UNIT -1

I**ntroduction to data communications**: Components, Networks, Network Types, Protocol Layering, TCP/IP Protocol Suite, The OSI Model. (Chapter 1)

Physical Layer: Signals, Signal Impairment, Digital Transmission, Analog Transmission, Multiplexing. (Chapter 2)
Data-Link Layer: **Data-link control:** Framing, Error Control.

Media Access Protocols: Carrier Sense Multiple Access, CSMA/CD. Link-Layer Addressing: Three Types of Addresses (Chapter 3)

Local Area Networks: Ethernet, Standard Ethernet Frame Format (Chapter 4)

CHAPTER 1

1.1 DATA COMMUNICATIONS

When we communicate, we are sharing information or data. This sharing can be local or remote. Local communication usually occurs face to face, while remote communication takes place over a distance. The word ***data*** refers to information presented in whatever form is agreed upon by the parties creating and using it.

Data communications is the exchange of data between two devices via some form of transmission medium such as a wire cable. For data communications to occur, the communicating devices must be part of a communications system made up of a combination of hardware (physical equipment) and software (programs). **The effectiveness of a data communications system depends on four fundamental characteristics:** delivery, accuracy, timeliness, and jitter.

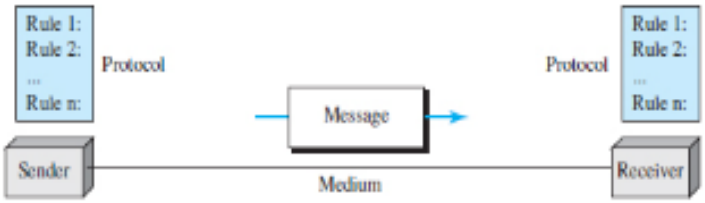
1. **Delivery.** The system must deliver data to the correct destination. Data must be received by the intended device or user and only by that device or user.
2. **Accuracy.** The system must deliver the data accurately. Data that have been altered in transmission and left uncorrected are unusable**.**
3. **Timeliness.** The system must deliver data in a timely manner. Data delivered late are useless. In the case of video and audio, timely delivery means delivering data as they are produced, in the same order that they are produced, and without significant delay. This kind of delivery is called *real-time* transmission.
4. **Jitter.** Jitter refers to the variation in the packet arrival time. It is the uneven delay in the delivery of audio or video packets. For example, let us assume that video packets are sent every 30 ms. If some of the packets arrive with a 30-ms delay and others with a 40-ms delay, the video will have an uneven quality.

1.1.1 Components

A data communications system has five components (see Figure 1.1).

Figure 1.1 *Five components of a data communications system*

Figure 1.1 *Five components of a data communications system*

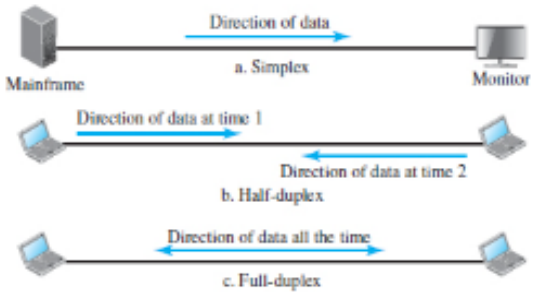


1. **Message.** The message is the information (data) to be communicated. Popular forms of information include text, numbers, pictures, audio, and video.
2. **Sender.** The sender is the device that sends the data message. It can be a computer, a telephone handset, a video camera, and so on.
3. **Receiver.**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

1.1.3 Data Flow

Communication between two devices can be simplex, half-duplex, or full-duplex as shown in Figure 1.2.

Figure 1.2 *Data flow (simplex, half-duplex, and full-duplex)*



Simplex

In **simplex mode,** the communication is unidirectional, as on a one-way street. Only one of the two devices on a link can transmit; the other can only receive (see Figure 1.2a). Keyboards and traditional monitors are examples of simplex devices. The keyboard can only introduce input; the monitor can only accept output.

Half-Duplex

In **half-duplex mode**, each station can both transmit and receive, but not at the same time. When one device is sending, the other can only receive, and vice versa (see Figure 1.2b). Walkie-talkies and CB (citizens band) radios are both half-duplex systems.

Full-Duplex

In **full-duplex mode**, both stations can transmit and receive simultaneously (see Figure 1.2c). The full-duplex mode is like a two-way street with traffic flowing in both directions at the same time. In full-duplex mode, signals going in one direction share the capacity of the link with signals going in the other direction. One common example of full-duplex communication is the telephone network. When two people are communicating by a telephone line, both can talk and listen at the same time.

1.2 NETWORKS

A **network** is the interconnection of a set of devices capable of communication. In this definition, a device can be a **host**, 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 that connects the network to other networks, a switch that connects devices together, or a modem (modulator-demodulator) that changes the form of data.

1.2.1 Network Criteria

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

Performance

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

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.

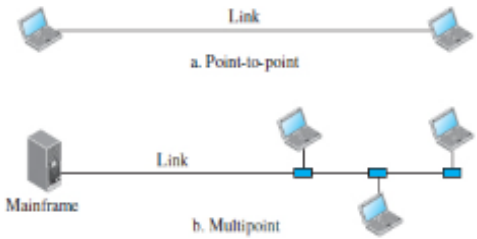
1.2.2 Physical Structures

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 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* (see Figure 1.3 on next page).

Figure 1.3 *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.

Multipoint

A **multipoint** (also called **multidrop**) **connection** is one in which more than two devices share a single link. In a multipoint environment, the capacity of the channel is shared, either spatially or temporally. If several devices can use the link simultaneously, it is a *spatially shared* connection. If users must take turns, it is a *timeshared* connection.

Physical Topology

The term ***physical topology*** refers to the way in which a network is laid out physically. Two or more devices connect to a link; two or more links form a topology. The topology of a network is the geometric representation of the relationship of all the links and linking devices (usually called **nodes**) to one another. There are four basic topologies possible: *mesh*, *star*, *bus*, and *ring*.

Mesh Topology

In a **mesh topology**, every device has a dedicated point-to-point link to every other device. The term *dedicated* means that the link carries traffic only between the two devices it connects. To find the number of physical links in a fully connected mesh network with n nodes, we first consider that each node must be connected to every other node. Node 1 must be connected to $n - 1$ nodes,

node 2 must be connected to $n - 1$ nodes, and finally node n must be connected to $n - 1$ nodes. We need $n(n - 1)$ physical links. To accommodate that many links, every device on the network must have $n - 1$ input/output (I/O) ports (see Figure 1.4 on next page) to be connected to the other $n - 1$ stations.

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 1.5).

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

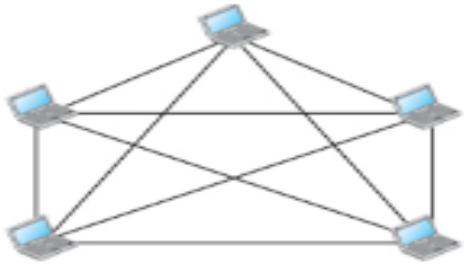
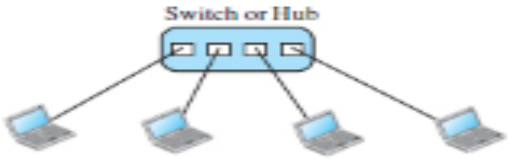


Figure 1.5 *A star topology connecting four stations*



Bus Topology

The preceding topology 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 1.6).

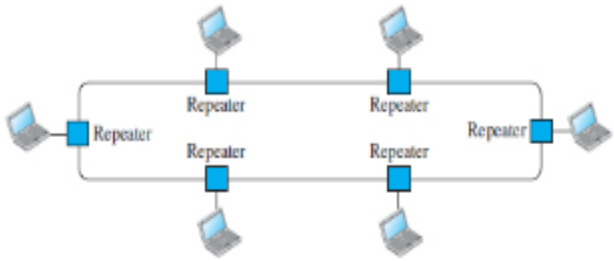
Figure 1.6 *A bus topology connecting three stations*

Nodes are connected to the bus cable by drop lines and taps. A *drop line* is a connection running between the device and the main cable. A *tap* is a connector that either splices into the main cable or punctures the sheathing of a cable to create a contact with the metallic core.

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, which regenerates the bits and passes them along (see Figure 1.7).

Figure 1.7 *A ring topology connecting six stations*



1.3 NETWORK TYPES

Now we discuss different types of networks: LANs and WANs.

1.3.1 Local Area Network

A **local area network (LAN)** is usually privately owned and connects some hosts in a single office, building, or campus.

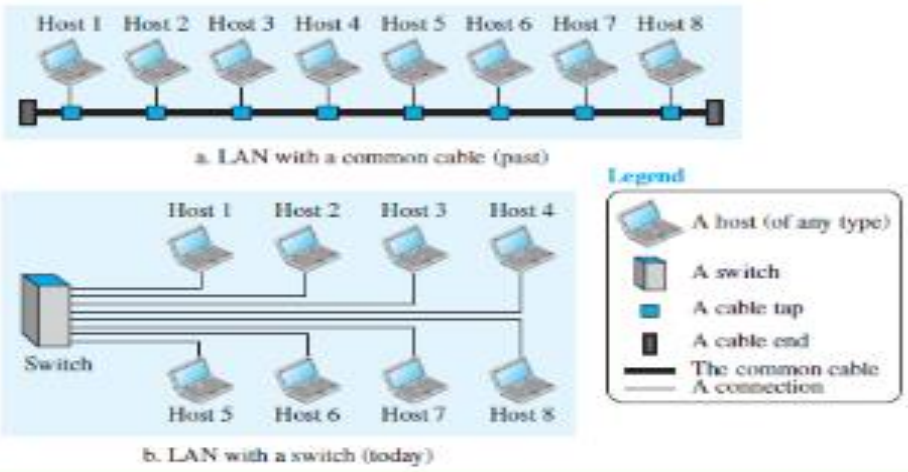
Each host in a LAN has an identifier, which is an address that uniquely defines the host in the LAN. A packet sent by a host to another host carries both the source host's and the destination host's addresses.

When LANs were used in isolation (which is rare today), they were designed to allow resources to be shared between the hosts. As we will see shortly, LANs today are connected to each other and to WANs (discussed next) to create communication at a wider level (see Figure 1.8 on next page).

1.3.2 Wide Area Network (WAN)

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

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



Point-to-Point WAN

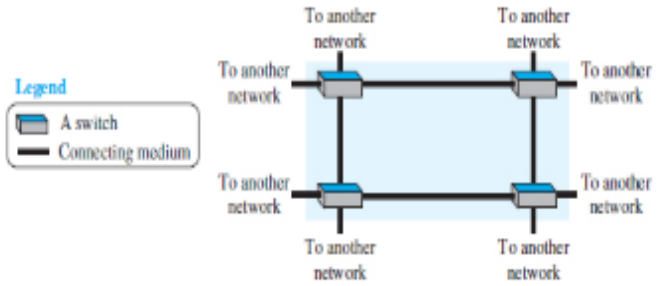
A point-to-point WAN is a network that connects two communicating devices through a transmission medium (cable or air). Figure 1.9 shows an example of a point-to-point WAN.

Figure 1.9 *A point-to-point WAN*

Switched WAN

A switched WAN is a network with more than two ends. It is used in the backbone of a global communications network today. Figure 1.10 shows an example of a switched WAN.

Figure 1.10 *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 1.11 shows this internet.

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

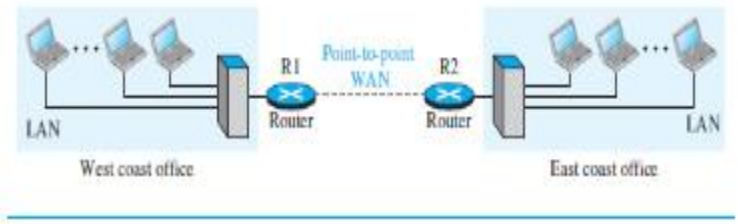
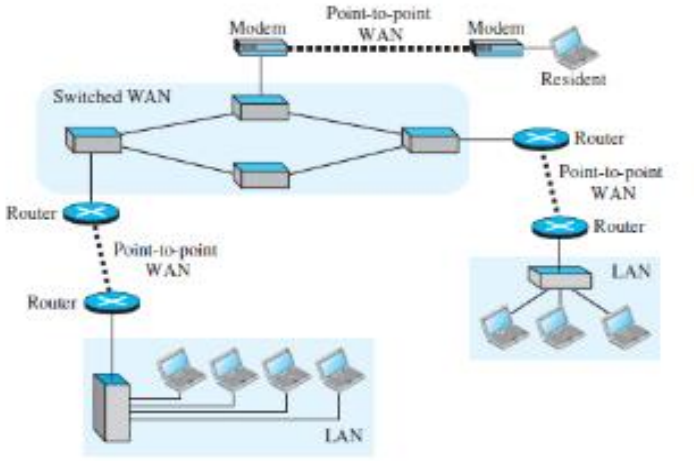


Figure 1.12 shows another internet with several LANs and WANs connected. One of the WANs is a switched WAN with four switches.

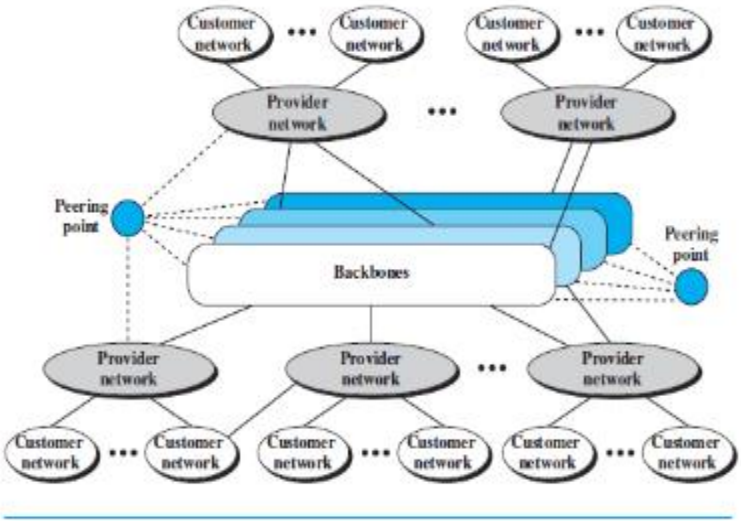
Figure 1.12 *A heterogeneous internetwork made of four WANs and two LANs*



1.3.3 The Internet

As we discussed before, 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 (fiber optical cable with high end routers to transfer data in the network efficiently within geographic regions i.e. Tata communications, AT&T etc). , provider networks, and customer networks. At the top level, the *backbones* are large networks owned by some communication companies. 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.*

1.3.4 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 (such as a telephone network, a cable network, a wireless network, or other types of networks).

Using Telephone Networks

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

- ***Dial-up service.*** The first solution is to add a modem that converts data to voice to the telephone line. 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 an Internet connection, it cannot be used for a telephone (voice) connection. It is only useful for small residences and businesses with occasional connection to the Internet.

- ***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 digital subscriber line (DSL) service also allows the line to be used simultaneously for voice and data communications.

Using Cable Networks

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

Using Wireless Networks

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

Direct Connection to the Internet

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

1.4 PROTOCOL LAYERING

We defined the term *protocol* before. In data communications 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**.

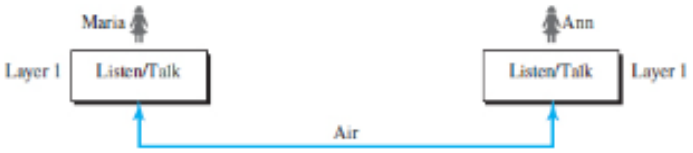
1.4.1 Scenarios

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 1.14.

