2. Protocols

2.1 Difference between Protocol, Algorithm and Standard

- *Protocol*: A protocol is a set of rules that governs how a system operates. The rules establish the basic functioning of the different parts, how they interact with each other, and what conditions are necessary for a healthy implementation. The different parts of a protocol are not sensitive to order or chronology – it does not matter which part is enacted first. And a protocol does not tell the system how to produce a result. It does not have an objective other than a smooth execution. It does not produce an output.
- *Algorithm*: An algorithm, on the other hand, is a set of instructions that produces an output or a result. It can be a simple script, or a complicated program. The order of the instructions is important, and the algorithm specifies what that order is. It tells the system what to do in order to achieve the desired result. It may not know what the result is beforehand, but it knows that it wants one.
- *Protocol vs Algorithm*: A protocol is like the engine of a car, how a car works. An algorithm is what you need to do to drive the car, the actions that the driver performs. The protocol is a set of rules that determines how the system functions. The algorithm tells the system what to do. The protocol is. The algorithm does. In the kitchen, the protocol would be a set of conditions and instructions such as: The knife cuts, the flame heats, olive oil is delicious, frying pans are good for sauteing onions, wash your hands before handling food, burnt food tastes bad. An algorithm in the same kitchen could be: First, chop the onion, then, heat up the olive oil in the pan, put the onion in the pan, add some salt, and stir until the onion is translucent.
- *Standard*: Standards are essential in creating and maintaining an open and competitive market for equipment manufacturers and in guaranteeing national and international interoperability of data and telecommunications technology and processes. Standards provide guidelines to manufacturers, vendors, government agencies, and other service providers to ensure the kind of inter-connectivity necessary in today's marketplace and in international communications.

2.2 The need for a Protocol Architecture

When computers, terminals, and/or other data processing devices exchange data, the procedures involved can be quite complex. Consider, for example, the transfer of a file between two computers. There must be a data path between the two computers, either directly or via a communication network. But more is needed.Typical tasks to be performed are as follows:

1. The source system must either activate the direct data communication path or inform the communication network of the identity of the desired destination system.
2. The source system must ascertain that the destination system is prepared to receive data.
3. The file transfer application on the source system must ascertain that the file management program on the destination system is prepared to accept and store the file for this particular user.
4. If the file formats used on the two systems are different, one or the other system must perform a format translation function.

It is clear that there must be a high degree of cooperation between the two computer systems. Instead of implementing the logic for this as a single module, the task is broken up into sub-tasks, each of which is implemented separately. In a protocol architecture, the modules are arranged in a vertical stack. Each layer in the stack performs a related subset of the functions required to communicate with another system. It relies on the next lower layer to perform more primitive functions and to conceal the details of those functions. It provides services to the next higher layer. Ideally, layers should be defined so that changes in one layer do not require changes in other layers. Of course, it takes two to communicate, so the same set of layered functions must exist in two systems. Communication is achieved by having the corresponding, or peer, layers in two systems communicate.The peer layers communicate by means of formatted blocks of data that obey a set of rules or conventions known as a protocol. The key features of a protocol are as follows:

- Syntax: Concerns the format of the data blocks
- Semantics: Includes control information for coordination and error handling
- Timing: Includes speed matching and sequencing

2.3 Layered Tasks

To reduce design complexity, most networks are organized as a stack of layers or levels, each one built upon the one below it. For example, let us consider two friends who communicate through postal mail. The process of sending a letter to a friend would be complex if there were no services available from the post office. Fig. 2.1 shows the steps in this task. In Fig. 2.1, we have a sender, a receiver, and a carrier that transports the letter. There is a hierarchy of tasks.

Let us first describe, in order, the activities that take place at the sender site. The sender writes the letter, inserts the letter in an envelope, writes the sender and receiver addresses, and drops the letter in a mailbox (Higher Layer). The letter is picked up by a letter carrier and delivered to the post office (Middle layer). The letter is sorted at the post office; a carrier transports the letter (Lower layer).

The letter is then on its way to the recipient. On the way to the recipient's local post office, the letter may actually go through a central office. In addition, it may be transported by truck, train, airplane, boat, or a combination of these.

At the Receiver Site, The carrier transports the letter to the post office (Lower layer). The letter is sorted and delivered to the recipient's mailbox (Middle layer). The receiver picks up the letter, opens the envelope, and reads it (Higher layer).

According to our analysis, there are three different activities at the sender site and another three activities at the receiver site. The task of transporting the letter between the sender and the receiver is done by the carrier. Something that is not obvious immediately is that the tasks must be done in

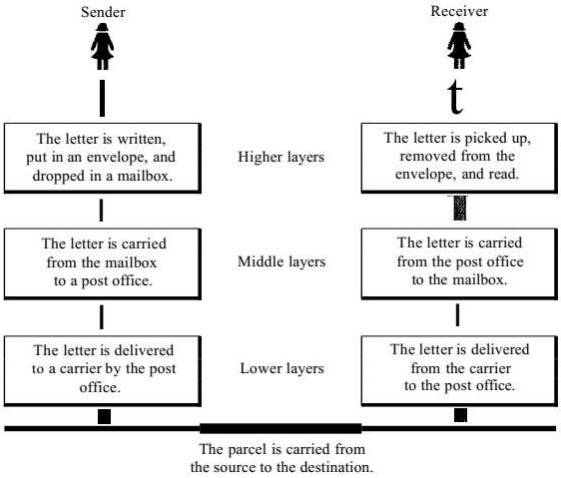

Figure 2.1: Use of Layers in delivering a letter

the order given in the hierarchy. At the sender site, the letter must be written and dropped in the mailbox before being picked up by the letter carrier and delivered to the post office. At the receiver site, the letter must be dropped in the recipient mailbox before being picked up and read by the recipient.

Let us take another example, Imagine two philosophers (peer processes in layer 3), one of whom speaks Urdu and English and one of whom speaks Chinese and French. Since they have no common language, they each engage a translator (peer processes at layer 2), each of whom in turn contacts a secretary (peer processes in layer 1). Philosopher 1 wishes to convey his affection for oryctolagus cuniculus to his peer. To do so, he passes a message (in English) across the 2/3 interface to his translator, saying "I like rabbits," as illustrated in Fig. 2.2. The translators have agreed on a neutral language known to both of them, Dutch, so the message is converted to "Ik vind konijnen leuk." The choice of the language is the layer 2 protocol and is up to the layer 2 peer processes.

The translator then gives the message to a secretary for transmission, for example, by email (the layer 1 protocol). When the message arrives at the other secretary, it is passed to the local translator, who translates it into French and passes it across the 2/3 interface to the second philosopher. Note that each protocol is completely independent of the other ones as long as the interfaces are not changed. The translators can switch from Dutch to, say, Finnish, at will, provided that they both agree and neither changes his interface with either layer 1 or layer 3. Similarly, the secretaries can switch from email to telephone without disturbing (or even informing) the other layers. Each process may add some information intended only for its peer. This information is not passed up to the layer above.

The number of layers, the name of each layer, the contents of each layer, and the function of each layer differ from network to network. The purpose of each layer is to offer certain services to the higher layers while shielding those layers from the details of how the offered services are actually implemented. In a sense, each layer is a kind of virtual machine, offering certain services to the layer above it.

This concept is actually a familiar one and is used throughout computer science, where it is variously known as information hiding, abstract data types, data encapsulation, and object-oriented programming. The fundamental idea is that a particular piece of software (or hardware) provides a service to its users but keeps the details of its internal state and algorithms hidden from them.

When layer n on one machine carries on a conversation with layer n on another machine, the rules and conventions used in this conversation are collectively known as the layer n protocol.

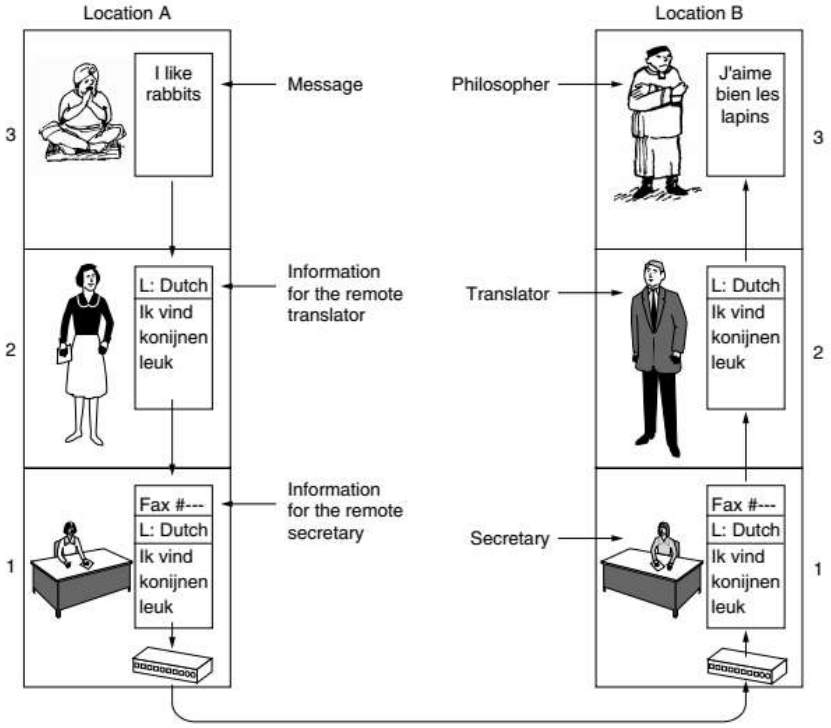

Figure 2.2: The philosopher-translator-secretary architecture.

Basically, a protocol is an agreement between the communicating parties on how communication is to proceed. As an analogy, when a woman is introduced to a man, she may choose to stick out her hand. He, in turn, may decide to either shake it or kiss it, depending, for example, on whether she is an American lawyer at a business meeting or a European princess at a formal ball. Violating the protocol will make communication more difficult, if not completely impossible.

In reality, no data are directly transferred from layer n on one machine to layer n on another machine. Instead, each layer passes data and control information to the layer immediately below it, until the lowest layer is reached. Below layer 1 is the physical medium through which actual communication occurs. In Fig. 2.3, virtual communication between *peers* is shown by dotted lines and physical communication by solid lines. Between each pair of adjacent layers is an *interface*. The *interface* defines which primitive operations and services the lower layer makes available to the upper one. When network designers decide how many layers to include in a network and what each one should do, one of the most important considerations is defining clean interfaces between the layers. Doing so, in turn, requires that each layer perform a specific collection of well-understood functions. In addition to minimizing the amount of information that must be passed between layers, clear cut interfaces also make it simpler to replace one layer with a completely different protocol or implementation (e.g., replacing all the telephone lines by satellite channels) because all that is required of the new protocol or implementation is that it offer exactly the same set of services to its upstairs neighbor as the old one did. It is common that different hosts use different implementations of the same protocol (often written by different companies). In fact, the protocol itself can change in some layer without the layers above and below it even noticing.

