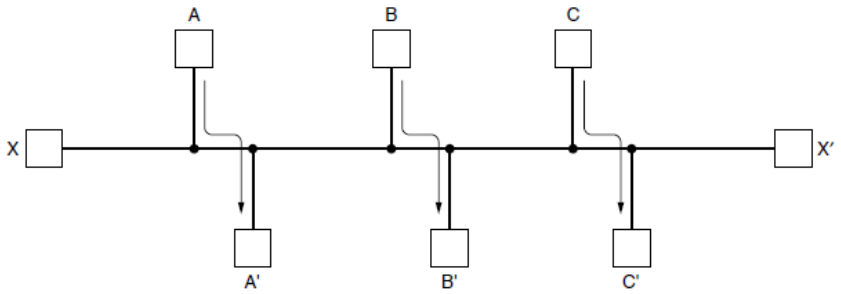


FIGURE 3.5: NETWORK WITH A CONFLICT BETWEEN FAIRNESS AND EFFICIENCY

To maximize the total flow, the *X* to *X*′ traffic should be shut off altogether. Unfortunately, *X* and *X*′ may not see it that way. Evidently, some compromise between global efficiency and fairness to individual connections is needed.

Routing algorithms can be grouped into two major classes: nonadaptive and adaptive. **Nonadaptive algorithms** do not base their routing decisions on any measurements or estimates of the current topology and traffic.

Instead, the choice of the route to use to get from *I* to *J* (for all *I* and *J*) is computed in advance, offline, and downloaded to the routers when the network is booted. This procedure is sometimes called **static routing**. Because it does not respond to failures, static routing is mostly useful for situations in which the routing choice is clear.

Adaptive algorithms, in contrast, change their routing decisions to reflect changes in the topology, and sometimes changes in the traffic as well. These **dynamic routing** algorithms differ in where they get their information (e.g., locally, from adjacent routers, or from all routers), when they change the routes and what metric is used for optimization.

THE OPTIMALITY PRINCIPLE: Before we get into specific algorithms, it may be helpful to note that one can make a general statement about optimal routes without regard to network topology or traffic. This statement is known as the **optimality principle** (Bellman, 1957).

It states that if router *J* is on the optimal path from router *I* to router *K*, then the optimal path from *J* to *K* also falls along the same route.

As a direct consequence of the optimality principle, we can see that the set of optimal routes from all sources to a given destination form a tree rooted at the destination. Such a tree is called a **sink tree** and is illustrated in Fig. 3.6(b), where the distance metric is the number of hops. The goal of all routing algorithms is to discover and use the sink trees for all routers.

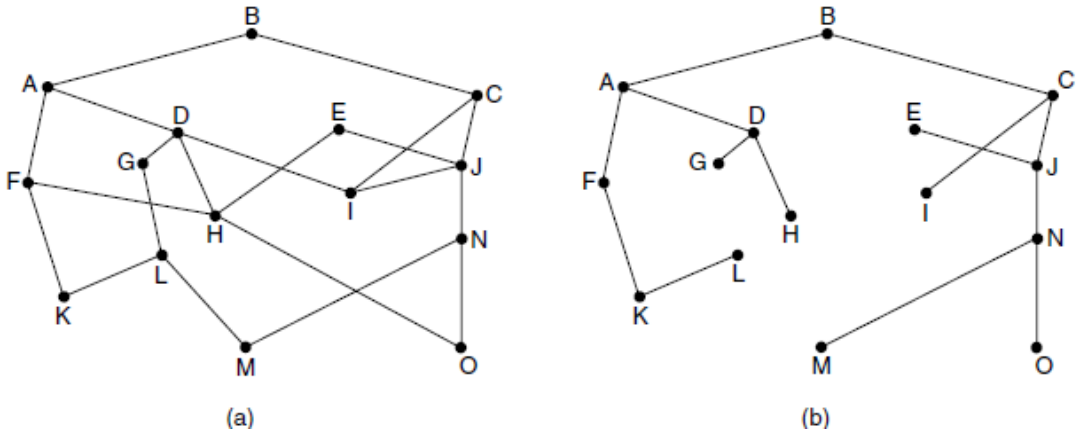


FIGURE 3.6: (A) A NETWORK. (B) A SINK TREE FOR ROUTER B

Note that a sink tree is not necessarily unique; 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 convenient shorthand for both cases. Both cases also depend on the technical assumption 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. Since a sink tree is indeed a tree, it does not contain any loops, so each packet will be delivered within a finite and bounded number of hops.

SHORTEST PATH ALGORITHM:

The concept of a **shortest path** deserves some explanation. One way of measuring path length is the number of hops. Using this metric, the paths *ABC* and *ABE* in Fig. 3.7 are equally long. Another metric is the geographic distance in kilometers, in which case *ABC* is clearly much longer than *ABE* (assuming the figure is drawn to scale).

However, many other metrics besides hops and physical distance are also possible. For example, each edge could be labeled with the mean delay of a standard test packet, as measured by hourly runs. With this graph labeling, the shortest path is the fastest path rather than the path with the fewest edges or kilometers.

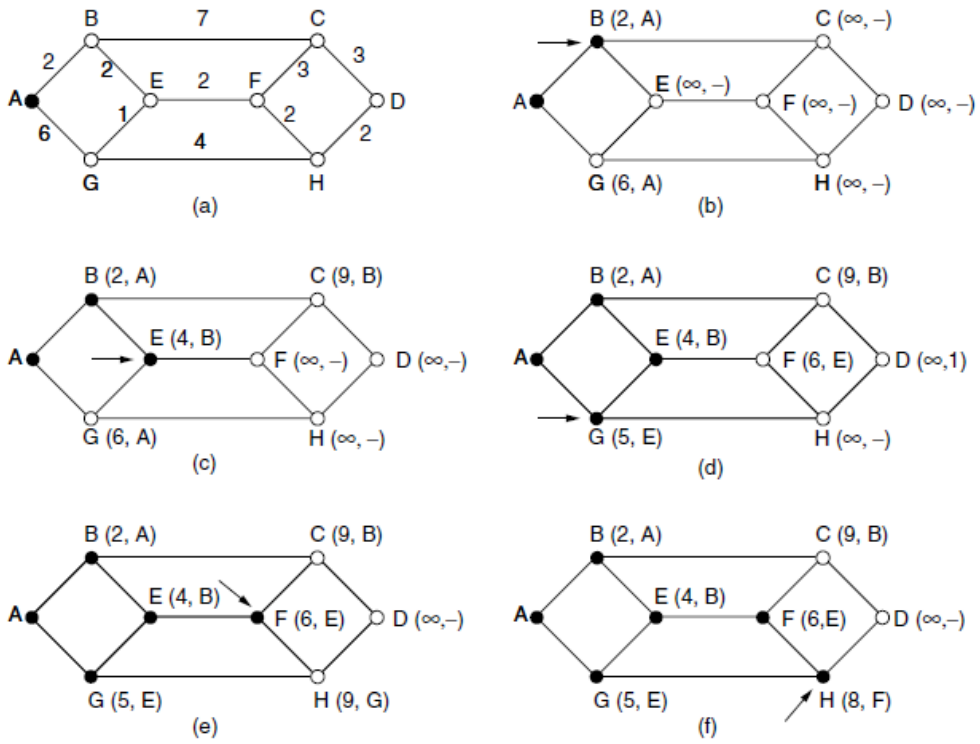


Figure 3.7: The first six steps used in computing the shortest path from *A* to *D*. The arrows indicate the working node.

Several algorithms for computing the shortest path between two nodes of a graph are known. This one is due to Dijkstra (1959) and finds the shortest paths between a source and all destinations in the network. Each node is labeled (in parentheses) with its distance from the source node along the best known path.

The distances must be non-negative, as they will be if they are based on real quantities like bandwidth and delay. Initially, no paths are known, so all nodes are labeled with infinity. As the algorithm proceeds and paths are found, the labels may change, reflecting better paths.

A label may be either tentative or permanent. Initially, all labels are tentative. When it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter.

To illustrate how the labeling algorithm works, look at the weighted, undirected graph of Fig. 3.7(a), where the weights represent, for example, distance. We want to find the shortest path from *A* to *D*. We start out by marking node *A* as permanent, indicated by a filled-in circle.

Then we examine, in turn, each of the nodes adjacent to *A* (the working node), relabeling each one with the distance to *A*. Whenever a node is relabeled, we also label it with the node from which the probe was made so that we can reconstruct the final path later.

If the network had more than one shortest path from *A* to *D* and we wanted to find all of them, we would need to remember all of the probe nodes that could reach a node with the same distance.

Having examined each of the nodes adjacent to *A*, we examine all the tentatively labeled nodes in the whole graph and make the one with the smallest label permanent, as shown in Fig. 3.7(b). This one becomes the new working node.

We now start at *B* and examine all nodes adjacent to it. If the sum of the label on *B* and the distance from *B* to the node being considered is less than the label on that node, we have a shorter path, so the node is relabeled.

After all the nodes adjacent to the working node have been inspected and the tentative labels changed if possible, the entire graph is searched for the tentatively labeled node with the smallest value. This node is made permanent and becomes the working node for the next round. Figure 3.7 shows the first six steps of the algorithm.

To see why the algorithm works, look at Fig. 3.7(c). At this point we have just made *E* permanent. Suppose that there were a shorter path than *ABE*, say *AXYZE* (for some *X* and *Y*).

There are two possibilities: either node *Z* has already been made permanent, or it has not been. If it has, then *E* has already been probed (on the round following the one when *Z* was made permanent), so the *AXYZE* path has not escaped our attention and thus cannot be a shorter path.

Now consider the case where *Z* is still tentatively labeled. If the label at *Z* is greater than or equal to that at *E*, then *AXYZE* cannot be a shorter path than *ABE*. If the label is less than that of *E*, then *Z* and not *E* will become permanent first, allowing *E* to be probed from *Z*.

FLOODING:

When a routing algorithm is implemented, each router must make decisions based on local knowledge, not the complete picture of the network. A simple local technique is **flooding**, in which every incoming packet is sent out on every outgoing line except the one it arrived on.

Flooding obviously generates vast numbers of duplicate packets, in fact, an infinite number unless some measures are taken to damp the process. One such measure is to have a hop counter contained in the header of each packet that is decremented at each hop, with the packet being discarded when the counter reaches zero. Ideally, the hop counter should be initialized to the length of the path from source to destination. If the sender does not know how long the path is, it can initialize the counter to the worst case, namely, the full diameter of the network.

Flooding with a hop count can produce an exponential number of duplicate packets as the hop count grows and routers duplicate packets they have seen before. A better technique for damming the flood is to have routers keep track of which packets have been flooded, to avoid sending them out a second time.

One way to achieve this goal is to have the source router put a sequence number in each packet it receives from its hosts. Each router then needs a list per source router telling which sequence numbers originating at that source have already been seen. If an incoming packet is on the list, it is not flooded.

Flooding is not practical for sending most packets, but it does have some important uses. First, it ensures that a packet is delivered to every node in the network. This may be wasteful if there is a single destination that needs the packet, but it is effective for broadcasting information. In wireless networks, all messages transmitted by a station can be received by all other stations within its radio range, which is, in fact, flooding, and some algorithms utilize this property.

Second, flooding is tremendously robust. Even if large numbers of routers are blown to bits (e.g., in a military network located in a war zone), flooding will find a path if one exists, to get a packet to its destination. Flooding also requires little in the way of setup. The routers only need to know their neighbors.

This means that flooding can be used as a building block for other routing algorithms that are more efficient but need more in the way of setup. Flooding can also be used as a metric against which other routing algorithms can be compared. Flooding always chooses the shortest path because it chooses every possible path in parallel. Consequently, no other algorithm can produce a shorter delay (if we ignore the overhead generated by the flooding process itself).

DISTANCE VECTOR ROUTING:

A **distance vector routing** algorithm operates by having each router maintain a table (i.e., a vector) giving the best known distance to each destination and which link to use to get there. These tables are updated by exchanging information with the neighbors. Eventually, every router knows the best link to reach each destination.

The distance vector routing algorithm is sometimes called by other names, most commonly the distributed **Bellman-Ford** routing algorithm, after the researchers who developed it (Bellman, 1957; and Ford and Fulkerson, 1962). It was the original ARPANET routing algorithm and was also used in the Internet under the name RIP.

In distance vector routing, each router maintains a routing table indexed by, and containing one entry for each router in the network. This entry has two parts: the preferred outgoing line to use for that destination and an estimate of the

distance to that destination. The distance might be measured as the number of hops or using another metric, as we discussed for computing shortest paths.

The router is assumed to know the ‘‘distance’’ to each of its neighbors. If the metric is hops, the distance is just one hop. If the metric is propagation delay, the router can measure it directly with special ECHO packets that the receiver just timestamps and sends back as fast as it can.

LINK STATE ROUTING:

Distance vector routing was used in the ARPANET until 1979, when it was replaced by link state routing. The primary problem that caused its demise was that the algorithm often took too long to converge after the network topology changed (due to the count-to-infinity problem). Consequently, it was replaced by an entirely new algorithm, now called **link state routing**.

Variants of link state routing called IS-IS and OSPF are the routing algorithms that are most widely used inside large networks and the Internet today. The idea behind link state routing is fairly simple and can be stated as five parts. Each router must do the following things to make it work:

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

In effect, the complete topology is distributed to every router. Then Dijkstra’s algorithm can be run at each router to find the shortest path to every other router.

Link state routing is widely used in actual networks, so a few words about some example protocols are in order. Many ISPs use the **IS-IS** (**Intermediate System-Intermediate System**) link state protocol (Oran, 1990). It was designed for an early network called DECnet, later adopted by ISO for use with the OSI protocols and then modified to handle other protocols as well, most notably, IP.

OSPF (**Open Shortest Path First**) is the other main link state protocol. It was designed by IETF several years after IS-IS and adopted many of the innovations designed for IS-IS. These innovations include a self-stabilizing method of flooding link state updates, the concept of a designated router on a LAN, and the method of computing and supporting path splitting and multiple metrics.

As a consequence, there is very little difference between IS-IS and OSPF. The most important difference is that IS-IS can carry information about multiple

network layer protocols at the same time (e.g., IP, IPX, and AppleTalk). OSPF does not have this feature, and it is an advantage in large multiprotocol environments.

HIERARCHICAL ROUTING:

As networks grow in size, the router routing tables grow proportionally. Not only is router memory consumed by ever-increasing tables, but more CPU time is needed to scan them and more bandwidth is needed to send status reports about them. At a certain point, the network may grow to the point where it is no longer feasible for every router to have an entry for every other router, so the routing will have to be done hierarchically, as it is in the telephone network.

When hierarchical routing is used, the routers are divided into what we will call **regions**. Each router knows all the details about how to route packets to destinations within its own region but knows nothing about the internal structure of other regions. When different networks are interconnected, it is natural to regard each one as a separate region to free the routers in one network from having to know the topological structure of the other ones.

For huge networks, a two-level hierarchy may be insufficient; it may be necessary to group the regions into clusters, the clusters into zones, the zones into groups, and so on, until we run out of names for aggregations.

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 sending to all machines and letting those that are interested read the data. Sending a packet to all destinations simultaneously is called **broadcasting**.

Various methods have been proposed for doing it. One broadcasting method that requires no special features from the network is for the source to simply send a distinct packet to each destination.

Not only is the method wasteful of bandwidth and slow, but it also requires the source to have a complete list of all destinations. This method is not desirable in practice, even though it is widely applicable.

An improvement is **multidestination routing**, in which 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 like a normal packet. Multidestination routing is like using separately addressed packets, except that when several packets must follow the same route, one of them pays full fare and the rest ride free.

The network bandwidth is therefore used more efficiently. However, this scheme still requires the source to know all the destinations, plus it is as much work for a router to determine where to send one multidestination packet as it is for multiple distinct packets.

MULTICAST ROUTING:

Sending a message to such a group is called **multicasting**, and the routing algorithm used is called **multicast routing**. All multicasting schemes require some way to create and destroy groups and to identify which routers are members of a group. How these tasks are accomplished is not of concern to the routing algorithm.

For now, we will assume that each group is identified by a multicast address and that routers know the groups to which they belong.

Multicast routing schemes build on the broadcast routing schemes we have already studied, sending packets along spanning trees to deliver the packets to the members of the group while making efficient use of bandwidth. However, the best spanning tree to use depends on whether the group is dense, with receivers scattered over most of the network, or sparse, with much of the network not belonging to the group.

If the group is dense, broadcast is a good start because it efficiently gets the packet to all parts of the network. But broadcast will reach some routers that are not members of the group, which is wasteful.

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.

Each router can then construct its own pruned spanning tree for each sender to the group in question by constructing a sink tree for the sender as usual and then removing all links that do not connect group members to the sink node. **MOSPF** (**Multicast OSPF**) is an example of a link state protocol that works in this way.

ANYCAST ROUTING:

So far, we have covered delivery models in which a source sends to a single destination (called **unicast**), to all destinations (called broadcast), and to a group of destinations (called multicast). Another delivery model, called **anycast** is sometimes also useful. In anycast, a packet is delivered to the nearest member of a group. Schemes that find these paths are called **anycast routing**.

ROUTING FOR MOBILE HOSTS:

Millions of people use computers while on the go, from truly mobile situations with wireless devices in moving cars, to nomadic situations in which laptop computers are used in a series of different locations. We will use the term **mobile hosts** to mean either category, as distinct from stationary hosts that never move.

Increasingly, people want to stay connected wherever in the world they may be, as easily as if they were at home. These mobile hosts introduce a new complication: to route a packet to a mobile host, the network first has to find it.

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 hosts also has a permanent home address that can be used to determine its home location, analogous to the way the telephone number 1-212-5551212 indicates the United States (country code 1) and Manhattan (212).

The routing goal in systems with mobile hosts is to make it possible to send packets to mobile hosts using their fixed home addresses and have the packets efficiently reach them wherever they may be. The trick, of course, is to find them.

ROUTING IN AD HOC NETWORKS: We have now seen how to do routing when the hosts are mobile but the routers are fixed. An even more extreme case is one in which the routers themselves are mobile. Among the possibilities are emergency workers at an earthquake site, military vehicles on a battlefield, a fleet of ships at sea, or a gathering of people with laptop computers in an area lacking 802.11.

In all these cases, and others, each node communicates wirelessly and acts as both a host and a router. 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 the topology is suddenly tossed out the window. Nodes 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 barring failures, which are hopefully rare. With an ad hoc network, the topology may be changing all the time,

so the desirability and even the validity of paths can change spontaneously without warning. Needless to say, these circumstances make routing in ad hoc networks more challenging than routing in their fixed counterparts.

CONGESTION CONTROL ALGORITHMS:

Too many packets present in (a part of) the network causes packet delay and loss that degrades performance. This situation is called **congestion**. The network and transport layers share the responsibility for handling congestion.

Since congestion occurs within the network, it is the network layer that directly experiences it and must ultimately determine what to do with the excess packets. However, the most effective way to control congestion is to reduce the load that the transport layer is placing on the network. This requires the network and transport layers to work together.

Figure 3.8 depicts the onset of congestion. When the number of packets hosts send into the network is well within its carrying capacity, the number delivered is proportional to the number sent. If twice as many are sent, twice as many are delivered.

However, as the offered load approaches the carrying capacity, bursts of traffic occasionally fill up the buffers inside routers and some packets are lost. These lost packets consume some of the capacity, so the number of delivered packets falls below the ideal curve. The network is now congested.

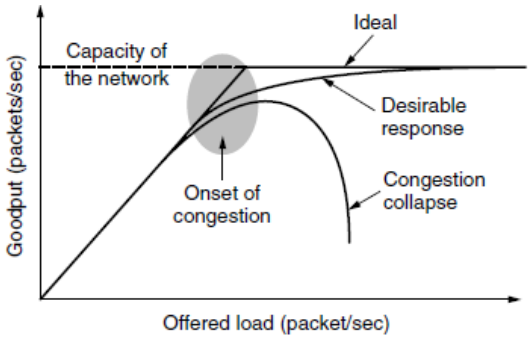


FIGURE 3.8: WITH TOO MUCH TRAFFIC, PERFORMANCE DROPS SHARPLY

Unless the network is well designed, it may experience a **congestion collapse**, in which performance plummets as the offered load increases beyond the capacity. This can happen because packets can be sufficiently delayed inside the network that they are no longer useful when they leave the network.

For example, in the early Internet, the time a packet spent waiting for a backlog of packets ahead of it to be sent over a slow 56-kbps link could reach the maximum time it was allowed to remain in the network. It then had to be thrown away.

A different failure mode occurs when senders retransmit packets that are greatly delayed, thinking that they have been lost. In this case, copies of the same packet will be delivered by the network, again wasting its capacity.

To capture these factors, the y-axis of Fig. 3.8 is given as **goodput**, which is the rate at which *useful* packets are delivered by the network. We would like to design networks that avoid congestion where possible and do not suffer from congestion collapse if they do become congested. Unfortunately, congestion cannot wholly be avoided.

APPROACHES TO CONGESTION CONTROL:

The presence of congestion means that the load is (temporarily) greater than the resources (in a part of the network) can handle. Two solutions come to mind: increase the resources or decrease the load. As shown in Fig. 3.9, these solutions are usually applied on different time scales to either prevent congestion or react to it once it has occurred.

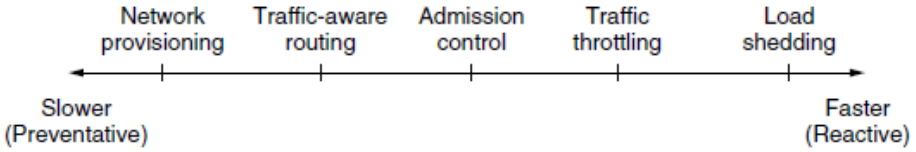


FIGURE 3.9: TIMESCALES OF APPROACHES TO CONGESTION CONTROL

The most basic way to avoid congestion is to build a network that is well matched to the traffic that it carries. If there is a low-bandwidth link on the path along which most traffic is directed, congestion is likely. Sometimes resources can be added dynamically when there is serious congestion.