Figure 1.14 *A single-layer protocol*



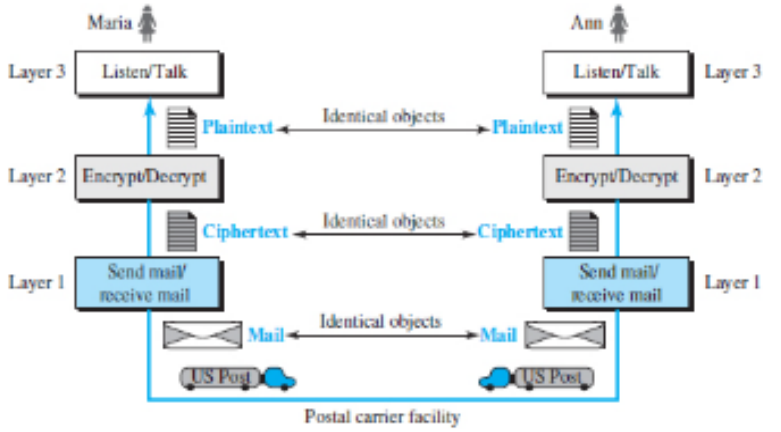
Even in this simple scenario, we can see that a set of rules needs to be followed. First, Maria and Ann know that they should greet each other when they meet. Second, they know that they should confine their vocabulary to the level of their friendship. Third, each party knows that she should refrain from speaking when the other party is speaking. Fourth, each party knows that the conversation should be a dialog, not a monolog: Both should have the opportunity to talk about the issue. Fifth, they should exchange some nice words when they leave.

We can see that the protocol used by Maria and Ann is different from the communication between a professor and the students in a lecture hall. The communication in the second case is mostly monolog; the professor talks most of the time unless a student has a question, a situation in which the protocol dictates that she should raise her hand and wait for permission to speak. In this case, the communication is normally very formal and limited to the subject being taught.

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 conversations using regular mail through the post office. However, they do not want their ideas to be revealed to other people if the letters are intercepted. They agree on an encryption/decryption technique. The sender of the letter encrypts it to make it unreadable by an intruder; the receiver of the letter decrypts it to get the original letter. We discuss the encryption/decryption methods later in the book, but for the moment we assume that Maria and Ann use one technique to make 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 1.15. We assume that Ann and Maria each have three machines (or robots) that can perform the task at each layer.

Figure 1.15 *A three-layer protocol*



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 ciphertext, which is passed to the first-layer machine. The first-layer machine, presumably a robot, takes the ciphertext, 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 mailbox, recognizing the letter from Maria by the sender address. The machine takes out the ciphertext 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. For example, in Figure 1.15, we could have used only one machine to do the job of all three machines. However, if Maria and Ann decide that the encryption/decryption done by the machine is not enough to protect their secrecy, they have to change the whole machine. In the present situation, they need to change only the second-layer machine; the other two can remain the same. This is referred to as *modularity*. Modularity in this case means independent layers. A layer (module) can be defined as a black box with inputs and outputs, without concern about how inputs are changed to outputs. If two machines provide the same outputs when given the same inputs, they can replace each other. For example, Ann and Maria can buy the second-layer machine from two different manufacturers. As long as the two machines create the same ciphertext from the same plaintext and vice versa, they do the job.

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. For example, Maria may decide not to buy the machine (robot) for the first layer; she can do the job herself. As long as Maria can do the tasks provided by the first layer, in both directions, the communications system works.

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.

Is there any disadvantage to protocol layering? One can argue that having a single layer makes the job easier. There is no need for each layer to provide a service to the upper layer and give service to the lower layer. For example, Ann and Maria could find or build one machine that could do all three tasks. However, as mentioned above, if one day they found that their code was broken, each would have to replace the whole machine with a new one instead of just changing the machine in the second layer.

1.4.2 Principles of Protocol Layering

Let us discuss the 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.

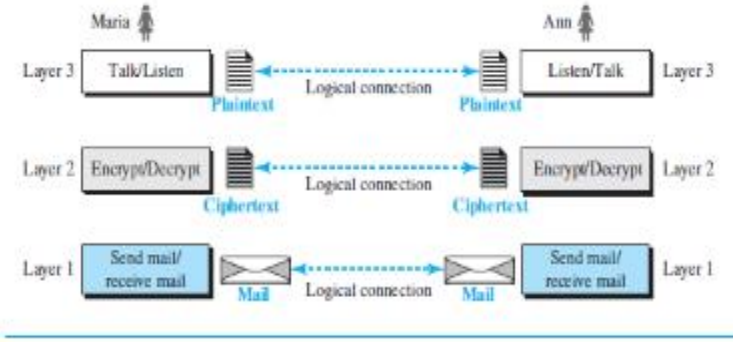
Second Principle

The second important 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 the third layer at both sites should be a plaintext letter. The object under the second layer at both sites should be a ciphertext letter. The object under the first layer at both sites should be a piece of mail.

1.4.3 Logical Connections

After following the above two principles, we can think about logical connections between each layer as shown in Figure 1.16. 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. We will see that the concept of logical connection will help us better understand the task of layering we encounter in data communications and networking.

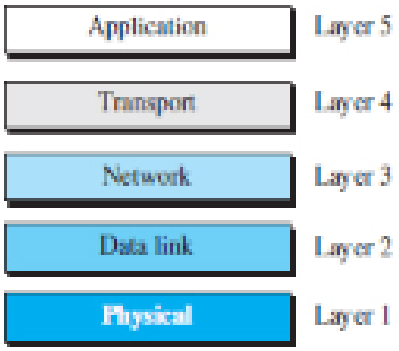
Figure 1.16 *Logical connections between peer layers*



1.5 TCP/IP PROTOCOL SUITE

Now that we know about the concept of protocol layering and the logical connections between layers in our second scenario, we can introduce the **Transmission Control Protocol/Internet Protocol (TCP/IP).** 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 **TCP/IP protocol suite** is defined as five layers as shown in Figure 1.17.

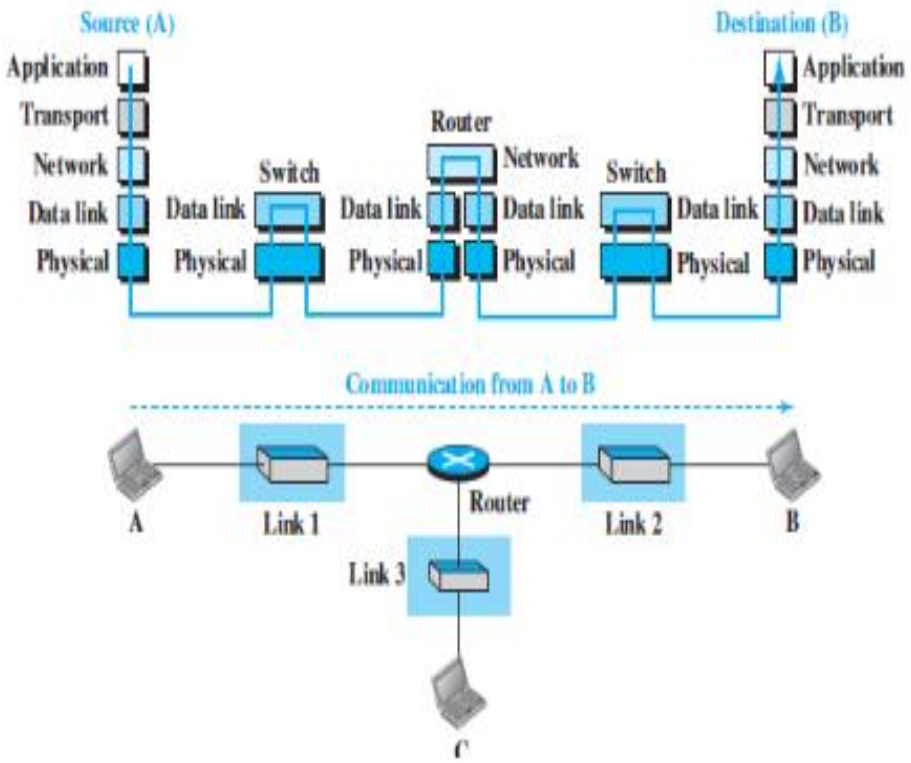
Figure 1.17 *Layers in the TCP/IP protocol suite*



1.5.1 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 1.18 .

Figure 1.18 *Communication through an internet*



Let us assume that computer A communicates with computer B. As Figure 1.18 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 only in three layers; there is no transport or application layer in a router as long as the router is used only for routing. Although a router is always involved in one network layer, it is involved in *n* combinations of link and physical layers in which *n* is the number of links the router is connected to. The reason is that each link may use its own data-link or physical protocol. For example, in Figure 1.18, the router is involved in three links, but the message sent from source computer A to destination computer B is involved in two links. Each link may be using different link-layer and physical-layer protocols; the router needs to receive a packet from link 1 based on one pair of protocols and deliver it to link 2 based on another pair of protocols.

A link-layer switch in a link, however, is involved only in two layers, data-link and physical protocols. Although each switch in Figure 1.18 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.

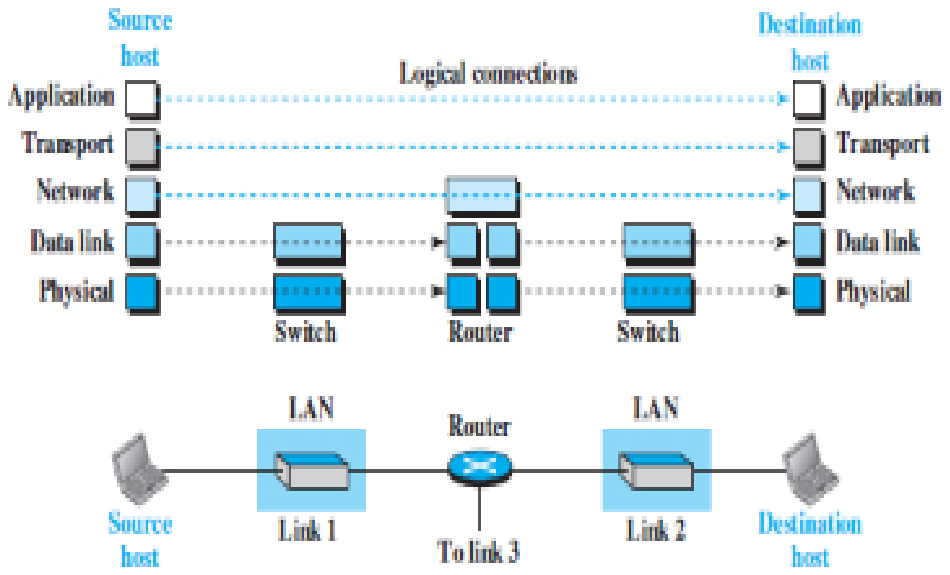
1.5.2 Brief Description of Layers

We now briefly discuss the functions and duties of layers in the TCP/IP protocol suite. Each layer is discussed in detail in a separate chapter of the book. To better understand the duties of each layer, we need to think about the logical connections between the layers. Figure 1.19 shows the logical connections in our simple internet.

Using logical connections makes it easier for us to think about the duty of each layer. As Figure 1.19 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.

Figure 1.19 *Logical connections between layers of the TCP/IP protocol suite*

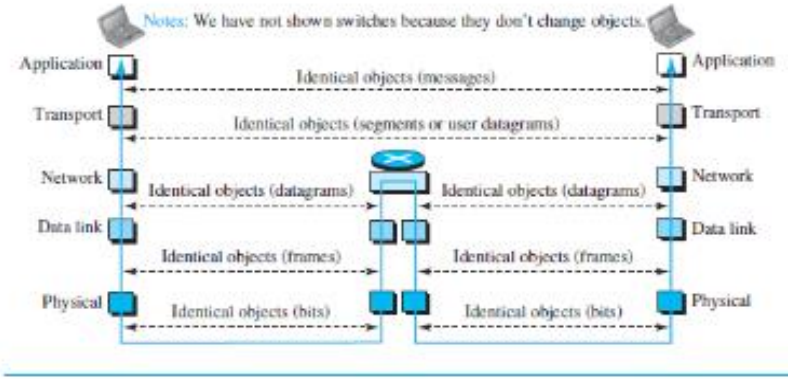
Figure 1.19 Logical connections between layers of the TCP/IP protocol suite



Another way of thinking about 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 1.20 shows the second principle discussed previously for protocol layering. We show the identical objects below each layer related to each device.

Figure 1.20 *Identical objects in the TCP/IP protocol suite*



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. (See Chapter 4 for a discussion of fragmentation.) Note that the link between two hops does not change the object.

1.5.3 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. 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. We discuss the physical layer in Chapter 2.

Data-Link Layer

We have seen that an *internet* is made up of several links (LANs and WANs) connected by routers. 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. We discuss the data-link layer in Chapters 3, 4, 5, and 6.

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, because 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 discuss the network layer in Chapters 7 and 8.

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 (imaginary) connection, to the transport layer at the destination host. In other words, the transport layer is responsible for giving services to the application layer: to get a message from an application program running on the source host and deliver it to the corresponding application program on the destination host. We need to say that we discuss the transport layer in Chapter 9.

Application Layer

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. We discuss the application layer in Chapter 10.

1.6 THE OSI MODEL

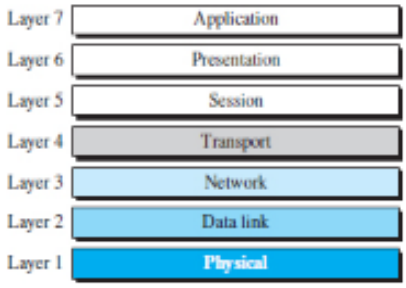
Although, when speaking of the Internet, everyone talks about the TCP/IP protocol suite, it is not the only suite of protocols defined. Established in 1947, the **International Organization for Standardization (ISO)** is a multinational body dedicated to worldwide agreement on international standards. 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.

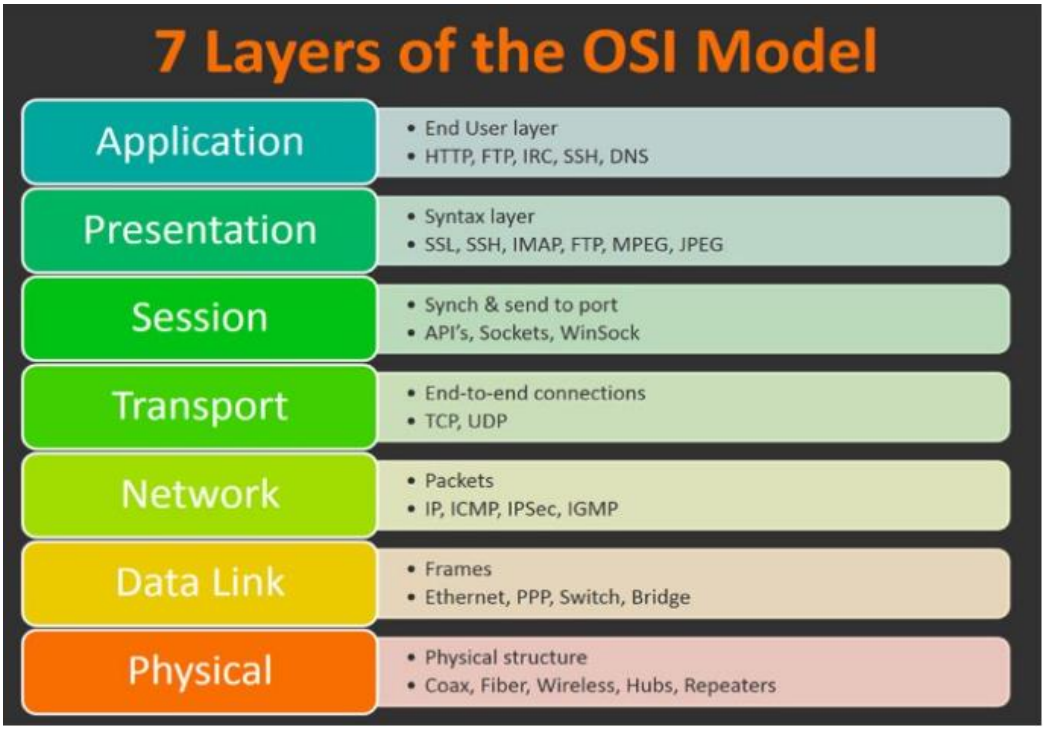
ISO is the organization; OSI is the model.

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 1.21).

Figure 1.21 *The OSI model*

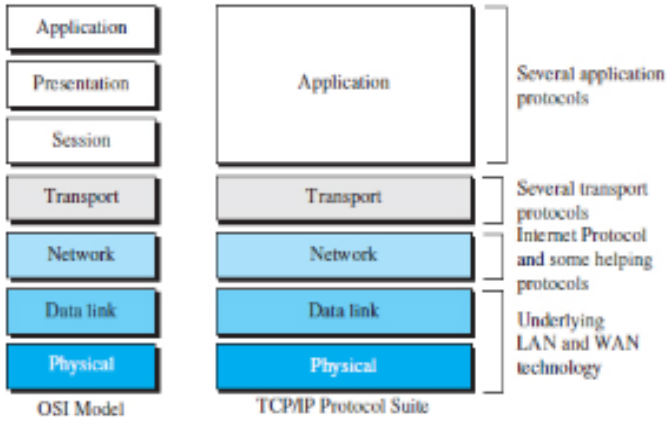




1.6.1 OSI versus TCP/IP

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 1.22.