A set of layers and protocols is called a *network architecture*. The specification of an architecture must contain enough information to allow an implementer to write the program or build the hardware for each layer so that it will correctly obey the appropriate protocol. Neither the details of the implementation nor the specification of the interfaces is part of the architecture because these are hidden away inside the machines and not visible from the outside. It is not even necessary that the

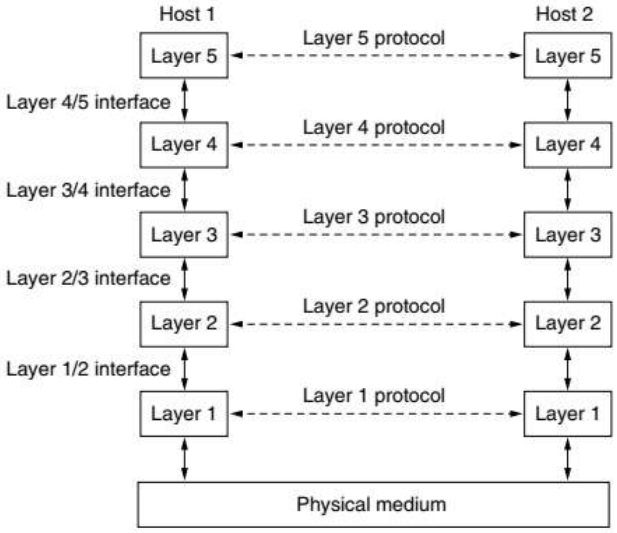

Figure 2.3: Layers, protocols, and interfaces.

interfaces on all machines in a network be the same, provided that each machine can correctly use all the protocols. *A list of the protocols used by a certain system, one protocol per layer, is called a* ***protocol stack***.

2.3.1 Design Issues for the Layers

Some of the key design issues that occur in computer networks will come up in layer after layer. Below, we will briefly mention the more important ones. ***Reliability*** is the design issue of making a network that operates correctly even though it is made up of a collection of components that are themselves unreliable. Think about the bits of a packet traveling through the network. There is a chance that some of these bits will be received damaged (inverted) due to fluke electrical noise, random wireless signals, hardware flaws, software bugs and so on. How is it possible that we find and fix these errors? One mechanism for finding errors in received information uses codes for error detection. Information that is incorrectly received can then be retransmitted until it is received correctly.

More powerful codes allow for ***error correction***, where the correct message is recovered from the possibly incorrect bits that were originally received. Both of these mechanisms work by adding redundant information. They are used at low layers, to protect packets sent over individual links, and high layers, to check that the right contents were received.

Another reliability issue is finding a working path through a network. Often there are multiple paths between a source and destination, and in a large network, there may be some links or routers that are broken. Suppose that the network is down in Germany. Packets sent from London to Rome via Germany will not get through, but we could instead send packets from London to Rome via Paris. The network should automatically make this decision. This topic is called ***routing***.

A second design issue concerns the ***evolution of the network***. Over time, networks grow larger and new designs emerge that need to be connected to the existing network. We have recently seen the key structuring mechanism used to support change by dividing the overall problem and hiding implementation details: protocol layering. There are many other strategies as well.

Since there are many computers on the network, every layer needs a mechanism for identifying the senders and receivers that are involved in a particular message. This mechanism is called ***addressing or naming***, in the low and high layers, respectively.

An aspect of growth is that different network technologies often have different limitations. For

example, not all communication channels preserve the order of messages sent on them, leading to solutions that number messages. Another example is differences in the maximum size of a message that the networks can transmit. This leads to mechanisms for disassembling, transmitting, and then reassembling messages. This overall topic is called ***internetworking***.

When networks get large, new problems arise. Cities can have traffic jams, a shortage of telephone numbers, and it is easy to get lost. Not many people have these problems in their own neighborhood, but citywide they may be a big issue. Designs that continue to work well when the network gets large are said to be ***scalable***.

A third design issue is ***resource allocation***. Networks provide a service to hosts from their underlying resources, such as the capacity of transmission lines. To do this well, they need mechanisms that divide their resources so that one host does not interfere with another too much.

Many designs share network bandwidth dynamically, according to the short term needs of hosts, rather than by giving each host a fixed fraction of the bandwidth that it may or may not use. This design is called ***statistical multiplexing***, meaning sharing based on the statistics of demand. It can be applied at low layers for a single link, or at high layers for a network or even applications that use the network.

An allocation problem that occurs at every level is how to keep a fast sender from swamping a slow receiver with data. Feedback from the receiver to the sender is often used. This subject is called ***flow control***. Sometimes the problem is that the network is oversubscribed because too many computers want to send too much traffic, and the network cannot deliver it all. This overloading of the network is called ***congestion***. One strategy is for each computer to reduce its demand when it experiences congestion. It, too, can be used in all layers.

It is interesting to observe that the network has more resources to offer than simply bandwidth. For uses such as carrying live video, the timeliness of delivery matters a great deal. Most networks must provide service to applications that want this real-time delivery at the same time that they provide service to applications that want high throughput. Quality of service is the name given to mechanisms that reconcile these competing demands.

The last major design issue is to ***secure the network*** by defending it against different kinds of threats. One of the threats we have mentioned previously is that of eavesdropping on communications. Mechanisms that provide confidentiality defend against this threat, and they are used in multiple layers. Mechanisms for authentication prevent someone from impersonating someone else. They might be used to tell fake banking Web sites from the real one, or to let the cellular network check that a call is really coming from your phone so that you will pay the bill. Other mechanisms for integrity prevent surreptitious changes to messages, i.e., altering.

2.3.2 Connection-Oriented Versus Connectionless Service

Layers can offer two different types of service to the layers above them: connection-oriented and connectionless. In this section we will look at these two types and examine the differences between them. Connection-oriented service is modeled after the telephone system. To talk to someone, you pick up the phone, dial the number, talk, and then hang up. Similarly, to use a connection-oriented network service, the service user first establishes a connection, uses the connection, and then releases the connection. The essential aspect of a connection is that it acts like a tube: the sender pushes objects (bits) in at one end, and the receiver takes them out at the other end. In most cases the order is preserved so that the bits arrive in the order they were sent.

In some cases when a connection is established, the sender, receiver, and subnet conduct a negotiation about the parameters to be used, such as maximum message size, quality of service required, and other issues. Typically, one side makes a proposal and the other side can accept it, reject it, or make a counter proposal. A circuit is another name for a connection with associated resources, such as a fixed bandwidth. This dates from the telephone network in which a circuit was

a path over copper wire that carried a phone conversation.

In contrast to connection-oriented service, connectionless service is modeled after the postal system. Each message (letter) carries the full destination address, and each one is routed through the intermediate nodes inside the system independent of all the subsequent messages. There are different names for messages in different contexts; a packet is a message at the network layer. When the intermediate nodes receive a message in full before sending it on to the next node, this is called store-and-forward switching. The alternative, in which the onward transmission of a message at a node starts before it is completely received by the node, is called cut-through switching. Normally, when two messages are sent to the same destination, the first one sent will be the first one to arrive. However, it is possible that the first one sent can be delayed so that the second one arrives first.

Each kind of service can further be characterized by its reliability. Some services are reliable in the sense that they never lose data. Usually, a reliable service is implemented by having the receiver acknowledge the receipt of each message so the sender is sure that it arrived. The acknowledgement process introduces overhead and delays, which are often worth it but are sometimes undesirable.

A typical situation in which a reliable connection-oriented service is appropriate is file transfer. The owner of the file wants to be sure that all the bits arrive correctly and in the same order they were sent. Very few file transfer customers would prefer a service that occasionally scrambles or loses a few bits, even if it is much faster.

Reliable connection-oriented service has two minor variations: message sequences and byte streams. In the former variant, the message boundaries are preserved. When two 1024-byte messages are sent, they arrive as two distinct 1024-byte messages, never as one 2048-byte message. In the latter, the connection is simply a stream of bytes, with no message boundaries. When 2048 bytes arrive at the receiver, there is no way to tell if they were sent as one 2048-byte message, two 1024-byte messages, or 2048 1-byte messages. If the pages of a book are sent over a network to a phototypesetter as separate messages, it might be important to preserve the message boundaries. On the other hand, to download a DVD movie, a byte stream from the server to the user's computer is all that is needed. Message boundaries within the movie are not relevant.

For some applications, the transit delays introduced by acknowledgements are unacceptable. One such application is digitized voice traffic for voice over IP. It is less disruptive for telephone users to hear a bit of noise on the line from time to time than to experience a delay waiting for acknowledgements. Similarly, when transmitting a video conference, having a few pixels wrong is no problem, but having the image jerk along as the flow stops and starts to correct errors is irritating. Not all applications require connections.

For example, spammers send electronic junk-mail to many recipients. The spammer probably does not want to go to the trouble of setting up and later tearing down a connection to a recipient just to send them one item. Nor is 100 percent reliable delivery essential, especially if it costs more. All that is needed is a way to send a single message that has a high probability of arrival, but no guarantee. Unreliable (meaning not acknowledged) connectionless service is often called datagram service, in analogy with telegram service, which also does not return an acknowledgement to the sender. Despite it being unreliable, it is the dominant form in most networks for reasons that will become clear later. In other situations, the convenience of not having to establish a connection to send one message is desired, but reliability is essential. The acknowledged datagram service can be provided for these applications. It is like sending a registered letter and requesting a return receipt. When the receipt comes back, the sender is absolutely sure that the letter was delivered to the intended party and not lost along the way. Text messaging on mobile phones is an example.

Still another service is the request-reply service. In this service the sender transmits a single datagram containing a request; the reply contains the answer. Request-reply is commonly used to implement communication in the client-server model: the client issues a request and the server responds to it. For example, a mobile phone client might send a query to a map server to retrieve

the map data for the current location. Fig. 2.4 summarizes the types of services discussed above.

	Service	Example
Connection-oriented	Reliable message stream	Sequence of pages
	Reliable byte stream	Movie download
	Unreliable connection	Voice over IP
Connection-less	Unreliable datagram	Electronic junk mail
	Acknowledged datagram	Text messaging
	Request-reply	Database query

Figure 2.4: Six different types of service

The concept of using unreliable communication may be confusing at first. After all, why would anyone actually prefer unreliable communication to reliable communication? First of all, reliable communication (in our sense, that is, acknowledged) may not be available in a given layer. For example, Ethernet does not provide reliable communication. Packets can occasionally be damaged in transit. It is up to higher protocol levels to recover from this problem. In particular, many reliable services are built on top of an unreliable datagram service. Second, the delays inherent in providing a reliable service may be unacceptable, especially in real-time applications such as multimedia. For these reasons, both reliable and unreliable communication coexist.

2.3.3 Service Primitives and Parameters

A service is formally specified by a set of primitives (operations) available to user processes to access the service. These primitives tell the service to perform some action or report on an action taken by a peer entity. If the protocol stack is located in the operating system, as it often is, the primitives are normally system calls. These calls cause a trap to kernel mode, which then turns control of the machine over to the operating system to send the necessary packets. The set of primitives available depends on the nature of the service being provided. The primitives for connection-oriented service are different from those of connectionless service. As a minimal example of the service primitives that might provide a reliable byte stream, consider the primitives listed in Fig. 2.6. They will be familiar to fans of the Berkeley socket interface, as the primitives are a simplified version of that interface.

These primitives might be used for a request-reply interaction in a client-server environment. To illustrate how, We sketch a simple protocol that implements the service using acknowledged datagrams. First, the server executes LISTEN to indicate that it is prepared to accept incoming connections. A common way to implement LISTEN is to make it a blocking system call. After executing the primitive, the server process is blocked until a request for connection appears.