For example, turning on spare routers or enabling lines that are normally used only as backups (to make the system fault tolerant) or purchasing bandwidth on the open market. More often, links and routers that are regularly heavily utilized are upgraded at the earliest opportunity. This is called **provisioning** and happens on a time scale of months, driven by long-term traffic trends.

To make the most of the existing network capacity, routes can be tailored to traffic patterns that change during the day as network user's wake and sleep in different time zones. For example, routes may be changed to shift traffic away from heavily used paths by changing the shortest path weights.

Some local radio stations have helicopters flying around their cities to report on road congestion to make it possible for their mobile listeners to route their packets (cars) around hotspots. This is called **traffic-aware routing**. Splitting traffic across multiple paths is also helpful.

However, sometimes it is not possible to increase capacity. The only way then to beat back the congestion is to decrease the load. In a virtual-circuit network, new connections can be refused if they would cause the network to become congested. This is called **admission control**.

TRAFFIC-AWARE ROUTING:

The first approach we will examine is traffic-aware routing. These schemes adapted to changes in topology, but not to changes in load; the goal in taking load into account when computing routes is to shift traffic away from hotspots that will be the first places in the network to experience congestion.

The most direct way to do this is to set the link weight to be a function of the (fixed) link bandwidth and propagation delay plus the (variable) measured load or average queuing delay. Least-weight paths will then favor paths that are more lightly loaded, all else being equal.

ADMISSION CONTROL:

One technique that is widely used in virtual-circuit networks to keep congestion at bay is **admission control**. The idea is simple: do not set up a new virtual circuit unless the network can carry the added traffic without becoming congested. Thus, attempts to set up a virtual circuit may fail. This is better than the alternative, as letting more people in when the network is busy just makes matters worse.

By analogy, in the telephone system, when a switch gets overloaded it practices admission control by not giving dial tones. The trick with this approach is working out when a new virtual circuit will lead to congestion. The task is straightforward in the telephone network because of the fixed bandwidth of calls (64 kbps for uncompressed audio).

However, virtual circuits in computer networks come in all shapes and sizes. Thus, the circuit must come with some characterization of its traffic if we are to apply admission control.

Traffic is often described in terms of its rate and shape. The problem of how to describe it in a simple yet meaningful way is difficult because traffic is typically bursty—the average rate is only half the story.

For example, traffic that varies while browsing the Web is more difficult to handle than a streaming movie with the same long-term throughput because the bursts of Web traffic are more likely to congest routers in the network.

A commonly used descriptor that captures this effect is the **leaky bucket** or **token bucket**. A leaky bucket has two parameters that bound the average rate

and the instantaneous burst size of traffic. Leaky buckets are widely used for quality of service.

TRAFFIC THROTTLING:

In the Internet and many other computer networks, senders adjust their transmissions to send as much traffic as the network can readily deliver. In this setting, the network aims to operate just before the onset of congestion.

When congestion is imminent, it must tell the senders to throttle back their transmissions and slow down. This feedback is business as usual rather than an exceptional situation. The term **congestion avoidance** is sometimes used to contrast this operating point with the one in which the network has become (overly) congested.

Let us now look at some approaches to throttling traffic that can be used in both datagram networks and virtual-circuit networks. Each approach must solve two problems. First, routers must determine when congestion is approaching, ideally before it has arrived. To do so, each router can continuously monitor the resources it is using.

Three possibilities are the utilization of the output links, the buffering of queued packets inside the router, and the number of packets that are lost due to insufficient buffering. Of these possibilities, the second one is the most useful.

Averages of utilization do not directly account for the burstiness of most traffic—a utilization of 50% may be low for smooth traffic and too high for highly variable traffic. Counts of packet losses come too late. Congestion has already set in by the time that packets are lost.

Choke Packets:

The most direct way to notify a sender of congestion is to tell it directly. In this approach, the router selects a congested packet and sends a **choke packet** back to the source host, giving it the destination found in the packet.

The original packet may be tagged (a header bit is turned on) so that it will not generate any more choke packets farther along the path and then forwarded in the usual way. To avoid increasing load on the network during a time of congestion, the router may only send choke packets at a low rate.

When the source host gets the choke packet, it is required to reduce the traffic sent to the specified destination, for example, by 50%. In a datagram network, simply picking packets at random when there is congestion is likely to

cause choke packets to be sent to fast senders, because they will have the most packets in the queue.

The feedback implicit in this protocol can help prevent congestion yet not throttle any sender unless it causes trouble. For the same reason, it is likely that multiple choke packets will be sent to a given host and destination.

The host should ignore these additional chokes for the fixed time interval until its reduction in traffic takes effect. After that period, further choke packets indicate that the network is still congested.

Explicit Congestion Notification:

Instead of generating additional packets to warn of congestion, a router can tag any packet it forwards (by setting a bit in the packet's header) to signal that it is experiencing congestion.

When the network delivers the packet, the destination can note that there is congestion and inform the sender when it sends a reply packet. The sender can then throttle its transmissions as before. This design is called **ECN** (**Explicit Congestion Notification shown in figure 3.10**) and is used in the Internet.

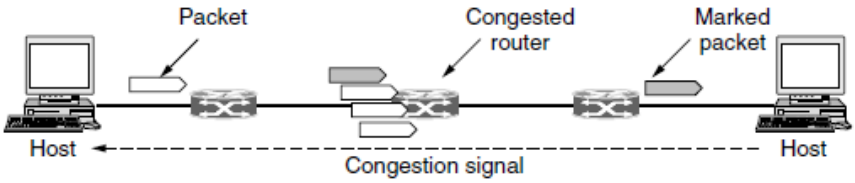


FIGURE 3.10: EXPLICIT CONGESTION NOTIFICATION

LOAD SHEDDING: It is a fancy way of saying that when routers are being inundated by packets that they cannot handle, they just throw them away. The term comes from the world of electrical power generation, where it refers to the practice of utilities intentionally blacking out certain areas to save the entire grid from collapsing on hot summer days when the demand for electricity greatly exceeds the supply.

QUALITY OF SERVICE:

An easy solution to provide good quality of service is to build a network with enough capacity for whatever traffic will be thrown at it. The name for this solution is **over provisioning**. The resulting network will carry application traffic without significant loss and, assuming a decent routing scheme, will deliver packets with low latency. Performance doesn't get any better than this.

To some extent, the telephone system is over provisioned because it is rare to pick up a telephone and not get a dial tone instantly. There is simply so much

capacity available that demand can almost always be met. The trouble with this solution is that it is expensive.

Four issues must be addressed to ensure quality of service:

1. What applications need from the network?
2. How to regulate the traffic that enters the network.
3. How to reserve resources at routers to guarantee performance.
4. Whether the network can safely accept more traffic.

No single technique deals efficiently with all these issues. Instead, a variety of techniques have been developed for use at the network (and transport) layer. Practical quality-of-service solutions combine multiple techniques. To this end, we will describe two versions of quality of service for the Internet called Integrated Services and Differentiated Services.

APPLICATION REQUIREMENTS:

A stream of packets from a source to a destination is called a **flow.** A flow might be all the packets of a connection in a connection-oriented network, or all the packets sent from one process to another process in a connectionless network. The needs of each flow can be characterized by four primary parameters: ***bandwidth***, ***delay***, ***jitter***, and ***loss***. Together, these determine the **QoS** (**Quality of Service**) the flow requires.

Several common applications and the stringency (meaning toughness/flexibility) of their network requirements are listed in Fig. 3.11. The applications differ in their bandwidth needs, with email, audio in all forms, and remote login not needing much, but file sharing and video in all forms needing a great deal.

More interesting are the delay requirements. File transfer applications, including email and video, are not delay sensitive. If all packets are delayed uniformly by a few seconds, no harm is done.

Interactive applications, such as Web surfing and remote login, are more delay sensitive. Real-time applications, such as telephony and videoconferencing, have strict delay requirements. If all the words in a telephone call are each delayed by too long, the users will find the connection unacceptable. On the other hand, playing audio or video files from a server does not require low delay.

The variation (i.e., standard deviation) in the delay or packet arrival times is called **jitter**. The first three applications in Fig. 3.11 are not sensitive to the packets arriving with irregular time intervals between them. Remote login is somewhat

sensitive to that, since updates on the screen will appear in little bursts if the connection suffers much jitter.

Video and especially audio are extremely sensitive to jitter. If a user is watching a video over the network and the frames are all delayed by exactly 2.000 seconds, no harm is done. But if the transmission time varies randomly between 1 and 2 seconds, the result will be terrible unless the application hides the jitter. For audio, a jitter of even a few milliseconds is clearly audible.

Application	Bandwidth	Delay	Jitter	Loss
Email	Low	Low	Low	Medium
File sharing	High	Low	Low	Medium
Web access	Medium	Medium	Low	Medium
Remote login	Low	Medium	Medium	Medium
Audio on demand	Low	Low	High	Low
Video on demand	High	Low	High	Low
Telephony	Low	High	High	Low
Videoconferencing	High	High	High	Low

FIGURE 3.11: STRINGENCY OF APPLICATIONS' QUALITY-OF-SERVICE REQUIREMENTS

To accommodate a variety of applications, networks may support different categories of QoS. An influential example comes from ATM networks. They support:

1. Constant bit rate (e.g., telephony).
2. Real-time variable bit rate (e.g., compressed videoconferencing).
3. Non-real-time variable bit rate (e.g., watching a movie on demand).
4. Available bit rate (e.g., file transfer).

These categories are also useful for other purposes and other networks.

TRAFFIC SHAPING: Before the network can make QoS guarantees, it must know what traffic is being guaranteed. In the telephone network, this characterization is simple. For example, a voice call (in uncompressed format) needs 64 kbps and consists of one 8-bit sample every 125 μsec.

However, traffic in data networks is **bursty**. It typically arrives at nonuniform rates as the traffic rate varies (e.g., videoconferencing with compression), users interact with applications (e.g., browsing a new Web page), and computers switch between tasks. Bursts of traffic are more difficult to handle than constant-rate traffic because they can fill buffers and cause packets to be lost.

Traffic shaping is a technique for regulating the average rate and burstiness of a flow of data that enters the network. The goal is to allow

applications to transmit a wide variety of traffic that suits their needs, including some bursts, yet have a simple and useful way to describe the possible traffic patterns to the network.

When a flow is set up, the user and the network (i.e., the customer and the provider) agree on a certain traffic pattern (i.e., shape) for that flow. In effect, the customer says to the provider "my transmission pattern will look like this; can you handle it?"

Sometimes this agreement is called an **SLA** (**Service Level Agreement**), especially when it is made over aggregate flows and long periods of time, such as all of the traffic for a given customer. As long as the customer fulfills her part of the bargain and only sends packets according to the agreed-on contract, the provider promises to deliver them all in a timely fashion.

Traffic shaping reduces congestion and thus helps the network live up to its promise. However, to make it work, there is also the issue of how the provider can tell if the customer is following the agreement and what to do if the customer is not. Packets in excess of the agreed pattern might be dropped by the network, or they might be marked as having lower priority. Monitoring a traffic flow is called **traffic policing**.

PACKET SCHEDULING:

Being able to regulate the shape of the offered traffic is a good start. However, to provide a performance guarantee, we must reserve sufficient resources along the route that the packets take through the network. To do this, we are assuming that the packets of a flow follow the same route. Spraying them over routers at random makes it hard to guarantee anything. As a consequence, something similar to a virtual circuit has to be set up from the source to the destination, and all the packets that belong to the flow must follow this route.

Algorithms that allocate router resources among the packets of a flow and between competing flows are called **packet scheduling algorithms**. Three different kinds of resources can potentially be reserved for different flows:

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

The first one, bandwidth, is the most obvious. If a flow requires 1 Mbps and the outgoing line has a capacity of 2 Mbps, trying to direct three flows through that line is not going to work. Thus, reserving bandwidth means not oversubscribing any output line.

A second resource that is often in short supply is buffer space. When a packet arrives, it is buffered inside the router until it can be transmitted on the chosen outgoing line. The purpose of the buffer is to absorb small bursts of traffic as the flows contend with each other.

If no buffer is available, the packet has to be discarded since there is no place to put it. For good quality of service, some buffers might be reserved for a specific flow so that flow does not have to compete for buffers with other flows. Up to some maximum value, there will always be a buffer available when the flow needs one.

Finally, CPU cycles may also be a scarce resource. It takes router CPU time to process a packet, so a router can process only a certain number of packets per second. While modern routers are able to process most packets quickly, some kinds of packets require greater CPU processing, such as the ICMP packets. Making sure that the CPU is not overloaded is needed to ensure timely processing of these packets.

INTERNETWORKING:

HOW NETWORKS DIFFER:

Networks can differ in many ways. Some of the differences, such as different modulation techniques or frame formats, are internal to the physical and data link layers. These differences will not concern us here. Instead, in Fig. 3.12 we list some of the differences that can be exposed to the network layer. It is papering over these differences that makes internetworking more difficult than operating within a single network.

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, the source needs to be able to address the destination.

What do we do if the source is on an Ethernet network and the destination is on a WiMAX network? Assuming we can even specify a WiMAX destination from an Ethernet network, packets would cross from a connectionless network to a connection-oriented one.

This may require that a new connection be set up on short notice, which injects a delay, and much overhead if the connection is not used for many more packets. Many specific differences may have to be accommodated as well. How do we multicast a packet to a group with some members on a network that does not support multicast?

The differing max packet sizes used by different networks can be a major nuisance, too. How do you pass an 8000-byte packet through a network whose

maximum size is 1500 bytes? If packets on a connection-oriented network transit a connectionless network, they may arrive in a different order than they were sent. That is something the sender likely did not expect, and it might come as an (unpleasant) surprise to the receiver as well.

Item	Some Possibilities
Service offered	Connectionless versus connection oriented
Addressing	Different sizes, flat or hierarchical
Broadcasting	Present or absent (also multicast)
Packet size	Every network has its own maximum
Ordering	Ordered and unordered delivery
Quality of service	Present or absent; many different kinds
Reliability	Different levels of loss
Security	Privacy rules, encryption, etc.
Parameters	Different timeouts, flow specifications, etc.
Accounting	By connect time, packet, byte, or not at all

FIGURE 3.12: SOME OF THE MANY WAYS NETWORKS CAN DIFFER.

How Networks Can Be Connected

There are two basic choices for connecting different networks: we can build devices that translate or convert packets from each kind of network into packets for each other network, or, like good computer scientists, we can try to solve the problem by adding a layer of indirection and building a common layer on top of the different networks. In either case, the devices are placed at the boundaries between networks.

Internetworking has been very successful at building large networks, but it only works when there is a common network layer. There have, in fact, been many network protocols over time. Getting everybody to agree on a single format is difficult when companies perceive it to their commercial advantage to have a proprietary format that they control.

A router that can handle multiple network protocols is called a **multiprotocol router**. It must either translate the protocols, or leave connection for a higher protocol layer. Neither approach is entirely satisfactory. Connection at a higher layer, say, by using TCP, requires that all the networks implement TCP (which may not be the case). Then, it limits usage across the networks to applications that use TCP (which does not include many real-time applications).

TUNNELING:

Handling the general case of making two different networks interwork is exceedingly difficult. However, there is a common special case that is manageable even for different network protocols. This case is where the source and destination

hosts are on the same type of network, but there is a different network in between. As an example, think of an international bank with an IPv6 network in Paris, an IPv6 network in London and connectivity between the offices via the IPv4 Internet. This situation is shown in Fig. 3.13.

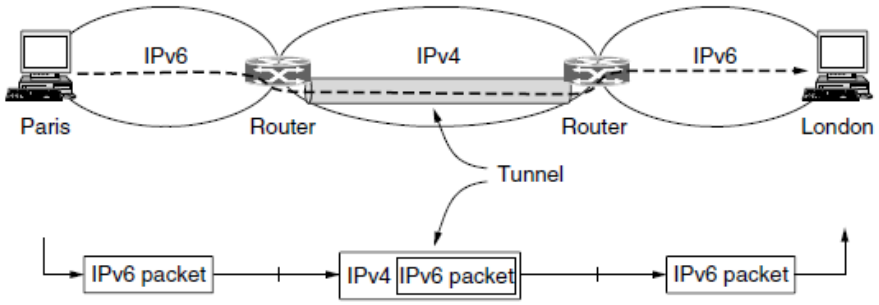


FIGURE 3.13: TUNNELING A PACKET FROM PARIS TO LONDON

The solution to this problem is a technique called **tunneling**. To send an IP packet to a host in the London office, a host in the Paris office constructs the packet containing an IPv6 address in London, and sends it to the multiprotocol router that connects the Paris IPv6 network to the IPv4 Internet.

When this router gets the IPv6 packet, it encapsulates the packet with an IPv4 header addressed to the IPv4 side of the multiprotocol router that connects to the London IPv6 network.

That is, the router puts a (IPv6) packet inside a (IPv4) packet. When this wrapped packet arrives, the London router removes the original IPv6 packet and sends it onward to the destination host. The path through the IPv4 Internet can be seen as a big tunnel extending from one multiprotocol router to the other.

The IPv6 packet just travels from one end of the tunnel to the other, snug in its nice box. It does not have to worry about dealing with IPv4 at all. Neither do the hosts in Paris or London. Only the multiprotocol routers have to understand both IPv4 and IPv6 packets.

In effect, the entire trip from one multiprotocol router to the other is like a hop over a single link. Tunneling is widely used to connect isolated hosts and networks using other networks.

INTERNETWORK ROUTING:

Routing through an internet poses the same basic problem as routing within a single network, but with some added complications. To start, the networks may internally use different routing algorithms. For example, one network may use link state routing and another distance vector routing. Since link state algorithms need to know the topology but distance vector algorithms do not, this difference alone would make it unclear how to find the shortest paths across the internet.

Networks run by different operators lead to bigger problems. First, the operators may have different ideas about what is a good path through the network. One operator may want the route with the least delay, while another may want the most inexpensive route. This will lead the operators to use different quantities to set the shortest-path costs.

Finally, the internet may be much larger than any of the networks that comprise it. It may therefore require routing algorithms that scale well by using a hierarchy, even if none of the individual networks need to use a hierarchy.

All of these considerations lead to a two-level routing algorithm. Within each network, an **intradomain** or **interior gateway protocol** is used for routing. (''Gateway'' is an older term for ''router.'') It might be a link state protocol of the Kind.

Across the networks that make up the internet, an **interdomain** or **exterior gateway protocol** is used. The networks may all use different intradomain protocols, but they must use the same interdomain protocol.

*In the Internet, the interdomain routing protocol is called **BGP** (**Border Gateway Protocol**).*

There is one more important term to introduce. Since each network is operated independently of all the others, it is often referred to as an **AS** (**Autonomous System**). A good mental model for an AS is an ISP network. In fact, an ISP network may be comprised of more than one AS, if it is managed, or, has been acquired, as multiple networks. But the difference is usually not significant.

PACKET FRAGMENTATION: Each network or link imposes some maximum size on its packets. These limits have various causes, among them:

1. Hardware (e.g., the size of an Ethernet frame).
2. Operating system (e.g., all buffers are 512 bytes).
3. Protocols (e.g., the number of bits in the packet length field).
4. Compliance with some (inter)national standard.
5. Desire to reduce error-induced retransmissions to some level.
6. Desire to prevent one packet from occupying the channel too long.

The result of all these factors is that the network designers are not free to choose any old maximum packet size they wish. Maximum payloads for some common technologies are 1500 bytes for Ethernet and 2272 bytes for 802.11. IP is more generous, allows for packets as big as 65,515 bytes.

Hosts usually prefer to transmit large packets because this reduces packet overheads such as bandwidth wasted on header bytes. An obvious internetworking problem appears when a large packet wants to travel through a network whose maximum packet size is too small. This nuisance has been a persistent issue, and solutions to it have evolved along with much experience gained on the Internet.

One solution is to make sure the problem does not occur in the first place. However, this is easier said than done. A source does not usually know the path a packet will take through the network to a destination, so it certainly does not know how small packets must be to get there. This packet size is called the **Path MTU** (**Path Maximum Transmission Unit**).

The alternative solution to the problem is to allow routers to break up packets into **fragments**, sending each fragment as a separate network layer packet. However, as every parent of a small child knows, converting a large object into small fragments is considerably easier than the reverse process.

THE NETWORK LAYER IN THE INTERNET

THE IP VERSION 4 PROTOCOL:

An appropriate place to start our study of the network layer in the Internet is with the format of the IP datagrams themselves. An IPv4 datagram consists of a header part and a body or payload part. The header has a 20-byte fixed part and a variable-length optional part. The header format is shown in Fig. 3.14. The bits are transmitted from left to right and top to bottom, with the high-order bit of the *Version* field going first. (This is a ‘‘big-endian’’ network byte order.

On little-endian machines, such as Intel x86 computers, a software conversion is required on both transmission and reception.) In retrospect, little-endian would have been a better choice, but at the time IP was designed, no one knew it would come to dominate computing.

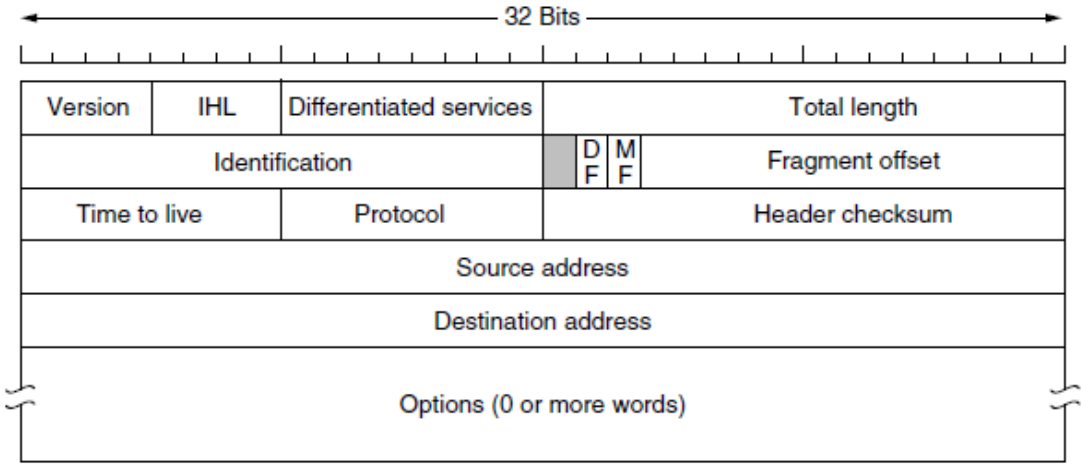


FIGURE 3.14: THE IPV4 (INTERNET PROTOCOL) HEADER

The *Version* field keeps track of which version of the protocol the datagram belongs to.