Figure 1.22 *TCP/IP and OSI model*



Two reasons were mentioned for this decision. First, TCP/IP has more than one transport-layer protocol. Some of the functionalities of the session layer are available in some of the transport-layer protocols. Second, the application layer is not only one piece of software. Many applications can be developed at this layer. If some of the functionalities mentioned in the session and presentation layers are needed for a particular application, they can be included in the development of that piece of software.

1.6.2 Lack of OSI Model's Success

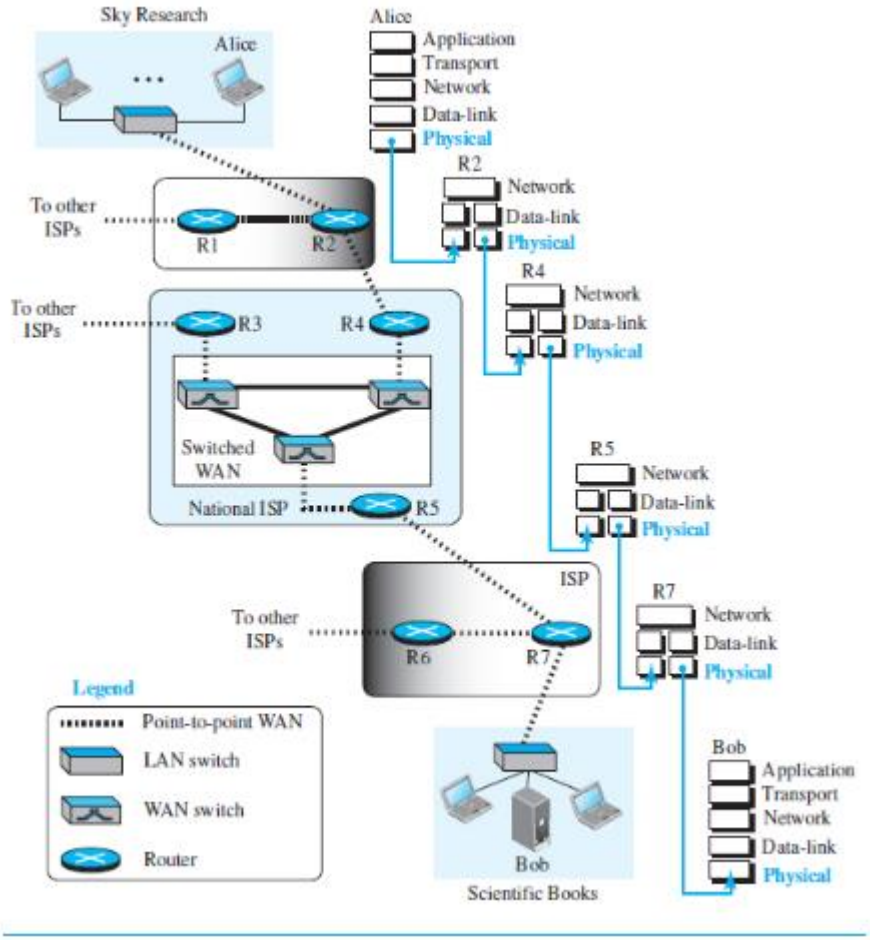
The OSI model appeared after the TCP/IP protocol suite. Most experts were at first excited and thought that the TCP/IP protocol would be fully replaced by the OSI model. This did not happen for several reasons, but we describe only three, which are agreed upon by all experts in the field. First, **OSI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

CHAPTER 2

PHYSICAL LAYER

Figure 2.1 shows a scenario in which a scientist, Alice, working at Sky Research, needs to order a book for her research from Bob, the manager of Scientific Books, an online bookstore.

Figure 2.1 *Communication at the physical layer*



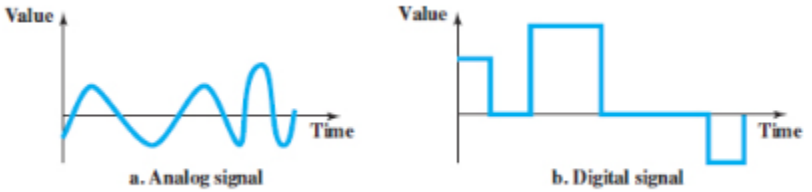
We assume that there are only three Internet service providers (ISPs) between Alice and Bob. An ISP is a company that provides Internet services to an area. In our scenario, Alice is connected to her ISP, and Bob is connected to his ISP. The two local ISPs are connected through a national ISP. Figure 2.1 shows seven routers, R1 to R7, and some switches. A router is a device that routes data when there is more than one possible route. A switch is a device that connects some devices when there is no need for routing. We discuss routers and switches in later chapters. For the

moment, we are interested in a physical route between Alice and Bob that contains only four routers: R2, R4, R5, and R7.

2.1 SIGNALS

What are really exchanged between Alice and Bob are *data* (information). However, what goes through the network connecting Alice to Bob at the physical layer are *signals*. When Alice sends a message to Bob, the message is changed to electrical signals; when Bob receives the message, the signals are changed back to the message. The reverse situation occurs when Bob sends a message to Alice. The signals can be *analog* or *digital*. An analog signal takes many values; a digital signal takes a limited number of values as shown in Figure 2.2.

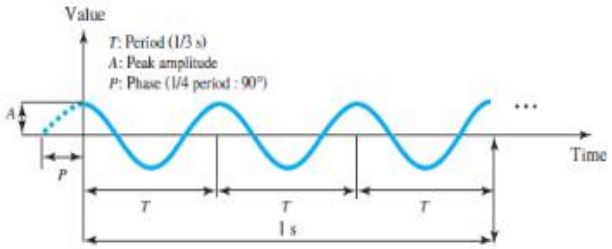
Figure 2.2 *Comparison of analog and digital signals*



2.1.1 Analog Signals

Let us first concentrate on analog signals. An analog signal can take one of two forms: *periodic* or *aperiodic* (*nonperiodic*). In data communications, we commonly use periodic analog signals. They can be classified as simple or composite. **A simple periodic analog signal, a sine wave, cannot be decomposed into simpler signals.** Figure 2.3 shows a sine wave.

Figure 2.3 *A sine wave*



The sine wave is the most fundamental form of a periodic analog signal. When we visualize it as a simple oscillating curve, its change over the course of a cycle is smooth and consistent. A sine wave can be represented by three parameters: *period*, *peak amplitude*, and *phase*. These three parameters fully describe a sine wave. The frequency is not an independent parameter: It is the inverse of the period.

Peak Amplitude

The ***peak amplitude*** of a signal is **the absolute value of its highest intensity**. For electrical signals, peak amplitude is normally measured in volts.

Period and Frequency

The ***period*** (***T***) refers to the **amount of time, in seconds, a signal needs to complete one cycle**. The ***frequency*** ***(f)***, measured in hertz (Hz), refers to **the number of periods in 1 s**. Note that period and frequency are just one characteristic defined in two ways. Period and frequency are inverses of each other, in other words ($f = 1/T$).

Example 2.2

The electrical voltage in our homes in the United States is periodic with a peak value between 110 to 120 V. Its frequency is 60 Hz.

Phase

The term **phase** describes the **position of the waveform relative to time 0**. If we think of the wave as something that can be shifted backward or forward along the time axis, phase describes the amount of that shift. It indicates the status of the first cycle. Phase is measured in degrees or radians (360° is 2π rad).

Wavelength

The term **wavelength** is another characteristic of a signal traveling through a transmission medium. **The wavelength is the distance a simple signal can travel in one period**. The wavelength binds the period or the frequency of a simple sine wave to the propagation speed in the medium.

While the frequency of a signal is independent of the medium, the wavelength depends on both the frequency and the medium. The wavelength can be calculated if one is given the propagation speed of the medium and the frequency of the signal. If we represent wavelength by λ, propagation speed by *c*, and frequency by *f*, we get

$$\lambda = c / f = c \times T$$

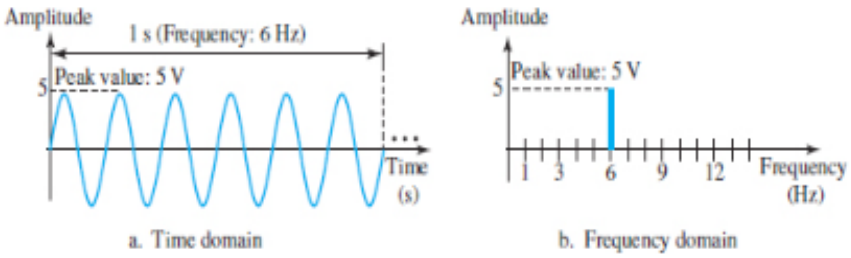
$$\lambda = c / f = c \times T$$

Time and Frequency Domains

A sine wave is comprehensively defined by its amplitude, frequency, and phase. We have been **showing a sine wave** by using what is called a **time-domain plot**, which shows changes in signal amplitude with respect to time. To show the relationship between amplitude and frequency, we can use a **frequency-domain plot**. Figure 2.4 shows a signal in both the time and frequency domains.

In the frequency domain, a sine wave is represented by one spike. The position of the spike shows the frequency; its height shows the peak amplitude.

Figure 2.4 *The time-domain and frequency-domain plots of a sine wave*



a. Time domain b. Frequency domain

Composite Signals

So far, we have focused on simple sine waves. Simple sine waves have many applications in daily life, such as sending energy from one place to another. However, if we had only one single sine wave to convey a conversation over the phone, it would make no sense and carry no information. We would just hear a buzz. We need to send a composite signal to communicate data. A **composite signal** is **made up of many simple sine waves**.

Bandwidth

The range of frequencies contained in a composite signal is its **bandwidth**. **The bandwidth is the difference between the lowest and highest frequencies in the signal**. For example, if a composite signal contains frequencies between 1000 and 5000, its bandwidth is 5000 – 1000, or 4000.

The bandwidth of a composite signal is the difference between the highest and the lowest frequencies contained in that signal.

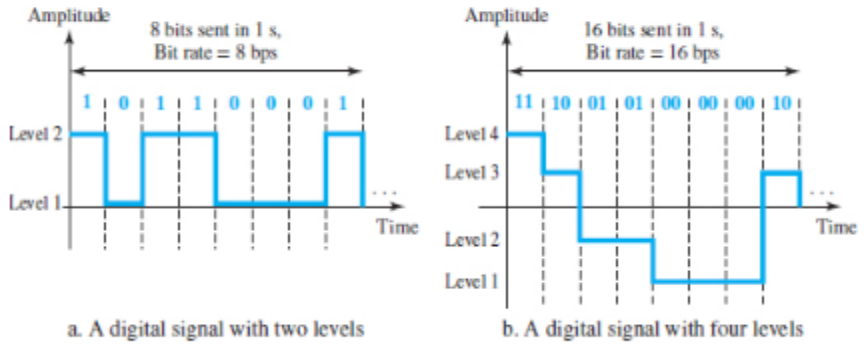
2.1.2 Digital Signals

In addition to being represented by an analog signal, information can also be represented by a digital signal. For example, a value 1 can be encoded as a positive voltage and a value 0 as zero voltage. A digital signal can have more than two levels. In this case, we can send more than 1 bit for each level. Figure 2.5 shows two signals, one with two levels and the other with four.

We send 1 bit per level in Figure 2.5a and 2 bits per level in Figure 2.5b.

In general, if a signal has *L* levels, each level needs $\log_2 L$ bits.

Figure 2.5 *Two digital signals: one with two signal levels and the other with four signal levels*



a. A digital signal with two levels

b. A digital signal with four levels

Bit Rate

Most digital signals are nonperiodic, and thus period and frequency are not appropriate characteristics of digital signals. Another term—*bit rate* (instead of *frequency*)—is used to describe digital signals. The **bit rate** is **the number of bits sent in 1 s**, expressed in bits per second (bps). The bit rate can be represented as kbps (kilo bits per second, where kilo means one thousand) or Mbps (Mega bits per second, where mega means one million).

Example 2.3

Assume we need to download text documents at the rate of 100 pages per second. What is the required bit rate of the channel? A page is an average of 24 lines with 80 characters in each line.

What is bit rate and bit length?

If we assume that one character requires 8 bits, the bit rate is

Bit rate = 100 × 24 × 80 × 8 = 1,536,000 bps = 1.536 Mbps

Bit Length

We discussed the concept of the wavelength for an analog signal: **the distance one cycle occupies on the transmission medium.** We can define something similar for a digital signal: the bit length. The **bit length** is **the distance 1 bit occupies on the transmission medium**.

Bit length = 1 / (bit rate)

Example 2.4

The length of the bit in Example 2.3 is

1/1,536,000 = 0.000000651 s = 0.651 µs

Transmission of Digital Signals

The fundamental question is, how can we send a digital signal from point A to point B?

We can transmit a digital signal by using one of two different approaches: ***baseband transmission* or *broadband transmission***.

Baseband transmission means sending a digital signal over a channel without changing it to an analog signal.

Broadband transmission or modulation means changing the digital signal to an analog signal for transmission.

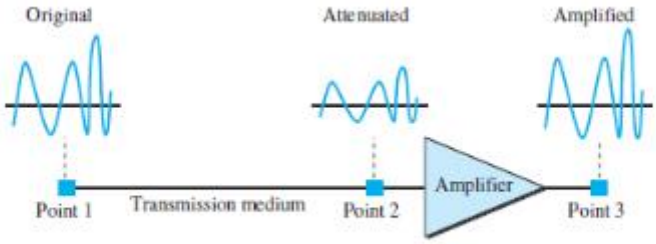
2.2 SIGNAL IMPAIRMENT

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***.

2.2.1 Attenuation and Amplification

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. To compensate for this loss, we need **amplification**. Figure 2.6 shows the effect of attenuation and amplification.

Figure 2.6 *Attenuation and amplification*



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.

Variables P_1 and P_2 are the powers of a signal at points 1 and 2, respectively.

$$\mathbf{dB = 10 \log_{10} (P_2/P_1)}$$

Example 2.5

Suppose a signal travels through a transmission medium and its power is reduced to one-half. This means that $P_2 = 0.5P_1$. In this case, the attenuation (loss of power) can be calculated as

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

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

2.2.2 Distortion

Distortion means that the **signal changes its form or shape**. Distortion can occur in a composite signal made up 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.

Eg: speech signal processing

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. **Corrupts the actual signal while transmission through communication medium.** Eg: Coaxial Cables, audio electronics and ICs. **Impulse noise** is a spike (a signal with high energy and very short duration) that comes from power lines, lightning, and so on.

Signal-to-Noise Ratio (SNR)

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

$$\text{SNR} = \text{(average signal power)} / \text{(average noise power)}$$

We need to consider the average signal power and the average noise power because these may change with time. Because SNR is the ratio of two powers, it is often described in decibel units, SNR_{dB}, as

$$\text{SNRdB} = 10 \log_{10} \text{SNR}$$

2.2.3 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.

$$\textbf{BitRate} = 2 \times B \times \log_2 L$$

In this formula, B is the bandwidth of the channel, L is the number of signal levels used to represent data, and BitRate is the number of 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. If the number of levels in a signal is just 2, the receiver can easily distinguish between a 0 and a 1. If the level of a signal is 64, the receiver must be very sophisticated to distinguish between 64 different levels. In other words, **increasing the levels of a signal reduces the reliability of the system.**

Example 2.6

We need to send 265 kbps over a noiseless (ideal) channel with a bandwidth of 20 kHz. How many signal levels do we need?

We can use the Nyquist formula as shown:

$$265{,}000 = 2 \times 20{,}000 \times \log_2 L \rightarrow \log_2 L = 6.625 \; L = 2^{6.625} = 98.7 \text{ levels}$$

Because this result is not a power of 2, we need to either increase the number of levels or reduce the bit rate. If we have 128 levels, the bit rate is 280 kbps. If we have 64 levels, the bit rate is 240 kbps.