Next, the client process executes CONNECT to establish a connection with the server. The CONNECT call needs to specify who to connect to, so it might have a parameter giving the server's address. The operating system then typically sends a packet to the peer asking it to connect, as shown by (1) in Fig. 2.5. The client process is suspended until there is a response. When the packet arrives at the server, the operating system sees that the packet is requesting a connection. It checks to see if there is a listener, and if so it unblocks the listener. The server process can then establish the connection with the ACCEPT call. This sends a response (2) back to the client process to accept the connection. The arrival of this response then releases the client. At this point the client and server are both running and they have a connection established.

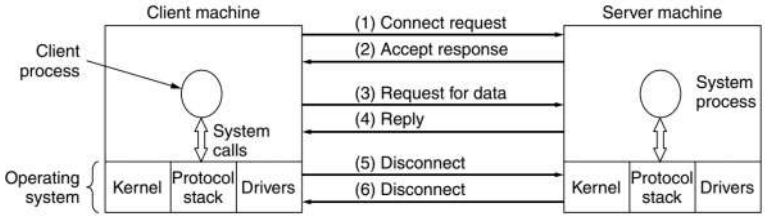

Figure 2.5: A simple client-server interaction using acknowledged datagrams

Primitive	Meaning
LISTEN	Block waiting for an incoming connection
CONNECT	Establish a connection with a waiting peer
ACCEPT	Accept an incoming connection from a peer
RECEIVE	Block waiting for an incoming message
SEND	Send a message to the peer
DISCONNECT	Terminate a connection

Figure 2.6: Six service primitives that provide a simple connection-oriented service

The obvious analogy between this protocol and real life is a customer (client) calling a company's customer service manager. At the start of the day, the service manager sits next to his telephone in case it rings. Later, a client places a call. When the manager picks up the phone, the connection is established. The next step is for the server to execute RECEIVE to prepare to accept the first request. Normally, the server does this immediately upon being released from the LISTEN, before the acknowledgement can get back to the client. The RECEIVE call blocks the server.

Then the client executes SEND to transmit its request (3) followed by the execution of RECEIVE to get the reply. The arrival of the request packet at the server machine unblocks the server so it can handle the request. After it has done the work, the server uses SEND to return the answer to the client (4). The arrival of this packet unblocks the client, which can now inspect the answer. If the client has additional requests, it can make them now. When the client is done, it executes DISCONNECT to terminate the connection (5). Usually, an initial DISCONNECT is a blocking call, suspending the client and sending a packet to the server saying that the connection is no longer needed. When the server gets the packet, it also issues a DISCONNECT of its own, acknowledging the client and releasing the connection (6). When the server's packet gets back to the client machine, the client process is released and the connection is broken. In a nutshell, this is how connection-oriented communication works. Of course, life is not so simple. Many things can go wrong here. The timing can be wrong (e.g., the CONNECT is done before the LISTEN), packets can get lost, and much more. We will look at these issues in great detail later, but for the moment, Fig. 2.5 briefly summarizes how client-server communication might work with acknowledged datagrams so that we can ignore lost packets.

Given that six packets are required to complete this protocol, one might wonder why a connectionless protocol is not used instead. The answer is that in a perfect world it could be, in which case only two packets would be needed: one for the request and one for the reply. However, in the face of large messages in either direction (e.g., a megabyte file), transmission errors, and lost packets, the situation changes. If the reply consisted of hundreds of packets, some of which could be lost during transmission, how would the client know if some pieces were missing? How would the client know whether the last packet actually received was really the last packet sent? Suppose the client wanted a second file. How could it tell packet 1 from the second file from a lost packet 1 from the first file that suddenly found its way to the client? In short, in the real world, a simple

request-reply protocol over an unreliable network is often inadequate.

The services between adjacent layers in the OSI architecture are expressed in terms of primitives and parameters. A primitive specifies the function to be performed, and the parameters are used to pass data and control information. The actual form of a primitive is implementation dependent. An example is a procedure call. Four types of primitives are used in standards to define the interaction between adjacent layers in the architecture. These are defined as:

- Request: A primitive issued by a service user to invoke some service and to pass the parameters needed to specify fully the requested service
- Indication: A primitive issued by a service provider either to indicate that a procedure has been invoked by the peer service user on the connection and to provide the associated parameters, or notify the service user of a provider-initiated action
- Response: A primitive issued by a service user to acknowledge or complete some procedure previously invoked by an indication to that user
- Confirm: A primitive issued by a service provider to acknowledge or complete some procedure previously invoked by a request by the service user

The layout of Fig. 2.7(a) suggests the time ordering of these events. This sequence of events is referred to as a confirmed service, as the initiator receives confirmation that the requested service has had the desired effect at the other end. If only request and indication primitives are involved (corresponding to steps 1 through 3), then the service dialogue is a nonconfirmed service; the initiator receives no confirmation that the requested action has taken place (Fig. 2.7(b)).

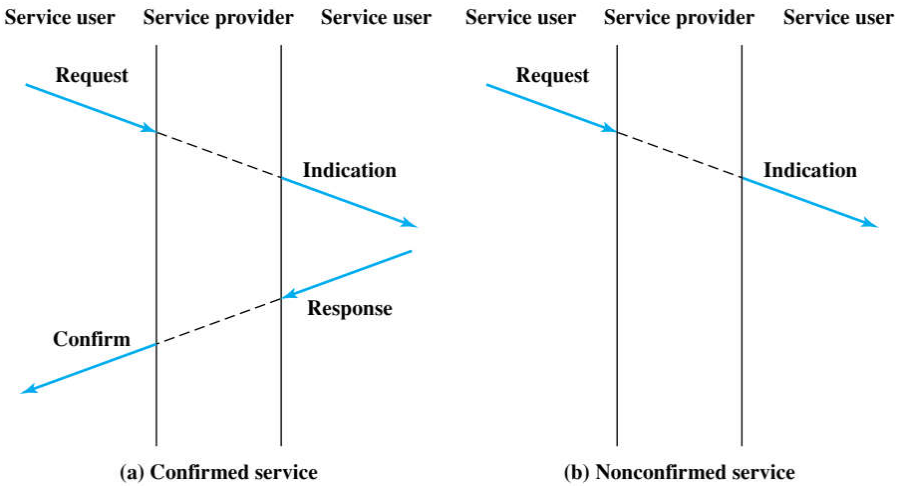

Figure 2.7: Time Sequence Diagrams for Service Primitives

For example, consider the transfer of data from an N entity to a peer N entity in another system. The following steps occur:

1. The source N entity invokes its $N-1$ entity with a request primitive. Associated with the primitive are the parameters needed, such as the data to be transmitted and the destination address.
2. The source $N-1$ entity prepares an $N-1$ PDU to be sent to its peer $N-1$ entity.
3. The destination $N-1$ entity delivers the data to the appropriate destination N entity via an indication primitive, which includes the data and source address as parameters.
4. If an acknowledgment is called for, the destination N entity issues a response primitive to its $N-1$ entity.
5. The $N-1$ entity conveys the acknowledgment in an $N-1$ PDU.
6. The acknowledgment is delivered to the N entity as a confirm primitive

2.3.4 The Relationship of Services to Protocols

Services and protocols are distinct concepts. This distinction is so important that we emphasize it again here. A service is a set of primitives (operations) that a layer provides to the layer above it. The service defines what operations the layer is prepared to perform on behalf of its users, but it says nothing at all about how these operations are implemented. A service relates to an interface between two layers, with the lower layer being the service provider and the upper layer being the service user.

A protocol, in contrast, is a set of rules governing the format and meaning of the packets, or messages that are exchanged by the peer entities within a layer. Entities use protocols to implement their service definitions. They are free to change their protocols at will, provided they do not change the service visible to their users. In this way, the service and the protocol are completely decoupled. This is a key concept that any network designer should understand well. To repeat this crucial point, services relate to the interfaces between layers, as illustrated in Fig. 2.8. In contrast, protocols relate to the packets sent between peer entities on different machines. It is very important not to confuse the two concepts.

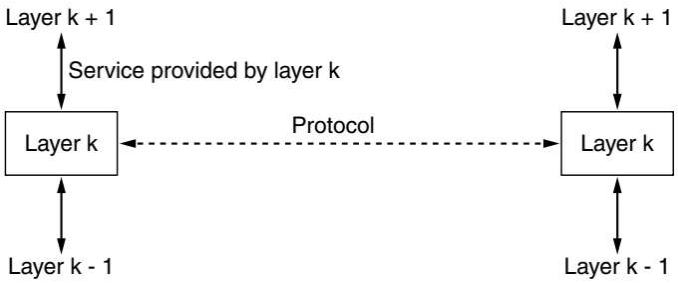

Figure 2.8: The relationship between a service and a protocol

An analogy with programming languages is worth making. A service is like an abstract data type or an object in an object-oriented language. It defines operations that can be performed on an object but does not specify how these operations are implemented. In contrast, a protocol relates to the implementation of the service and as such is not visible to the user of the service. Many older protocols did not distinguish the service from the protocol. In effect, a typical layer might have had a service primitive SEND PACKET with the user providing a pointer to a fully assembled packet. This arrangement meant that all changes to the protocol were immediately visible to the users. Most network designers now regard such a design as a serious blunder.

2.4 ISO-OSI Reference Model

The OSI model (minus the physical medium) is shown in Fig. 2.9. This model is based on a proposal developed by the International Standards Organization (ISO) as a first step toward international standardization of the protocols used in the various layers. It was revised in 1995. The model is called the ISO OSI (Open Systems Interconnection) Reference Model because it deals with connecting open systems-that is, systems that are open for communication with other systems. We will just call it the OSI model for short. The OSI model has seven layers. The principles that were applied to arrive at the seven layers can be briefly summarized as follows:

1. A layer should be created where a different abstraction is needed.
2. Each layer should perform a well-defined function.
3. The function of each layer should be chosen with an eye toward defining internationally standardized protocols.
4. The layer boundaries should be chosen to minimize the information flow across the interfaces.

5. The number of layers should be large enough that distinct functions need not be thrown together in the same layer out of necessity and small enough that the architecture does not become unwieldy

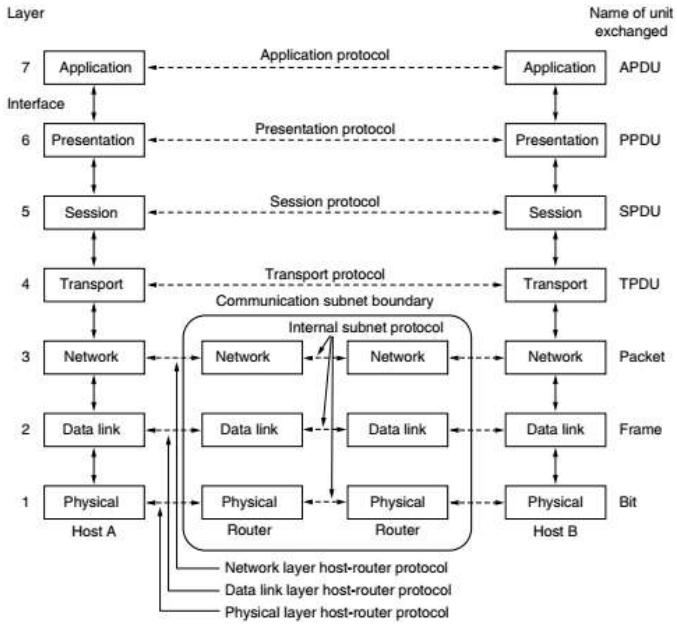

Figure 2.9: The OSI reference model

Below we will discuss each layer of the model in turn, starting at the bottom layer. Note that the OSI model itself is not a network architecture because it does not specify the exact services and protocols to be used in each layer. It just tells what each layer should do. Fig. 2.10 illustrates the functionality of each layer in jist.