Since the header length is not constant, a field in the header, *IHL*, is provided to tell how long the header is, in 32-bit words. The minimum value is 5, which applies when no options are present. The maximum value of this 4-bit field is 15, which limits the header to 60 bytes, and thus the *Options* field to 40 bytes.

The *Differentiated services* field is one of the few fields that have changed its meaning (slightly) over the years. Originally, it was called the *Type of service* field. Various combinations of reliability and speed are possible. For digitized voice, fast delivery beats accurate delivery.

For file transfer, error-free transmission is more important than fast transmission. The *Type of service* field provided 3 bits to signal priority and 3 bits to signal whether a host cared more about delay, throughput, or reliability.

The *Total length* includes everything in the datagram—both header and data. The maximum length is 65,535 bytes. At present, this upper limit is tolerable, but with future networks, larger datagrams may be needed.

The *Identification* field is needed to allow the destination host to determine which packet a newly arrived fragment belongs to. All the fragments of a packet contain the same *Identification* value.

DF stands for Don't Fragment. It is an order to the routers not to fragment the packet. Originally, it was intended to support hosts incapable of putting the pieces back together again.

MF stands for More Fragments. All fragments except the last one have this bit set. It is needed to know when all fragments of a datagram have arrived.

The *Fragment offset* tells where in the current packet this fragment belongs. All fragments except the last one in a datagram must be a multiple of 8 bytes, the elementary fragment unit. Since 13 bits are provided, there is a maximum of 8192 fragments per datagram, supporting a maximum packet length up to the limit of the *Total length* field. Working together, the *Identification*, *MF*, and *Fragment offset* fields are used to implement fragmentation.

The *TtL (Time to live)* field is a counter used to limit packet lifetimes. It was originally supposed to count time in seconds, allowing a maximum lifetime of 255 sec.

When the network layer has assembled a complete packet, it needs to know what to do with it. The *Protocol* field tells it which transport process to give the packet to. TCP is one possibility, but so are UDP and some others.

Since the header carries vital information such as addresses, it rates its own checksum for protection, the *Header checksum*. The algorithm is to add up all the 16-bit halfwords of the header as they arrive, using one's complement arithmetic, and then take the one's complement of the result. For purposes of this algorithm, the *Header checksum* is assumed to be zero upon arrival. Such a checksum is useful for detecting errors while the packet travels through the network.

The *Source address* and *Destination address* indicate the IP address of the source and destination network interfaces.

The *Options* field was designed to provide an escape to allow subsequent versions of the protocol to include information not present in the original design, to permit experimenters to try out new ideas, and to avoid allocating header bits to information that is rarely needed. The options are of variable length. The *Options* field is padded out to a multiple of 4 bytes. Originally, the five options listed in Fig. 3.15.

Option	Description
Security	Specifies how secret the datagram is
Strict source routing	Gives the complete path to be followed
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

FIGURE 3.15: SOME OF THE IP OPTIONS

IPV4 ADDRESSES:

The identifier used in the IP layer of the TCP/IP protocol suite to identify the connection of each device to the Internet is called the Internet address or IP address. An IPv4 address is a 32-bit address that uniquely and universally defines the connection of a host or a router to the Internet. The IP address is the address of the connection, not the host or the router, because if the device is moved to another network, the IP address may be changed.

IPv4 addresses are unique in the sense that each address defines one, and only one, connection to the Internet. If a device has two connections to the Internet, via two networks, it has two IPv4 addresses. IPv4 addresses are universal in the sense that the addressing system must be accepted by any host that wants to be connected to the Internet.

Address Space

A protocol like IPv4 that defines addresses has an address space. An **address space** is the total number of addresses used by the protocol. If a protocol uses *b* bits to define an address, the address space is 2^b because each bit can have two different values (0 or 1). IPv4 uses 32-bit addresses, which means that the

address space is 232 or 4,294,967,296 (more than four billion). If there were no restrictions, more than 4 billion devices could be connected to the Internet.

Notation

There are three common notations to show an IPv4 address: binary notation (base 2), dotted-decimal notation (base 256), and hexadecimal notation (base 16). In *binary notation,* an IPv4 address is displayed as 32 bits. To make the address more readable, one or more spaces are usually inserted between each octet (8 bits). Each octet is often referred to as a byte. To make the IPv4 address more compact and easier to read, it is usually written in decimal form with a decimal point (dot) separating the bytes.

This format is referred to as *dotted-decimal notation*. Note that because each byte (octet) is only 8 bits, each number in the dotted-decimal notation is between 0 and 255. We sometimes see an IPv4 address in hexadecimal notation. Each hexadecimal digit is equivalent to four bits. This means that a 32-bit address has 8 hexadecimal digits. This notation is often used in network programming. Figure 3.16 shows an IP address in the three discussed notations.

HIERARCHY IN ADDRESSING: A 32-bit IPv4 address is also hierarchical, but divided only into two parts. The first part of the address, called the *prefix*, defines the network; the second part of the address, called the *suffix*, defines the node (connection of a device to the Internet).

Figure 3.17 shows the prefix and suffix of a 32-bit IPv4 address. The prefix length is *n* bits and the suffix length is (32 − *n*) bits.

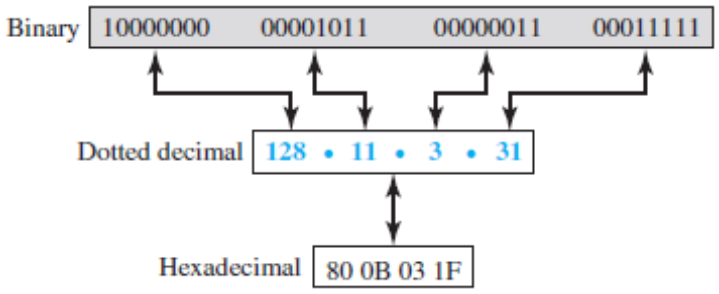


FIGURE 3.16: THREE DIFFERENT NOTATIONS IN IPV4 ADDRESSING

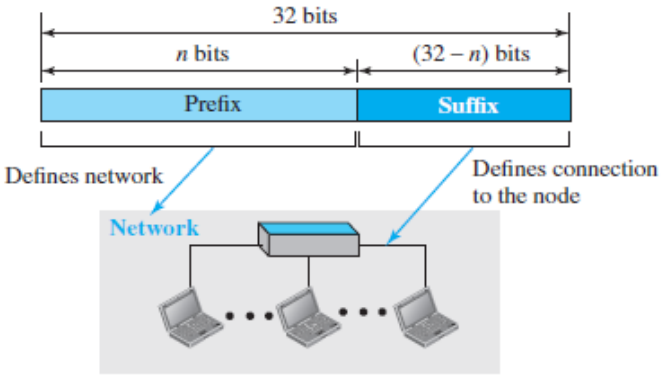


FIGURE 3.17: HIERARCHY IN ADDRESSING

A prefix can be fixed length or variable length. The network identifier in the IPv4 was first designed as a fixed-length prefix. This scheme, which is now obsolete, is referred to as classful addressing. The new scheme, which is referred to as classless addressing, uses a variable-length network prefix. First, we briefly discuss Classful addressing; then we concentrate on classless addressing.

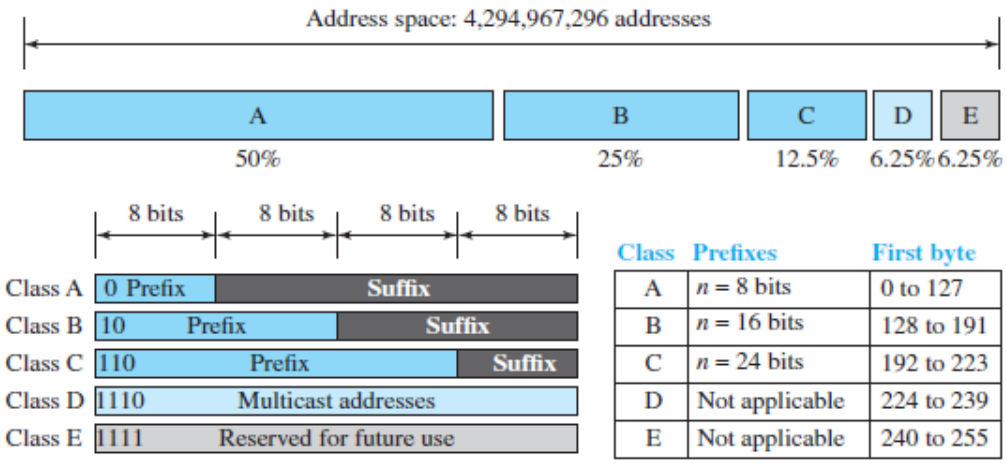
Classful Addressing:

When the Internet started, an IPv4 address was designed with a fixed-length prefix, but to accommodate both small and large networks, three fixed-length prefixes were designed instead of one (n = 8, n = 16, and n = 24). The whole address space was divided into five classes (class A, B, C, D, and E), as shown in Figure 3.18. This scheme is referred to as **classful addressing.**

In class A, the network length is 8 bits, but since the first bit, which is 0, defines the class, we can have only seven bits as the network identifier. This means there are only 27 = 128 networks in the world that can have a class A address.

In class B, the network length is 16 bits, but since the first two bits, which are (10)2, define the class, we can have only 14 bits as the network identifier. This means there are only 214 = 16,384 networks in the world that can have a class B address.

All addresses that start with (110)2 belong to class C. In class C, the network length is 24 bits, but since three bits define the class, we can have only 21 bits as the network identifier. This means there are 221 = 2,097,152 networks in the world that can have a class C address.



Class	Prefixes	First byte
A	$n = 8$ bits	0 to 127
B	$n = 16$ bits	128 to 191
C	$n = 24$ bits	192 to 223
D	Not applicable	224 to 239
E	Not applicable	240 to 255

FIGURE 3.18: OCCUPATION OF THE ADDRESS SPACE IN CLASSFUL ADDRESSING

Class D is not divided into prefix and suffix. It is used for multicast addresses. All addresses that start with 1111 in binary belong to class E. As in Class D, Class E is not divided into prefix and suffix and is used as reserve.

Advantage of Classful Addressing:

Although classful addressing had several problems and became obsolete, it had one advantage: Given an address, we can easily find the class of the address and, since the prefix length for each class is fixed, we can find the prefix length immediately. In other words, the prefix length in classful addressing is inherent in the address; no extra information is needed to extract the prefix and the suffix.

Address Depletion: The reason that classful addressing has become obsolete is address depletion. Since the addresses were not distributed properly, the Internet was faced with the problem of the addresses being rapidly used up, resulting in no more addresses available for organizations and individuals that needed to be connected to the Internet.

Subnetting and Supernetting: To alleviate address depletion, two strategies were proposed and, to some extent, implemented: subnetting and Supernetting. In subnetting, a class A or class B block is divided into several subnets.

Each subnet has a larger prefix length than the original network. While subnetting was devised to divide a large block into smaller ones, Supernetting was devised to combine several class C blocks into a larger block to be attractive to organizations that need more than the 256 addresses available in a class C block. This idea did not work either because it makes the routing of packets more difficult.

Classless Addressing:

Subnetting and Supernetting in classful addressing did not really solve the address depletion problem. With the growth of the Internet, it was clear that a larger address space was needed as a long-term solution. The larger address space, however, requires that the length of IP addresses also be increased, which means the format of the IP packets needs to be changed.

Although the long-range solution has already been devised and is called IPv6, a short-term solution was also devised to use the same address space but to change the distribution of addresses to provide a fair share to each organization. The short-term solution still uses IPv4 addresses, but it is called *classless addressing*. In other words, the class privilege was removed from the distribution to compensate for the address depletion.

In classless addressing, the whole address space is divided into variable length blocks. The prefix in an address defines the block (network); the suffix defines the node (device). Theoretically, we can have a block of 20, 21, 22, . . . , 232 addresses. One of the restrictions, as we discuss later, is that the number of addresses in a block needs to be a power of 2. An organization can be granted one block of addresses. Figure 3.19 shows the division of the whole address space into nonoverlapping blocks.

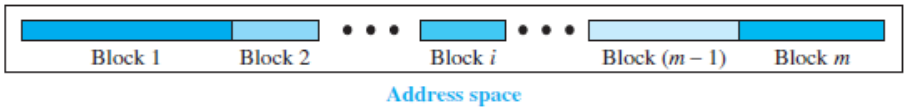


FIGURE 3.19: VARIABLE-LENGTH BLOCKS IN CLASSLESS ADDRESSING

Unlike classful addressing, the prefix length in classless addressing is variable. We can have a prefix length that ranges from 0 to 32. The size of the network is inversely proportional to the length of the prefix. A small prefix means a larger network; a large prefix means a smaller network.

We need to emphasize that the idea of classless addressing can be easily applied to classful addressing. An address in class A can be thought of as a classless address in which the prefix length is 8. An address in class B can be thought of as a classless address in which the prefix is 16, and so on. In other words, classful addressing is a special case of classless addressing.

Prefix Length: Slash Notation:

The first question that we need to answer in classless addressing is how to find the prefix length if an address is given. Since the prefix length is not inherent in the address, we need to separately give the length of the prefix. In this case, the prefix length, *n*, is added to the address, separated by a slash. The notation is informally referred to as *slash notation* and formally as ***classless interdomain***

routing or ***CIDR*** (pronounced cider) strategy. An address in classless addressing can then be represented as shown in Figure 3.20.

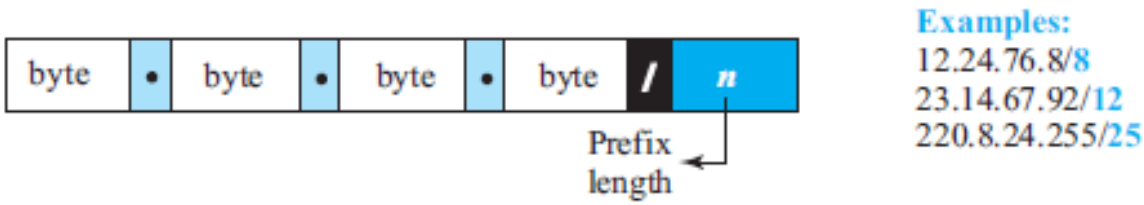


FIGURE 3.20: SLASH NOTATION (CIDR)

Extracting Information from an Address:

Given any address in the block, we normally like to know three pieces of information about the block to which the address belongs: the number of addresses, the first address in the block, and the last address. Since the value of prefix length, *n*, is given, we can easily find these three pieces of information, as shown in Figure 3.21.

1. The number of addresses in the block is found as $N = 2^{32-n}$.
2. To find the first address, we keep the *n* leftmost bits and set the (32 − *n*) rightmost bits all to 0s.
3. To find the last address, we keep the *n* leftmost bits and set the (32 − *n*) rightmost bits all to 1s.

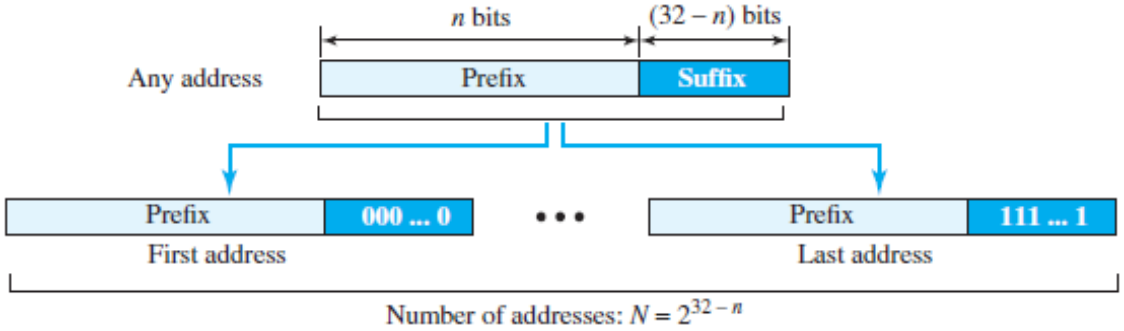


FIGURE 3.21: INFORMATION EXTRACTION IN CLASSLESS ADDRESSING

Example:

A classless address is given as 167.199.170.82/**27**. We can find the above three pieces of information as follows. The number of addresses in the network is $2^{32-n} = 25 = 32$ addresses.

The first address can be found by keeping the first 27 bits and changing the rest of the bits to 0s.

Address: 167.199.170.82/**27** **10100111 11000111 10101010 01010010**

First address: 167.199.170.64/**27** **10100111 11000111 10101010 01000000**

The last address can be found by keeping the first 27 bits and changing the rest of the bits to 1s.

Address: 167.199.170.82/**27** **10100111 11000111 10101010 01011111**

Last address: 167.199.170.95/**27** **10100111 11000111 10101010 01011111**

IP VERSION 6:

IP has been in heavy use for decades. It has worked extremely well, as demonstrated by the exponential growth of the Internet. Unfortunately, IP has become a victim of its own popularity: it is close to running out of addresses. Even with CIDR and NAT using addresses more sparingly, the last IPv4 addresses are expected to be assigned by ICANN before the end of 2012.

IPv6 (**IP version 6**) is a replacement design that does just that. It uses 128-bit addresses; a shortage of these addresses is not likely any time in the foreseeable future. However, IPv6 has proved very difficult to deploy. It is a different network layer protocol that does not really interwork with IPv4, despite many similarities. Also, companies and users are not really sure why they should want IPv6 in any case.

In 1990 IETF started work on a new version of IP, one that 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 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).
5. Pay more attention to the type of service, particularly for real-time data.
6. Aid multicasting by allowing scopes to be specified.
7. Make it possible for a host to roam without changing its address.
8. Allow the protocol to evolve in the future.

9. Permit the old and new protocols to coexist for years.

The design of IPv6 presented a major opportunity to improve all of the features in IPv4 that fall short of what is now wanted. One proposal was to run TCP over CLNP, the network layer protocol designed for OSI. With its 160-bit addresses, CLNP would have provided enough address space forever.

IPv6 meets IETF's 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, with small modifications being required to deal with longer addresses.

The main features of IPv6 are discussed below.

- First and foremost, IPv6 has longer addresses than IPv4. They are 128 bits long, which solves the problem that IPv6 set out to solve: providing an effectively unlimited supply of Internet addresses.
- The second major improvement of IPv6 is the simplification of the header. It contains only seven fields (versus 13 in IPv4). This change allows routers to process packets faster and thus improves throughput and delay.
- The third major improvement is better support for options. This change was essential with the new header because fields that previously were required are now optional (because they are not used so often).
    - In addition, the way options are represented is different, making it simple for routers to skip over options not intended for them. This feature speeds up packet processing time.
- A fourth area in which IPv6 represents a big advance is in security.
- Finally, more attention has been paid to quality of service.

The Main IPv6 Header:

The IPv6 header is shown in Fig. 3.22. The *Version* field is always 6 for IPv6 (and 4 for IPv4). During the transition period from IPv4, which has already taken more than a decade, routers will be able to examine this field to tell what kind of packet they have.

As an aside, making this test wastes a few instructions in the critical path, given that the data link header usually indicates the network protocol for demultiplexing, so some routers may skip the check.

The *Differentiated services* field (originally called *Traffic class*) is used to distinguish the class of service for packets with different real-time delivery requirements.

The *Flow label* field provides a way for a source and destination to mark groups of packets that have the same requirements and should be treated in the same way by the network, forming a pseudo connection.

The *Payload length* field tells how many bytes follow the 40-byte header of Fig. 3.22. The name was changed from the IPv4 *Total length* field because the meaning was changed slightly: the 40 header bytes are no longer counted as part of the length (as they used to be). This change means the payload can now be 65,535 bytes instead of a mere 65,515 bytes.

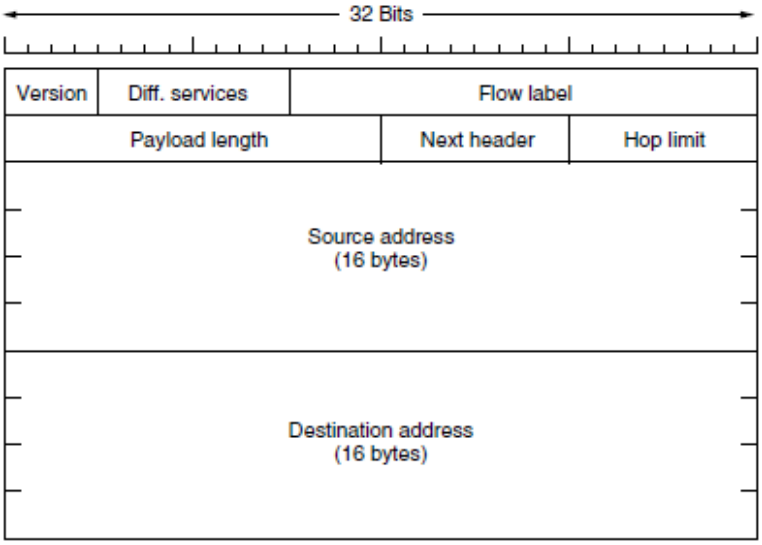


FIGURE 3.22: THE IPV6 FIXED HEADER (REQUIRED)

The *Next header* field tells which transport protocol handler (e.g., TCP, UDP) to pass the packet to.

The *Hop limit* field is used to keep packets from living forever. It is, in practice, the same as the *Time to live* field in IPv4, namely, a field that is decremented on each hop. In

Next come the *Source address* and *Destination address* fields. A new notation has been devised for writing 16-byte addresses. They are written as eight groups of four hexadecimal digits with colons between the groups, like this:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Since many addresses will have many zeros inside them, three optimizations have been authorized. First, leading zeros within a group can be omitted, so 0123 can be written as 123. Second, one or more groups of 16 zero bits can be replaced by a pair of colons. Thus, the above address now becomes

8000::123:4567:89AB:CDEF

INTERNET CONTROL PROTOCOLS:

In addition to IP, which is used for data transfer, the Internet has several companion control protocols that are used in the network layer. They include ICMP, ARP, and DHCP.