Noisy Channel: Shannon Capacity

In reality, we cannot have a noiseless channel; the channel is always noisy. Claude Shannon introduced a formula, called the **Shannon capacity**, to determine the theoretical highest data rate for a noisy channel:

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

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

Example 2.7

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

$$C = B \log_2 (1 + \text{SNR}) = B \log_2 (1 + 0) = B \log_{2\,1} = B \times 0 = 0$$

This means that the capacity of this channel is zero regardless of the bandwidth. In other words, the data are so corrupted in this channel that they are useless when received.

Example 2.8

Calculate the theoretical highest bit rate(capacity) of a regular telephone line. A telephone line normally has a bandwidth of 3000 Hz (300 to 3300 Hz) assigned for data communications. The signal-to-noise ratio is usually 3162.

For this channel the capacity is calculated as

$$C = B \log_2 (1 + \text{SNR}) = 3000 \log_2 (1 + 3162) = 34{,}881 \text{ bps}$$

This means that the highest bit rate for a telephone line is 34.881 kbps. If we want to send data faster than this, we can either increase the bandwidth of the line or improve the signal-to-noise ratio.

Using Both Limits

In practice, we need to use both methods to find the limits and signal levels. Let us show this with an example.

Example 2.9

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

Solution

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

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

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

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

The Shannon capacity gives us the upper limit; the Nyquist formula tells us how many signal levels we need

2.2.4 Performance

Up to now, we have discussed the tools of transmitting data (signals) over a network and how the data behave. One important issue in networking is the performance of the network—how good is it?

Bandwidth

Measure of maximum amount of data transmitted in the network.

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 is the range of *frequencies* involved. Bandwidth in bits per second can be defined as *the number of bits a channel* can pass.

Example 2.10

The bandwidth of a subscriber line is 4 kHz for voice or data. The bandwidth of this line for data transmission can be up to 56 kbps, using a sophisticated device to change the digital signal to analog. If the telephone company improves the quality of the line and increases the bandwidth to 8 kHz, we can send 112 kbps.

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 send only *T* bps through this link,

with *T* always less than *B*. In other words, the **bandwidth is a potential measurement of a link; the throughput is an actual measurement of how fast we can send data**.

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

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 say that normally there are four types of delay: *propagation delay, transmission delay, queuing delay, and processing delay*. The latency or total delay is

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

Bandwidth-Delay Product

Bandwidth and delay are two performance metrics of a link. However, what is very important in data communications is the product of the two, the bandwidth-delay product. Let us elaborate on this issue, using two hypothetical cases as examples.

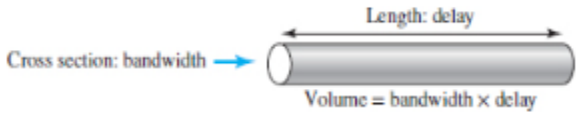
- **Case 1.** Let us assume that we have a link with a bandwidth of 1 bps. We also assume that the delay of the link is 5 s (also unrealistic). We want to see what the bandwidth-delay product means in this case. We can see that the bandwidth-delay product (1 × 5) is the maximum number of bits that can fill the link. There can be no more than 5 bits at any one time on the link.
- **Case 2.** Now assume we have a bandwidth of 5 bps with a delay of 5 s. We can see that there can be a maximum of 5 × 5 = 25 bits on the line. The reason is that, at each second, there are 5 bits on the line.

The bandwidth-delay product defines the number of bits that can fill the link.

Example 2.12

We can think about the link between two points as a pipe. The cross section of the pipe represents the bandwidth, and the length of the pipe represents the delay. We can say the volume of the pipe defines the bandwidth-delay product, as shown in Figure 2.7.

Figure 2.7 *Bandwidth-delay product*



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.

2.3 DIGITAL TRANSMISSION

A computer network is designed to send information from one point to another. This information needs to be converted to either a digital signal or an analog signal for transmission. In this section, we discuss the first choice; in Section 2.4, we discuss the second choice.

In digital transmission, if data are digital, we need to use **digital-to-digital conversion** techniques. If data are analog, we need to use **analog-to-digital conversion**.

2.3.1 Digital-to-Digital Conversion

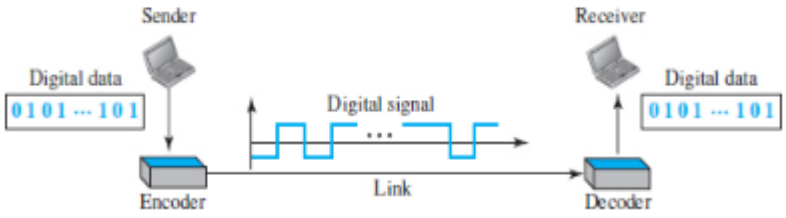
If we use digital transmission and our data are already digital, we need digital-to-digital conversion. The conversion involves three techniques: ***line coding***, ***block coding***, **and** ***scrambling***. Line coding is always needed; block coding and scrambling may or may not be needed. The computers used the digital form to store the information. Therefore, the data needs to be converted in digital form so that it can be used by a computer.

Line Coding

Line coding is **the process of converting digital data to digital signals**. We assume that data, in the form of text, numbers, graphical images, audio, or video, are stored in computer memory as sequences of bits (see Figure 2.8).

Figure 2.8 *Line coding and decoding*

Figure 2.8 *Line coding and decoding*



Line coding converts a sequence of bits to a digital signal. At the sender, digital data are encoded into a digital signal; at the receiver, the digital data are re-created by decoding the digital signal.

Block Coding

Helps in error detection and re-transmission of the signal. Block coding can give redundancy and improve the performance of line coding.

In general, **block coding** changes a block of *m* bits into a block of *n* bits, where *n* is larger than *m*. Thus, its adds extra bits (redundancy bits) which helps in synchronization at receiver's and sender's end and also providing some kind of error detecting capability.

Block coding is referred to as an *mB*/*nB* encoding technique. Block coding normally involves three steps: division, substitution, and combination. In the division step, a sequence of bits is divided into groups of *m* bits. For example, in 4B/5B encoding, the original bit sequence is divided into 4-bit groups. The heart of block coding is the substitution step. In this step, we substitute an *m*-bit group for an *n*-bit group. For example, in the case of 4B/5B encoding, we substitute a 4-bit code for a 5-bit group. Finally, the *n*-bit groups are combined to form a stream.

Scrambling

Scrambling is a technique implemented during encoding that does not increase the number of bits and does provide synchronization.

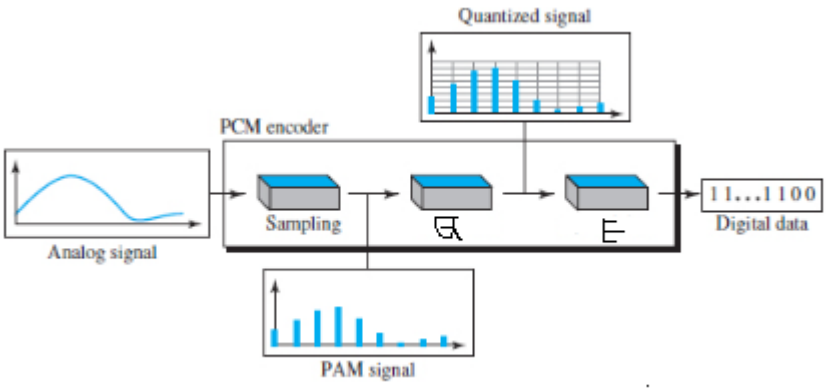
2.3.2 Analog-to-Digital Conversion

Sometimes we have an analog signal such as one created by a microphone or camera. The tendency today is to change an analog signal to digital data because the digital signal is less susceptible to noise. In this section we describe two techniques, **pulse code modulation and delta modulation**. After the digital data are created (digitization), we can use one of the techniques described in Section 2.3.1, to convert the digital data to a digital signal.

Pulse Code Modulation (PCM)

The most common technique used to change an analog signal to digital data (**digitization**) is called **pulse code modulation (PCM)**. A PCM encoder has three processes, as shown in Figure 2.9.

Figure 2.9 *Components of a PCM encoder*



Sampling: "Measuring the instantaneous values of continuous-time signal in a discrete form."

Quantization: Representing the sampled values of the amplitude by a finite set of levels

The three processes are:

1. **The analog signal is sampled every *T* s.**
2. **The sampled signal is quantized, which means every sample is considered as a pulse.**
3. **The quantized values (pulses) are encoded as streams of bits.**

Example 2.13

We want to digitize the human voice. What is the bit rate, assuming 8 bits per sample?

Solution

The human voice normally contains frequencies from 0 to 4000 Hz. So the sampling rate and bit rate are calculated as

The sampling rate that is applied to voice is typically twice the maximum frequency of the voice.

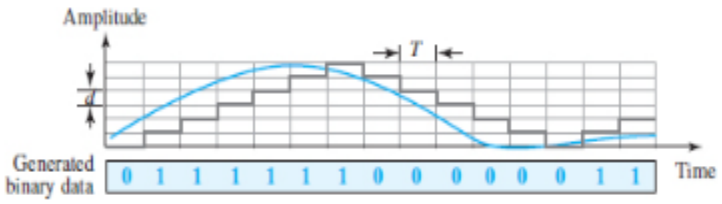
Sampling rate = 4000 × 2 = 8000samples/s
Bit rate = 8000 × 8 = 64,000 bps = 64 kbps

Delta Modulation (DM)

PCM is a very complex technique. Other techniques have been developed to reduce its complexity. The simplest is **delta modulation (DM)**. PCM finds the value of the signal amplitude for each sample; DM finds the change from the previous sample. Figure 2.10 shows the process. Note that there are no code words here; bits are sent one after the other.

Figure 2.10 *The process of delta modulation*

Figure 2.10 *The process of delta modulation*



2.4 ANALOG TRANSMISSION

Although **digital transmission** is desirable, it needs **a low-pass channel** (a channel that starts from 0); **analog transmission** is the only choice if we have a **bandpass channel** (a channel that does not start from zero). Converting digital data to a bandpass analog signal is traditionally called *digital-to-analog conversion*. Converting a low-pass analog signal to a bandpass analog

signal is traditionally called *analog-to-analog conversion*. In this section, we discuss these two types of conversions.

2.4.1 Digital-to-Analog Conversion

Digital-to-analog conversion is the process of changing one of the characteristics of an analog signal based on the information in digital data. A **sine wave** is defined by three characteristics: ***amplitude, frequency, and phase***. Any of the three characteristics can be altered in this way, giving us at least three mechanisms for modulating digital data into an analog signal: **amplitude shift keying (ASK)**, **frequency shift keying (FSK)**, and **phase shift keying (PSK)**. In addition, there is a fourth (and better) mechanism that combines changing both the amplitude and phase, called **quadrature amplitude modulation (QAM)**. QAM is the most efficient of these options and is the mechanism commonly used today.

Used in music players to convert digital data streams into analog audio signals. They are also used in televisions and mobile phones to convert digital video data into analog video signals.

Amplitude Shift Keying

In amplitude shift keying, the **amplitude of the carrier signal is varied** to create signal elements. Both frequency and phase remain constant while the amplitude changes.

Binary ASK (BASK)

ASK is normally implemented using **only two levels**. This is referred to as *binary amplitude shift keying* or *on-off keying* (OOK). The peak amplitude of one signal level is 0; the other is the same as the amplitude of the carrier frequency. Figure 2.11 gives a conceptual view of binary ASK.

Figure 2.11 *Binary amplitude shift keying*

Figure 2.11 *Binary amplitude shift keying*

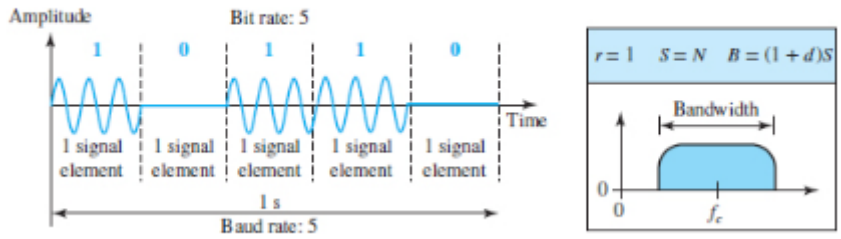


Figure 2.11 also shows the bandwidth for ASK. Although the carrier signal is only one simple sine wave, the process of modulation produces a nonperiodic composite signal. This signal, as was discussed before, has a continuous set of frequencies. As we expect, the bandwidth is proportional to the signal rate (baud). However, there is normally another factor involved, called *d*, which depends on the modulation and filtering process. The value of *d* is between 0 and 1.

This means that the **bandwidth can be expressed as shown, where *S* is the signal rate and *B* is the bandwidth** The formula $B = (1 + d)S$

Shows that the required bandwidth has a minimum value of S and a maximum value of $2S$. The most important point here is the location of the bandwidth. The middle of the bandwidth is where f_c, the carrier frequency, is located. This means if we have a bandpass channel available (signal not start with 0), we can choose our f_c so that the modulated signal occupies that bandwidth. This is, in fact, the most important advantage of digital-to-analog conversion. We can shift the resulting bandwidth to match what is available.

Multilevel ASK

The above discussion uses only two amplitude levels. We can have multilevel ASK in which there are more than two levels. We can use 4, 8, 16, or more different amplitudes for the signal and modulate the data using 2, 3, 4, or more bits at a time. In these cases, $r = 2$, $r = 3$, $r = 4$, and so on. Although this is not implemented with pure ASK, it is implemented with QAM (as we will see later).

Frequency Shift Keying (FSK)

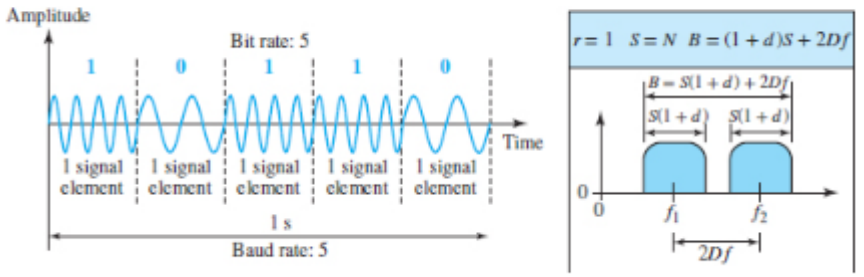
In frequency shift keying, the **frequency of the carrier signal is varied to represent data.** The frequency of the modulated signal is constant for the duration of one signal element, but changes for the next signal element if the data element changes. Both the peak amplitude and phase remain constant for all signal elements.

Binary FSK (BFSK)

One way to think about binary FSK (BFSK) is to consider two carrier frequencies. In Figure 2.12, we have selected **two carrier frequencies, f_1 and f_2.** We use the **first carrier frequency if the data element is 0; we use the second if the data element is 1.** However, note that this example is unrealistic and used only for demonstration purposes. Normally the carrier frequencies are very high, and the difference between them is very small.

Figure 2.12 *Binary frequency shift keying*

Figure 2.12 *Binary frequency shift keying*



Phase Shift Keying

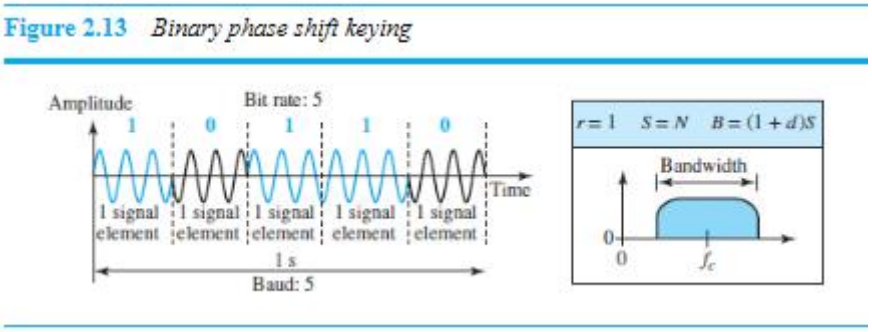
In phase shift keying, the **phase of the carrier is varied to represent two or more different signal elements. Both peak amplitude and frequency remain constant as the phase changes**.

Today, PSK is more common than ASK or FSK. However, we will see shortly that QAM, which combines ASK and PSK, is the dominant method of digital-to-analog modulation.