2.4.1 Physical Layer

- The physical layer is concerned with ***transmitting raw bits over a communication channel***.
- The design issues have to do with making sure that when one side sends a 1 bit it is received by the other side as a 1 bit, not as a 0 bit. They largely deal with mechanical, electrical, and timing interfaces, as well as the physical transmission medium, which lies below the physical layer.
- Typical questions here are:
  - what electrical signals should be used to represent a 1 and a 0
  - how many nanoseconds a bit lasts
  - whether transmission may proceed simultaneously in both directions
  - how the initial connection is established
  - how it is torn down when both sides are finished
  - how many pins the network connector has
  - what each pin is used for
- Fig. 2.11 illustrates the function of the Physical Layer.

Thus, the physical layer is concerned with:

- **Physical characteristics of interfaces and medium**: The physical layer defines the characteristics of the interface between the devices and the transmission medium. It also defines the type of transmission medium.
- **Representation of bits**: The physical layer data consists of a stream of bits (sequence of 0s or 1s) with no interpretation. The physical layer defines the type of encoding (how 0s and 1s

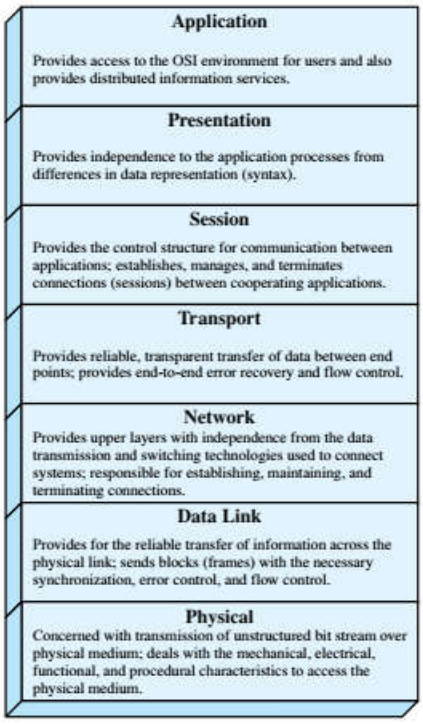

Figure 2.10: The OSI Layer functions

are changed to signals).

- **Data rate**: The physical layer defines the duration of a bit, which is how long it lasts.
- **Synchronization of bits.** The sender and receiver not only must use the same bit rate but also their clocks must be synchronized.
- **Line configuration.** The physical layer is concerned with the connection of devices to the media, i.e., point-to-point or multi-point configuration.
- **Physical topology.** The physical topology defines how devices are connected to make a network.
- **Transmission mode.** The physical layer also defines the direction of transmission between two devices: simplex, half-duplex, or full-duplex.

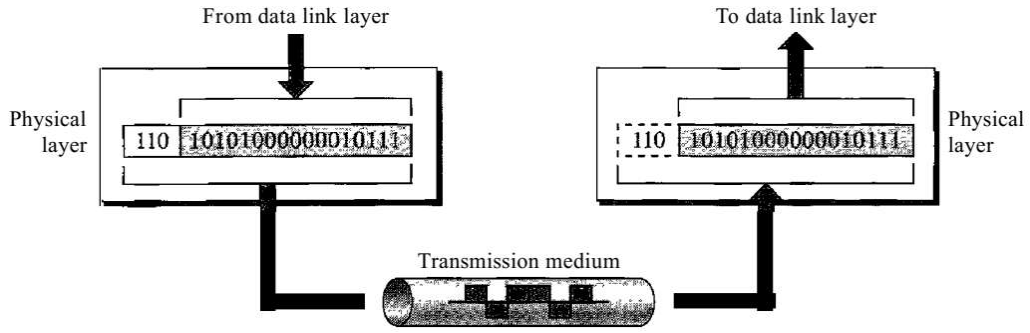

Figure 2.11: Physical Layer

2.4.2 Data Link Layer

The data link layer *controls the transmission errors*. It does so by masking the real errors so the network layer does not see them. It accomplishes this task by having the sender break up the input data into data frames (typically a few hundred or a few thousand bytes) and transmit the frames sequentially. If the service is reliable, the receiver confirms correct receipt of each frame by sending back an acknowledgment frame.

The data link layer is also responsible for *flow control*. It manages to keep a fast transmitter from drowning a slow receiver with data. It uses traffic regulation mechanism to let the transmitter know when the receiver can accept more data.

Broadcast networks have an additional issue in the data link layer: how to *control access* to the shared channel. A special sublayer of the data link layer, the Medium Access Control (MAC) sublayer, deals with this problem. The data link layer is responsible for hop-to-hop delivery as shown in Fig. 2.12.

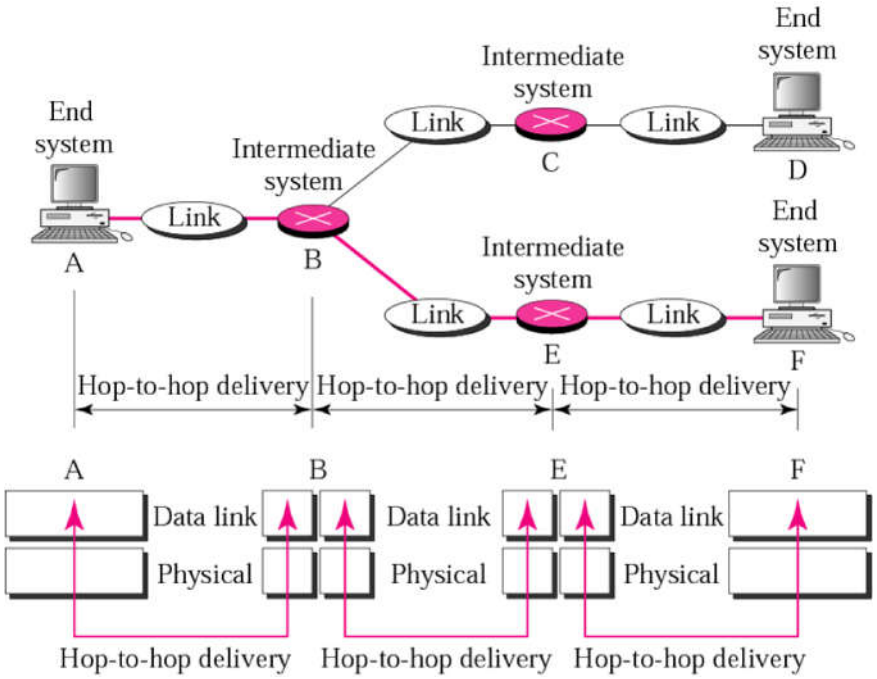

Figure 2.12: Hop-to-Hop Delivery

Thus, the data link layer is responsible for:

- Framing: The data link layer divides the stream of bits received from the network layer into manageable data units called frames.
- Physical addressing: If frames are to be distributed to different systems on the network, the data link layer adds a header to the frame to define the sender and/or receiver of the frame. If the frame is intended for a system outside the sender's network, the receiver address is the address of the device that connects the network to the next one.
- Flow control: If the rate at which the data are absorbed by the receiver is less than the rate at which data are produced in the sender, the data link layer imposes a flow control mechanism to avoid overwhelming the receiver.
- Error control: The data link layer adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged or lost frames. It also uses a mechanism to recognize duplicate frames. Error control is normally achieved through a trailer added to the end of the frame.

- Access control: When two or more devices are connected to the same link, data link layer protocols are necessary to determine which device has control over the link at any given time.

2.4.3 Network Layer

The network layer controls the operation of the sub-network or *subnet*. A key design issue is determining how packets are routed from source to destination. *Routes* can be based on static tables that are ***wired*** into the network and rarely changed, or more often they can be updated automatically to avoid failed components. They can also be determined at the start of each conversation, for example, a terminal session, such as a login to a remote machine. Finally, they can be highly dynamic, being determined anew for each packet to reflect the current network load.

If too many packets are present in the subnet at the same time, they will get in one another's way, forming bottlenecks. Handling congestion is also a responsibility of the network layer, in conjunction with higher layers that adapt the load they place on the network. More generally, the quality of service provided (delay, transit time, jitter, etc.) is also a network layer issue. When a packet has to travel from one network to another to get to its destination, many problems can arise. The addressing used by the second network may be different from that used by the first one. The second one may not accept the packet at all because it is too large. The protocols may differ, and so on. It is up to the network layer to overcome all these problems to allow heterogeneous networks to be interconnected. In broadcast networks, the routing problem is simple, so the network layer is often thin or even nonexistent.

The network layer is responsible for the source-to-destination delivery of a packet, possibly across multiple networks (links). Unlike the data link layer, which deals with the delivery of the packet between two systems on the same network (links), the network layer ensures that each packet gets from its point of origin to its final destination. If two systems are connected to the same link, there is usually no need for a network layer. However, if the two systems are attached to different networks (links) with connecting devices between the networks (links), there is often a need for the network layer to accomplish source-to-destination delivery as shown in Fig. **??**.

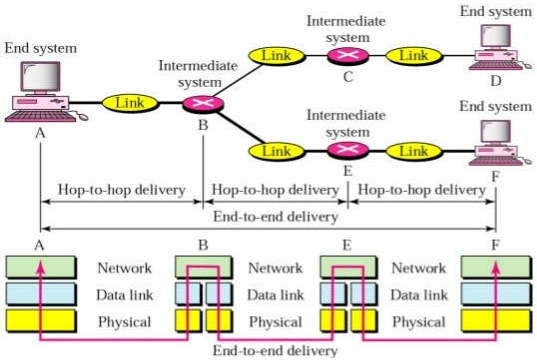

Figure 2.13: Source-to-Destination Delivery

Thus, the network layer is responsible for:

- Logical addressing: The physical addressing implemented by the data link layer handles the addressing problem locally. If a packet passes the network boundary, we need another addressing system to help distinguish the source and destination systems. The network layer adds a header to the packet coming from the upper layer that, among other things, includes the logical addresses of the sender and receiver.
- Routing: When independent networks or links are connected to create inter-networks(network of networks) or a large network, the connecting devices (called routers or switches) route or switch the packets to their final destination. One of the functions of the network layer is to

provide this mechanism.

2.4.4 Transport Layer

The basic function of the transport layer is to accept data from above it, split it up into smaller units if need be, pass these to the network layer, and ensure that the pieces all arrive correctly at the other end. Furthermore, all this must be done efficiently and in a way that isolates the upper layers from the inevitable changes in the hardware technology over the course of time.

The transport layer also determines what type of service to provide to the session layer, and, ultimately, to the users of the network. The most popular type of transport connection is an error-free point-to-point channel that delivers messages or bytes in the order in which they were sent. However, other possible kinds of transport service exist, such as the transporting of isolated messages with no guarantee about the order of delivery, and the broadcasting of messages to multiple destinations. The type of service is determined when the connection is established. (As an aside, an error-free channel is completely impossible to achieve; what people really mean by this term is that the error rate is low enough to ignore in practice.)