IMCP—The Internet Control Message Protocol:

The operation of the Internet is monitored closely by the routers. When something unexpected occurs during packet processing at a router, the event is reported to the sender by the **ICMP** (**Internet Control Message Protocol**). ICMP is also used to test the Internet. About a dozen types of ICMP messages are defined. Each ICMP message type is carried encapsulated in an IP packet. The most important ones are listed in Fig. 3.23.

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 and echo reply	Check if a machine is alive
Timestamp request/reply	Same as Echo, but with timestamp
Router advertisement/solicitation	Find a nearby router

FIGURE 3.23: THE PRINCIPAL ICMP MESSAGE TYPES

The DESTINATION UNREACHABLE message is used when the 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 *TtL (Time to live)* counter has reached zero. This event is a symptom that packets are looping, or that the counter values are being set too low.

The PARAMETER PROBLEM message indicates that an illegal value has been detected in a header field. This problem indicates a bug in the sending host’s IP software or possibly in the software of a router transited.

The SOURCE QUENCH message was long ago used to throttle hosts that were sending too many packets. When a host received this message, it was expected to slow down.

The REDIRECT message is used when a router notices that a packet seems to be routed incorrectly. It is used by the router to tell the sending host to update to a better route.

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 can be used to measure network performance.

OSPF—AN INTERIOR GATEWAY ROUTING PROTOCOL:

The Internet is made up of a large number of independent networks or **ASes** (**Autonomous Systems**) that are operated by different organizations, usually a company, university, or ISP. Inside of its own network, an organization can use its own algorithm for internal routing, or **intradomain routing**, as it is more commonly known. Nevertheless, there are only a handful of standard protocols that are popular.

An intradomain routing protocol is also called an **interior gateway protocol**. We will study the problem of routing between independently operated networks, or **interdomain routing**. For that case, all networks must use the same interdomain routing protocol or **exterior gateway protocol**. The protocol that is used in the Internet is BGP (Border Gateway Protocol).

Early intradomain routing protocols used a distance vector design, based on the distributed Bellman-Ford algorithm inherited from the ARPANET. It works well in small systems, but less well as networks get larger. It also suffers from the count-to-infinity problem and generally slow convergence.

The ARPANET switched over to a link state protocol in May 1979 because of these problems, and in 1988 IETF began work on a link state protocol for intradomain routing. That protocol, called **OSPF** (**Open Shortest Path First**), became a standard in 1990. It drew on a protocol called **IS-IS** (**Intermediate-System to Intermediate-System**), which became an ISO standard.

Given the long experience with other routing protocols, the group designing OSPF had a long list of requirements that had to be met. First, the algorithm had to be published in the open literature, hence the ‘‘O’’ in OSPF.

Second, the new protocol had to support a variety of distance metrics, including physical distance, delay, and so on. Third, it had to be a dynamic algorithm, one that adapted to changes in the topology automatically and quickly.

Fourth, and new for OSPF, it had to support routing based on type of service. The new protocol had to be able to route real-time traffic one way and other traffic a different way. At the time, IP had a *Type of service* field, but no existing routing protocol used it. This field was included in OSPF but still nobody used it, and it was eventually removed.

Fifth, and related to the above, OSPF had to do load balancing, splitting the load over multiple lines. Most previous protocols sent all packets over a single best route, even if there were two routes that were equally good. The other route was not used at all. In many cases, splitting the load over multiple routes gives better performance.

Sixth, support for hierarchical systems was needed. By 1988, some networks had grown so large that no router could be expected to know the entire topology. OSPF had to be designed so that no router would have to.

OSPF supports both point-to-point links (e.g., SONET) and broadcast networks (e.g., most LANs). Actually, it is able to support networks with multiple routers, each of which can communicate directly with the others (called **multi-access networks**) even if they do not have broadcast capability. Earlier protocols did not handle this case well.

OSPF works by exchanging information between adjacent routers, which is not the same as between neighboring routers. In particular, it is inefficient to have every router on a LAN talk to every other router on the LAN. To avoid this situation, one router is elected as the **designated router**. It is said to be **adjacent** to all the other routers on its LAN, and exchanges information with them.

In effect, it is acting as the single node that represents the LAN. Neighboring routers that are not adjacent do not exchange information with each other. A backup designated router is always kept up to date to ease the transition should the primary designated router crash and need to be replaced immediately.

During normal operation, each router periodically floods LINK STATE UPDATE messages to each of its adjacent routers. These messages gives its state and provide the costs used in the topological database. The flooding messages are acknowledged, to make them reliable.

Each message has a sequence number, so a router can see whether an incoming LINK STATE UPDATE is older or newer than what it currently has. Routers also send these messages when a link goes up or down or its cost changes.

DATABASE DESCRIPTION messages give the sequence numbers of all the link state entries currently held by the sender. By comparing its own values with those of the sender, the receiver can determine who has the most recent values. These messages are used when a link is brought up.

All these messages are sent directly in IP packets. The five kinds of messages are summarized in Fig. 3.24.

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

FIGURE 3.24: THE FIVE TYPES OF OSPF MESSAGES

BGP—THE EXTERIOR GATEWAY ROUTING PROTOCOL:

Within a single AS, OSPF and IS-IS are the protocols that are commonly used. Between ASes, a different protocol, called **BGP** (**Border Gateway Protocol**), is used. A different protocol is needed because the goals of an intradomain protocol and an interdomain protocol are not the same. All an intradomain protocol has to do is move packets as efficiently as possible from the source to the destination.

BGP is a form of distance vector protocol, but it is quite unlike intradomain distance vector protocols such as RIP. We have already seen that policy, instead of minimum distance, is used to pick which routes to use. Another large difference is that instead of maintaining just the cost of the route to each destination, each BGP router keeps track of the path used. This approach is called a **path vector protocol**.

The path consists of the next hop router (which may be on the other side of the ISP, not adjacent) and the sequence of ASes, or **AS path**, that the route has followed (given in reverse order). Finally, pairs of BGP routers communicate with each other by establishing TCP connections. Operating this way provides reliable communication and also hides all the details of the network being passed through.

An example of how BGP routes are advertised is shown in Fig. 3.25. There are three ASes and the middle one is providing transit to the left and right ISPs. A route advertisement to prefix *C* starts in *AS3*. When it is propagated across the link to *R2c* at the top of the figure, it has the AS path of simply *AS3* and the next hop router of *R3a*.

At the bottom, it has the same AS path but a different next hop because it came across a different link. This advertisement continues to propagate and crosses the boundary into *AS1*. At router *R1a*, at the top of the figure, the AS path is *AS2*, *AS3* and the next hop is *R2a*.

Carrying the complete path with the route makes it easy for the receiving router to detect and break routing loops. The rule is that each router that sends a

route outside of the AS prepends its own AS number to the route. (This is why the list is in reverse order.)

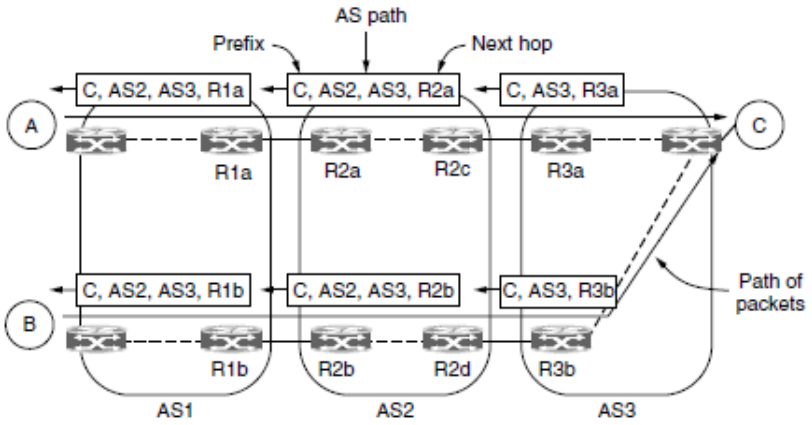


FIGURE 3.25: PROPAGATION OF BGP ROUTE ADVERTISEMENTS

When a router receives a route, it checks to see if its own AS number is already in the AS path. If it is, a loop has been detected and the advertisement is discarded.

INTERNET PROTOCOL (IP):

The network layer in version 4 can be thought of as one main protocol and three auxiliary ones. The main protocol, Internet Protocol version 4 (IPv4), is responsible for packetizing, forwarding, and delivery of a packet at the network layer.

The Internet Control Message Protocol version 4 (ICMPv4) helps IPv4 to handle some errors that may occur in the network-layer delivery. The Internet Group Management Protocol (IGMP) is used to help IPv4 in multicasting. The Address Resolution Protocol (ARP) is used to glue the network and data-link layers in mapping network-layer addresses to link-layer addresses. Figure 3.26 shows the positions of these four protocols in the TCP/IP protocol suite.

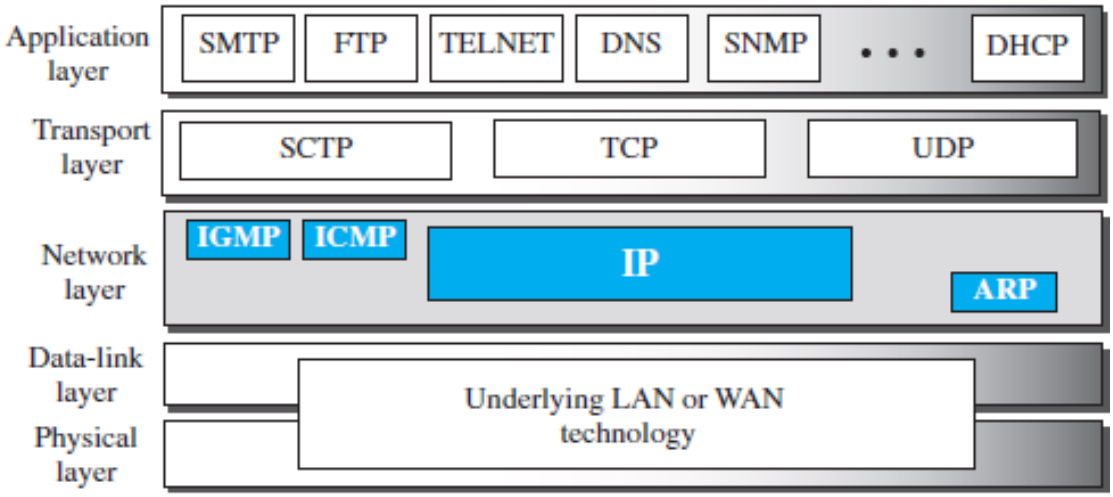


FIGURE 3.26: POSITION OF IP & OTHER NETWORK-LAYER PROTOCOLS IN TCP/IP PROTOCOL SUITE

IPv4 is an unreliable datagram protocol—a best-effort delivery service. The term *best-effort* means that IPv4 packets can be corrupted, be lost, arrive out of order, or be delayed, and may create congestion for the network. If reliability is important, IPv4 must be paired with a reliable transport-layer protocol such as TCP.

IPv4 is also a connectionless protocol that uses the datagram approach. This means that each datagram is handled independently, and each datagram can follow a different route to the destination. This implies that datagrams sent by the same source to the same destination could arrive out of order. Again, IPv4 relies on a higher-level protocol to take care of all these problems.

DATAGRAM FORMAT:

Packets used by the IP are called *datagrams*. Figure 3.27 shows the IPv4 datagram format. A datagram is a variable-length packet consisting of two parts: header and payload (data). The header is 20 to 60 bytes in length and contains information essential to routing and delivery. It is customary in TCP/IP to show the header in 4-byte sections.

- ***Version Number.*** The 4-bit version number (VER) field defines the version of the IPv4 protocol, which, obviously, has the value of 4.
- ***Header Length*.** The 4-bit header length (HLEN) field defines the total length of the datagram header in 4-byte words. The IPv4 datagram has a variable-length header.
- ***Service Type*.** In the original design of the IP header, this field was referred to as type of service (TOS), which defined how the datagram should be handled.

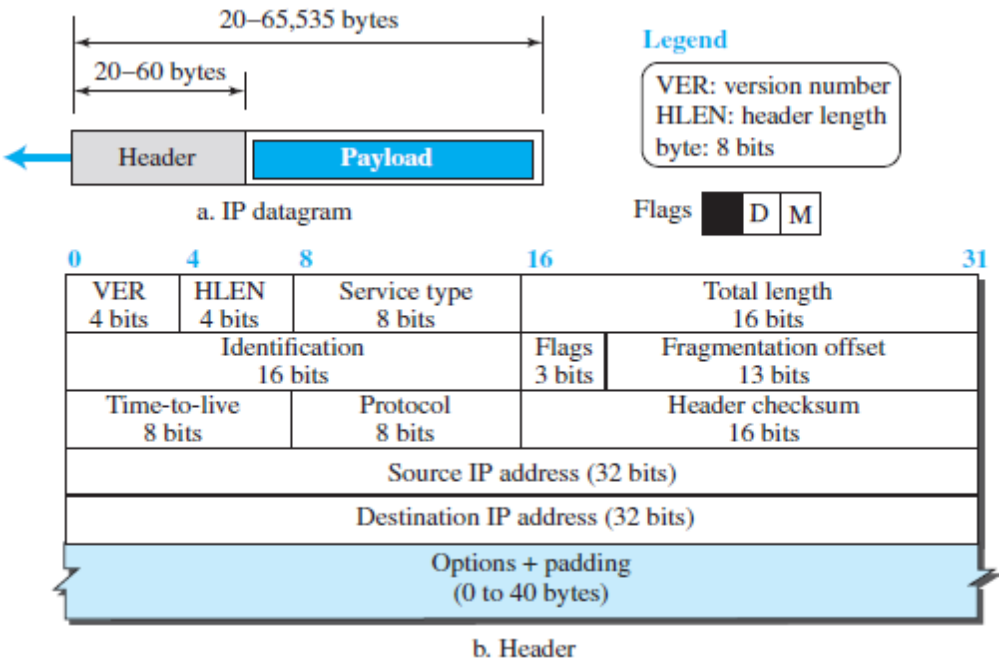


FIGURE 3.27: IP DATAGRAM

- ***Total Length*.** This 16-bit field defines the total length (header plus data) of the IP datagram in bytes. A 16-bit number can define a total length of up to 65,535 (when all bits are 1s).

- ***Identification, Flags, and Fragmentation Offset*.** These three fields are related to the fragmentation of the IP datagram when the size of the datagram is larger than the underlying network can carry.

- ***Time-to-live*.** The time-to-live (TTL) field is used to control the maximum number of hops (routers) visited by the datagram. When a source host sends the datagram, it stores a number in this field. This value is approximately two times the maximum number of routers between any two hosts. Each router that processes the datagram decrements this number by one. If this value, after being decremented, is zero, the router discards the datagram.

- ***Protocol*.** In TCP/IP, the data section of a packet, called the *payload,* carries the whole packet from another protocol. A datagram, for example, can carry a packet belonging to any transport-layer protocol such as UDP or TCP. A datagram can also carry a packet from other protocols that directly use the service of the IP, such as some routing protocols or some auxiliary protocols.

- ***Header checksum*.** IP is not a reliable protocol; it does not check whether the payload carried by a datagram is corrupted during the transmission. IP puts the burden of error checking of the payload on the protocol that owns the payload, such as UDP or TCP. The datagram header, however, is added by IP, and its error-checking is the responsibility of IP.

- ***Source and Destination Addresses*.** These 32-bit source and destination address fields define the IP address of the source and destination respectively. The source host should know its IP address. The destination IP address is either known by the protocol that uses the service of IP or is provided by the DNS.

- ***Options.*** A datagram header can have up to 40 bytes of options. Options can be used for network testing and debugging. Although options are not a required part of the IP header, option processing is required of the IP software.

- ***Payload*.** Payload, or data, is the main reason for creating a datagram. Payload is the packet coming from other protocols that use the service of IP. Comparing a datagram to a postal package, payload is the content of the package; the header is only the information written on the package.

ICMPv4:

The IPv4 has no error-reporting or error-correcting mechanism. The IP protocol also lacks a mechanism for host and management queries. A host sometimes needs to determine if a router or another host is alive. And sometimes a network manager needs information from another host or router.

The **Internet Control Message Protocol version 4 (ICMPv4)** has been designed to compensate for the above two deficiencies. It is a companion to the IP protocol. ICMP itself is a network-layer protocol.

However, its messages are not passed directly to the data-link layer as would be expected. Instead, the messages are first encapsulated inside IP datagrams before going to the lower layer. When an IP datagram encapsulates an ICMP message, the value of the protocol field in the IP datagram is set to 1 to indicate that the IP payroll is an ICMP message.

MESSAGES:

ICMP messages are divided into two broad categories: ***error-reporting messages*** and ***query messages***.

The ***error-reporting messages*** report problems that a router or a host (destination) may encounter when it processes an IP packet.

The ***query messages***, which occur in pairs, help a host or a network manager get specific information from a router or another host. For example, nodes can discover their neighbors. Also, hosts can discover and learn about routers on their network and routers can help a node redirect its messages.

An ICMP message has an 8-byte header and a variable-size data section. Although the general format of the header is different for each message type, the first 4 bytes are common to all.

As Figure 3.28 shows, the first field, ICMP type, defines the type of the message. The code field specifies the reason for the particular message type. The last common field is the checksum field (to be discussed later in the chapter). The rest of the header is specific for each message type.

The data section in error messages carries information for finding the original packet that had the error. In query messages, the data section carries extra information based on the type of query.

Error Reporting Messages: Since IP is an unreliable protocol, one of the main responsibilities of ICMP is to report some errors that may occur during the processing of the IP datagram. ICMP does not correct errors, it simply reports them.

Error correction is left to the higher-level protocols. Error messages are always sent to the original source because the only information available in the datagram about the route is the source and destination IP addresses.

ICMP uses the source IP address to send the error message to the source (originator) of the datagram. To make the error-reporting process simple, ICMP follows some rules in reporting messages:

- First, no error message will be generated for a datagram having a multicast address or special address (such as *this host* or *loopback*).
- Second, no ICMP error message will be generated in response to a datagram carrying an ICMP error message.
- Third, no ICMP error message will be generated for a fragmented datagram that is not the first fragment.

Destination Unreachable: The most widely used error message is the destination unreachable (type 3). This message uses different codes (0 to 15) to define the type of error message and the reason why a datagram has not reached its final destination.

Source Quench: Another error message is called the *source quench* (type 4) message, which informs the sender that the network has encountered congestion and the datagram has been dropped; the source needs to slow down sending more datagrams.

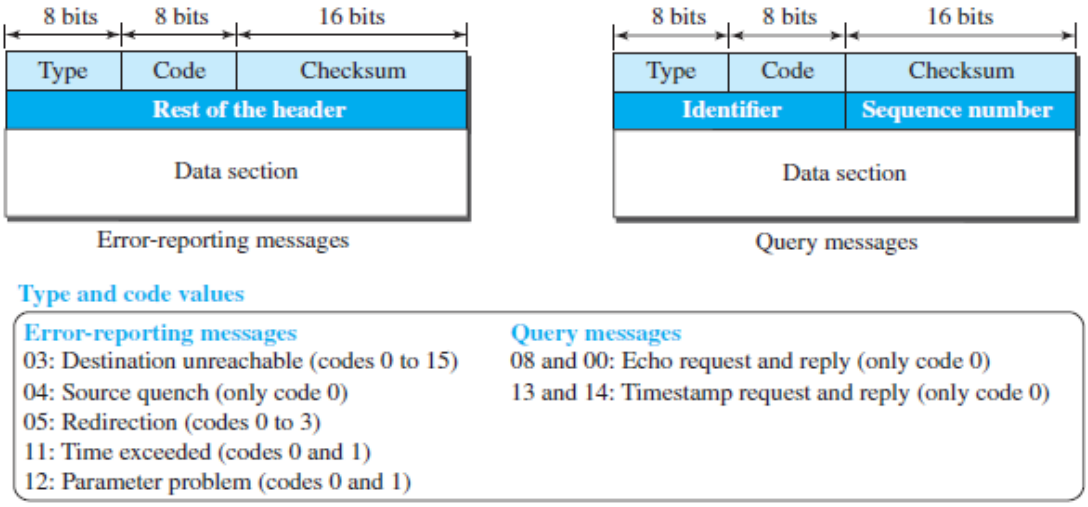


FIGURE 3.28: GENERAL FORMAT OF ICMP MESSAGES

Redirection Message: The *redirection message* (type 5) is used when the source uses a wrong router to send out its message. The router redirects the message to the appropriate router, but informs the source that it needs to change its default router in the future. The IP address of the default router is sent in the message.

Parameter Problem: A *parameter problem message* (type 12) can be sent when either there is a problem in the header of a datagram (code 0) or some options are missing or cannot be interpreted (code 1).

Query Messages: Query messages in ICMP can be used independently without relation to an IP datagram. Of course, a query message needs to be encapsulated in a datagram, as a carrier.

Query messages are used to probe or test the liveliness of hosts or routers in the Internet, find the one-way or the round-trip time for an IP datagram between two devices, or even find out whether the clocks in two devices are synchronized. Naturally, query messages come in pairs: request and reply.

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.

Messages:

There are only two types of messages in IGMP version 3, query and report messages, as shown in Figure 3.29. 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 3.29: IGMP OPERATION

Query Message:

The query message is sent by a router to all hosts in each interface to collect information about their membership. There are three versions of query messages, as described below:

a. A ***general* query** message is sent about membership in any group. It is encapsulated in a datagram with the destination address 224.0.0.1 (all hosts and routers). Note that all routers attached to the same network receive this message to inform them that this message is already sent and that they should refrain from resending it.

b. A ***group-specific*** query message is sent from a router to ask about the membership related to a specific group. This is sent when a router does not receive a response about a specific group and wants to be sure that there is no active member of that group in the network. The group identifier (multicast address) is mentioned in the message. The message is encapsulated in a datagram with the destination address set to the corresponding multicast address. Although all hosts receive this message, those not interested drop it.

c. A ***source-and-group-specific*** query message is sent from a router to ask about the membership related to a specific group when the message comes from a specific source or sources. Again the message is sent when the router does not hear about a specific group related to a specific host or hosts. The message is encapsulated in a datagram with the destination address set to the corresponding multicast address. Although all hosts receive this message, those not interested drop it.