Binary PSK (BPSK)

The simplest PSK is binary BPSK, in which we have only two signal elements, one with a phase of 0°, and the other with a phase of 180°. Figure 2.13 gives a conceptual view of PSK. Binary PSK is as simple as binary ASK with one big advantage—it is less susceptible to noise.

Figure 2.13 *Binary phase shift keying*



2.4.2 Analog-to-Analog Conversion

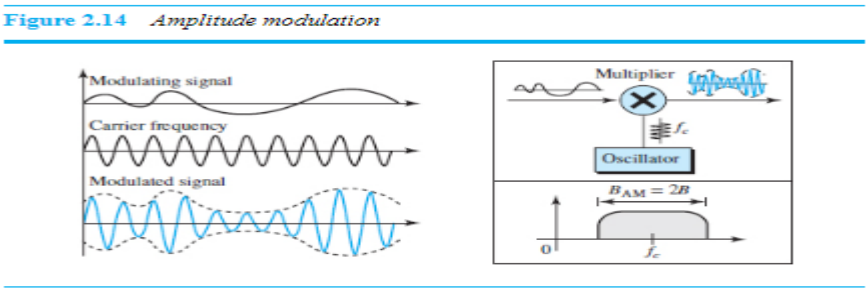
Analog-to-analog conversion, or **analog modulation**, is the **representation of analog information by an analog signal.** One may ask why we need to modulate an analog signal; it is already analog. Modulation is needed if the medium is bandpass in nature or if only a bandpass channel is available to us. An example is radio. The government assigns a narrow bandwidth to each radio station. The analog signal produced by each station is a low-pass signal, all in the same range. To be able to listen to different stations, the low-pass signals need to be shifted, each to a different range.

Analog-to-analog conversion can be accomplished in three ways: **amplitude modulation (AM)**, **frequency modulation (FM)**, and **phase modulation (PM)**. FM and PM are usually categorized together.

Amplitude Modulation

In AM transmission, **the carrier signal is modulated so that its amplitude varies with the changing amplitudes of the modulating signal.** The frequency and phase of the carrier remain the same; only the amplitude changes to follow variations in the information. Figure 2.14 shows how this concept works. The modulating signal is the envelope of the carrier.

Figure 2.14 *Amplitude modulation*

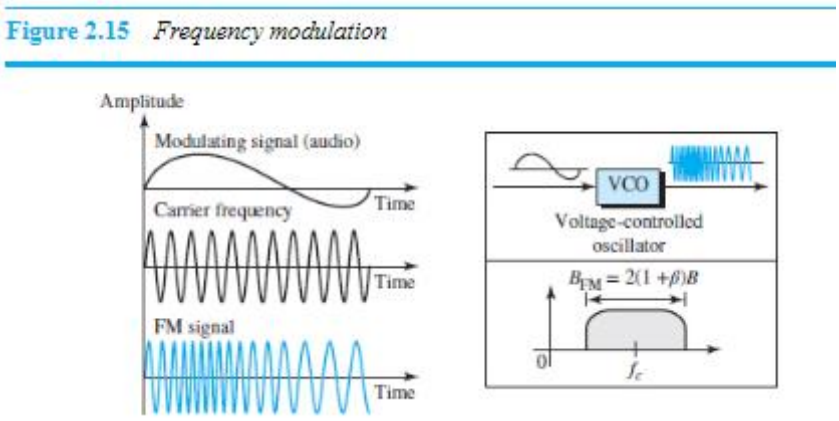


As Figure 2.14 shows, AM is normally implemented by using a **simple multiplier** because **the amplitude of the carrier signal needs to be changed according to the amplitude of the modulating signal.** Figure 2.14 also shows the bandwidth of an AM signal. The modulation creates a bandwidth that is twice the bandwidth of the modulating signal and covers a range centered on the carrier frequency. However, the signal components above and below the carrier frequency carry exactly the same information. For this reason, some implementations discard one-half of the signals and cut the bandwidth in half.

Frequency Modulation

In FM transmission, **the frequency of the carrier signal is modulated to follow the changing voltage level (amplitude) of the modulating signal**. The peak amplitude and phase of the carrier signal remain constant, but as the amplitude of the information signal changes, the frequency of the carrier changes correspondingly. Figure 2.15 shows the relationships of the modulating signal, the carrier signal, and the resultant FM signal.

Figure 2.15 *Frequency modulation*



As Figure 2.15 shows, FM is normally implemented by using a voltage-controlled oscillator as with FSK. The frequency of the oscillator changes according to the input voltage, which is the

amplitude of the modulating signal. Figure 2.15 also shows the bandwidth of an FM signal. The actual bandwidth is difficult to determine exactly, but it can be shown empirically as $B_{FM} = 2(1 + \beta)B$ where β is a factor that depends on modulation technique with a common value of 4.

This is the main reason that many radio stations use FM to broadcast music over the radio.Additionally, some of its uses are also found in radar, telemetry, music synthesis as well as in video-transmission instruments.

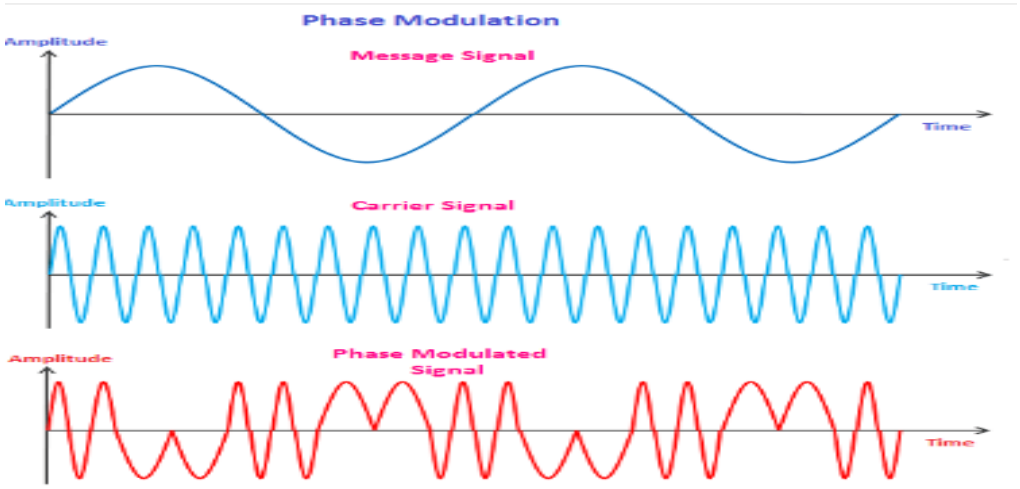
Phase Modulation

In PM transmission, the **phase of the carrier signal is modulated to follow the changing voltage level (amplitude) of the message signal.** The peak amplitude and frequency of the carrier signal remain constant, but as the amplitude of the information signal changes, the phase of the carrier changes correspondingly. It can be proven mathematically that PM is the same as FM with one difference. In FM, the instantaneous change in the carrier frequency is proportional to the amplitude of the modulating signal; in PM the instantaneous change in the carrier frequency is proportional to the derivative of the amplitude of the modulating signal. Figure 2.16 shows the relationships of the modulating signal, the carrier signal, and the resultant PM signal.

Phase modulation is widely used for transmitting radio waves and is an integral element of many digital transmission coding schemes that support an ample range of wireless technologies such as GSM, Satellite television, and Wi-Fi.

In Figure 2.16, third figure shows that the phase of both the positive and negative half cycles of the carrier signal varies as per amplitude variations of the modulating signal. During the positive half cycle, the carrier signal phase shifts in one direction, whereas during the negative half cycle, the carrier signal phase shifts in the opposite direction.

Figure 2.16 *Phase modulation*



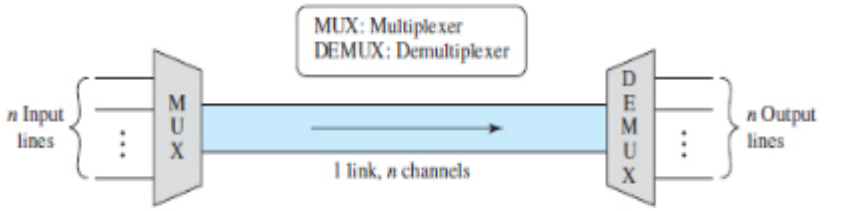
2.5 MULTIPLEXING

Whenever the bandwidth of a medium linking two devices is greater than the bandwidth needs of the devices, the link can be shared. **Multiplexing** is **the set of techniques that allows the simultaneous transmission of multiple signals across a single data link.** As data and telecommunications use increases, so does traffic. We can accommodate this increase by continuing to add individual links each time a new channel is needed, or we can install higher-bandwidth links and use each to carry multiple signals. Today's technology includes high-bandwidth media such as optical fiber and terrestrial and satellite microwaves. Each has a bandwidth far in excess of that needed for the average transmission signal. If the bandwidth of a link is greater than the bandwidth needs of the devices connected to it, the bandwidth is wasted. An efficient system maximizes the utilization of all resources; bandwidth is one of the most precious resources we have in data communications.

In a multiplexed system, *n* lines share the bandwidth of one link. Figure 2.17 shows the basic format of a multiplexed system. The lines on the left direct their transmission streams to a *multiplexer*, which combines them into a single stream (many-to-one). At the receiving end, that stream is fed into a *demultiplexer*, which separates the stream back into its component transmissions (one-to-many) and directs them to their corresponding lines. In Figure 2.17, the word *link* refers to the physical path. The word *channel* refers to the portion of a link that carries a transmission between a given pair of lines. One link can have many (*n*) channels.

Figure 2.17 *Dividing a link into channels*

Figure 2.17 *Dividing a link into channels*



There are three basic multiplexing techniques: frequency-division multiplexing (FDM), wavelength-division multiplexing (WDM), and time-division multiplexing (TDM).
The first two are techniques designed for analog signals; the third, for digital signals.

2.5.1 Frequency-Division Multiplexing

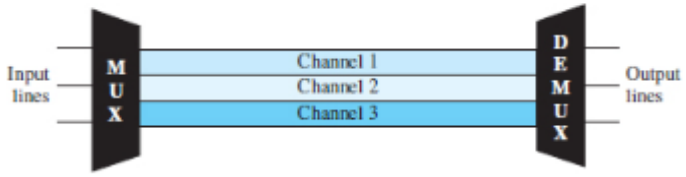
Frequency-division multiplexing (FDM) is an **analog technique** that can **be applied when the bandwidth of a link (in hertz) is greater than the combined bandwidths of the signals to be transmitted.** In FDM, signals generated by each sending device modulate different carrier frequencies. These modulated signals are then combined into a single composite signal that can be transported by the link.

Carrier frequencies are separated by sufficient bandwidth to accommodate the modulated signal. These bandwidth ranges can be thought of as channels through which the various signals travel. Channels can be separated by strips of unused bandwidth—**guard bands**—to prevent signals from overlapping. In addition, carrier frequencies must not interfere with the original data frequencies.

Figure 2.18 gives a conceptual view of FDM. In this illustration, the transmission path is divided into three parts, each representing a channel that carries one transmission.

Figure 2.18 *Frequency-division multiplexing*

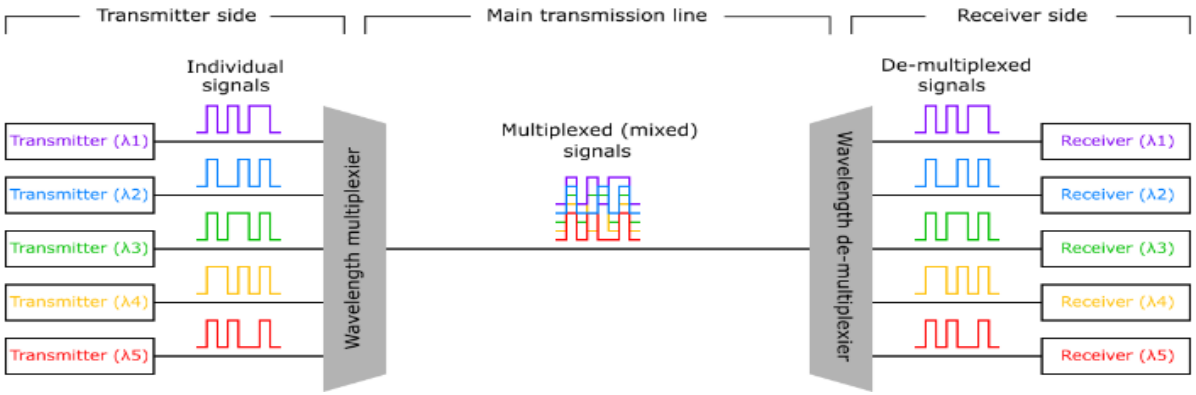
Figure 2.18 *Frequency-division multiplexing*



We consider FDM to be an analog multiplexing technique; however, this does not mean that FDM cannot be used to combine sources sending digital signals. Digital signals can be converted to analog signals before FDM is used to multiplex them.

2.5.2 Wavelength Division Multiplexing

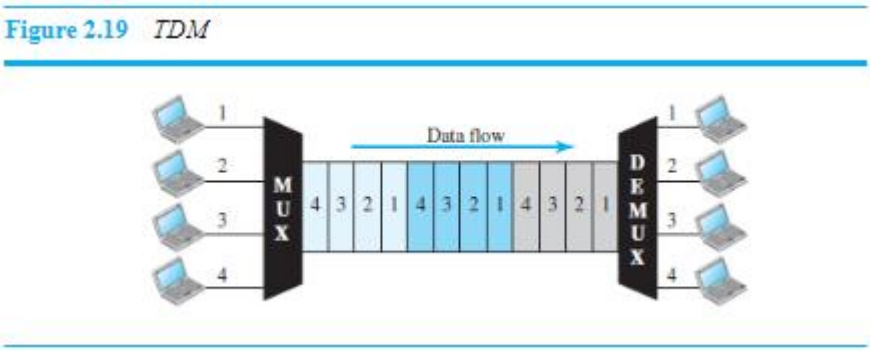
Light has different wavelength (colors). In fiber optic mode, multiple optical carrier signals are multiplexed into an optical fiber by using different wavelengths. This is an analog multiplexing technique and is done conceptually in the same manner as FDM but uses light as signals.



2.5.3 Time-Division Multiplexing

Time-division multiplexing (TDM) is a digital process that allows several connections to share the high bandwidth of a link. Instead of sharing a portion of the bandwidth as in FDM, time is shared. **Each connection occupies a portion of time in the link**. Figure 2.19 gives a conceptual view of TDM. Note that the same link is used as in FDM; here, however, the link is shown sectioned by time rather than by frequency. In the figure, portions of signals 1, 2, 3, and 4 occupy the link sequentially.

Figure 2.19 *TDM*



Note that in Figure 2.19 we are concerned with only multiplexing, not switching. This means that all the data in a message from source 1 always go to one specific destination, be it 1, 2, 3, or 4. The delivery is fixed and unvarying, unlike switching.

We also need to remember that TDM is, in principle, **a digital multiplexing technique. Digital data from different sources are combined into one timeshared link**. However, this does not mean that the sources cannot produce analog data; analog data can be sampled, changed to digital data, and then multiplexed by using TDM.

CHAPTER 3

3.2 DATA-LINK CONTROL

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. Its functions include *framing* and *error control*. In this section, we first discuss framing, or how to organize the bits that are carried by the physical layer. We then discuss flow and error control. Techniques for error detection are discussed at the end of this section.

3.2.1 Framing

Data transmission in the physical layer means ***moving bits in the form of a signal from the source to the destination***. The physical layer provides **bit synchronization** to ensure that *the sender and receiver use the same bit durations and timing*. We discussed the physical layer in Chapter 2.

The data-link layer, on the other hand, needs to **pack bits into frames,** so that each frame is distinguishable from another. Our postal system practices a type of framing. The simple act of inserting a letter into an envelope separates one piece of information from another; the envelope serves as the delimiter. In addition, each envelope defines the sender and receiver addresses, which is necessary since the postal system is a many-to-many carrier facility.

Framing* in the data-link layer separates a message *from one source to a destination by adding a sender address and a destination address. The destination address defines where the packet is to go; the sender address helps the recipient acknowledge the receipt.

Although the whole message could be packed in one frame, that is not normally done. One reason is that a frame can be very large, making flow and error control very inefficient. When a message is carried in one very large frame, even a single-bit error would require the retransmission of the whole frame. When a message is divided into smaller frames, a single-bit error affects only that small frame.