The transport layer is a true end-to-end layer; it carries data all the way from the source to the destination. In other words, a program on the source machine carries on a conversation with a similar program on the destination machine, using the message headers and control messages. In the lower layers, each protocols is between a machine and its immediate neighbors, and not between the ultimate source and destination machines, which may be separated by many routers. The difference between layers 1 through 3, which are chained, and layers 4 through 7,which are end-to-end, is illustrated in Fig. 2.9.

The transport layer is responsible for process-to-process delivery of the entire message as shown in Fig. 2.14. A process is an application program running on a host. Whereas the network layer oversees source-to-destination delivery of individual packets, it does not recognize any relationship between those packets. It treats each one independently, as though each piece belonged to a separate message, whether or not it does. The transport layer, on the other hand, ensures that the whole message arrives intact and in order, overseeing both error control and flow control at the source-to-destination level.

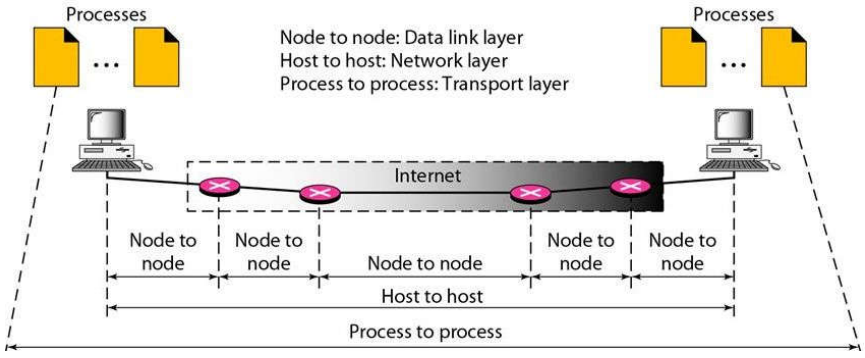

Figure 2.14: Process-to-Process Delivery

Thus, the duties of the Transport Layer are as follows:

- Service-point addressing: Computers often run several programs at the same time. For this reason, source-to-destination delivery means delivery not only from one computer to the next but also from a specific process (running program) on one computer to a specific process (running program) on the other. The transport layer header must therefore include a type of address called a service-point address (or port address). The network layer gets each packet to the correct computer; the transport layer gets the entire message to the correct process on

that computer.

- Segmentation and reassembly: A message is divided into transmittable segments,with each segment containing a sequence number. These numbers enable the transport layer to reassemble the message correctly upon arriving at the destination and to identify and replace packets that were lost in transmission.
- Connection control: The transport layer can be either connectionless or connection oriented. A connectionless transport layer treats each segment as an independent packet and delivers it to the transport layer at the destination machine. A connectionoriented transport layer makes a connection with the transport layer at the destination machine first before delivering the packets. After all the data are transferred, the connection is terminated.
- Flow control: Like the data link layer, the transport layer is responsible for flow control. However, flow control at this layer is performed end to end rather than across a single link.
- Error control: Like the data link layer, the transport layer is responsible for error control. However, error control at this layer is performed process-to process rather than across a single link. The sending transport layer makes sure that the entire message arrives at the receiving transport layer without error (damage, loss, or duplication). Error correction is usually achieved through retransmission.

2.4.5 Session Layer

The session layer is the *network dialog controller*. It establishes, maintains, and synchronizes the interaction among communicating systems. It allows users on different machines to establish sessions between them. Sessions offer various services, including dialog control (keeping track of whose turn it is to transmit), token management (preventing two parties from attempting the same critical operation simultaneously), and synchronization (check pointing long transmissions to allow them to pick up from where they left off in the event of a crash and subsequent recovery).

Specific responsibilities of the session layer include the following:

- Dialog control. The session layer allows two systems to enter into a dialog. It allows the communication between two processes to take place in either half duplex (one way at a time) or full-duplex (two ways at a time) mode.
- Synchronization. The session layer allows a process to add checkpoints, or synchronization points, to a stream of data. For example, if a system is sending a file of 2000 pages, it is advisable to insert checkpoints after every 100 pages to ensure that each 100-page unit is received and acknowledged independently. In this case, if a crash happens during the transmission of page 523, the only pages that need to be resent after system recovery are pages 501 to 523. Pages previous to 501 need not be present.

2.4.6 Presentation Layer

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

Specific responsibilities of the presentation layer include the following:

- Translation: The processes (running programs) in two systems are usually exchanging information in the form of character strings, numbers, and so on. The information must be changed to bit streams before being transmitted. Because different computers use different encoding systems, the presentation layer is responsible for interoperability between these different encoding methods. The presentation layer at the sender changes the information

from its sender-dependent format into a common format. The presentation layer at the receiving machine changes the common format into its receiver-dependent format.

- Encryption: To carry sensitive information, a system must be able to ensure privacy. Encryption means that the sender transforms the original information to another form and sends the resulting message out over the network. Decryption reverses the original process to transform the message back to its original form.
- Compression: Data compression reduces the number of bits contained in the information. Data compression becomes particularly important in the transmission of multimedia such as text, audio, and video.

2.4.7 Application Layer

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

Specific services provided by the application layer include the following:

- Network virtual terminal. A network virtual terminal is a software version of a physical terminal, and it allows a user to log on to a remote host. To do so, the application creates a software emulation of a terminal at the remote host. The user's computer talks to the software terminal which, in turn, talks to the host, and vice-versa. The remote host believes it is communicating with one of its own terminals and allows the user to log on. File transfer, access, and management. This application allows a user to access files in a remote host (to make changes or read data), to retrieve files from a remote computer for use in the local computer, and to manage or control files in a remote computer locally.
- Mail services. This application provides the basis for e-mail forwarding and storage.
- Directory services. This application provides distributed database sources and access for global information about various objects and services.

The summary of duties for each layer is thus given in Fig. 2.15.

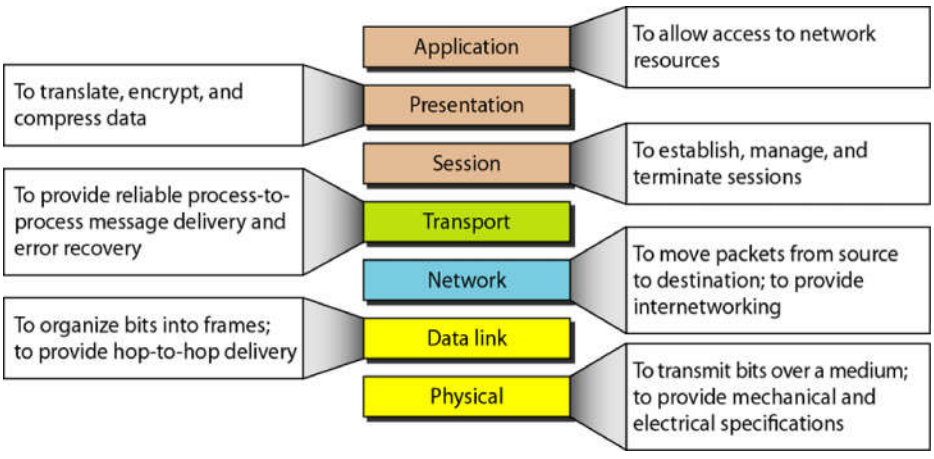

Figure 2.15: Summary of the responsibilities of the layers

2.5 TCP/IP Reference Model

The TCP/IP Reference Model has evolved from the original ARPANET. ARPANET started as a research network sponsored by the DoD (U.S. Department of Defense). It eventually connected

hundreds of universities and government installations, using leased telephone lines. When satellite and radio networks were added later, the existing protocols had trouble interworking with them, so a new reference architecture was needed. Thus, from nearly the beginning, the ability to connect multiple networks in a seamless way was one of the major design goals. This architecture later became known as the TCP/IP Reference Model. The structure of the TCP/IP reference model compared to the ISO-OSI Model is given in Fig. 2.16.

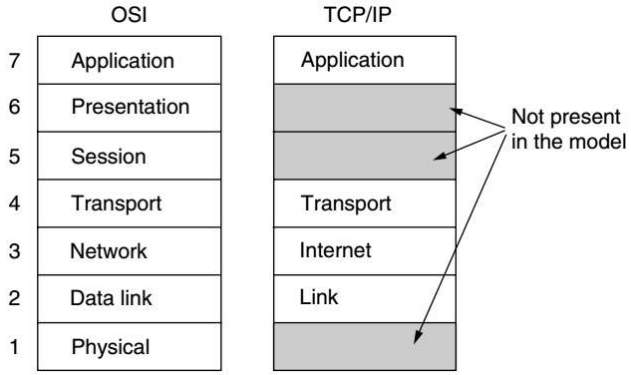

Figure 2.16: TCP/IP Reference Model

2.5.1 Link Layer

At the physical and data link layers, TCP/IP does not define any specific protocol. It supports all the standard and proprietary protocols. A network in a TCP/IP internetwork can be a LAN or a WAN. The incompatibility issues discussed earlier led to the choice of a packet-switching network based on a connectionless layer that runs across different networks. The lowest layer in the model, the link layer describes what links such as serial lines and classic Ethernet must do to meet the needs of this connectionless internet layer. It is not really a layer at all, in the normal sense of the term, but rather an interface between hosts and transmission links. Early material on the TCP/IP model has little to say about it.

2.5.2 Internet Layer

The internet layer is the linchpin that holds the whole architecture together. It is shown in Fig. 2.16 as corresponding roughly to the OSI network layer. Its job is to permit hosts to inject packets into any network and have them travel independently to the destination (potentially on a different network). They may even arrive in a completely different order than they were sent, in which case it is the job of higher layers to rearrange them, if in-order delivery is desired. Note that *internet* is used here in a generic sense, even though this layer is present in the Internet.

The analogy here is with the (snail) mail system. A person can drop a sequence of international letters into a mailbox in one country, and with a little luck most of them will be delivered to the correct address in the destination country. The letters will probably travel through one or more international mail gateways along the way, but this is transparent to the users. Furthermore, that each country (i.e., each network) has its own stamps, preferred envelope sizes, and delivery rules is hidden from the users. The internet layer defines an official packet format and protocol called IP(Internet Protocol), plus a companion protocol called ICMP (Internet Control Message Protocol) that helps it function. The job of the internet layer is to deliver IP packets where they are supposed to go. Packet routing is clearly a major issue here, as is congestion (though IP has not proven effective at avoiding congestion). There are 3 other protocols to help IP. They are ARP (Address Resolution Protocol), RARP (Reverse Address Resolution Protocol) and IGMP (Internet Group

Message Protocol). The ARP is used to associate a logical address with a physical address. On a typical physical network, such as a LAN, each device on a link is identified by a physical or station address, usually imprinted on the network interface card (NIC). ARP is used to find the physical address of the node when its Internet address is known. The RARP allows a host to discover its Internet address when it knows only its physical address. It is used when a computer is connected to a network for the first time or when a disk-less computer is booted. The IGMP is used to facilitate the simultaneous transmission of a message to a group of recipients.

2.5.3 Transport Layer

The layer above the internet layer in the TCP/IP model is now usually called the transport layer. It is designed to allow peer entities on the source and destination hosts to carry on a conversation, just as in the OSI transport layer. Two end-to-end transport protocols have been defined here, namely TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). There is another protocol Stream Control Transmission Protocol (SCTP) which provides support for newer applications such as voice over the Internet. It is a transport layer protocol that combines the best features of UDP and TCP. All the 3 protocols will be discussed in details later.