Report Message

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).

In IGMPv3, if a host needs to join a group, it waits until it receives a query message and then sends a report message. If a host needs to leave a group, it does not respond to a query message. If no other host responds to the corresponding message, the group is purged from the router database.

Propagation of Membership Information:

After a router has collected membership information from the hosts and other routers at its own level in the tree, it can propagate it to the router located in a higher level of the tree. Finally, the router at the tree root can get the membership information to build the multicast tree. The process, however, is more complex than what we can explain in one paragraph. Interested readers can check the book website for the complete description of this protocol.

Encapsulation: The IGMP message is encapsulated in an IP datagram with the value of the protocol field set to 2 and the TTL field set to 1. The destination IP address of the datagram, however, depends on the type of message, as shown in figure 3.30.

Message Type	*IP Address*
General Query	224.0.0.1
Other Queries	Group address
Report	224.0.0.22

FIGURE 3.30: DESTINATION IP ADDRESSES

UNIT-III
NETWORK LAYER
OBJECTIVE QUESTIONS

1. The network layer concerns with
a) bits b) frames
c) packets d) none of the mentioned

2. Which one of the following is not a function of network layer?
a) routing b) inter-networking
c) congestion control **d) none of the mentioned**

3. The 4 byte IP address consists of
a) network address b) host address
c) both (a) and (b) d) none

4. In virtual circuit network each packet contains
a) full source and destination address **b) a short VC number**
c) both (a) and (b) d) none

5. Which one of the following routing algorithm can be used for network layer design?
a) shortest path algorithm b) distance vector routing
c) link state routing **d) all the above**

6. 6. Multidestination routing
a) is same as broadcast routing b) contains the list of all destinations
c) data is not sent by packets d) none

7. A subset of a network that includes all the routers but contains no loops is called
a) spanning tree b) spider structure
c) spider tree d) none

8. Which one of the following algorithm is not used for congestion control?
a) traffic aware routing b) admission control
c) load shedding **d) none of the mentioned**

9. The network layer protocol of internet is
a) Ethernet **b) internet protocol**
c) hypertext transfer protocol d) none of the mentioned

10. ICMP is primarily used for
a) error and diagnostic functions b) addressing
c) forwarding
d) none of the mentioned

11. The operation of subnet is controlled by _____.
a. Network Layer.
b. Data Link Layer

c. Data Layer
d. Transport Layer

12. _____are two popular examples of distance vector routing protocols.

a. OSPF and RIP
b. RIP and BGP
c. BGP and OSPF
d. BGP and SPF

13._____ deals with the issues of creating and maintaining routing tables.

a. Forwarding
b. Routing
c. Directing
d. None directing

14. During an adverse condition, the length of time for every device in the network to produce an accurate routing table is called the _____.

a. Accurate time
b. integrated time
c. Convergence time
d. Average time

15. A _____ routing table contains information entered manually.

a. Static
b. Dynamic
c. Hierarchical
d. Non static

16. Which of the following is/are the uses of static routing methods?
a. To manually define a default route.
b. To provide more secure network environment.
c. To provide more efficient resource utilization.
d. All of the above

17. A _____routing table is updated periodically using one of the dynamic routing protocols.
a. static
b. dynamic
c. hierarchical
d. non static

18.Which of the following is not the category of dynamic routing algorithm?
a. Distance vector protocols
b. Link state protocols
c. Hybrid protocols
d. Automatic state protocols

19. In _____ forwarding, the full IP address of a destination is given in the routing table.
a. next-hop
b. network-specific
c. host-specific
d. default

20. To build the routing table, _____algorithms allow routers to automatically discover and maintain awareness or the paths through the network.

a. Static routing **b. dynamic routing**

c. Hybrid routing d. automatic routing

21. In _____ forwarding, the mask and destination addresses are both 0.0.0.0 in the routing table.

a. next-hop b. network-specific

c. host-specific **d. default**

22).To build the routing table, _____method use preprogrammed definitions representing paths through the network.

a. Static routing b. dynamic routing

c. Hybrid routing d. automatic routing

23).In _____forwarding, the destination addresses is a network address in the routing table.

a. next-hop **b. network-specific**

c. host-specific d. default

24)._____ allows routers to exchange information within an AS.

a. Interior Gateway Protocol (IGP) b. Exterior Gateway Protocol (EGP)

c. Border Gateway Protocol (BGP) d. Static Gateway Protocol (SGP)

25. In _____forwarding, the routing table holds the address of just the next hop instead of complete route information.

a. next-hop b. network-specific

c. host-specific d. default

26. Which of the following is an example of Exterior Gateway Protocol?

a. Open Short Path First (OSPF) **b. Border Gateway Protocol (BGP)**

c. Routing Information Protocol (RIP) d. All of the above

27. A one-to-all communication between one source and all hosts on a network is classified as a____.

a. unicast b. multicast

c. broadcast d. point to point

28. _____allow the exchange of summary information between autonomous systems.

a. Interior Gateway Protocol (IGP) **b. Exterior Gateway Protocol (EGP)**

c. Border Gateway Protocol (BGP) d. Dynamic Gateway Protocol (DGP)

29).A robust routing protocol provides the ability to ..... build and manage the information in the IP routing table.

a. Dynamically b. Statically

c. Hierarchically d. All of the above

30.State True of False for definition of an autonomous system(AS).

i) An AS is defined as a physical portion of a larger IP network.

ii) An AS is normally comprised of an internetwork within an organization.

a. i-True, ii-True b. i-True, ii-False

c. i-False, ii-True d. i-False, ii-False

31. What are the parameters on which two networks differ.

a) Packet sized used b) use flow and error control technique

c) Connectionless control and security mechanism **d) all**

32. _____ are the limitations that cause different networks have different packet size.

a) hardware b) operating system

c) protocols **d) all**

33. Fragmentation means_____

a) adding of sma;ll packets to form large packets

b) breaking the large packet into small packets

c) forwarding packet through different networks

d) None

34. The header part of a fragment contains_____ number of fields

a) 2 **b)3** c)1 d)4

35.The header checksum is the IP header is used to verify _____.

a) only header b) only data c) both d) None

36. The higest IPV4 address in digital notation is

a) 255.0.0.0 b)255.255.0.0 **c)255.255.255.255**

d)255.255.255.255

37. Which class of IP address is used for used for more networks and less hosts.

a) class-A b) class-B **c)class-C** d) class-E

38. In IPV4 addressing, the digital notation address 222.255.255.255 belongs to

a) class-A b) class-B **c)class-C** d) class-E

39.classless inter domain routing contains_____.
a)32 bit IP address b)32 bit mask address **c)both** d)None

40.To indicate a sender that,it has no data,to the receiver bit is set
a)PSH b)RST **c)FIN** d) ACK

41.connecting different networks is called _____
a) intranet working **b)internetworking** c)multiple networking d)ALL

42.Bridges are used in _____ layer.
a) physical **b)MAC** c)network d)application

43.Which is a intranet working device
a) router b)gateway c)bridge **d)ALL**

44. Gateways are used at _____layer.
a) datalink layer b)network layer **c)application** d)ALL

45.The max length of the option load field in IP datagram is _____.
a)40 bytes b)80 bytes c)16 bytes d)any number of bytes multipleof 4

46. The length of the subnet mask is _____ bits .
a)16 bits **b)32bits** c)64bits d)any

47.Address resolution protocol is used to MAP the IP address on to the_____.
a) data link layer b)internet address c)network address d)port address

48.RARP is used to map the data link layer address onto _____address.
a)network b)port **c)IP** d)None

49.which are the following option are used in IPV4.
a) security b)timestamp c)source routing **d)ALL**

50.Which class of IP addressing provide more number of hosts in each network
a)class-A b)class-B c)class-c d)class-D

UNIT-III
Descriptive Questions

1. What are the Services provided by Network layer to the Transport layer ?

2. Discuss the functions of the communication subnet to provide datagram service.

3. What is meant by connection state information in a virtual circuit network?

4. Compare Virtual-Circuit and Datagram Subnets.

5. What is routing algorithm? What are the classifications of it?

6. What is the Optimality Principle?

7. With an example explain shortest path routing algorithm.

8. Explain flooding

9. Explain distance vector routing algorithm.

10. Explain count-to-infinity problem.
11. Write short notes on the following
    (a) IPV4 (b) IPV6
12. Write about Internet Control Protocols.

UNIT-IV

TRANSPORT LAYER:

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.

Introduction:

The transport layer is located between the application layer and the network layer. It provides a process-to-process communication between two application layers, one at the local host and the other at the remote host.

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.

THE TRANSPORT SERVICE:

Services provided to the upper layers:

The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective data transmission service to its users, normally processes in the application layer. To achieve this, the transport layer makes use of the services provided by the network layer. The software and/or hardware within the transport layer that does the work is called the **transport entity**.

The transport entity can be located in the operating system kernel, in a library package bound into network applications, in a separate user process, or even on the network interface card. The first two options are most common on the Internet. The (logical) relationship of the network, transport, and application layers is illustrated in Fig. 4.1.

Just as there are two types of network service, ***connection-oriented*** *and* ***connectionless****, there are also two types of transport service*. The ***connection-oriented transport service*** is similar to the connection-oriented network service in many ways. In both cases, connections have three phases: establishment, data transfer, and release. Addressing and flow control are also similar in both layers.

Furthermore, the ***connectionless transport service*** is also very similar to the connectionless network service. However, note that it can be difficult to provide a connectionless transport service on top of a connection-oriented network service, since it is inefficient to set up a connection to send a single packet and then tear (*meaning run/rip/rush*) it down immediately afterwards.

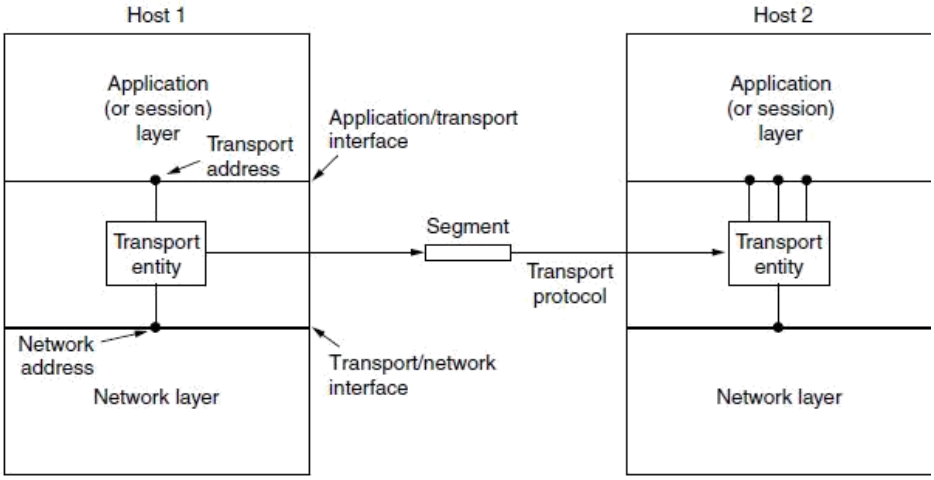


Figure 4.1: The network, transport, and application layers

Transport service primitives:

To allow users to access the transport service, the transport layer must provide some operations to application programs, that is, a transport service interface. Each transport service has its own interface.

The transport service is similar to the network service, but there are also some important differences. The main difference is that the network service is intended to model the service offered by real networks and all. Real networks can lose packets, so the network service is generally unreliable.

The connection-oriented transport service, in contrast, is reliable. Of course, real networks are not error-free, but that is precisely the purpose of the transport layer—to provide a reliable service on top of an unreliable network.

A second difference between the network service and transport service is whom the services are intended for. The network service is used only by the transport entities. Few users write their own transport entities, and thus few users or programs ever (*meaning always/forever/still*) see the bare network service.

Berkeley sockets: Let us now briefly inspect another set of transport primitives, the socket primitives as they are used for TCP. Sockets were first released as part of the Berkeley UNIX 4.2BSD software distribution in 1983. They quickly became popular.

The primitives are now widely used for Internet programming on many operating systems, especially UNIX-based systems, and there is a socket-style API for Windows called "winsock." The primitives are listed in Fig. 4.2.

Primitive	Meaning
SOCKET	Create a new communication endpoint
BIND	Associate a local address with a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Passively establish an incoming connection
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

Figure 4.2: The socket primitives for TCP

Note: An Example of Socket Programming: An Internet File Server

ELEMENTS OF TRANSPORT PROTOCOLS:

The transport service is implemented by a **transport protocol** used between the two transport entities. In some ways, transport protocols resemble the data link protocols. Both have to deal with error control, sequencing, and flow control, among other issues.

However, significant differences between the two also exist. These differences are due to major dissimilarities between the environments in which the two protocols operate, as shown in Fig. 4.3.

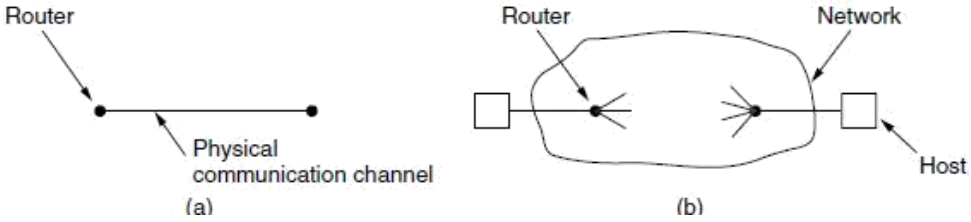


Figure 4.3: Environment of the (a) data link layer (b) transport layer

At the data link layer, two routers communicate directly via a physical channel, whether wired or wireless, whereas at the transport layer, this physical channel is replaced by the entire network.

For one thing, over point-to-point links such as wires or optical fiber, it is usually not necessary for a router to specify which router it wants to talk to—each outgoing line leads directly to a particular router. In the transport layer, explicit addressing of destinations is required.

For another thing, the process of establishing a connection over the wire of Fig. 4.3(a) is simple: the other end is always there (unless it has crashed, in which case it is not there). Either way, there is not much to do.

Even on wireless links, the process is not much different. Just sending a message is sufficient to have it reach all other destinations. If the message is not acknowledged due to an error, it can be resent. In the transport layer, initial connection establishment is complicated.

Addressing:

When an application (e.g., a user) process wishes to set up a connection to a remote application process, it must specify which one to connect to. (Connectionless transport has the same problem: to whom should each message be sent?) The method normally used is to define transport addresses to which processes can listen for connection requests.

In the Internet, these endpoints are called **ports**. We will use the generic term **TSAP** (**Transport Service Access Point**) to mean a specific endpoint in the transport layer. The analogous endpoints in the network layer (i.e., network layer addresses) are naturally called **NSAPs** (**Network Service Access Points**). IP addresses are examples of NSAPs.

Figure 4.4 illustrates the relationship between the NSAPs, the TSAPs, and a transport connection.

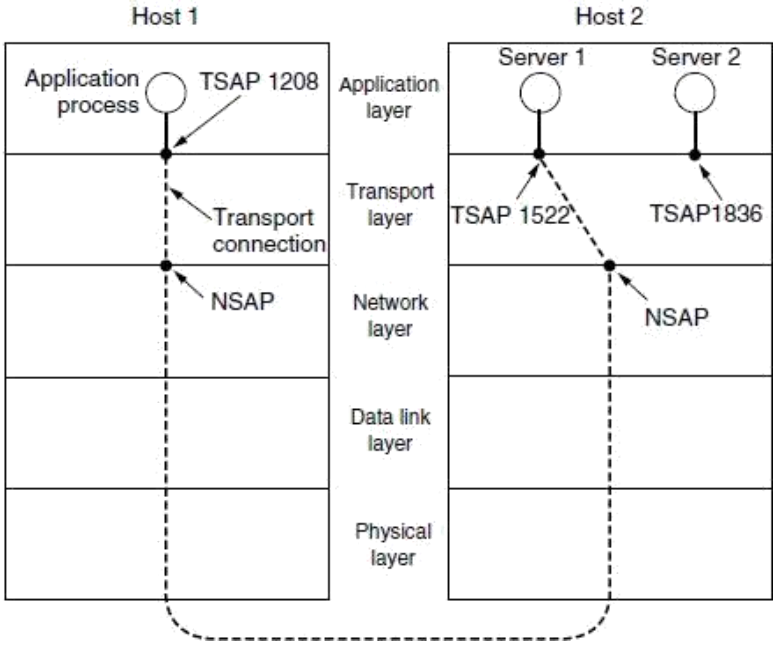


Figure 4.4: TSAPs, NSAPs, and Transport connections

Application processes, both clients and servers, can attach themselves to a local TSAP to establish a connection to a remote TSAP. These connections run through NSAPs on each host, as shown in figure 4.4.

A possible scenario for a transport connection is as follows:

1. A mail server process attaches itself to TSAP 1522 on host 2 to wait for an incoming call. A call such as our LISTEN might be used, for example.

2. An application process on host 1 wants to send an email message, so it attaches itself to TSAP 1208 and issues a CONNECT request.

    - The request specifies TSAP 1208 on host 1 as the source and TSAP 1522 on host 2 as the destination. This action ultimately results in a transport connection being established between the application process and the server.

3. The application process sends over the mail message.

4. The mail server responds to say that it will deliver the message.

5. The transport connection is released.

Connection Establishment:

Establishing a connection sounds easy, but it is actually surprisingly tricky. At first glance, it would seem sufficient for one transport entity to just send a CONNECTION REQUEST segment to the destination and wait for a CONNECTION ACCEPTED reply. The problem occurs when the network can lose, delay, corrupt, and duplicate packets. This behavior causes serious complications.

Imagine a network that is so congested that acknowledgements hardly ever get back in time and each packet times out and is retransmitted two or three times. Suppose that the network uses datagrams inside and that every packet follows a different route.

Some of the packets might get stuck in a traffic jam inside the network and take a long time to arrive. That is, they may be delayed in the network and pop out much later, when the sender thought that they had been lost.

The worst possible nightmare is as follows. A user establishes a connection with a bank, sends messages telling the bank to transfer a large amount of money to the account of a not-entirely-trustworthy person. Unfortunately, the packets decide to take the scenic route to the destination and go off exploring a remote corner of the network.

The sender then times out and sends them all again. This time the packets take the shortest route and are delivered quickly so the sender releases the connection.

Unfortunately, eventually the initial batch of packets finally come out of hiding and arrive at the destination in order, asking the bank to establish a new connection and transfer money (again). The bank has no way of telling that these are duplicates. It must assume that this is a second, independent transaction, and transfers the money again.

The crux (*meaning root*) of the problem is that the delayed duplicates are thought to be new packets. We cannot prevent packets from being duplicated and delayed. But if and when this happens, the packets must be rejected as duplicates and not processed as fresh packets.

The problem can be attacked in various ways, none of them very satisfactory. One way is to use throwaway 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. Delayed duplicate packets then never find their way to a transport process and can do no damage.

Note: However, this approach makes it more difficult to connect with a process in the first place.

Another possibility is to give each connection a unique identifier (i.e., a sequence number incremented for each connection established) chosen by the initiating party and put in each segment, including the one requesting the connection.

After each connection is released, each transport entity can update a table listing obsolete connections as (peer transport entity, connection identifier) pairs. Whenever a connection request comes in, it can be checked against the table to see if it belongs 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. This history must persist at both the source and destination machines. Otherwise, if a machine crashes and loses its memory, it will no longer know which connection identifiers have already been used by its peers.

Instead, we need to take a different tack to simplify the problem. Rather than allowing packets to live forever within the network, we devise a mechanism to kill off aged packets that are still hobbling about.

Packet lifetime can be restricted to a known maximum using one (or more) of the following techniques:

1. Restricted network design.

2. Putting a hop counter in each packet.

3. Timestamping each packet.

TCP uses three-way handshake to establish connections in the presence of delayed duplicate control segments as shown in figure 4.5.

Connection Release:

Releasing a connection is easier than establishing one. 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.

Asymmetric release is abrupt and may result in data loss. Consider the scenario of Fig. 4.6. After the connection is established, host 1 sends a segment that arrives properly at host 2. Then host 1 sends another segment.

Unfortunately, host 2 issues a DISCONNECT before the second segment arrives. The result is that the connection is released and data are lost.

Symmetric release does the job when each process has a fixed amount of data to send and clearly knows when it has sent it. In other situations, determining that all the work has been done and the connection should be terminated is not so obvious.

One can envision a protocol in which host 1 says ‘‘I am done. Are you done too?’’ If host 2 responds: ‘‘I am done too. Goodbye, the connection can be safely released.’’

In practice, we can avoid this quandary (*meaning dilemma/difficulty*) by foregoing the need for agreement and pushing the problem up to the transport user, letting each side independently decide when it is done. This is an easier problem to solve.

Figure 4.7 illustrates four scenarios of releasing using a three-way handshake. While this protocol is not infallible, it is usually adequate. In Fig. 4.7(a), we see the normal case in which one of the users sends a DR (DISCONNECTION REQUEST) segment to initiate the connection release.

When it arrives, the recipient sends back a DR segment and starts a timer, just in case its DR is lost. When this DR arrives, the original sender sends back an ACK segment and releases the connection.