Frame Size

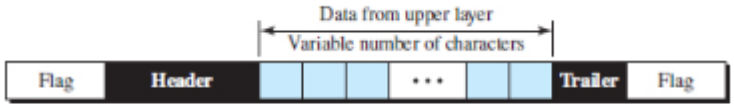
Frames can be of fixed or variable size. In ***fixed-size framing***, there is **no need to define the boundaries of the frames**; the size itself can be used as a delimiter. An example of this type of framing is the ATM WAN, which uses frames of fixed size called *cells*.

Our main discussion in this chapter concerns ***variable-size framing***, prevalent in local-area networks. In variable-size framing, we need a way to define the end of one frame and the beginning of the next. Historically, **two approaches have been 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 (see Appendix A). 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 3.4 shows the format of a frame in a character-oriented protocol.

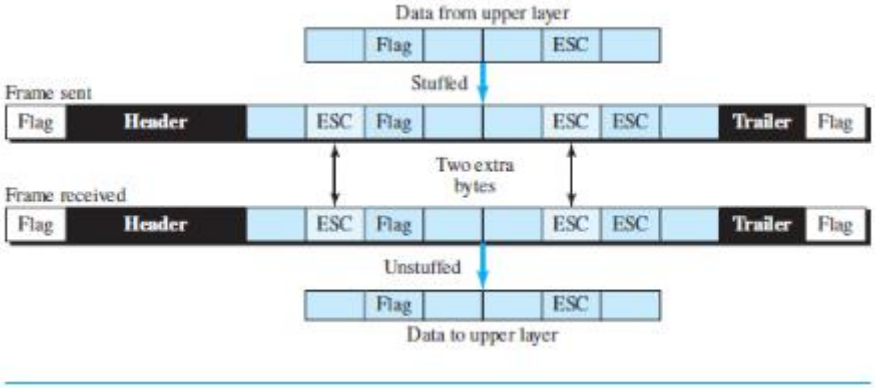
Figure 3.4 *A frame in a character-oriented protocol*



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 pattern 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.

Byte stuffing by the escape character allows the presence of the flag in the data section of the frame, but it creates another problem. What happens if the text contains one or more escape characters followed by a byte with the same pattern as the flag? The receiver removes the escape character, but keeps the next byte, which is incorrectly interpreted as the end of the frame. To solve this problem, the escape characters that are part of the text must also be marked by another escape character. In other words**, if the escape character is part of the text, an extra one is added to show that the second one is part of the text**. Figure 3.5 shows the situation.

Figure 3.5 *Byte stuffing and unstuffing*



Byte stuffing is the process of adding one extra byte whenever there is a flag or escape character in the text.

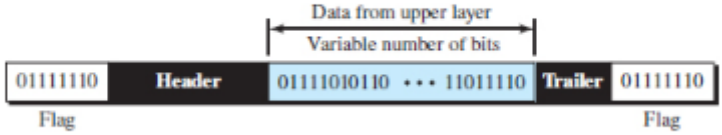
Character-oriented protocols present another problem in data communications. The universal coding systems in use today, such as Unicode, have 16-bit and 32-bit characters that conflict with 8-bit characters. We can say that, in general, the tendency is moving toward the bit-oriented protocols that we will discuss next.

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 end of the frame, as shown in Figure 3.6.

Figure 3.6 *A frame in a bit-oriented protocol*

Figure 3.6 *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 one single bit (instead of one 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 3.7 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.

Figure 3.7 *Bit stuffing and unstuffing*

Figure 11.4 ***Bit stuffing and unstuffing***

This means that if the flag like pattern 0111110 appears in the data, it will change to 01111100 (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.

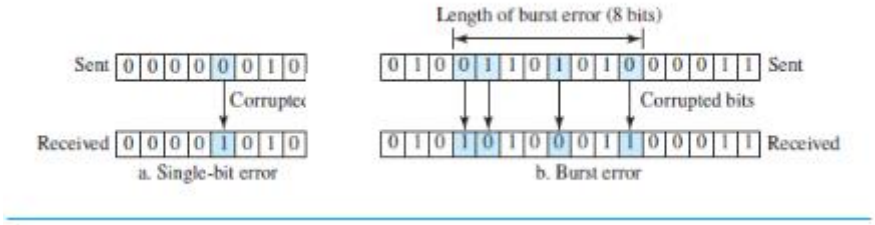
3.2.2 Error Control

Error control is both error detection and error correction. It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. In the data-link layer, the term ***error control*** refers primarily to methods of **error detection and retransmission** (error correction is done using retransmission of the corrupted frame).

Types of Errors

Whenever bits flow from one point to another, they are subject to unpredictable changes because of **interference**. This interference can change the shape of the signal. The term **single-bit error** means that **only 1 bit of a given data unit (such as a byte, character, or packet) is changed from 1 to 0 or from 0 to 1**. The term **burst error** means that **two or more bits in the data unit have changed from 1 to 0 or from 0 to 1**. Figure 3.8 shows the effect of a single-bit error and burst error, respectively, on a data unit.

Figure 3.8 *Single-bit error 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 one bit, which means that when noise affects data, it affects a set of bits. The number of bits affected depends on the data rate and duration of noise. For example, if we are sending data at 1 kbps, a noise of 1/100 s can affect 10 bits; if we are sending data at 1 Mbps, the same noise can affect 10,000 bits.

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 looking only 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 the 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. We concentrate on error detection.

Coding –Error detection

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 the 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***. In this book, we concentrate on block coding; convolution coding is more complex and beyond the scope of this book.

Block Coding

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

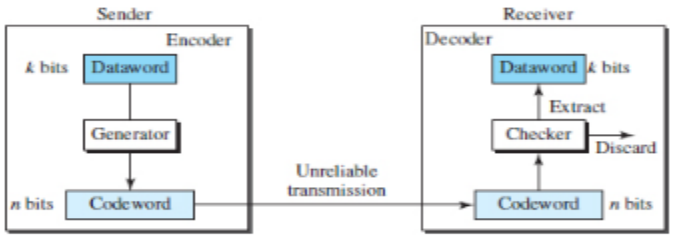
Error Detection

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

1. **The receiver has (or can find) a list of valid codewords.**
2. **The original codeword has changed to an invalid one.**

Figure 3.9 shows the role of block coding in error detection.

Figure 3.9 *Process of error detection in block coding*



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

Example 3.1

Let us assume that $k = 2$ and $n = 3$. Table 3.1 shows the list of datawords and codewords. Later, we will see how to derive a codeword from a dataword.

Table 3.1 *A code for error detection in Example 3.1*

Datawords	*Codewords*	*Datawords*	*Codewords*
00	000	10	101
01	011	11	110

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

1. **The receiver receives 011. It is a valid codeword. The receiver extracts the dataword 01 from it.**
2. **The codeword is corrupted during transmission, and 111 is received (the leftmost bit is corrupted). This is not a valid codeword and is discarded.**
3. **The codeword is corrupted during transmission, and 000 is received (the right 2 bits are corrupted). This is a valid codeword. The receiver incorrectly extracts the dataword 00. Two corrupted bits have made the error undetectable.**

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

Hamming Distance

One of the central concepts in coding for error control is the idea of the Hamming distance. The **Hamming distance between two words (of the same size) is the number of differences between the corresponding bits**. We show the Hamming distance between two words x and y as $d(x, y)$. We may wonder why the Hamming distance is important for **error detection.** The reason is that the Hamming distance between the received codeword and the sent codeword is the number of bits that are corrupted during transmission. For example, if the

codeword 00000 is sent and 01101 is received, 3 bits are in error and the Hamming distance between the two is $d(00000, 01101) = 3$. In other words, if the Hamming distance between the sent 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.

The Hamming distance between two words is the number of differences between corresponding bits.

Example 3.2

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

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

Linear Block Codes

Almost all block codes used today belong to a subset of block codes called *linear block codes*. The use of nonlinear block codes for error detection and correction is not as widespread because their structure makes theoretical analysis and implementation difficult. We therefore concentrate on linear block codes. The formal definition of linear block codes requires the knowledge of abstract algebra (particularly Galois fields), which is beyond the scope of this book. We therefore give an informal definition. For our purposes, **a linear block code is a code in which the exclusive OR (addition modulo-2) of two valid codewords creates another valid codeword.**

Example 3.5

The code in Table 3.1 is a linear block code because the result of XORing any codeword with any other codeword is a valid codeword. For example, the XORing of the second and third codewords creates the fourth one.

Minimum Distance for Linear Block Codes

The minimum Hamming distance for a linear block code is simple to find. It is the number of 1s in the nonzero valid codeword with the smallest number of 1s.

Example 3.6

In our first code (Table 3.1), the numbers of 1s in the nonzero codewords are 2, 2, and 2. So the minimum Hamming distance is $d_{min} = 2$.

Parity-Check Code

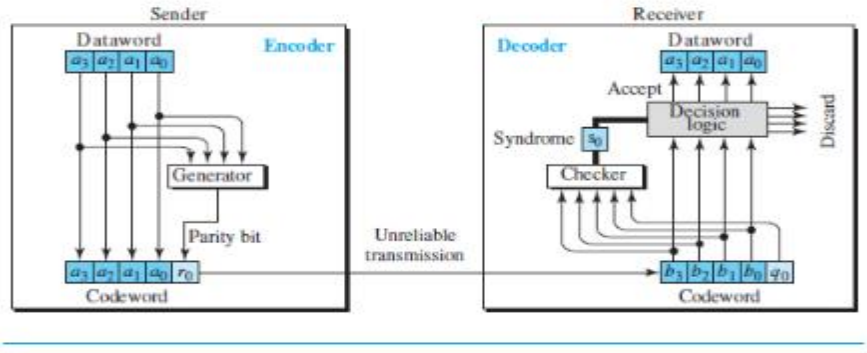
Perhaps the most familiar **error-detecting code** is the **parity-check code**. This code is a linear block code. In this code, a k-bit dataword is changed to an n-bit codeword where $n = k + 1$. The extra bit, called the *parity bit*, is selected to make the total number of 1s in the codeword even. 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 3.1) is a parity-check code ($k = 2$ and $n = 3$). The code in Table 3.2 is also a parity-check code with $k = 4$ and $n = 5$.

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

Figure 3.11 shows a possible structure of an encoder (at the sender) and a decoder (at the receiver). The encoder uses a generator that takes a copy of a 4-bit dataword (a_0, a_1, a_2, and a_3) and generates a parity bit r_0. The dataword bits and the parity bit create the 5-bit codeword. The parity bit that is added makes the number of 1s in the codeword even. This is normally done by adding the 4 bits of the dataword (modulo-2); the result is the parity bit. In other words,

$$r_0 = a_3 + a_2 + a_1 + a_0 \quad \text{(modulo-2)}$$

Figure 3.11 *Encoder and decoder for simple parity-check code*



If the number of 1s is even, the result is 0; if the number of 1s is odd, the result is 1. In both cases, the total number of 1s in the codeword is even.

The sender sends the codeword, which may be corrupted during transmission. The receiver receives a 5-bit word. The checker at the receiver does the same thing as the generator in the sender with one exception: The addition is done over all 5 bits. The result, which is called the **syndrome**, is just 1 bit. **The syndrome is 0 when the number of 1s in the received codeword is even(No Error);** otherwise, it is 1.

$$s_0 = b_3 + b_2 + b_1 + b_0 + q_0 \qquad \text{(modulo-2)}$$

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

Example 3.7

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

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

5. **Three bits—a_3, a_2, and a_1—are changed by errors. The received codeword is 01011. The syndrome is 1. The dataword is not created. This shows that the simple parity check, guaranteed to detect one single error, can also find an odd number of errors.**

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

Cyclic Codes

Cyclic codes are special linear block codes with one extra property. In a **cyclic code**, if a codeword is cyclically shifted (rotated), the result is another codeword. For example, if 1011000 is a codeword and we cyclically left-shift, then 0110001 is also a codeword.

In this case, if we call the bits in the first word 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. However, the theoretical background required is beyond the scope of this book. In this section, we simply discuss a subset of cyclic codes called the **cyclic redundancy check (CRC)** that is used in networks such as **LANs and WANs**.

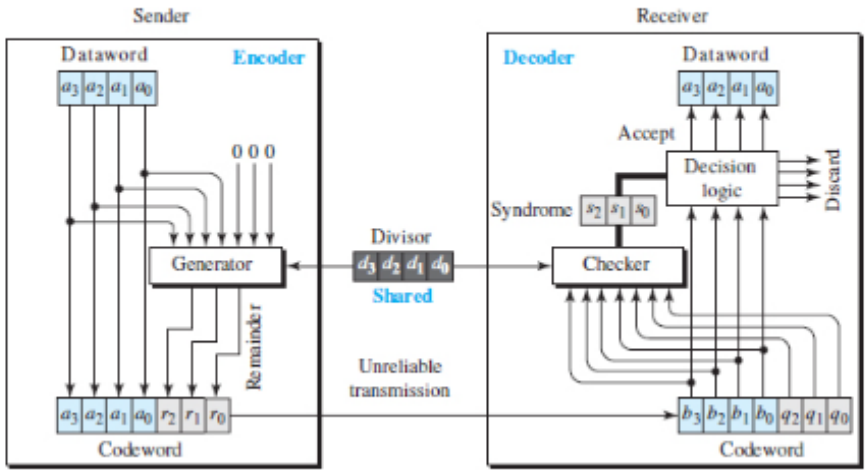
Table 3.3 shows an example of a CRC code. We can see both the linear and cyclic properties of this code.

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	111111

Figure 3.12 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 ($r_2r_1r_0$) is appended to the dataword to create the codeword.

Figure 3.12 *CRC encoder and decoder*

Figure 3.12 *CRC encoder and decoder*

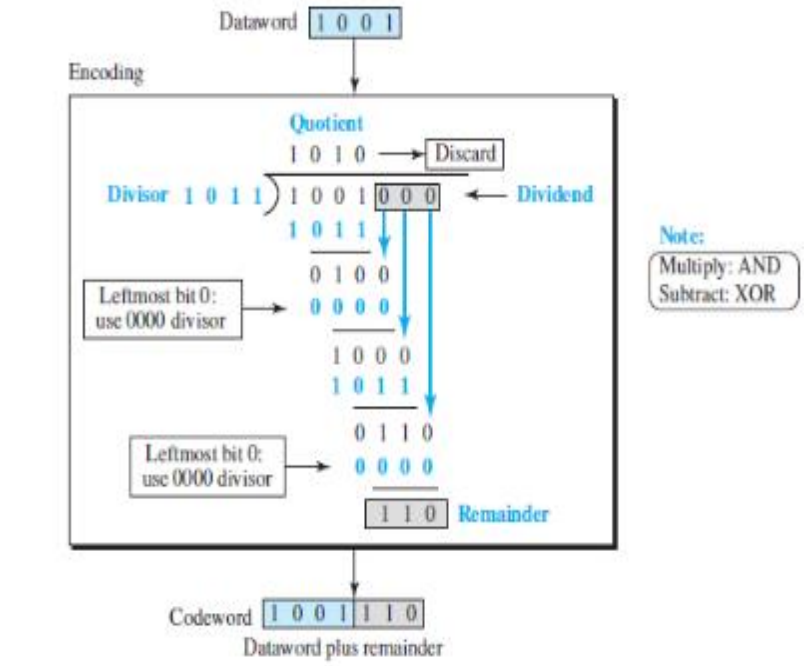


The decoder receives the codeword (possibly corrupted in transition). A copy of all n bits is fed to the checker, which is a replica of the generator. The remainder produced by the checker is a syndrome of $n - k$ (3 here) bits, which is fed to the decision logic analyzer. The analyzer has a simple function.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

Encoder Let us take a closer look at the encoder. The encoder takes a dataword and augments it with $n - k$ number of 0s. It then divides the augmented dataword by the divisor, as shown in Figure 3.13.