TCP (Transmission Control Protocol):

- Reliable
- connection-oriented protocol that
- allows a byte stream originating on one machine to be delivered without error on any other machine in the internet.
- It segments the incoming byte stream into discrete messages and passes each one on to the internet layer.
- At the destination, the receiving TCP process reassembles the received messages into the output stream.
- TCP also handles flow control to make sure a fast sender cannot swamp a slow receiver with more messages than it can handle.

UDP (User Datagram Protocol):

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

The relation of IP, TCP, and UDP is shown in Fig. 2.17.

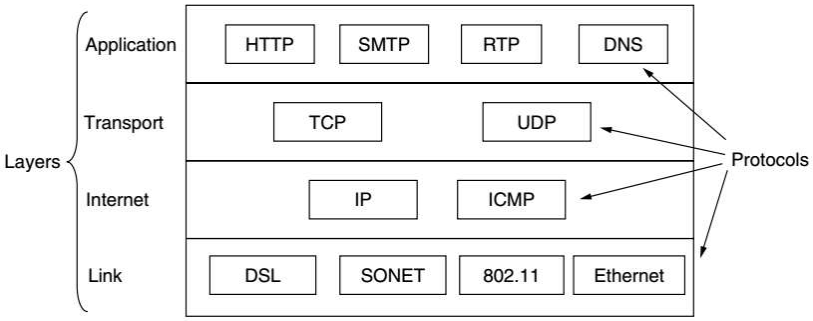

Figure 2.17: Protocols in TCP/IP

2.5.4 Application Layer

The TCP/IP model does not have session or presentation layers. No need for them was perceived. Instead, applications simply include any session and presentation functions that they require. Experience with the OSI model has proven this view correct: these layers are of little use to most applications. On top of the transport layer is the application layer. It contains all the higher-level protocols. The early ones included virtual terminal (TELNET), file transfer (FTP), and electronic mail (SMTP). Many other protocols have been added to these over the years. Some important ones that we will study, shown in Fig. 2.17, include the Domain Name System (DNS), for mapping host names onto their network addresses, HTTP, the protocol for fetching pages on the World Wide Web, and RTP, the protocol for delivering real-time media such as voice or movies.

2.6 Comparison of ISO-OSI and TCP/IP Reference Models

2.7 Addressing

Four levels of addresses are used in an internet employing the TCP/IP protocols:

- physical (link) addresses,
- logical (IP) addresses,
- port addresses, and
- specific addresses

Each address is related to a specific layer in the TCPIIP architecture, as shown in Fig. 2.18.

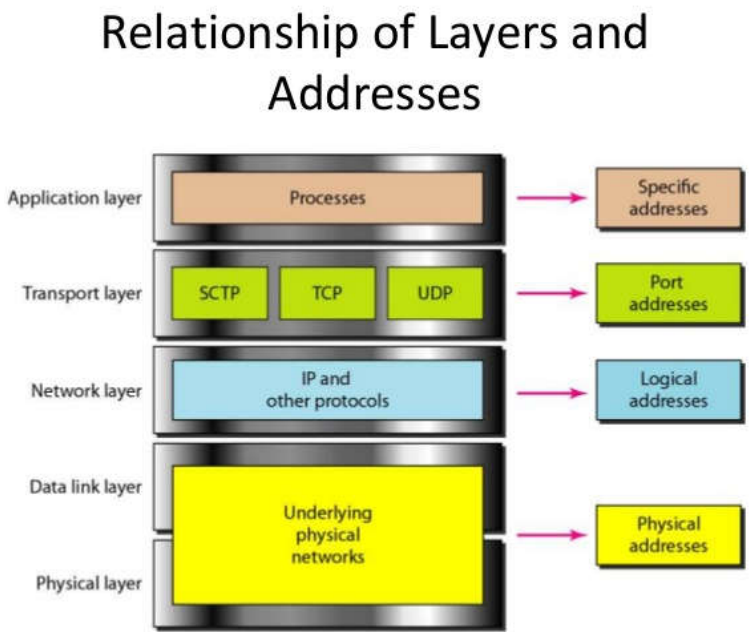

Figure 2.18: Relationship of layers and addresses in TCP/IP

2.7.1 Physical Addresses

The physical address, also known as the link address, is the address of a node as defined by its LAN or WAN. It is included in the frame used by the data link layer. It is the lowest-level address. The physical addresses have authority over the network (LAN or WAN). The size and format of these addresses vary depending on the network. For example, Ethernet uses a 6-byte (48-bit) physical address that is imprinted on the network interface card (NIC). LocalTalk (Apple), however, has a I-byte dynamic address that changes each time the station comes up.

Example: In Fig. 2.19 a node with physical address 10 sends a frame to a node with physical address 87. The two nodes are connected by a link (bus topology LAN). At the data link layer, this frame contains physical (link) addresses in the header. These are the only addresses needed. The rest of the header contains other information needed at this level. The trailer usually contains extra bits needed for error detection. As the figure shows, the computer with physical address 10 is the sender, and the computer with physical address 87 is the receiver. The data link layer at the sender receives data from an upper layer. It encapsulates the data in a frame, adding a header and a trailer. The header, among other pieces of information, carries the receiver and the sender physical (link) addresses. Note that in most data link protocols, the destination address, 87 in this case, comes before the source address (10 in this case).

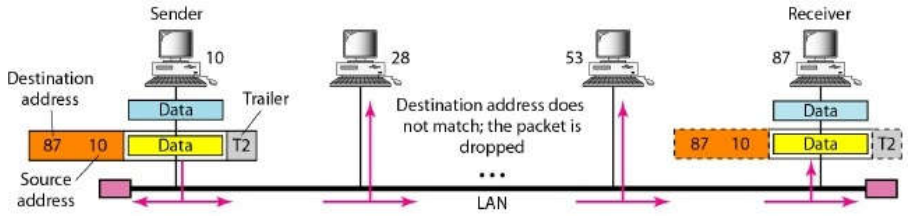

Figure 2.19: Physical Address (MAC Address)

In a bus topology, the frame is propagated in both directions (left and right). The frame propagated to the left dies when it reaches the end of the cable if the cable end is terminated appropriately. The frame propagated to the right is sent to every station on the network. Each station with a physical addresses other than 87 drops the frame because the destination address in the frame does not match its own physical address. The intended destination computer, however, finds a match between the destination address in the frame and its own physical address. The frame is checked, the header and trailer are dropped, and the data part is decapsulated and delivered to the upper layer. Most LANs use a 48-bit (6-byte) physical address written as 12 hexadecimal digits; every byte (2 hexadecimal digits) is separated by a colon, as 07:01:02:01:2C:4B which is a 6-byte (12 hexadecimal digits) physical address.

2.7.2 Logical Addresses

Logical addresses are necessary for universal communications that are independent of underlying physical networks. Physical addresses are not adequate in an internetwork environment where different networks can have different address formats. A universal addressing system is needed in which each host can be identified uniquely, regardless of the underlying physical network. The logical addresses are designed for this purpose. A logical address in the Internet is currently a 32-bit address that can uniquely define a host connected to the Internet. No two publicly addressed and visible hosts on the Internet can have the same IP address.

Example: Fig. 2.20 shows a part of an internet with two routers connecting three LANs. Each device (computer or router) has a pair of addresses (logical and physical) for each connection. In this case, each computer is connected to only one link and therefore has only one pair of addresses. Each router, however, is connected to three networks (only two are shown in the figure). So each router has three pairs of addresses, one for each connection. Although it may obvious that each router must have a separate physical address for each connection, it may not be obvious why it needs a logical address for each connection.

The computer with logical address A and physical address 10 needs to send a packet to the computer with logical address P and physical address 95. We use letters to show the logical addresses and numbers for physical addresses, but note that both are actually numbers, as we will see later in the chapter. The sender encapsulates its data in a packet at the network layer and adds

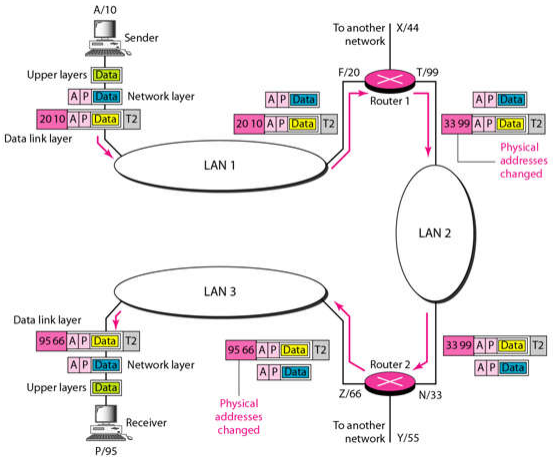

Figure 2.20: Logical Address (IP Address)

two logical addresses (A and P). Note that in most protocols, the logical source address comes before the logical destination address (contrary to the order of physical addresses). The network layer, however, needs to find the physical address of the next hop before the packet can be delivered. The network layer consults its routing table (see Chapter 22) and finds the logical address of the next hop (router I) to be F. The ARP discussed previously finds the physical address of router 1 that corresponds to the logical address of 20. Now the network layer passes this address to the data link layer, which in tum, encapsulates the packet with physical destination address 20 and physical source address 10. The frame is received by every device on LAN 1, but is discarded by all except router 1, which finds that the destination physical address in the frame matches with its own physical address. The router decapsulates the packet from the frame to read the logical destination address P. Since the logical destination address does not match the router's logical address, the router knows that the packet needs to be forwarded. The router consults its routing table and ARP to find the physical destination address of the next hop (router 2), creates a new frame, encapsulates the packet, and sends it to router 2. Note the physical addresses in the frame. The source physical address changes from 10 to 99. The destination physical address changes from 20 (router 1 physical address) to 33 (router 2 physical address). The logical source and destination addresses must remain the same; otherwise the packet will be lost.

At router 2 we have a similar scenario. The physical addresses are changed, and a new frame is sent to the destination computer. When the frame reaches the destination, the packet is decapsulated. The destination logical address P matches the logical address of the computer. The data are decapsulated from the packet and delivered to the upper layer. Note that although physical addresses will change from hop to hop, logical addresses remain the same from the source to destination. However, there are some exceptions to this rule that we discover later.

2.7.3 Port Address

The IP address and the physical address are necessary for a quantity of data to travel from a source to the destination host. However, arrival at the destination host is not the final objective of data communications on the Internet. A system that sends nothing but data from one computer to another is not complete. Today, computers are devices that can run multiple processes at the same time. The end objective of Internet communication is a process communicating with another process. For example, computer A can communicate with computer C by using TELNET. At the same time, computer A communicates with computer B by using the File Transfer Protocol (FTP). For these processes to receive data simultaneously, we need a method to label the different processes. In

other words, they need addresses. In the TCP/IP architecture, the label assigned to a process is called a port address. A port address in TCP/IP is 16 bits in length.