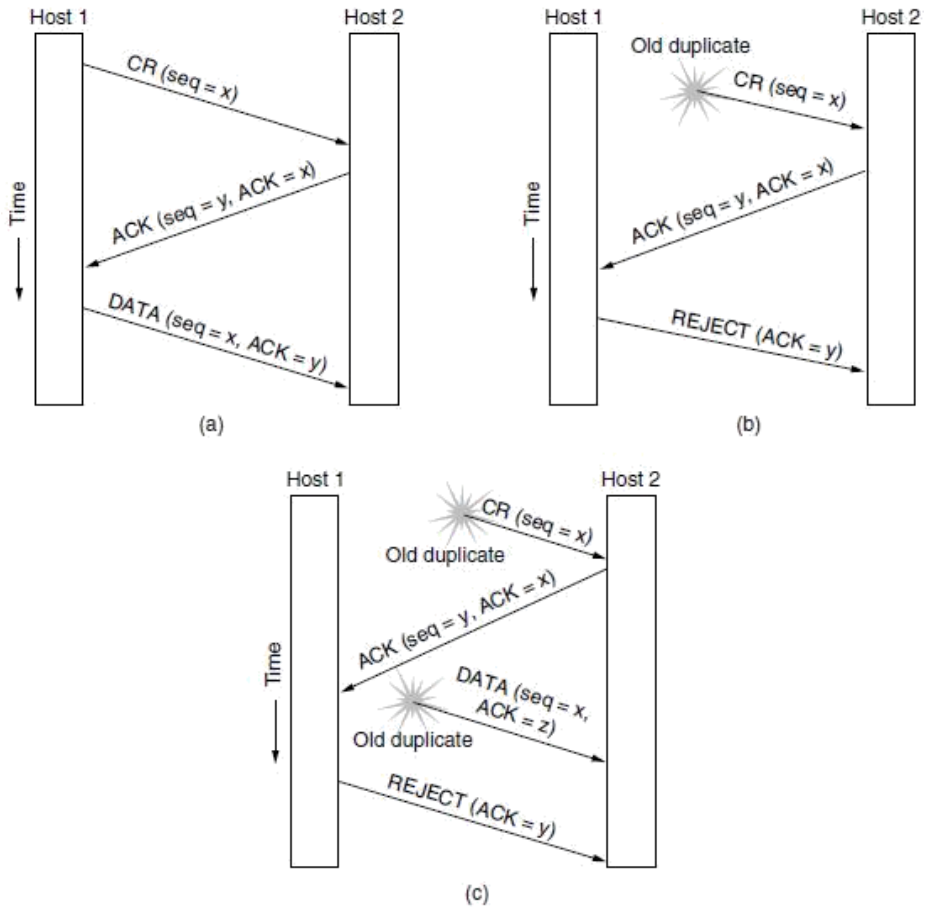


Figure 4.5: Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes Connection Request. (a) normal operation. (b) old duplicate connection request appearing out of nowhere. (c) duplicate connection request and duplicate ack.

Finally, when the ACK segment arrives, the receiver also releases the connection. Releasing a connection means that the transport entity removes the information about the connection from its table of currently open connections and signals the connection's owner (the transport user) somehow.

If the final ACK segment is lost, as shown in Fig. 4.7(b), the situation is saved by the timer. When the timer expires, the connection is released anyway. Now consider the case of the second DR being lost.

The user initiating the disconnection will not receive the expected response, will time out, and will start all over again. In Fig. 4.7(c), we see how this works, assuming that the second time no segments are lost and all segments are delivered correctly and on time.

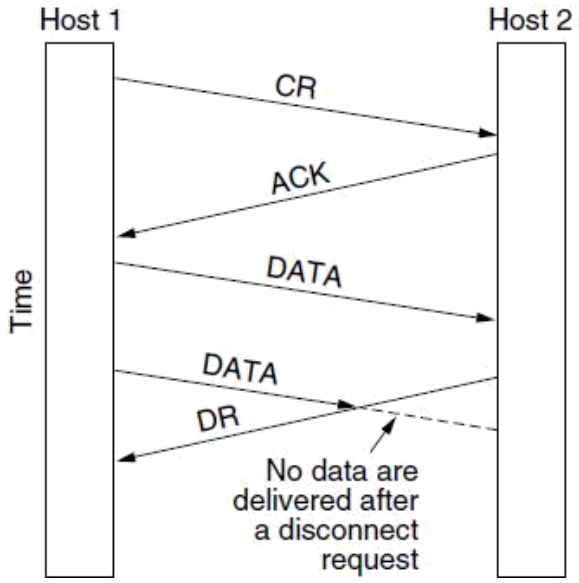


Figure 4.6: Abrupt disconnection with loss of data

Our last scenario, Fig. 4.7(d), is the same as Fig. 4.7(c) except that now we assume all the repeated attempts to retransmit the DR also fail due to lost segments. After *N* retries, the sender just gives up and releases the connection. Meanwhile, the receiver times out and also exits.

Error control and Flow control:

Error control is ensuring that the data is delivered with the desired level of reliability, usually that all of the data is delivered without any errors. Flow control is keeping a fast transmitter from overrunning a slow receiver.

MULTIPLEXING:

Multiplexing, or sharing several conversations over connections, virtual circuits, and physical links plays a role in several layers of the network architecture. In the transport layer, the need for multiplexing can arise in a number of ways. For example, if only one network address is available on a host, all transport connections on that machine have to use it.

When a segment comes in, some way is needed to tell which process to give it to. This situation, called **multiplexing**, is shown in Fig. 4.8(a). In this figure, four distinct transport connections all use the same network connection (e.g., IP address) to the remote host.

Multiplexing can also be useful in the transport layer for another reason. Suppose, for example, that a host has multiple network paths that it can use. If a user needs more bandwidth or more reliability than one of the network paths can provide, a way out is to have a connection that distributes the traffic among multiple network paths on a round-robin basis, as indicated in Fig. 4.8(b).

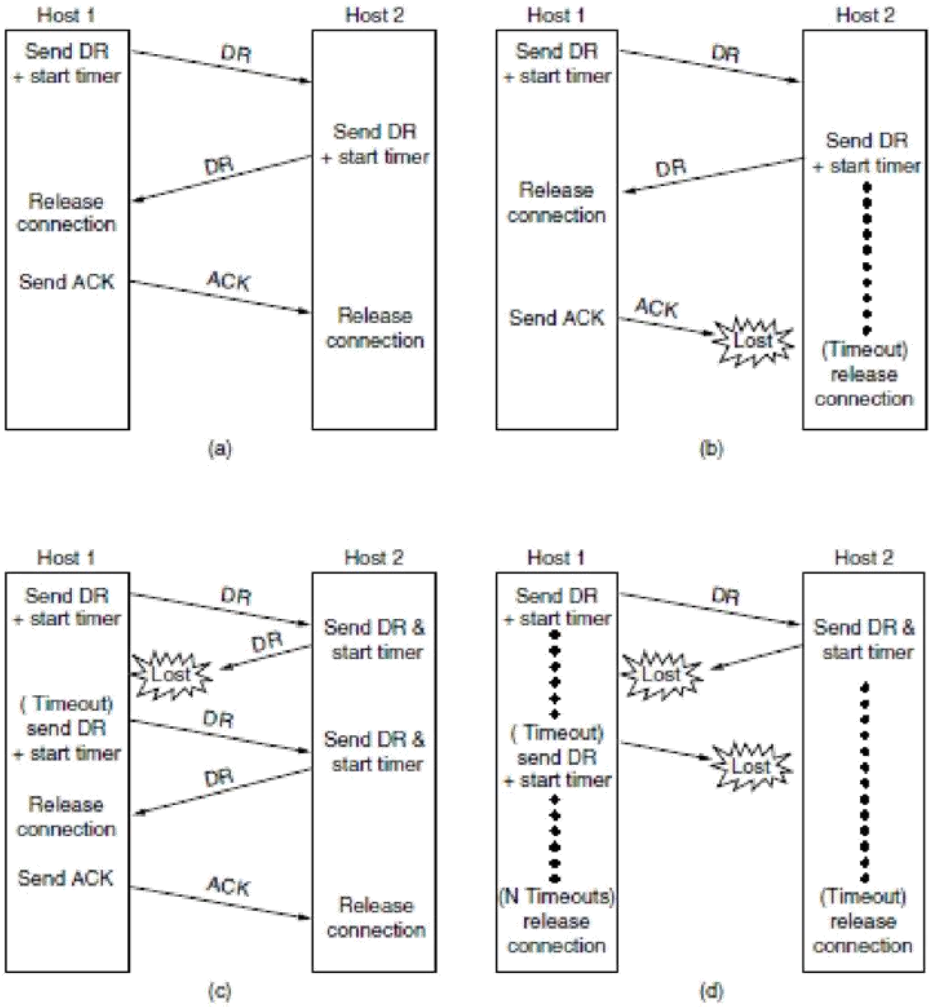


Figure 4.7: Four protocol scenarios for releasing a connection. (a) normal case of three-way handshake. (b) final ACK lost. (c) response lost. (d) response lost and subsequent DRs lost.

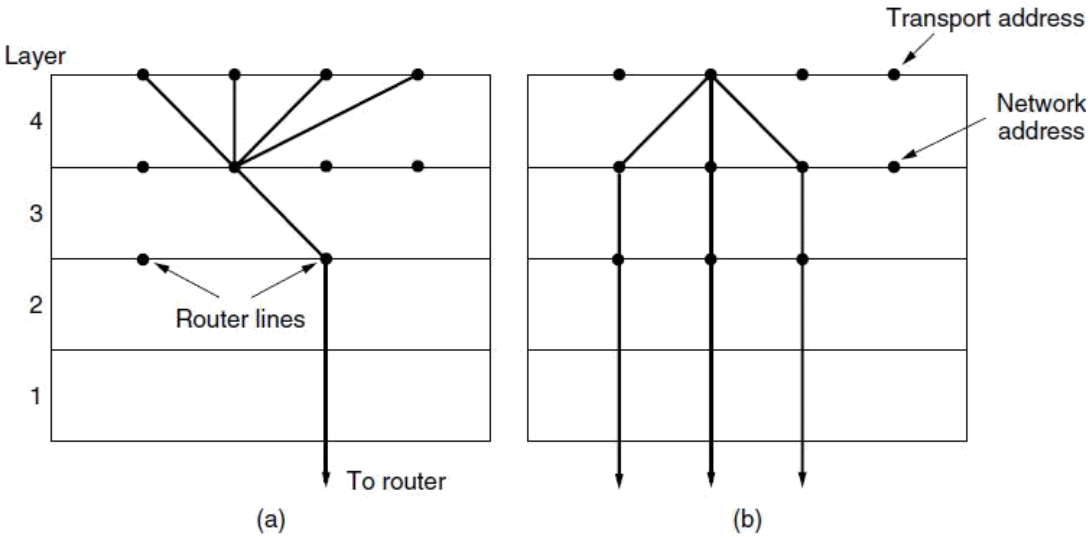


Figure 4.8: (A) Multiplexing (B) Inverse Multiplexing

CRASH RECOVERY:

If hosts and routers are subject to crashes or connections are long-lived (e.g., large software or media downloads), recovery from these crashes becomes an issue.

If the transport entity is entirely within the hosts, recovery from network and router crashes is straightforward. The transport entities expect lost segments all the time and know how to cope with them by using retransmissions.

A more troublesome problem is how to recover from host crashes. In particular, it may be desirable for clients to be able to continue working when servers crash and quickly reboot.

CONGESTION CONTROL:

If the transport entities on many machines send too many packets into the network too quickly, the network will become congested, with performance degraded as packets are delayed and lost.

Controlling congestion to avoid this problem is the combined responsibility of the network and transport layers. Congestion occurs at routers, so it is detected at the network layer.

However, congestion is ultimately caused by traffic sent into the network by the transport layer. The only effective way to control congestion is for the transport protocols to send packets into the network more slowly.

DESIRABLE BANDWIDTH ALLOCATION:

Before we describe how to regulate traffic, we must understand what we are trying to achieve by running a congestion control algorithm. That is, we must specify the state in which a good congestion control algorithm will operate the network.

The goal is more than to simply avoid congestion. It is to find a good allocation of bandwidth to the transport entities that are using the network. A good allocation will deliver good performance because it uses all the available bandwidth but avoids congestion, it will be fair across competing transport entities, and it will quickly track changes in traffic demands.

Efficiency and Power:

An efficient allocation of bandwidth across transport entities will use all of the network capacity that is available. However, it is not quite right to think that if there is a 100-Mbps link, five transport entities should get 20 Mbps each. They should usually get less than 20 Mbps for good performance.

Max-Min Fairness:

In the preceding discussion, we did not talk about how to divide bandwidth between different transport senders. This sounds like a simple question to answer—give all the senders an equal fraction of the bandwidth—but it involves several considerations.

Perhaps the first consideration is to ask what this problem has to do with congestion control.

A second consideration is what a fair portion means for flows in a network. It is simple enough if *N* flows use a single link, in which case they can all have 1*/N* of the bandwidth (although efficiency will dictate that they use slightly less if the traffic is bursty).

But what happens if the flows have different, but overlapping, network paths? For example, one flow may cross three links, and the other flows may cross one link. The three-link flow consumes more network resources. It might be fairer in some sense to give it less bandwidth than the one-link flows.

The form of fairness that is often desired for network usage is **max-min fairness**. An allocation is max-min fair if the bandwidth given to one flow cannot be increased without decreasing the bandwidth given to another flow with an allocation that is no larger.

Convergence:

A final criterion is that the congestion control algorithm converge quickly to a fair and efficient allocation of bandwidth. The discussion of the desirable operating point above assumes a static network environment.

However, connections are always coming and going in a network, and the bandwidth needed by a given connection will vary over time too. Because of the variation in demand, the ideal operating point for the network varies over time.

A good congestion control algorithm should rapidly converge to the ideal operating point, and it should track that point as it changes over time. If the convergence is too slow, the algorithm will never be close to the changing operating point. If the algorithm is not stable, it may fail to converge to the right point in some cases, or even oscillate around the right point.

Regulating the sending rate:

Now it is time to regulate the sending rates to obtain a desirable bandwidth allocation. The sending rate may be limited by two factors.

- The first is flow control, in the case that there is insufficient buffering at the receiver.
- The second is congestion, in the case that there is insufficient capacity in the network.

In Fig. 4.9, we see this problem illustrated hydraulically. In Fig. 4.9(a), we see a thick pipe leading to a small-capacity receiver. This is a flow-control limited situation. As long as the sender does not send more water than the bucket can contain, no water will be lost.

In Fig. 4.9(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).

The way that a transport protocol should regulate the sending rate depends on the form of the feedback returned by the network. Different network layers may return different kinds of feedback. The feedback may be explicit or implicit, and it may be precise or imprecise.

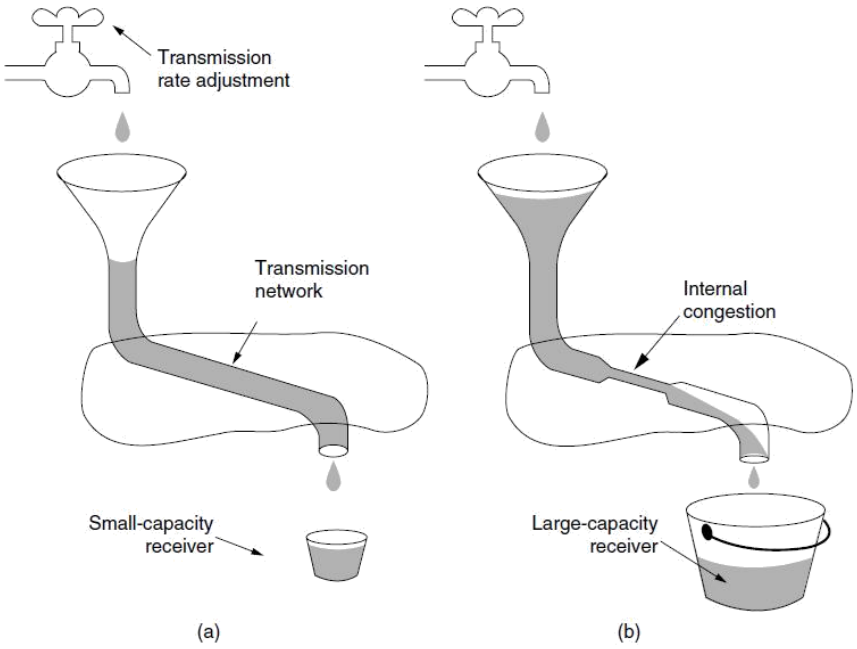


Figure 4.9: (a) a fast network feeding a low-capacity receiver. (b) a slow network feeding a high-capacity receiver.

Wireless issues:

Transport protocols such as TCP that implement congestion control should be independent of the underlying network and link layer technologies. That is a good theory, but in practice there are issues with wireless networks. The main issue is that packet loss is often used as a congestion signal, including by TCP.

Wireless networks lose packets all the time due to transmission errors. To function well, the only packet losses that the congestion control algorithm should observe are losses due to insufficient bandwidth, not losses due to transmission errors. One solution to this problem is to mask the wireless losses by using retransmissions over the wireless link.

THE INTERNET TRANSPORT PROTOCOLS:

UDP:

The Internet has two main protocols in the transport layer, a connectionless protocol and a connection-oriented one. The protocols complement each other.

The connectionless protocol is UDP. It does almost nothing beyond sending packets between applications, letting applications build their own protocols on top as needed.

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

INTRODUCTION TO UDP:

The Internet protocol suite supports a connectionless transport protocol called **UDP** (**User Datagram Protocol**).

UDP provides a way for applications to send encapsulated IP datagrams without having to establish a connection. UDP is described in RFC 768.

UDP transmits **segments** consisting of an 8-byte header followed by the payload. The header is shown in Fig. 4.10. The two **ports** serve to identify the endpoints within the source and destination machines.

When a UDP packet arrives, its payload is handed to the process attached to the destination port. This attachment occurs when the BIND primitive or something similar is used.

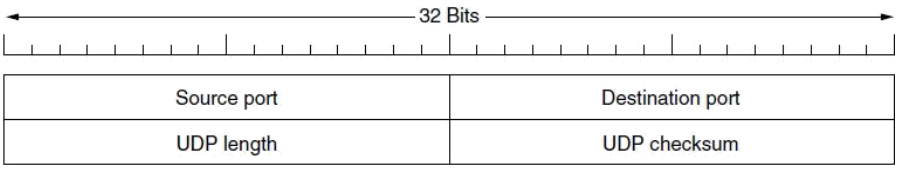


Figure 4.10: the UDP header

Think of ports as mailboxes that applications can rent to receive packets. In fact, the main value of UDP over just using raw IP is the addition of the source and destination ports.

Without the port fields, the transport layer would not know what to do with each incoming packet. With them, it delivers the embedded segment to the correct application.

The source port is primarily needed when a reply must be sent back to the source. By copying the *Source port* field from the incoming segment into the *Destination port* field of the outgoing segment, the process sending the reply can specify which process on the sending machine is to get it.

The *UDP length* field includes the 8-byte header and the data. The minimum length is 8 bytes, to cover the header. The maximum length is 65,515 bytes, which is lower than the largest number that will fit in 16 bits because of the size limit on IP packets.

An optional *Checksum* is also provided for extra reliability. It checksums the header, the data, and a conceptual IP pseudoheader. When performing this computation, the *Checksum* field is set to zero and the data field is padded out with an additional zero byte if its length is an odd number.

The checksum algorithm is simply to add up all the 16-bit words in one's complement and to take the one's complement of the sum.

Remote procedure call:

In a certain sense, sending a message to a remote host and getting a reply back is a lot like making a function call in a programming language. The idea behind RPC is to make a remote procedure call look as much as possible like a local one.

In the simplest form, to call a remote procedure, the client program must be bound with a small library procedure, called the **client stub**, that represents the server procedure in the client's address space.

Similarly, the server is bound with a procedure called the **server stub**. These procedures hide the fact that the procedure call from the client to the server is not local. The actual steps in making an RPC are shown in Fig. 4.12.

- Step 1 is the client calling the client stub. This call is a local procedure call, with the parameters pushed onto the stack in the normal way.
- Step 2 is the client stub packing the parameters into a message and making a system call to send the message. Packing the parameters is called **marshaling**.
- Step 3 is the operating system sending the message from the client machine to the server machine.

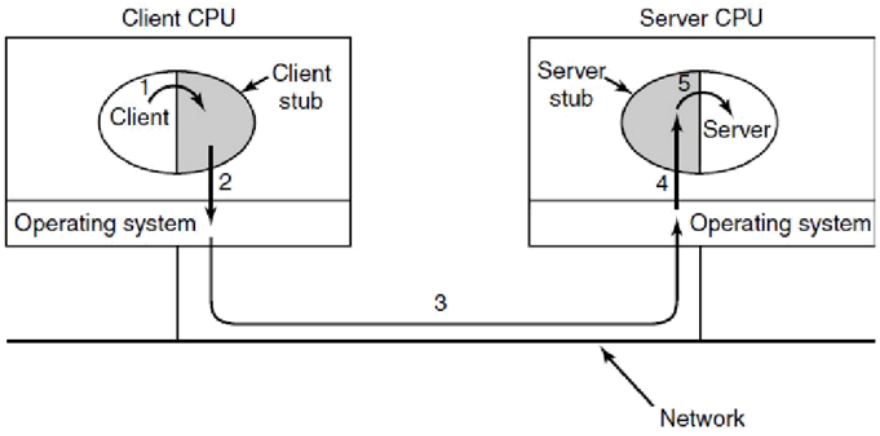


Figure 4.12: Steps in making a remote procedure call, the stubs are shaded

- Step 4 is the operating system passing the incoming packet to the server stub.

- Finally, step 5 is the server stub calling the server procedure with the unmarshaled parameters.

The reply traces the same path in the other direction.

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

Similarly, the server procedure is called by a procedure in its address space with the parameters it expects. To the server procedure, nothing is unusual.

Real-Time Transport Protocols

Client-server RPC is one area in which UDP is widely used. Another one is for real-time multimedia applications.

In particular, as Internet radio, Internet telephony, music-on-demand, videoconferencing, video-on-demand, and other multimedia applications became more commonplace, people have discovered that each application was reinventing more or less the same real-time transport protocol. Thus was **RTP** (**Real-time Transport Protocol**) born.

It is described in RFC 3550 and is now in widespread use for multimedia applications. There are two aspects of real-time transport . The first is the RTP protocol for transporting audio and video data in packets. The second is the processing that takes place, mostly at the receiver, to play out the audio and video

at the right time.

RTP—The Real-Time Transport Protocol:

The basic function of RTP is to multiplex several real-time data streams onto a single stream of UDP packets. The UDP stream can be sent to a single destination (unicasting) or to multiple destinations (multicasting).