Figure 3.13 *Division in CRC encoder*

Figure 3.13 *Division in CRC encoder*



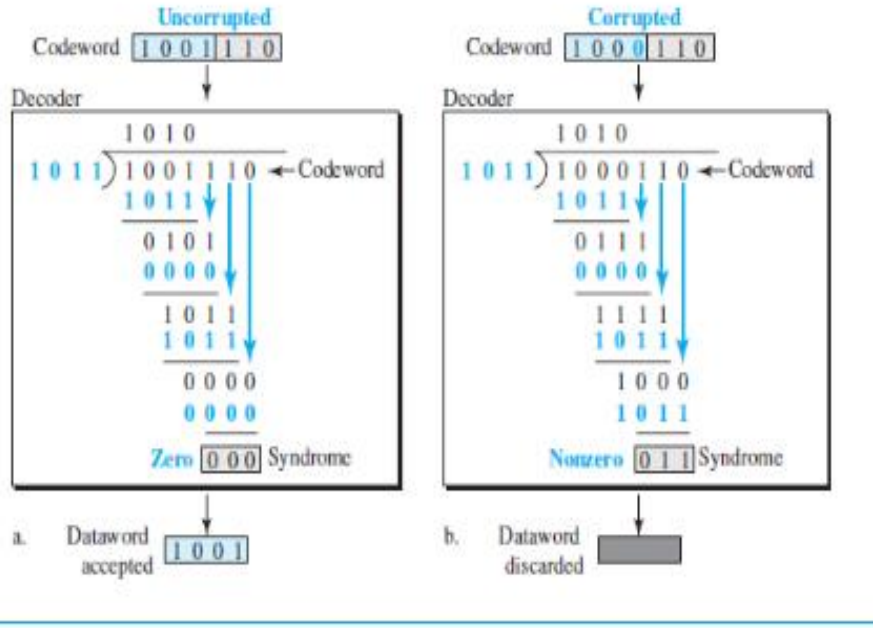
The process of modulo-2 binary division is the same as the familiar division process we use for decimal numbers. However, addition and subtraction in this case are the same; we use the XOR operation to do both.

As in decimal division, the process is done step by step. In each step, a copy of the divisor is XORed with the 4 bits of the dividend. The result of the XOR operation (remainder) is 3 bits (in this case), which is used for the next step after 1 extra bit is pulled down to make it 4 bits long. There is one important point we need to remember in this type of division. If the leftmost bit of the dividend (or the part used in each step) is 0, the step cannot use the regular divisor; we need to use an all-0s divisor.

When there are no bits left to pull down, we have a result. The 3-bit remainder forms the check bits (r_2, r_1, and r_0). They are appended to the dataword to create the codeword.

Decoder The codeword can change during transmission. The decoder does the same division process as the encoder. The remainder of the division is the syndrome. If the syndrome is all 0s, there is no error with a high probability; the dataword is separated from the received codeword and accepted. Otherwise, everything is discarded. Figure 3.14 shows two cases: Figure 3.14a shows the value of the syndrome when no error has occurred; the syndrome is 000. Figure 3.14b shows the case in which there is a single error. The syndrome is not all 0s (it is 011).

Figure 3.14 *Division in the CRC decoder for two cases*



Divisor You may be wondering how the divisor 1011 is chosen. This depends on the expectation we have from the code. Some of the standard divisors used in networking are shown in Table 3.4. The number in the name of the divisor (for example, CRC-32) refers to the degree of the polynomial (the highest power) representing the divisor. The number of bits is always one more than the degree of the polynomial. For example, CRC-8 has 9 bits and CRC-32 has 33 bits.

Name	*Binary*	*Application*
CRC-8	100000111	ATM header
CRC-10	11000110101	ATM AAL
CRC-16	10001000000100001	HDLC
CRC-32	100000100110000010001110110110111	LANs

Requirement

We can mathematically prove that a bit pattern needs to have at least two properties to be considered a generator (divisor):

1. **The pattern should have at least 2 bits.**
2. **The rightmost and leftmost bits should both be 1s.**

Performance

The following shows the performance of CRC.

- **Single errors.** All qualified generators (see above) can detect any single-bit error.
- **Odd number of errors.** All qualified generators can detect any odd number of errors if the generator can be evenly divided by $(11)_2$ using binary division in modulo-2 arithmetic; otherwise, only some odd number errors will be detected.
- **Burst errors.** If we assume the length of the burst error is L bits and r is the length of the remainder (r is the length of the generator minus 1; it is also the value of the highest power in the polynomial representing the generator):
    - All burst errors of the size $L \leq r$ are detected.
    - All burst errors of the size $L = r + 1$ are detected with probability $1 - (0.5)^{r-1}$.
    - All burst errors of the size $L > r + 1$ are detected with probability $1 - (0.5)^{r}$.

Advantages of Cyclic Codes

Cyclic codes 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. The book website shows how division can be done by a shift register that is included in the hardware of the node.

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**. We discuss it when we discuss the network layer.

3.3 MEDIA ACCESS PROTOCOLS

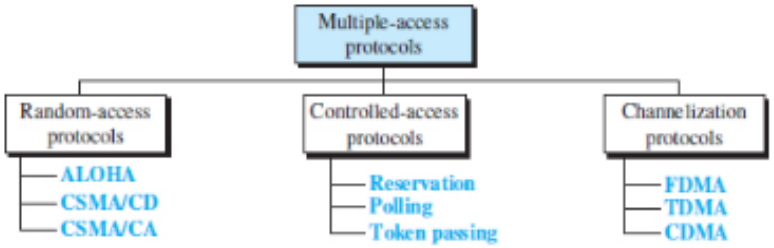
We said that the **data-link layer is divided into two sublayers**: **data-link control (DLC)** and **media access control (MAC).** We discussed DLC in Section 3.2; we talk about **media access control (MAC)** in this section. When we are using a dedicated link, such as a **dial-up telephone line**, we need **only** a **data-link-control protocol, such as the Point-to-Point Protocol (PPP),** that *manages the data transfer between the two ends*. On the other hand, if we are *sharing the media, wire or air, with other users, we need to have a protocol to first manage the sharing process and then to do the data transfer*. For example, **if we use our cellular phone to connect to another cellular phone**, the channel (the band allocated to the vendor company) is not dedicated. A person a few feet away from us may be using the same band to talk to her friend.

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 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. The situation is similar for multipoint networks. **We need to be sure that each node gets access to**

the link. The first goal is to **prevent any collision** between nodes. If somehow a collision does occur, the second goal is **to handle the collision**.

Many protocols have been devised to handle access to a shared link. We categorize them into three groups. Protocols belonging to each group are shown in Figure 3.23.

Figure 3.23 *Taxonomy of multiple-access protocols discussed in this chapter*



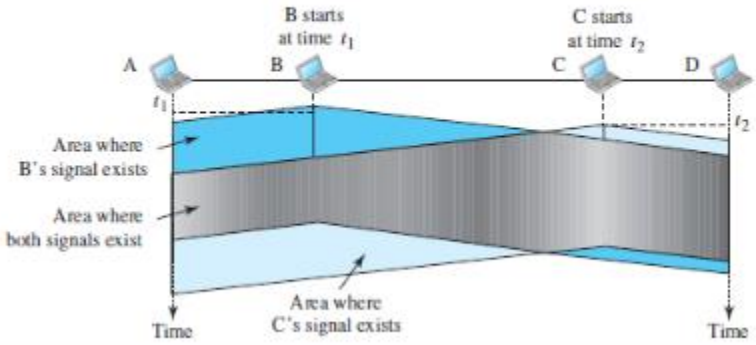
Carrier Sense Multiple Access (CSMA)

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

CSMA can reduce the possibility of collision, but it cannot eliminate it. The reason for this is shown in Figure 3.29, a space and time model of a CSMA network. Stations are connected to a shared channel (usually a dedicated medium).

Figure 3.29 *Space and time model of a collision in CSMA*

Figure 3.29 *Space and time model of a collision in CSMA*



The possibility of collision still exists because of propagation delay; when a station sends a frame, it still takes time (although very short) for the first bit to reach every station and for every station to sense it. In other words, a station may sense the medium and find it idle, only because the first bit sent by another station has not yet been received.

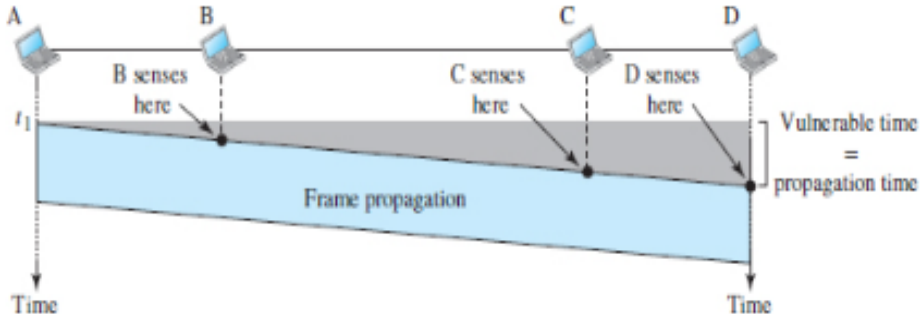
At time t_1, station B senses the medium and finds it idle, so it sends a frame. At time t_2 ($t_2 > t_1$), station C senses the medium and finds it idle because, at this time, the first bits from station B have not reached station C. Station C also sends a frame. The two signals collide, and both frames are destroyed.

Vulnerable Time

The vulnerable time for CSMA is the ***propagation time*** *T*p. **This is the time needed for a signal to propagate from one end of the medium to the other.** When a station sends a frame and any other station tries to send a frame during this time, a collision will result. But if the first bit of the frame reaches the end of the medium, every station will already have heard the bit and will refrain from sending. Figure 3.30 shows the worst case. The leftmost station A sends a frame at time t_1, which reaches the rightmost station D at time $t_1 + T$p. The gray area shows the vulnerable area in time and space.

Figure 3.30 *Vulnerable time in CSMA*

Figure 3.30 *Vulnerable time in CSMA*



Persistence Methods

What should a station do if the channel is busy? What should a station do if the channel is idle? Three methods have been devised to answer these questions: **1-persistent method**, **nonpersistent method**, and ***p-persistent method***. Figure 3.31 shows the behavior of these three persistence methods when a station finds a channel busy.

Figure 3.31 *Behavior of three persistence methods*

Figure 3.31 *Behavior of three persistence methods*

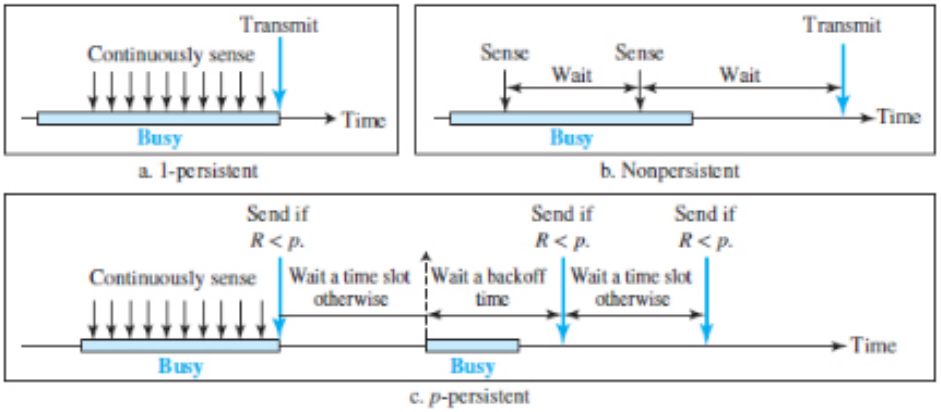


Figure 3.32 shows the flow diagrams for these methods.

Figure 3.32 shows the flow diagrams for these methods.

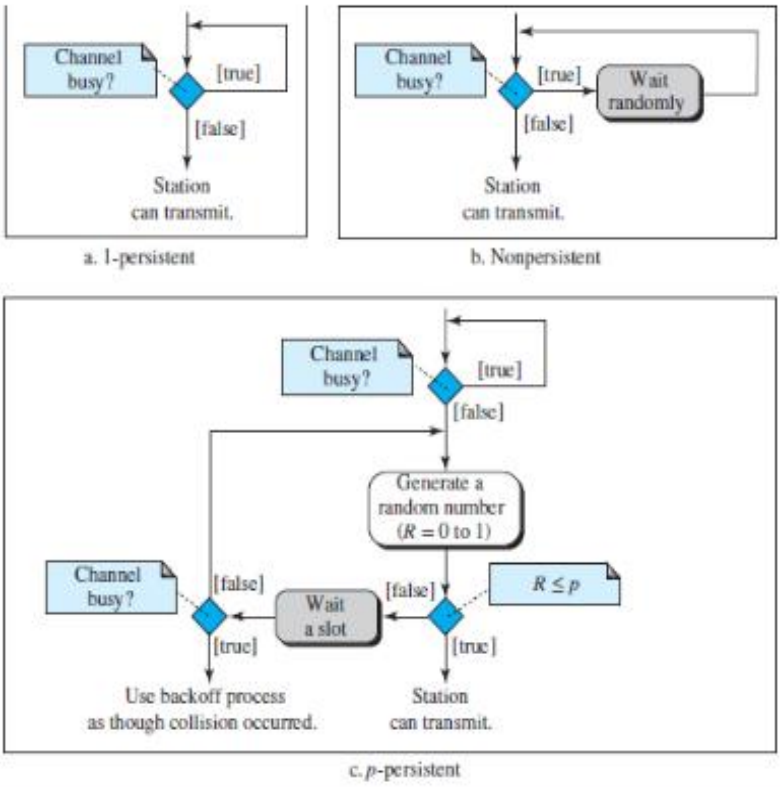
1-Persistent The *1-persistent method* is simple and straightforward. In this method, after the station finds the line idle, it sends its frame immediately (with probability 1). This method has the highest chance of collision because two or more stations may **find the line idle and send their frames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

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

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

1. **With probability *p*, the station sends its frame.**

Figure 3.32 *Flow diagram for three persistence methods*



2. **With probability *q* = 1 − *p*, the station waits for the beginning of the next time slot and checks the line again.**
   a. **If the line is idle, it goes to step 1.**
   b. **If the line is busy, it acts as though a collision has occurred and uses the backoff procedure.**

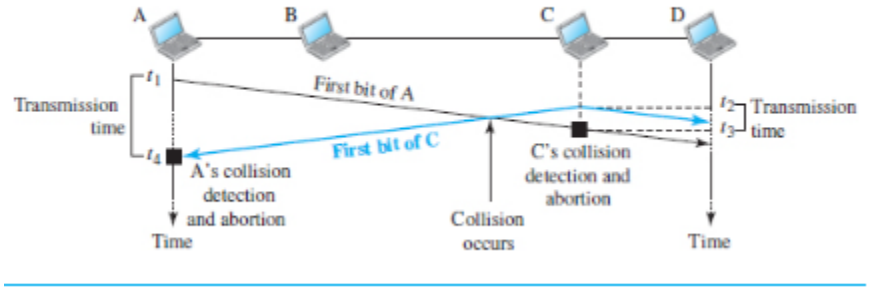
CSMA/CD

The CSMA method does not specify the procedure following a collision. **Carrier sense multiple access with collision detection (CSMA/CD)** augments the algorithm to handle the collision.

In this method, a station monitors the medium after it sends a frame to see if the transmission was successful. If so, the station is finished. If, however, there is a collision, the frame is sent again.

To better understand CSMA/CD, let us look at the first bits transmitted by the two stations involved in the collision. Although each station continues to send bits in the frame until it detects the collision, we show what happens as the first bits collide. In Figure 3.33, stations A and C are involved in the collision.

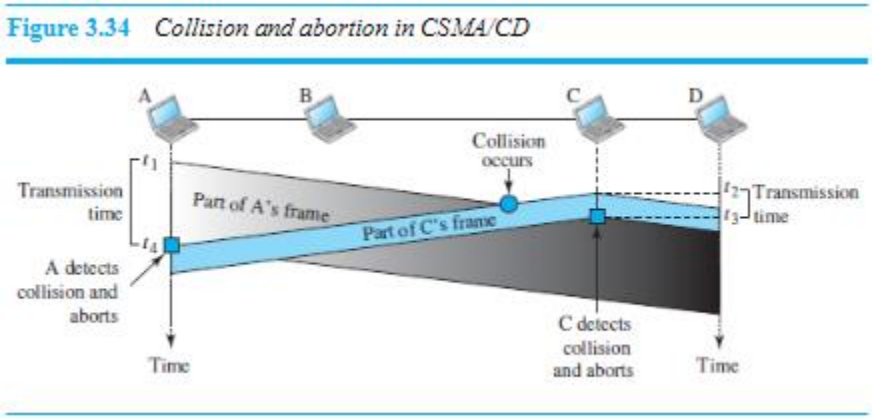
Figure 3.33 *Collision of the first bits in CSMA/CD*



At time t_1, station A has executed its persistence procedure and starts sending the bits of its frame. At time t_2, station C has not yet sensed the first bit sent by A. Station C executes its persistence procedure and starts sending the bits in its frame, which propagate both to the left and to the right. The collision occurs sometime after time t_2. Station C detects a collision at time t_3 when it receives the first bit of A's frame. Station C immediately (or after a short time, but we assume immediately) aborts transmission. Station A detects a collision at time t_4 when it receives the first bit of C's frame; it also immediately aborts transmission. Looking at Figure 3.33, we see that **A transmits for the duration $t_4 - t_1$; C transmits for the duration $t_3 - t_2$.**

Now that we know the time durations for the two transmissions, we can show a more complete graph in Figure 3.34.