Example: Fig. 2.21 shows two computers communicating via the Internet. The sending computer is running three processes at this time with port addresses a, b, and c. The receiving computer is running two processes at this time with port addresses j and k. Process a in the sending computer needs to communicate with process j in the receiving computer. Note that although both computers are using the same application, FTP, for example, the port addresses are different because one is a client program and the other is a server program, as we will see later. To show that data from process a need to be delivered to process j, and not k, the transport layer encapsulates data from the application layer in a packet and adds two port addresses (a and j), source and destination. The packet from the transport layer is then encapsulated in another packet at the network layer with logical source and destination addresses (A and P). Finally, this packet is encapsulated in a frame with the physical source and destination addresses of the next hop. We have not shown the physical addresses because they change from hop to hop inside the cloud designated as the Internet. Note that although physical addresses change from hop to hop, logical and port addresses remain the same from the source to destination. There are some exceptions to this rule that we discuss later in the book.

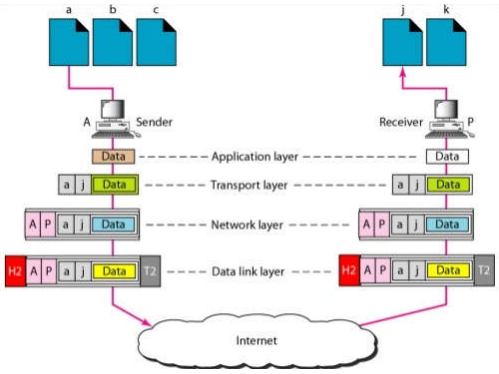

Figure 2.21: Port Addresses

2.8 Assignments

3. Physical Layer

The successful transmission of data depends principally on two factors: the quality of the signal being transmitted and the characteristics of the transmission medium.

3.1 Frequency, Spectrum, and Bandwidth

A signal is generated by the transmitter and transmitted over a medium to reach the receiver. A signal is a function of time, but it can also be expressed as a function of frequency; i.e., the signal consists of components of different frequencies. It turns out that the frequency domain view of a signal is more important to an understanding of data transmission than a time domain view.

Time Domain Concepts

Viewed as a function of time, an electromagnetic signal can be either analog or digital as shown in Fig. 3.1. An **analog signal** is one in which the signal intensity varies in a smooth fashion over time. In other words, there are no breaks or discontinuities in the signal. A **digital signal** is one in which the signal intensity maintains a constant level for some period of time and then abruptly changes to another constant level. The continuous signal might represent speech, and the discrete signal might represent binary 1s and 0s. The simplest sort of signal is a **periodic** signal, in which the same signal pattern repeats over time. Fig. 3.2 shows an example of a periodic continuous signal (sine wave) and a periodic discrete signal (square wave). Mathematically, a signal $s(t)$ is defined to be periodic if and only if $s(t+T) = s(t)$ for $-\inf < t < +\inf$ where the constant T is the period of the signal (T is the smallest value that satisfies the equation). Otherwise, a signal is **aperiodic**.

The sine wave is the fundamental periodic signal. A general sine wave can be represented by three parameters: peak amplitude (A), frequency (f), and phase (ϕ). The peak amplitude is the maximum value or strength of the signal over time; typically, this value is measured in volts. The frequency is the rate [in cycles per second, or Hertz (Hz)] at which the signal repeats. An equivalent parameter is the period (T) of a signal, which is the amount of time it takes for one repetition; therefore, Phase is a measure of the relative position in time within a single period of a signal, as is illustrated subsequently. More formally, for a periodic signal $f(t)$, phase is the fractional part t/T of the period T through which t has advanced relative to an arbitrary origin. The origin is

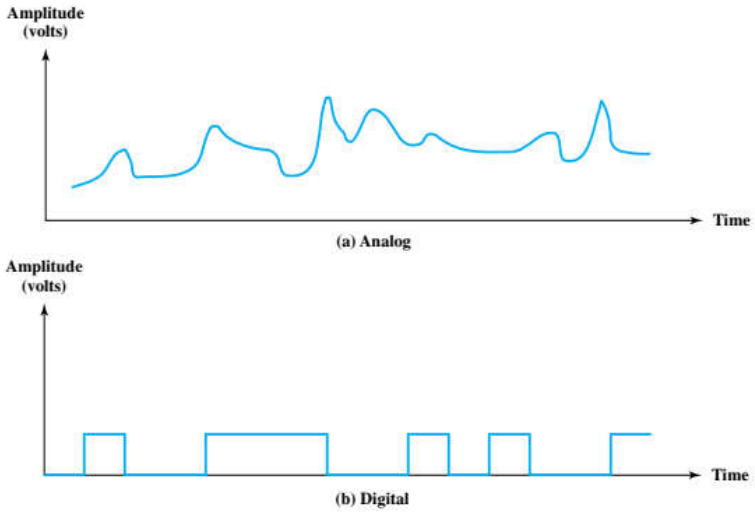

Figure 3.1: Analog and Digital Signal

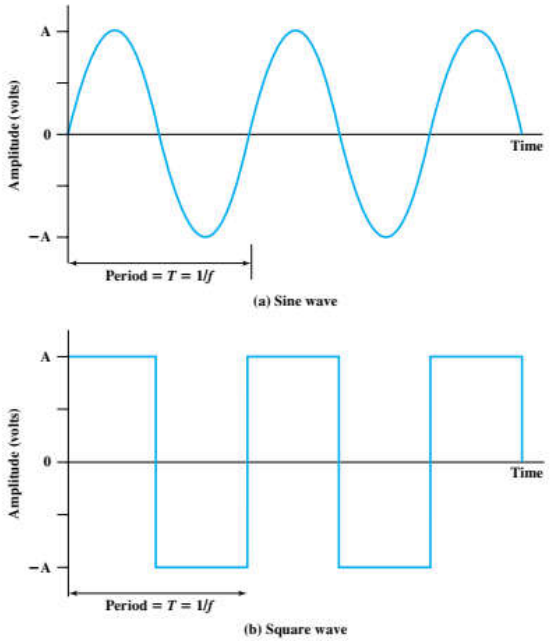

Figure 3.2: Examples of Periodic Signals

usually taken as the last previous passage through zero from the negative to the positive direction. The general sine wave can be written as $s(t) = Asin(2\pi ft) + \phi$. A function with the form of the preceding equation is known as a sinusoid.

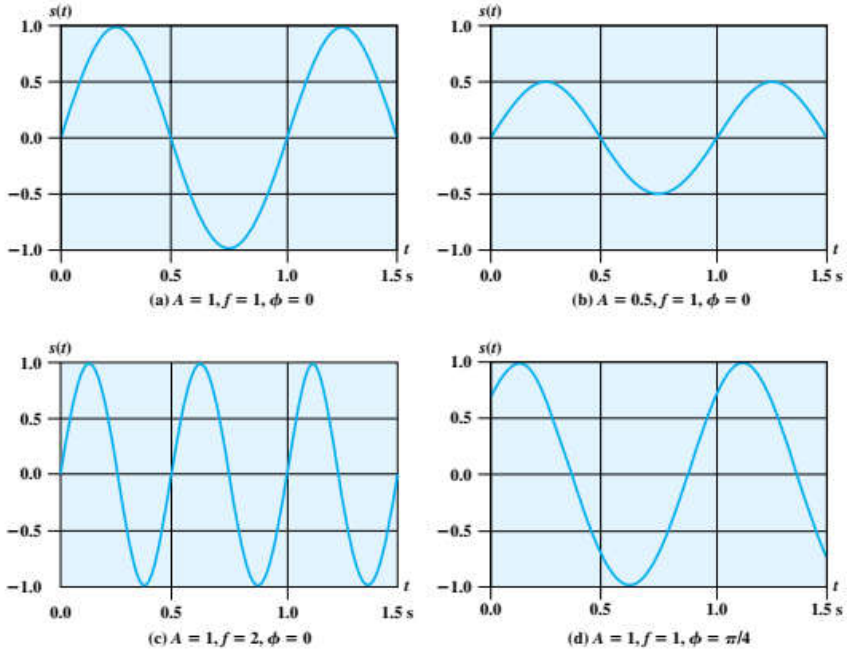

Figure 3.3: $s(t) = Asin(2\pi ft + \phi)$

Fig. 3.3 shows the effect of varying each of the three parameters. In part (a) of the figure, the frequency is 1 Hz; thus the period is second. Part (b) has the same frequency and phase but a peak amplitude of 0.5. In part (c) we have $f = 2$ which is equivalent to $T = 0.5$ Finally, part (d) shows the effect of a phase shift of $\pi/4$ radians, which is 45 degrees.

In Fig. 3.3, the horizontal axis is time; the graphs display the value of a signal at a given point in space as a function of time. These same graphs, with a change of scale, can apply with horizontal axes in space. In this case, the graphs display the value of a signal at a given point in time as a function of distance. For example, for a sinusoidal transmission (e.g., an electromagnetic radio wave some distance from a radio antenna, or sound some distance from a loudspeaker), at a particular instant of time, the intensity of the signal varies in a sinusoidal way as a function of distance from the source. There is a simple relationship between the two sine waves, one in time and one in space.The wavelength of a signal (λ) is the distance occupied by a single cycle, or, put another way, the distance between two points of corresponding phase of two consecutive cycles. Assume that the signal is traveling with a velocity v. Then the wavelength is related to the period as $\lambda = vT$ Equivalently, $\lambda f = v$. Of particular relevance to this discussion is the case where $v = c$, the speed of light in free space, which is approximately $3 \times 10^8 m/s$.

Frequency Domain Concepts

In practice, an electromagnetic signal will be made up of many frequencies. For example, the signal $s(t) = [4/\pi \times (sin(2\pi ft) + (1/3)sin(2\pi(3f)t))]$ is shown in Fig. 3.4(c).The components of this signal are just sine waves of frequencies f and 3f; parts (a) and (b) of the figure show these individual components. There are two interesting points that can be made about this figure:

- The second frequency is an integer multiple of the first frequency. When all of the frequency components of a signal are integer multiples of one frequency, the latter frequency is referred to as the fundamental frequency.

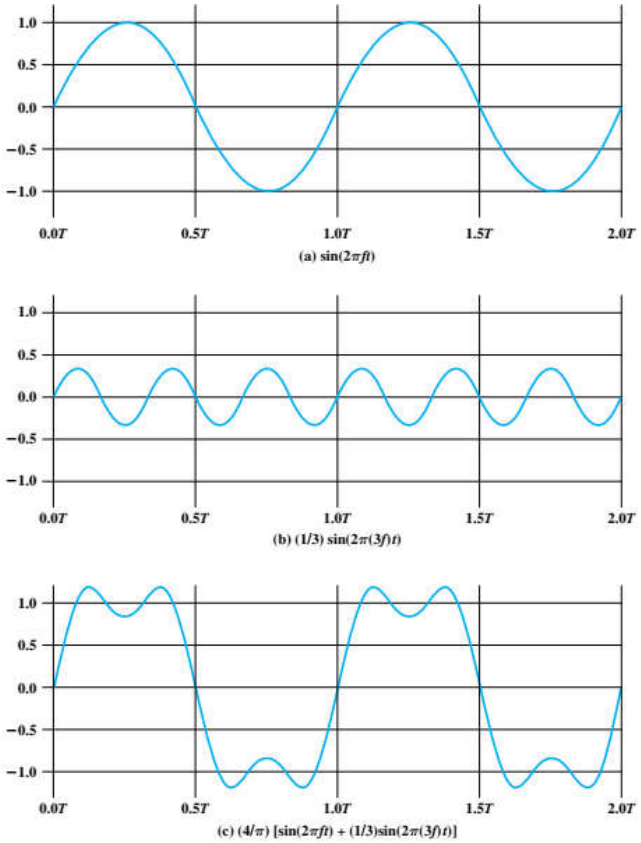

Figure 3.4: Addition of Frequency Components

- The period of the total signal is equal to the period of the fundamental frequency. The period of the component sin is and the period of $s(t)$ is also T, as can be seen from Fig. 3.4(c).