Because RTP just uses normal UDP, its packets are not treated specially by the routers unless some normal IP quality-of-service features are enabled. In particular, there are no special guarantees about delivery, and packets may be lost, delayed, corrupted, etc.

The RTP format contains several features to help receivers work with multimedia information. The RTP header is illustrated in Fig. 4.13. It consists of three 32-bit words and potentially some extensions.

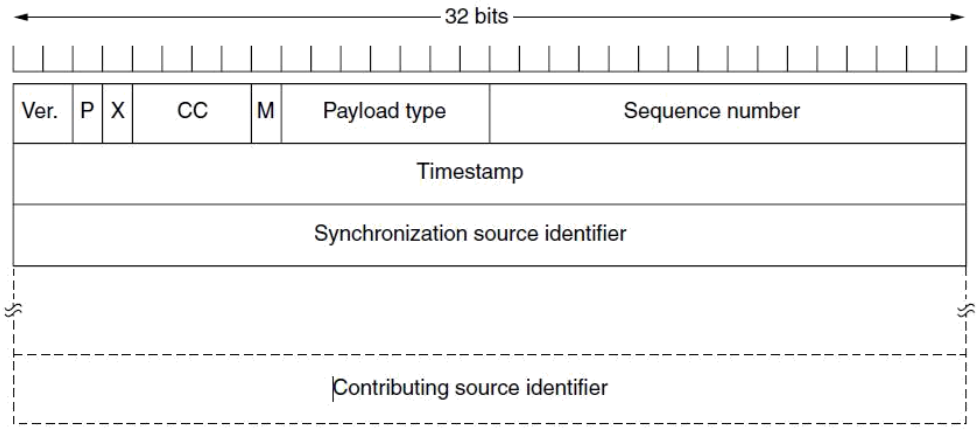


Figure 4.13: The RTP header

The first word contains the Version field, which is already at 2.

The *P* bit indicates that the packet has been padded to a multiple of 4 bytes.

The X bit indicates that an extension header is present.

The CC field tells how many contributing sources are present, from 0 to 15.

The M bit is an application-specific marker bit. It can be used to mark the start of a video frame, the start of a word in an audio channel, or something else that the application understands.

The Payload type field tells which encoding algorithm has been used (e.g., uncompressed 8-bit audio, MP3, etc.).

The Sequence number is just a counter that is incremented on each RTP packet sent. It is used to detect lost packets.

The Timestamp is produced by the stream's source to note when the first sample in the packet was made.

The Synchronization source identifier tells which stream the packet belongs to. It is the method used to multiplex and demultiplex multiple data streams onto a single stream of UDP packets.

Finally, the Contributing source identifiers, if any, are used when mixers are present.

RTCP—The Real-time Transport Control Protocol

RTP has a little sister protocol (little sibling protocol?) called **RTCP** (**Realtime Transport Control Protocol**). It is defined along with RTP in RFC 3550 and handles feedback, synchronization, and the user interface. It does not transport any media samples.

THE INTERNET TRANSPORT PROTOCOLS:

TCP

UDP is a simple protocol and it has some very important uses, such as clientserver interactions and multimedia, but for most Internet applications, reliable, sequenced delivery is needed. UDP cannot provide this, so another protocol is required. It is called TCP and is the main workhorse of the Internet.

Introduction to TCP:

TCP (**Transmission Control Protocol**) was specifically designed to provide a reliable end-to-end byte stream over an unreliable internetwork. An internetwork differs from a single network because different parts may have wildly different topologies, bandwidths, delays, packet sizes, and other parameters.

TCP was designed to dynamically adapt to properties of the internetwork and to be robust in the face of many kinds of failures. TCP was formally defined in RFC 793 in September 1981.

As time went on, many improvements have been made, and various errors and inconsistencies have been fixed. To give you a sense of the extent of TCP, the important RFCs are now RFC 793 plus: clarifications and bug fixes in RFC 1122; extensions for high-performance in RFC 1323.

Selective acknowledgements in RFC 2018; congestion control in RFC 2581; repurposing of header fields for quality of service in RFC 2873; improved retransmission timers in RFC 2988; and explicit congestion notification in RFC 3168. The IP layer gives no guarantee that datagrams will be delivered properly, nor any indication of how fast datagrams may be sent.

It is up to TCP to send datagrams fast enough to make use of the capacity but not cause congestion, and to time out and retransmit any datagrams that are not delivered. Datagrams that do arrive may well do so in the wrong order; it is also up to TCP to reassemble them into messages in the proper sequence.

The TCP Service Model:

TCP service is obtained by both the sender and the receiver creating end points, called **sockets.** Each socket has a socket number (address) consisting of the IP address of the host and a 16-bit number local to that host, called a **port**. A port is the TCP name for a TSAP.

For TCP service to be obtained, a connection must be explicitly established between a socket on one machine and a socket on another machine. A socket may be used for multiple connections at the same time. In other words, two or more connections may terminate at the same socket.

Port numbers below 1024 are reserved for standard services that can usually only be started by privileged users (e.g., root in UNIX systems). They are called **well-known ports**.

For example, any process wishing to remotely retrieve mail from a host can connect to the destination host's port 143 to contact its IMAP daemon. The list of well-known ports is given at *www.iana.org*. Over 700 have been assigned. A few of the better-known ones are listed in Fig. 4.14.

Port	Protocol	Use
20, 21	FTP	File transfer
22	SSH	Remote login, replacement for Telnet
25	SMTP	Email
80	HTTP	World Wide Web
110	POP-3	Remote email access
143	IMAP	Remote email access
443	HTTPS	Secure Web (HTTP over SSL/TLS)
543	RTSP	Media player control
631	IPP	Printer sharing

Figure 4.14: Some assigned ports

All TCP connections are full duplex and point-to-point. Full duplex means that traffic can go in both directions at the same time. Point-to-point means that each connection has exactly two end points. TCP does not support multicasting or broadcasting.

A TCP connection is a byte stream, not a message stream. Message oundaries are not preserved end to end.

The TCP Protocol:

A key feature of TCP, and one that dominates the protocol design, is that every byte on a TCP connection has its own 32-bit sequence number. When the Internet began, the lines between routers were mostly 56-kbps leased lines, so a host blasting away at full speed took over 1 week to cycle through the sequence numbers.

The sending and receiving TCP entities exchange data in the form of segments. A **TCP segment** consists of a fixed 20-byte header (plus an optional part) followed by zero or more data bytes. The TCP software decides how big segments should be.

It can accumulate data from several writes into one segment or can split data from one write over multiple segments. Two limits restrict the segment size. First, each segment, including the TCP header, must fit in the 65,515- byte IP payload. Second, each link has an **MTU** (**Maximum Transfer Unit**).

Each segment must fit in the MTU at the sender and receiver so that it can be sent and received in a single, unfragmented packet. However, it is still possible for IP packets carrying TCP segments to be fragmented when passing over a network path for which some link has a small MTU.

If this happens, it degrades performance and causes other problems. Instead, modern TCP implementations perform **path MTU discovery** by using the technique outlined in RFC 1191. This technique uses ICMP error messages to find the smallest MTU for any link on the path. TCP then adjusts the segment size downwards to avoid fragmentation.

The basic protocol used by TCP entities is the sliding window protocol with a dynamic window size. When a sender transmits a segment, it also starts a timer. When the segment arrives at the destination, the receiving TCP entity sends back a segment (with data if any exist, and otherwise without) bearing an acknowledgement number equal to the next sequence number it expects to receive and the remaining window size.

If the sender's timer goes off before the acknowledgement is received, the sender transmits the segment again.

The TCP Segment Header:

Figure 4.15 shows the layout of a TCP segment. Every segment begins with a fixed-format, 20-byte header. The fixed header may be followed by header options.

After the options, if any, up to 65,535 − 20 − 20 = 65,495 data bytes may follow, where the first 20 refer to the IP header and the second to the TCP header.

Segments without any data are legal and are commonly used for acknowledgements and control messages.

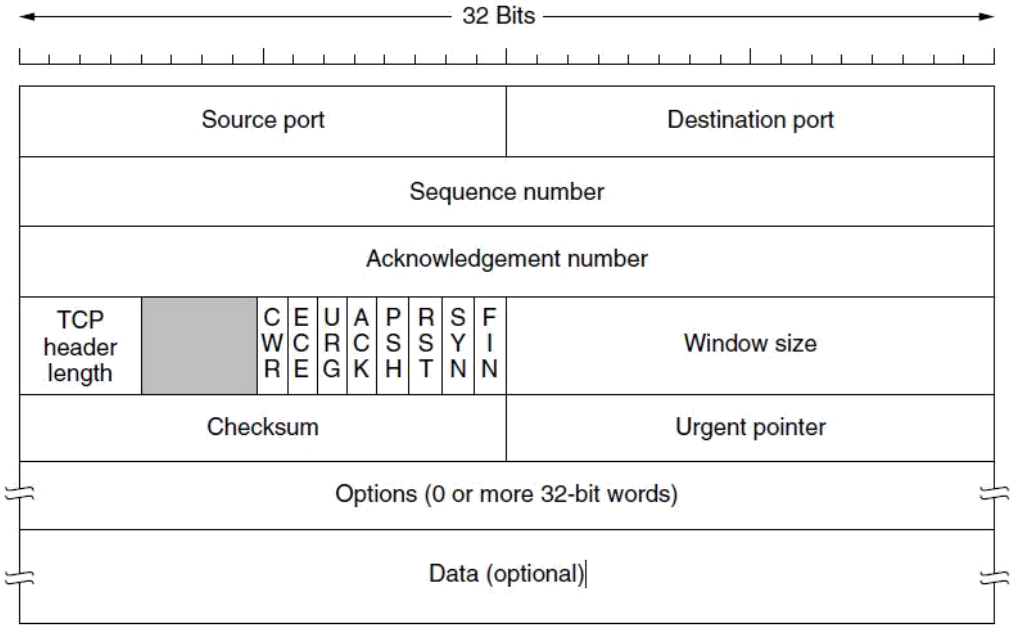


Figure 4.15: The TCP Header

The *Source port* and *Destination port* fields identify the local end points of the connection. The source and destination end points together identify the connection. This connection identifier is called a **5 tuple** because it consists of five pieces of information: the protocol (TCP), source IP and source port, and destination IP and destination port.

The *Sequence number* and *Acknowledgement number* fields perform their usual functions.

The *Sequence number* and *Acknowledgement number* fields perform their usual functions.

The *TCP header length* tells how many 32-bit words are contained in the TCP header. This information is needed because the *Options* field is of variable length, so the header is, too.

Now come eight 1-bit flags. *CWR* and *ECE* are used to signal congestion when ECN (Explicit Congestion Notification) is used. *CWR* is set to signal *Congestion Window Reduced* from the TCP sender to the TCP receiver so that it knows the sender has slowed down and can stop sending the *ECN-Echo*.

URG is set to 1 if the *Urgent pointer* is in use. The *Urgent pointer* is used to indicate a byte offset from the current sequence number at which urgent data are to be found.

The *ACK* bit is set to 1 to indicate that the *Acknowledgement number* is valid. This is the case for nearly all packets. If *ACK* is 0, the segment does not contain an acknowledgement, so the *Acknowledgement number* field is ignored.

The *PSH* bit indicates PUSHed data. The receiver is hereby kindly requested to deliver the data to the application upon arrival and not buffer it until a full buffer has been received (which it might otherwise do for efficiency).

The *RST* bit is used to abruptly reset a connection that has become confused due to a host crash or some other reason.

The *SYN* bit is used to establish connections. The *FIN* bit is used to release a connection.

The *Window size* field tells how many bytes may be sent starting at the byte acknowledged.

A *Checksum* is also provided for extra reliability. The *Options* field provides a way to add extra facilities not covered by the regular header.

TCP Connection Establishment:

Connections are established in TCP by means of the three-way handshake. To establish a connection, one side, say, the server, passively waits for an incoming connection by executing the LISTEN and ACCEPT primitives in that order, either specifying a specific source or nobody in particular.

The other side, say, the client, executes a CONNECT primitive, specifying the IP address and port to which it wants to connect, the maximum TCP segment size it is willing to accept, and optionally some user data (e.g., a password). The CONNECT primitive sends a TCP segment with the *SYN* bit on and *ACK* bit off and waits for a response.

When this segment arrives at the destination, the TCP entity there checks to see if there is a process that has done a LISTEN on the port given in the *Destination port* field. If not, it sends a reply with the *RST* bit on to reject the connection.

TCP Connection Release

Although TCP connections are full duplex, to understand how connections are released it is best to think of them as a pair of simplex connections. Each simplex connection is released independently of its sibling.

To release a connection, either party can send a TCP segment with the *FIN* bit set, which means that it has no more data to transmit. When the *FIN* is acknowledged, that direction is shut down for new data.

Data may continue to flow indefinitely in the other direction, however. When both directions have been shut down, the connection is released.

TCP Congestion Control:

The network layer detects congestion when queues grow large at routers and tries to manage it, if only by dropping packets. It is up to the transport layer to receive congestion feedback from the network layer and slow down the rate of traffic that it is sending into the network.

In the Internet, TCP plays the main role in controlling congestion, as well as the main role in reliable transport. That is why it is such a special protocol.

PERFORMANCE PROBLEMS IN COMPUTER NETWORKS

Some performance problems, such as congestion, are caused by temporary resource overloads. If more traffic suddenly arrives at a router than the router can handle, congestion will build up and performance will suffer.

Performance also degrades when there is a structural resource imbalance. For example, if a gigabit communication line is attached to a low-end PC, the poor host 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.

Overloads can also be synchronously triggered. As an example, if a segment contains a bad parameter , in many cases the receiver will thoughtfully send back an error notification.

Another tuning issue is setting timeouts. When a segment is sent, a timer is set to guard against loss of the segment. 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 segment is lost.

NETWORK PERFORMANCE MEASUREMENT:

When a network performs poorly, its users often complain to the folks running it, demanding improvements. To improve the performance, the operators must first determine exactly what is going on. To find out what is really happening, the operators must make measurements.

Measurements can be made in different ways and at many locations (both in the protocol stack and physically). The most basic kind of measurement is to start a timer when beginning some activity and see how long that activity takes.

Other measurements are made with counters that record how often some event has happened (e.g., number of lost segments).

Measuring network performance and parameters has many potential pitfalls. We list a few of them here. 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 segment, but repeat the measurement, say, one million times and take the average.

2) Make Sure That the Samples Are Representative

Ideally, the whole sequence of one million measurements should be repeated at different times of the day and the week to see the effect of different network conditions on the measured quantity.

3) Caching Can Wreak Havoc with Measurements

Repeating a measurement many times will return an unexpectedly fast answer if the protocols use caching mechanisms.

4) Be Sure That Nothing Unexpected Is Going On during Your Tests

Making measurements at the same time that some user has decided to run a video conference over your network will often give different results than if there is no video conference.

5) Be Careful When Using a Coarse-Grained Clock

Computer clocks function by incrementing some counter at regular intervals.

6) Be Careful about Extrapolating the Results

Suppose that you make measurements with simulated network loads running from 0 (idle) to 0.4 (40% of capacity).

UNIT- IV Objectives

1. The ..... is responsible for end to end delivery, segmentation and concatenation

a) Physical layer b) Data Link layer c) Network layer d) **Transport layer**

2. ..... needs ports or service access points.

a) Physical layer b) Data Link layer c) Network layer d) **Transport layer**

3. The task of ..... is to provide reliable, cost effective transport of data from source machine to destination machine.

a) Physical layer b) Data Link layer c) Network layer d) **Transport layer**

4. The hardware and/or software within the transport layer which does the work of making use of the services provided by the network layer is called as .....

a) transport media b) transport device **c) transport entity** d) network transporter

5. ..... measures the number of bytes of user data transferred per second, measured over some time interval. It is measured separately for each direction.

a) Throughput b) Transit delay c) Protection d) Resilience

6..... is the time between a message being sent by the transport user on the source machine and its being receive by the transport user on the destination machine.

a) Throughput b) **Transit delay** c) Protection d) Resilience

7. The time difference between the instant at which a transport connection is requested and the instant at which it is confirmed is called as .....

a) Connection establishment delay b) Transit delay c) Protection delay d) Priority delay

8. TCP is ---------- protocol.

a) connection oriented b) message oriented c) block oriented b) connection less

9. UDP is ---------- protocol.

a) connection oriented b) message oriented c) block oriented b) connection less

10. The message sent from transport entity to transport entity is called as .....

a) transport data unit b) transport display data unit

c) **transport protocol data unit** d) transport protocol display unit

11. ..... is designed for the connection oriented protocol such as Transmission Control Protocol(TCP).

a) Berkeley socket **b) Stream socket** c) Datagram socket d) Raw socket

12. ..... is used to implement the transport layer services between the two transport entities.

a) Transport service b) **Transport protocol** c) Transport address d) Transport control

13. Which of the following is/are the tasks of transport protocols

a) Error control b) Sequencing c) Flow control d) **All of the above**

14. The internet uses universal port numbers for services and these numbers are called as

a) **Well known port numbers** b) Fixed port numbers

c) Standard port numbers d) Ephemeral port numbers

15. ..... is designed for the connectionless protocol such as User Datagram Protocol(UDP).

a) Berkeley socket b) Stream socket **c) Datagram socket** d) Raw socket

16. ..... are known as well known ports and they are reserved for standard circuits.

a) below 1024 b**) above 1024** c) below 2048 d) below 512

17. In the TCP segment header, ... is a 32-bit number identifying the current position of the first data byte in the segment within the entire byte stream for the TCP connection.

ANS: sequence number

18. In the TCP segment header, ... is a 32-bit number identifying the next data byte the sender expects from the receiver.

ANS: acknowledgement number

19. A ..... is a special type of file handle, which is used by a process to request network services from the operating system.

ANS: socket

20. ..... is an optional 16-bit one's complement of the one's complement sum of a pseudo-IP header, the UDP header, and the UDP data.

ANS: Checksum

21. TCP groups a number of bytes together into a packet called a ....

ANS: Segment

22. A port address in TCP/IP is .....bits long.

ANS: 16

23. ..... is a process-to-process protocol that adds only port addresses, checksum error control, and length information to the data from upper layer.

ANS: UDP

24. UDP and TCP are both ..... layer protocols.

ANS: Transport

25. UDP packets have fixed-size header of ..... bytes.

ANS: 8

UNIT- IV Descriptive Questions

1. Write short notes on Transport layer
2. Write service primitives of transport layer
3. Explain congestion control in transport layer.
4. Explain connection management in transport layer.
5. Explain internet transport protocol TCP
6. Explain internet transport protocol UDP
7. Write about performance problems in computer Networks.
8. Explain network performance measurement.

UNIT-V

INTRODUCTION TO APPLICATION LAYER:

INTRODUCTION:

The application layer provides services to the user. Communication is provided using a logical connection, which means that the two application layers assume that there is an imaginary direct connection through which they can send and receive messages.

Providing Services:

All communication networks that started before the Internet were designed to provide services to network users. Most of these networks, however, were originally designed to provide one specific service. For example, the telephone network was originally designed to provide voice service: to allow people all over the world to talk to each other. This network, however, was later used for some other services, such as facsimile (fax), enabled by users adding some extra hardware at both ends.

The Internet was originally designed for the same purpose: to provide service to users around the world. The layered architecture of the TCP/IP protocol suite, however, makes the Internet more flexible than other communication networks such as postal or telephone networks.

Each layer in the suite was originally made up of one or more protocols, but new protocols can be added or some protocols can be removed or replaced by the Internet authorities. However, if a protocol is added to each layer, it should be designed in such a way that it uses the services provided by one of the protocols at the lower layer.

If a protocol is removed from a layer, care should be taken to change the protocol at the next higher layer that supposedly uses the services of the removed protocol. The application layer, however, is somewhat different from other layers in that it is the highest layer in the suite.

The protocols in this layer do not provide services to any other protocol in the suite; they only receive services from the protocols in the transport layer. This means that protocols can be removed from this layer easily. New protocols can be also added to this layer as long as the new protocols can use the services provided by one of the transport-layer protocols.

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.

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.

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.

Application-Layer Paradigms

It should be clear that to use the Internet we need two application programs to interact with each other: one running on a computer somewhere in the world, the other running on another computer somewhere else in the world. The two programs need to send messages to each other through the Internet infrastructure.

However, we have not discussed what the relationship should be between these programs.

Should both application programs be able to request services and provide services, or should the application programs just do one or the other?

Two paradigms have been developed during the lifetime of the Internet to answer this question: the client-server paradigm and the peer-to-peer paradigm.

Traditional Paradigm: Client-Server:

The traditional paradigm is called the **client-server paradigm**. It was the most popular paradigm until a few years ago. In this paradigm, the service provider is an application program, called the server process; it runs continuously, waiting for another application program, called the client process, to make a connection through the Internet and ask for service.

There are normally some server processes that can provide a specific type of service, but there are many clients that request service from any of these server processes. The server process must be running all the time; the client process is started when the client needs to receive service.

New Paradigm: Peer-to-Peer:

A new paradigm, called the **peer-to-peer paradigm** (often abbreviated *P2P paradigm*) has emerged to respond to the needs of some new applications.

In this paradigm, there is no need for a server process to be running all the time and waiting for the client processes to connect. The responsibility is shared between peers.

A computer connected to the Internet can provide service at one time and receive service at another time. A computer can even provide and receive services at the same time.

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.

Application Programming Interface:

A client process communicate with a server process with the help of a computer program which is normally written in a computer language with a predefined set of instructions that tells the computer what to do.

A computer language has a set of instructions for mathematical operations, a set of instructions for string manipulation, a set of instructions for input/output access, and so on.

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.

Several APIs have been designed for communication. One of the most common one is: **socket interface**. The socket interface is a set of instructions that provide communication between the application layer and the operating system, as shown in Figure 5.1.