Figure 3.34 *Collision and abortion in CSMA/CD*



Minimum Frame Size

For CSMA/CD to work, we need a **restriction on the frame size**. Before sending the last bit of the frame, the sending station must detect a collision, if any, and abort the transmission. This is so because the station, once the entire frame is sent, does not keep a copy of the frame and does not monitor the line for collision detection. Therefore, the **frame transmission time T_{fr}** must be at least 2 times the **maximum propagation time T_p**. To understand the reason, let us think about the

worst-case scenario. If the two stations involved in a collision are the maximum distance apart, the signal from the first takes time T_p to reach the second, and the effect of the collision takes another time T_P to reach the first. So the requirement is that the first station must still be transmitting after $2T_p$.

Example 3.12

A network using CSMA/CD has a bandwidth of 10 Mbps. If the maximum propagation time (including the delays in the devices and ignoring the time needed to send a jamming signal, as we see later) is 25.6 μs, what is the minimum size of the frame?

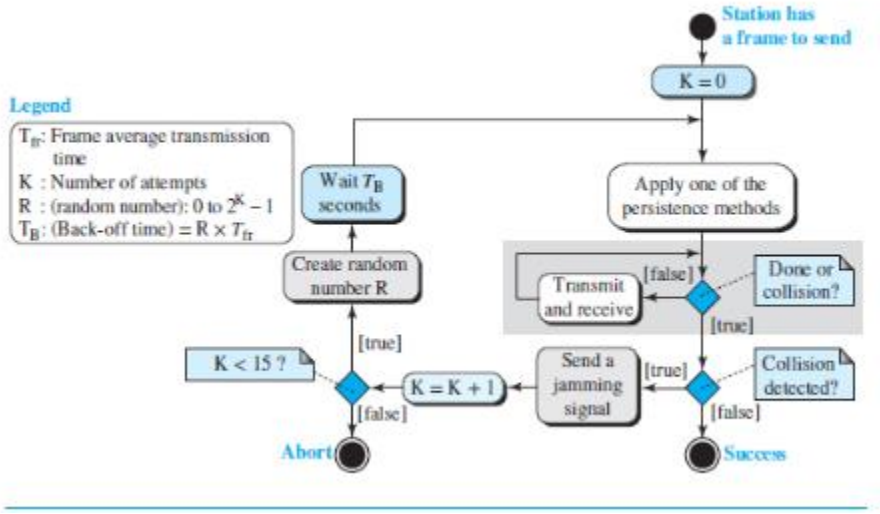
Solution

The minimum frame transmission time is $T_{fr} = 2 \times Tp = 51.2$ μs. This means, in the worst case, a station needs to transmit for a period of 51.2 μs to detect the collision. The minimum size of the frame is 10 Mbps × 51.2 μs = 512 bits or 64 bytes. This is actually the minimum size of the frame for Standard Ethernet, as we will see in Chapter 4.

Procedure

Now let us look at the flow diagram for CSMA/CD in Figure 3.35.

Figure 3.35 *Flow diagram for the CSMA/CD*



The first is of the persistence process. We need to sense the channel before we start sending the frame by using one of the persistence processes we discussed previously (nonpersistent, 1-persistent, or *p*-persistent). The corresponding box can be replaced by one of the persistence processes shown in Figure 3.32.

The second is the frame transmission. In ALOHA, we first transmit the entire frame and then wait for an acknowledgment. In CSMA/CD, transmission and collision detection is a continuous process. We do not send the entire frame and then look for a collision. The station transmits and receives continuously and simultaneously (using two different ports or a bidirectional port). We use a loop to show that transmission is a continuous process. **We constantly monitor to detect one of two conditions:** Either transmission is finished, or a collision is detected. Either event stops

transmission. When we come out of the loop, if a collision has not been detected, it means that transmission is complete; the entire frame is transmitted. Otherwise, a collision has occurred.

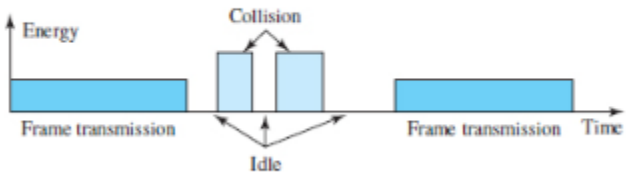
The third is the sending of a short *jamming signal* to make sure that all other stations become aware of the collision.

Energy Level

We can say that the level of energy in a channel can have three values: zero, normal, and abnormal. **At the zero level**, the channel is idle. **At the normal level**, a station has successfully captured the channel and is sending its frame. **At the abnormal level**, there is a collision and the level of the energy is twice the normal level. A station that has a frame to send or is sending a frame needs to monitor the energy level to determine if the channel is idle, busy, or in collision mode. Figure 3.36 shows the situation.

Figure 3.36 *Energy level during transmission, idleness, or collision*

Figure 3.36 *Energy level during transmission, idleness, or collision*



Traditional Ethernet

One of the LAN protocols that used CSMA/CD is the traditional Ethernet with the data rate of 10 Mbps. We discuss the Ethernet LANs in Chapter 4. but it is good to know that the traditional Ethernet was a broadcast LAN that used 1-persistence method to control access to the common media. Later versions of the Ethernet try to move from CSMA/CD access methods for the reason that we discuss in Chapter 4 when we discuss wired LANs.

3.4 LINK-LAYER ADDRESSING

The next issue we need to discuss about the data-link layer is the **link-layer addresses**. In Chapter 7, we will discuss IP addresses as the identifiers at the network layer that define the exact points in the Internet where the source and destination hosts are connected. However, in a connectionless internetwork such as the Internet we cannot make a datagram reach its destination using only IP addresses. The reason is that each datagram in the Internet, from the same source host to the same destination host, may take a different path. The source and destination IP addresses define the two ends but cannot define which links the datagram should pass through.

The above discussion shows that we need another addressing mechanism in a connectionless internetwork: the link-layer addresses of the two nodes. A ***link-layer address*** is sometimes called a ***link address***, sometimes a ***physical address***, and sometimes a ***MAC address.*** We use these terms interchangeably in this book.

Because 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.

3.4.1 Three Types of Addresses

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

Unicast Address

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

Example 3.13

As we will see in Chapter 4, the unicast link-layer addresses in the most **common LAN, the Ethernet**, are 48 bits (6 bytes) that are presented as 12 hexadecimal digits separated by colons; for example, the following is a link-layer address of a computer.

A3:34:45:11:92:F1

Multicast Address

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

Example 3.14

As we will see in Chapter 4, the multicast link-layer addresses in the **most common LAN**, the **Ethernet,** are 48 bits (6 bytes) that are presented as 12 hexadecimal digits separated by colons. The second digit, however needs to be an even number in hexadecimal. The following shows a multicast address:

A2:34:45:11:92:F1

Broadcast Address

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

Example 3.15

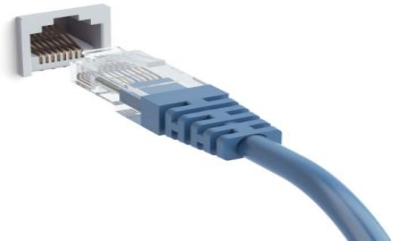
As we will see in Chapter 4, the broadcast link-layer addresses in the **most common LAN, the Ethernet,** are 48 bits that are all 1s and presented as 12 hexadecimal digits separated by colons. The following shows a broadcast address:

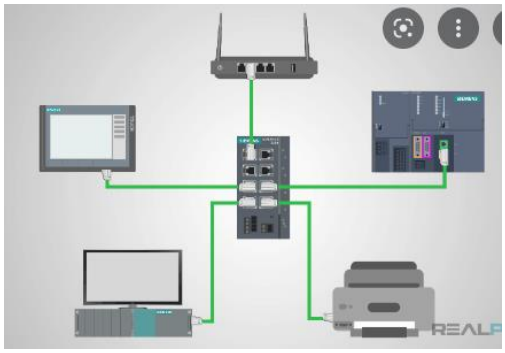
FF:FF:FF:FF:FF:FF

CHAPTER 4

Local Area Networks: LANs

4.1 ETHERNET





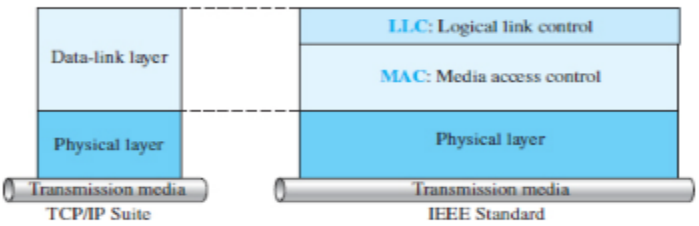
Ethernet, pronounced "E-thernet" (with a long "e"), is the standard way to connect computers on a network over a wired connection. It provides a simple interface and for connecting multiple devices, such computers, routers, and switches.

Local area network (LAN) is a computer network that is designed for a limited geographic area such as a building or campus. Although a LAN can be used as an isolated network to connect computers in an organization for the sole purpose of sharing resources, most LANs today are also linked to a wide area network (WAN) or the Internet.

In the 1980s and 1990s, several different types of wired LANs were used. The Institute of Electrical and Electronics Engineers (IEEE) has subdivided the data-link layer into two sublayers: **logical link control (LLC)** and **media access control (MAC)**. Note that logical link control is similar to data-link control (DLC), which we discussed in Chapter 3. The IEEE has also created several physical-layer standards for different LAN protocols. All these wired LANs use a media access method to solve the problem of sharing the media. The relationship of the IEEE 802 standard to the TCP/IP protocol suite is shown in Figure 4.1.

Figure 4.1 *IEEE standard for wired LANs*

Figure 4.1 *IEEE standard for wired LANs*

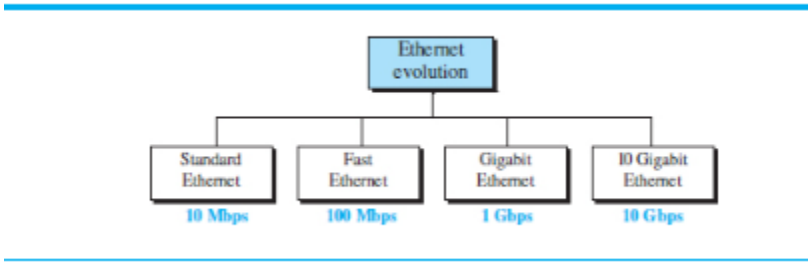


Almost every wired LAN technology except Ethernet has disappeared from the marketplace because Ethernet was able to update itself to meet the needs of the time. This means that we confine our discussion of wired LANs to the discussion of Ethernet.

Before we discuss the Ethernet protocol and all its generations, we need to briefly discuss the IEEE standard that we often encounter in text or real life. In 1985, the Computer Society of the IEEE started a project, called **Project 802**, to set standards to enable intercommunication among equipment from a variety of manufacturers. Project 802 does not seek to replace any part of the OSI model or TCP/IP protocol suite. Instead, it is a way of specifying functions of the physical layer and the data-link layer of major LAN protocols.

The Ethernet LAN was developed in the 1970s by Robert Metcalfe and David Boggs. Since then, it has gone through four generations: **Standard Ethernet** (10 Mbps), **Fast Ethernet** (100 Mbps), **Gigabit Ethernet** (1 Gbps), and **10 Gigabit Ethernet** (10 Gbps), as shown in Figure 4.2.

Figure 4.2 *Ethernet evolution through four generations*



Ethernet has multiple standards that all use the same interface. These include:

10BASE-Two - supports up to 10 Mbps
100BASE-T - supports up to 100 Mbps
1000BASE-T (also called "Gigabit Ethernet") - supports up to 1,000 Mbps
10000BASE-T (also called "10 Gigabit Ethernet") - supports up to 10,000 Mbps

4.1.1 Standard Ethernet (10 Mbps)

We refer to the original Ethernet technology with the data rate of 10 Mbps as the Standard Ethernet. Although most implementations have moved to other technologies in the Ethernet evolution, there are some features of the Standard Ethernet that have not changed during the evolution. We discuss this standard version to pave the way for understanding the other three technologies.

Connectionless and Unreliable Service

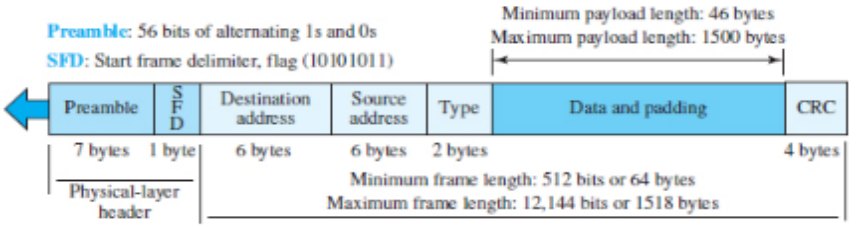
Ethernet provides a connectionless service, which means each frame sent is independent of the previous or next frame. Ethernet has no connection establishment or connection termination phases. The sender sends a frame whenever it has it; the receiver may or may not be ready for it. The sender may overwhelm the receiver with frames, which may result in dropped frames. If a frame drops, the sender data-link layer will not know about it unless an upper-layer protocol takes

care of it. Ethernet is also **unreliable.** If a frame is corrupted during transmission and the receiver finds out about the corruption, the receiver drops the frame silently. It is the duty of high-level protocols to find out about it.

Frame Format

The Ethernet frame contains seven fields, as shown in Figure 4.3.

Figure 4.3 *Ethernet frame*



- **Preamble.** This field contains 7 bytes (56 bits) of alternating 0s and 1s that alert the receiving system to the coming frame and enable it to synchronize its clock if it's out of synchronization. The pattern provides only an alert and a timing pulse. The 56-bit pattern allows the stations to miss some bits at the beginning of the frame. The *preamble* is actually added at the physical layer and is not (formally) part of the frame.
- **Start frame delimiter (SFD).** This field (1 byte: 10101011) signals the beginning of the frame. The SFD warns the station or stations that this is the last chance for synchronization. The last 2 bits are $(11)_2$ and alert the receiver that the next field is the destination address. This field is actually a flag that defines the beginning of the frame. We need to remember that an Ethernet frame is a variable-length frame. It needs a flag to define the beginning of the frame. The SFD field is also added at the physical layer.
- **Destination address (DA).** This field is 6 bytes (48 bits) and contains the link-layer address of the destination station or stations to receive the packet. We will discuss addressing shortly. When the receiver sees its own link-layer address, a multicast address for a group that the receiver is a member of, or a broadcast address, it decapsulates the data from the frame and passes the data to the upper-layer protocol defined by the value of the type field.
- **Source address (SA).** This field is also 6 bytes and contains the link-layer address of the sender of the packet.
- **Type.** This field defines the upper-layer protocol whose packet is encapsulated in the frame. This protocol can be IP, ARP, Open Shortest Path First (OSPF), and so on as we will see in the next chapters.
- **Data.** This field carries data encapsulated from the upper-layer protocols. It is a minimum of 46 and a maximum of 1500 bytes. If the data coming from the upper layer is more than 1500 bytes, it should be fragmented and encapsulated in more than one frame. If it is less than 46 bytes, it needs to be padded with extra 0s. A padded data frame is delivered to the upper-layer protocol as it is (without removing the padding), which means that it is the responsibility of the upper layer to remove or add the padding. The upper-layer protocol needs to know the length of its data.

- **CRC.** The last field contains error-detection information, in this case a CRC-32. The CRC is calculated over the addresses, types, and data field. If the receiver calculates the CRC and finds that it is not zero (corruption in transmission), it discards the frame.