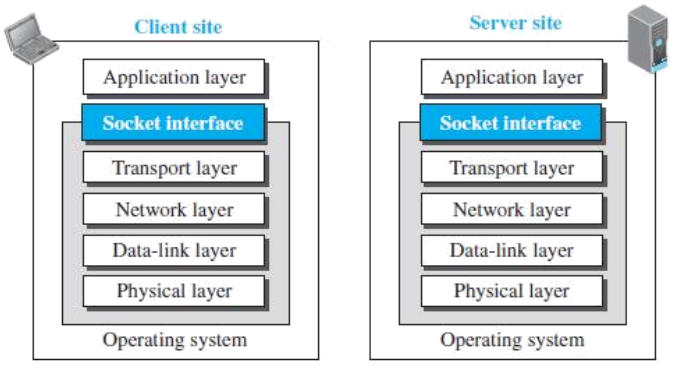


FIGURE 5.1: Position Of The Socket Interface

It is a set of instructions that can be used by a process to communicate with another process. The idea of sockets allows us to use the set of all instructions already designed in a programming language for other sources and sinks.

For example, in most computer languages, like C, C++, or Java, we have several instructions that can read and write data to other sources and sinks such as a keyboard (a source), a monitor (a sink), or a file (source and sink). We can use the same instructions to read from or write to sockets.

Sockets:

Although a socket is supposed to behave like a terminal or a file, it is not a physical entity like them; it is an abstraction. It is an object that is created and used by the application program.

Socket Addresses:

The interaction between a client and a server is two-way communication. In a two-way communication, we need a pair of addresses: local (sender) and remote (receiver). The local address in one direction is the remote address in the other direction and vice versa.

Since communication in the client-server paradigm is between two sockets, we need a pair of **socket addresses** for communication: a local socket address and a remote socket address. However, we need to define a socket address in terms of identifiers used in the TCP/IP protocol suite.

A socket address should first define the computer on which a client or a server is running. Socket address should be a combination of an IP address (32 bit) and a port number (16 bit).

Since a socket defines the end-point of the communication, we can say that a socket is identified by a pair of socket addresses, a local and a remote.

Finding Socket Addresses: How can a client or a server find a pair of socket addresses for communication? The situation is different for each site.

Server Site: The server needs a local (server) and a remote (client) socket address for communication.

Local Socket Address The local (server) socket address is provided by the operating system. The operating system knows the IP address of the computer on which the server process is running. The port number of a server process, however, needs to be assigned.

If the server process is a standard one defined by the Internet authority, a port number is already assigned to it. For example, the assigned port number for a Hypertext Transfer Protocol (HTTP) is the integer 80, which cannot be used by any other process.

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.

Client Site: The client also needs a local (client) and a remote (server) socket address for communication.

Local Socket Address The local (client) socket address is also provided by the operating system. The operating system knows the IP address of the computer on which the client is running. The port number, however, is a 16-bit temporary integer that is assigned to a client process each time the process needs to start the communication.

The port number, however, needs to be assigned from a set of integers defined by the Internet authority and called the ephemeral (temporary) port numbers. The operating system, however, needs to guarantee that the new port number is not used by any other running client process.

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.

Using Services of the Transport Layer:

A pair of processes provide services to the users of the Internet, human or programs. A pair of processes, however, need to use the services provided by the transport layer for communication because there is no physical communication at the application layer.

WORLD WIDE WEB AND HTTP:

World Wide Web:

The idea of the Web was first proposed by Tim Berners-Lee in 1989. 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*. 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.

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 pagecan 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 5.2).

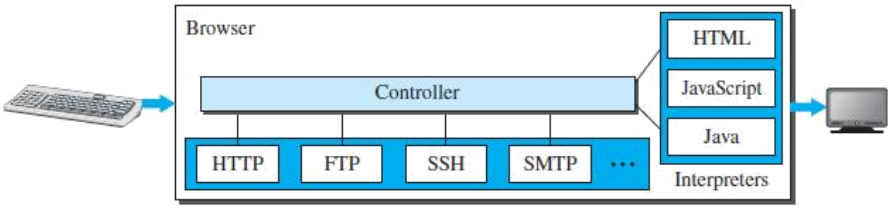


Figure 5.2: Browser

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.

***Port*.** The port, a 16-bit integer, is normally predefined for the client-server application.

***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.

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.

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.

Static documents are prepared using one of several languages: *HyperText Markup Language* (HTML), *Extensible Markup Language* (XML), *Extensible Style Language* (XSL), and *Extensible Hypertext Markup Language* (XHTML).

Dynamic Documents:

A **dynamic document** is created by a web server whenever a browser requests the document. When a request arrives, the web server runs an application program or a script that creates the dynamic document.

The server returns the result of the program or script as a response to the browser that requested the document. Because a fresh document is created for each request, the contents of a dynamic document may vary from one request to another. A very simple example of a dynamic document is the retrieval of the time and date from a server.

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.

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.

Nonpersistent versus Persistent Connections:

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*.

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.

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. Each message is made of four sections. The first section in the request message is called the *request line;* the first section in the response message is called the *status line*.

The other three sections have the same names in the request and response messages. However, the similarities between these sections are only in the names; they may have different contents. We discuss each message type separately.

Request Message:

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. Several methods are defined like GET, PUT, HEAD, POST, TRACE, DELETE, etc. The URL defines the address and name of the corresponding web page. The version field gives the version of the protocol; the most current version of HTTP is 1.1.

Response Message:

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.

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.

HTTP Security:

HTTP per se does not provide security. HTTP can be run over the Secure Socket Layer (SSL). In this case, HTTP is referred to as HTTPS. HTTPS provides confidentiality, client and server authentication, and data integrity.

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.

Although we can transfer files using HTTP, FTP is a better choice to transfer large files or to transfer files using different formats. Figure 5.3 shows the basic model of FTP. The client has three components: the user interface, the client control process, and the client data transfer process. The server has two components: the server control process and the server data transfer process.

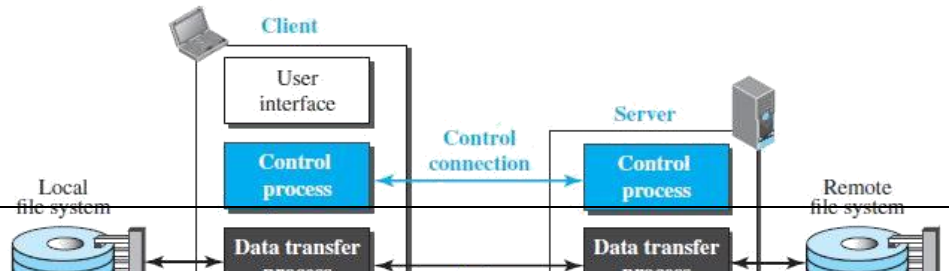


Figure 5.3: FTP

The control connection is made between the control processes. The data connection is made between the data transfer processes. Separation of commands and data transfer makes FTP more efficient. The control connection uses very simple rules of communication. We need to transfer only a line of command or a line of response at a time. The data connection, on the other hand, needs more complex rules due to the variety of data types transferred.

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.

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:

During this control connection, commands are sent from the client to the server and responses are sent from the server to the client. Commands, which are sent from the FTP client control process, are in the form of ASCII uppercase, which may or may not be followed by an argument. Some of the most common commands are shown in table below:

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

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.

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

ELECTRONIC MAIL:

Electronic mail (or e-mail) allows users to exchange messages. The nature of this application, however, is different from other applications discussed so far. In an application such as HTTP or FTP, the server program is running all the time, waiting for a request from a client. When the request arrives, the server provides the service. There is a request and there is a response.

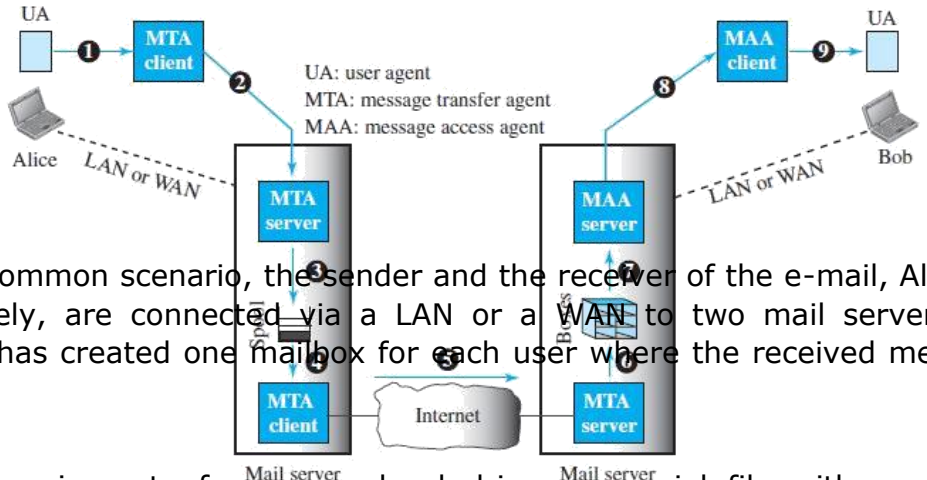
In the case of electronic mail, the situation is different. First, e-mail is considered a one-way transaction. When Alice sends an email to Bob, she may expect a response, but this is not a mandate. Bob may or may not respond. If he does respond, it is another one-way transaction.

Second, it is neither feasible nor logical for Bob to run a server program and wait until someone sends an e-mail to him. Bob may turn off his computer when he is not using it.

This means that the idea of client/server programming should be implemented in another way: using some intermediate computers (servers). 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 as shown in Figure 5.4.



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. 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.

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.

Message Transfer Agent: SMTP: Based on the common scenario, we can say that the e-mail is one of those applications that needs three uses of client-server paradigms to accomplish its task. It is important that we distinguish these three when we are dealing with e-mail.

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.

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.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.

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.

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.

TELNET:

A server program can provide a specific service to its corresponding client program. For example, the FTP server is designed to let the FTP client store or retrieve files on the server site. However, it is impossible to have a client/server pair for each type of service we need; the number of servers soon becomes intractable which is not scalable.

Another solution is to have a specific client/server program for a set of common scenarios, but to have some generic client/server programs that allow a user on the client site to log into the computer at the server site and use the services available there.

For example, if a student needs to use the Java compiler program at her university lab, there is no need for a Java compiler client and a Java compiler server. The student can use a client logging program to log into the university server and use the compiler program at the university. We refer to these generic client/server pairs as ***remote logging*** applications.

One of the original remote logging protocols is **TELNET,** which is an abbreviation for *TErminaL NETwork.* Although TELNET requires a logging name and password, it is vulnerable to hacking because it sends all data including the password in plaintext (not encrypted).

A hacker can eavesdrop and obtain the logging name and password. Because of this security issue, the use of TELNET has diminished in favor of another protocol, Secure Shell (SSH).

Although TELNET is almost replaced by SSH, we briefly discuss TELNET here for two reasons:

1. The simple plaintext architecture of TELNET allows us to explain the issues and challenges related to the concept of remote logging, which is also used in SSH when it serves as a remote logging protocol.

2. Network administrators often use TELNET for diagnostic and debugging purposes.

Local versus Remote Logging:

When a user logs into a local system, it is called *local logging*. As a user types at a terminal or at a workstation running a terminal emulator, the keystrokes are accepted by the terminal driver.

The terminal driver passes the characters to the operating system. The operating system, in turn, interprets the combination of characters and invokes the desired application program or utility.

However, 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.

NVT uses two sets of characters, one for data and one for control. Both are 8-bit bytes. 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.

User Interface:

The operating system (UNIX, for example) defines an interface with user-friendly commands. An example of such a set of commands can be found in Table beow:

Command Name	Meaning
open	Connect to a remote computer
close	Close the connections
display	Show the operating parameters
mode	Change to line or character mode
Quit	Exit TELNET
send	Send special characters

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. In this section, we discuss only SSH-2.

Components: SSH is an application-layer protocol with three components.

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 are:

1. Privacy or confidentiality of the message exchanged.

2. Data integrity, which means that it is guaranteed that the messages exchanged between the client and server are not changed by an intruder.

3. Server authentication, which means that the client is now sure that the server is the one that it claims to be.

4. Compression of the messages, which improves the efficiency of the system and makes attack more difficult.

SSH Authentication Protocol (SSH-AUTH):

After a secure channel is established between the client and the server and the server is authenticated for the client, SSH can call another procedure that can authenticate the client for the server. The client authentication process in SSH is very similar to what is done in Secure Socket Layer (SSL).

This layer defines a number of authentication tools similar to the ones used in SSL. Authentication starts with the client, which sends a request message to the server. The request includes the user name, server name, the method of authentication, and the required data. The server responds with either a success message, which confirms that the client is authenticated, or a failed message, which means that the process needs to be repeated with a new request message.

SSH Connection Protocol (SSH-CONN):

After the secured channel is established and both server and client are authenticated for each other, SSH can call a piece of software that implements the third protocol, SSHCONN.

One of the services provided by the SSH-CONN protocol is multiplexing. SSH-CONN takes the secure channel established by the two previous protocols and lets the client create multiple logical channels over it. Each channel can be used for a different purpose, such as remote logging, file transfer, and so on.

Applications:

Although SSH is often thought of as a replacement for TELNET, SSH is, in fact, a general-purpose protocol that provides a secure connection between a client and server.

SSH for Remote Logging:

Several free and commercial applications use SSH for remote logging. Among them, we can mention PuTTy, by Simon Tatham, which is a client SSH program that can be used for remote logging. Another application program is Tectia, which can be used on several platforms.

SSH for File Transfer:

One of the application programs that is built on top of SSH for file transfer is the *Secure File Transfer Program* (*sftp*). The *sftp* application program uses one of the channels provided by the SSH to transfer files. Another common application is called *Secure Copy* (*scp*). This application uses the same format as the UNIX copy command, *cp,* to copy files.

DOMAIN NAME SYSTEM (DNS):

Since the Internet is so huge today, a central directory system cannot hold all the mapping. In addition, if the central computer fails, the whole communication network will collapse.

A better solution is to distribute the information among many computers in the world. In this method, the host that needs mapping can contact the closest computer holding the needed information. This method is used by the **Domain Name System (DNS)**.

Figure 5.5 shows how TCP/IP uses a DNS client and a DNS server to map a name to an address. A user wants to use a file transfer client to access the corresponding file transfer server running on a remote host. The user knows only the file transfer server name, such as *afilesource.com.*

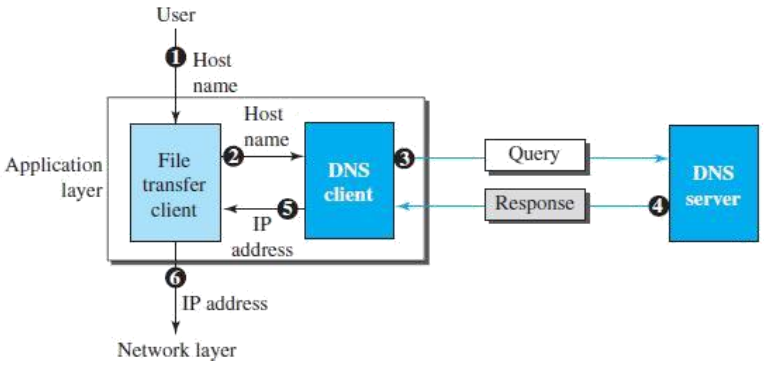


Figure 5.5: Purpose of DNS

Name Space:

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.

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 (see Figure 5.6).

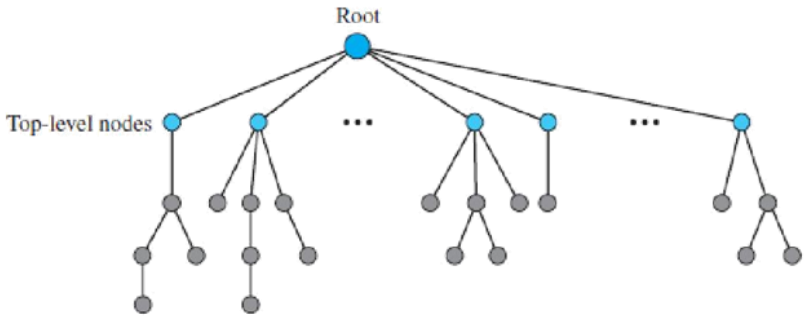


Figure 5.6: Domain name space

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 5.7 shows some domain names.

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 5.8 shows some domains. Note that a domain may itself be divided into domains.

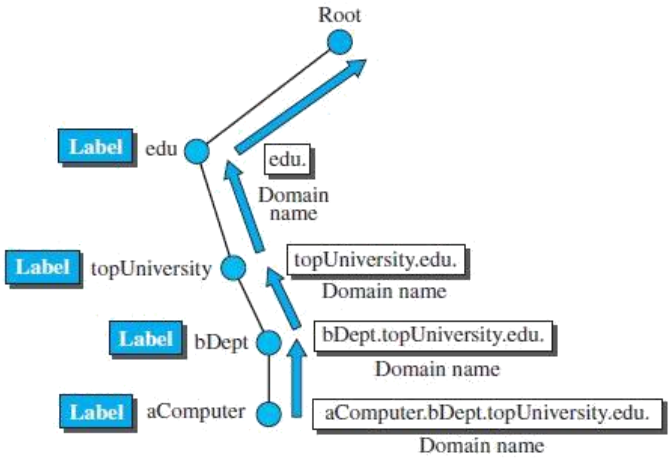


Figure 5.7: Domain names and labels

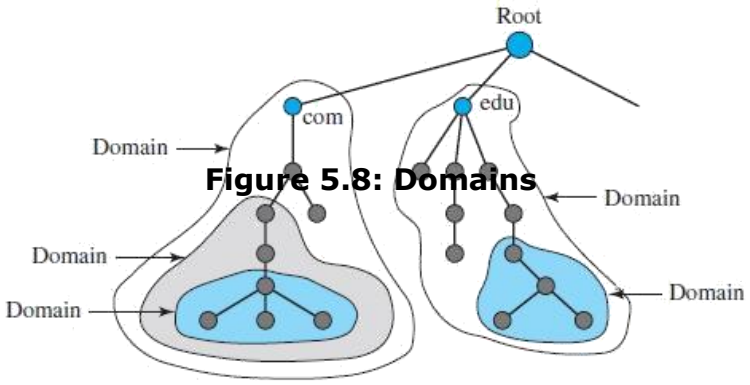


Figure 5.8: Domains

SNMP:

Several network management standards have been devised during the last few decades. The most important one is **Simple Network Management Protocol (SNMP),** used by the Internet.

SNMP is a framework for managing devices in an internet using the TCP/IP protocol suite. It provides a set of fundamental operations for monitoring and maintaining an internet. SNMP uses the concept of manager and agent. That is, a manager, usually a host, controls and monitors a set of agents, usually routers or servers (see Figure 5.9).

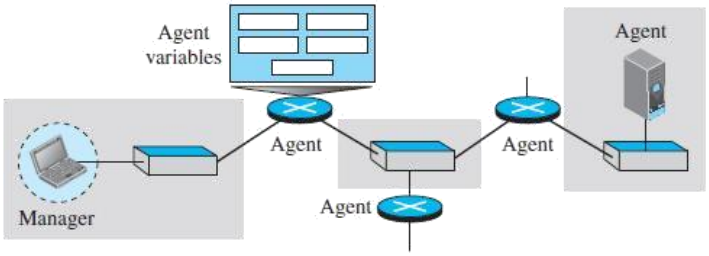


Figure 5.9: SNMP concept

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).**

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).

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. Part of a name can be inherited from the parent. We also need rules to define the types of objects.

Role of 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 & types without values).

Unit-V

1. The _____ translates internet domain and host names to IP address.
a) domain name system b) routing information protocol
c) network time protocol d) internet relay chat

2. Which one of the following allows a user at one site to establish a connection to another site and then pass keystrokes from local host to remote host?
a) HTTP b) FTP **c) telnet** d) none of the mentioned

3. Application layer protocol defines
a) types of messages exchanged b) message format, syntax and semantics
c) rules for when and how processes send and respond to messages **d) all of the mentioned**

4. Which one of the following protocol delivers/stores mail to reciever server?
a) simple mail transfer protocol b) post office protocol
c) internet mail access protocol d) hypertext transfer protocol

5. The ASCII encoding of binary data is called
a) base 64 encoding b) base 32 encoding c) base 16 encoding d) base 8 encoding

6. Which one of the following is an internet standard protocol for managing devices on IP network?
a) dynamic host configuration protocol **b) simple newtwork management protocol**
c) internet message access protocol d) media gateway protocol

7. Which one of the following is not an application layer protocol?
a) media gateway protocol b) dynamic host configuration protocol
c) resource reservation protocol d) session initiation protocol

8. Which one of the following is not correct?
a) application layer protocols are used by both source and destination devices during a communication session
b) application layer protocols implemented on the source and destination host must match
c) both (a) and (b)
d) none of the mentioned

9. When displaying a web page, the application layer uses the
a) HTTP protocol b) FTP protocol c) SMTP protocol d) none of the mentioned

10. This is not a application layer protocol
a) HTTP b) SMTP c) FTP **d) TCP**

11. The packet of information at the application layer is called
a) Packet **b) Message** c) Segment d) Frame

12) This is one of the architecture paradigm
a) Peer to peer b) Client-server c) HTTP **d) Both a and b**

13) Application developer has permission to decide the following on transport layer side
a) Transport layer protocol b) Maximum buffer size **c) Both of the mentioned** d) None

14) Application layer offers _____ service
a) End to end b) Process to process c) Both of the mentioned d) None

15) E-mail is
a) Loss-tolerant application b) Bandwidth-sensitive application
c) Elastic application d) None of the mentioned

16) Pick the odd one out
a) File transfer b) File download c) E-mail **d) Interactive games**

17) Which of the following is an application layer service ?
a) Network virtual terminal b) File transfer, access, and management
c) Mail service **d) All of the mentioned**

18) To deliver a message to the correct application program running on a host, the _____ address must be consulted
a) IP b) MAC **c) Port** d) None of the mentioned

19) This is a time-sensitive service
a) File transfer b) File download c) E-mail **d) Internet telephony**

20) Electronic mail uses this Application layer protocol
a) SMTP b) HTTP c) FTP d) SIP

UNIT- V Descriptive Questions

1. Give brief description about the architecture and services of an e – mail.
2. Explain the name servers with a neat sketch.
3. Explain the RFC 822 formats with suitable example.
4. Discuss in detail about the DNS name space.
5. Write a short note on resource records.
6. Draw and explain the architecture of WAP.
7.Explain the connections, methods and message header of HTTP.