It can be shown, using a discipline known as Fourier analysis, that any signal is made up of components at various frequencies, in which each component is a sinusoid. By adding together enough sinusoidal signals, each with the appropriate amplitude, frequency, and phase, any electromagnetic signal can be constructed. Put another way, any electromagnetic signal can be shown to consist of a collection of periodic analog signals (sine waves) at different amplitudes, frequencies, and phases. The importance of being able to look at a signal from the frequency perspective (frequency domain) rather than a time perspective (time domain) should become clear as the discussion proceeds.

So we can say that for each signal, there is a time domain function $s(t)$ that specifies the amplitude of the signal at each instant in time. Similarly, there is a frequency domain function $S(f)$ that specifies the peak amplitude of the constituent frequencies of the signal. Fig. 3.5(a) shows the frequency domain function for the signal of Fig. 3.4(c).

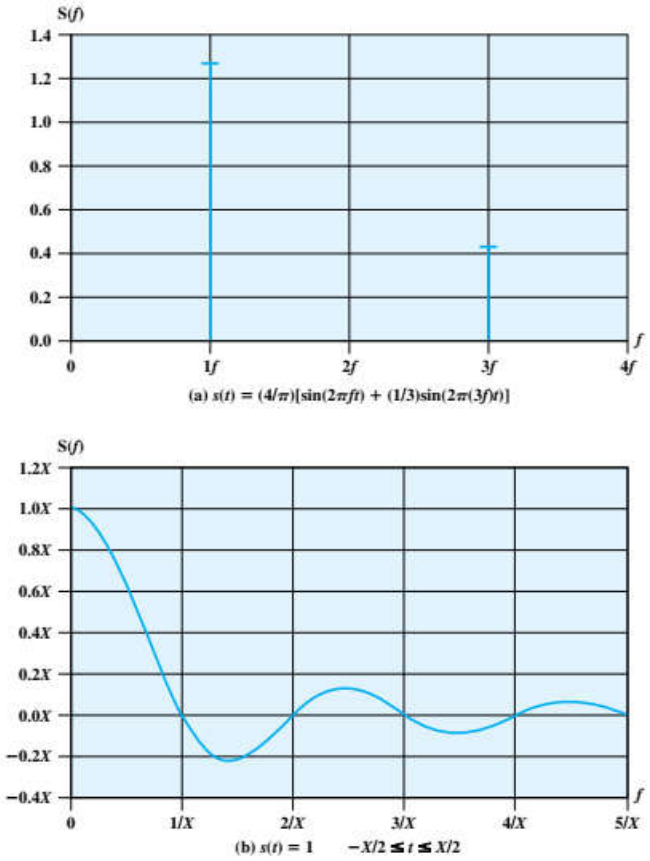

Figure 3.5: Frequency Domain Representations

Note that, in this case, S(f) is discrete. Fig. 3.5(b) shows the frequency domain function for a single square pulse that has the value 1 between $-X/2$ and $X/2$,and is 0 elsewhere. Note that in this case S(f) is continuous and that it has nonzero values indefinitely, although the magnitude of the frequency components rapidly shrinks for larger f. These characteristics are common for real signals.

The spectrum of a signal is the range of frequencies that it contains. For the signal of Fig. 3.4(c), the spectrum extends from f to 3f. The absolute bandwidth of a signal is the width of the spectrum. In the case of Fig. 3.4(c), the bandwidth is 2f. Many signals, such as that of Fig. 3.5(b), have an infinite bandwidth. However, most of the energy in the signal is contained in a relatively

narrow band of frequencies. This band is referred to as the effective bandwidth, or just bandwidth.

One final term to define is dc component. If a signal includes a component of zero frequency, that component is a direct current (dc) or constant component. For example, Fig. 3.6 shows the result of adding a dc component to the signal of Fig. 3.4(c). With no dc component, a signal has an average amplitude of zero, as seen in the time domain. With a dc component, it has a frequency term at $f = 0$ and a nonzero average amplitude.

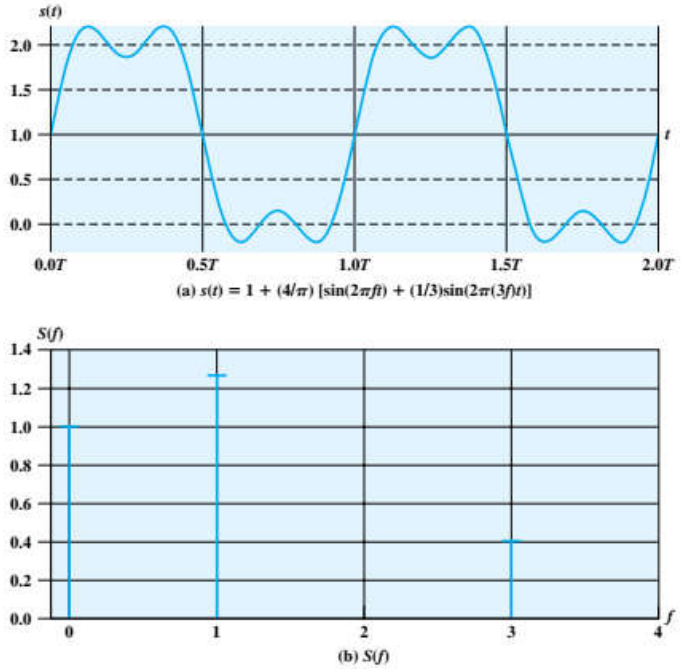

Figure 3.6: Signal with dc component

Relationship between Data Rate and Bandwidth

We have said that effective bandwidth is the band within which most of the signal energy is concentrated.The meaning of the term most in this context is somewhat arbitrary. The important issue here is that, although a given waveform may contain frequencies over a very broad range, as a practical matter any transmission system (transmitter plus medium plus receiver) will be able to accommodate only a limited band of frequencies.This, in turn, limits the data rate that can be carried on the transmission medium. To try to explain these relationships, consider the square wave of 3.2(b). Suppose that we let a positive pulse represent binary 0 and a negative pulse represent binary 1. Then the waveform represents the binary stream 0101. . . . The duration of each pulse is $1/(2f)$; thus the data rate is $2f$ bits per second (bps).What are the frequency components of this signal? To answer this question, consider again Fig. 3.4. By adding together sine waves at frequencies f and $3f$, we get a waveform that begins to resemble the original square wave. Let us continue this process by adding a sine wave of frequency $5f$, as shown in Fig. 3.7(a), and then adding a sine wave of frequency $7f$, as shown in Fig. 3.7(b).

As we add additional odd multiples of f, suitably scaled, the resulting waveform approaches that of a square wave more and more closely. Indeed, it can be shown that the frequency components of the square wave with amplitudes A and $-A$ can be expressed as $s(t) = A \times \frac{4}{\pi} \times \sum_{k_{odd},k=1}^{\infty} \frac{sin(2\pi kft)}{k}$.

Thus, this waveform has an infinite number of frequency components and hence an infinite bandwidth. However, the peak amplitude of the k^{th} frequency component, kf, is only $1/k$, so most of the energy in this waveform is in the first few frequency components. What happens if we limit

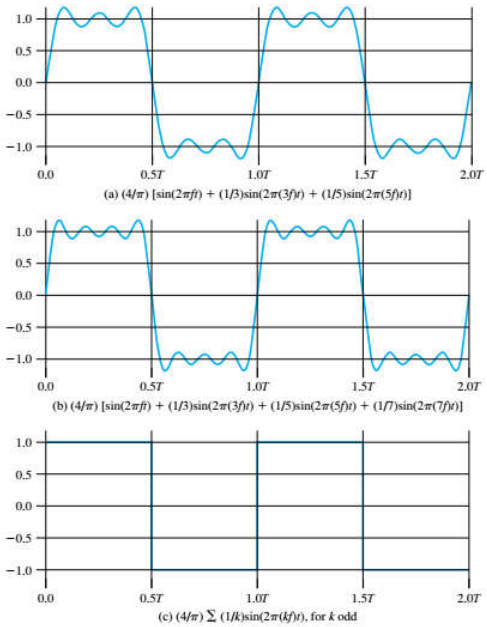

Figure 3.7: Frequency Components of Square Wave

the bandwidth to just the first three frequency components? We have already seen the answer, in Fig. 3.7(a). As we can see, the shape of the resulting waveform is reasonably close to that of the original square wave.

We can use Figs. 3.4 and 3.7 to illustrate the relationship between data rate and bandwidth. Suppose that we are using a digital transmission system that is capable of transmitting signals with a bandwidth of 4 MHz. Let us attempt to transmit a sequence of alternating 1s and 0s as the square wave of Fig. 3.7(c). What data rate can be achieved? We look at three cases.

1. Let us approximate our square wave with the waveform of Fig. 3.7(a). Although this waveform is a "distorted" square wave, it is sufficiently close to the square wave that a receiver should be able to discriminate between a binary 0 and a binary 1. If we let $f = 10^6 cycles/sec = 1Mhz$, then the bandwidth of the signal $s(t) = \frac{4}{\pi} \times [sin((2\pi \times 10^6)t) + \frac{1}{3}sin((2\pi \times 3 \times 10^6)t) + \frac{1}{5}sin((2\pi \times 5 \times 10^6)t)]$ is $5 \times 10^6 - 10^6 = 4Mhz$. Note that for $f = 1Mhz$, the period of the fundamental frequency is $T = 10^{-6} = 1\mu s$. If we treat this waveform as a bit string of 1s and 0s, one bit occurs every $0.5\mu s$ for a data rate of $2 \times 10^6 = 2Mbps$. Thus, for a bandwidth of $4MHz$, a data rate of $2Mbps$ is achieved.
2. Now suppose that we have a bandwidth of 8 MHz. Let us look again at Fig. 3.7(a), but now with $f = 2Mhz$ Using the same line of reasoning as before, the bandwidth of the signal is $(5 \times 2 \times 10^6) - (2 \times 10^6) = 8Mhz$. But in this case $T = 1/f = 0.5\mu s$. As a result, one bit occurs every $0.25\mu s$ for a data rate of $4Mbps$. Thus, other things being equal, by doubling the bandwidth, we double the potential data rate.
3. Now suppose that the waveform of Fig. 3.4(c) is considered adequate for approximating a square wave. That is, the difference between a positive and negative pulse in Fig. 3.4(c) is sufficiently distinct that the waveform can be successfully used to represent a sequence of 1s and 0s. Assuming $f = 2Mhz$ and $T = 1/f = 0.5\mu s$ like the previous case, so that one bit occurs every $0.25\mu s$ for a data rate of $4Mbps$. Using the waveform of Fig. 3.4(c), the bandwidth of the signal is $(3 \times 2 \times 10^6) - (2 \times 10^6) = 4Mhz$. Thus, a given bandwidth can support various data rates depending on the ability of the receiver to discern the difference between 0 and 1 in the presence of noise and other impairments.

We can draw the following conclusions from the preceding discussion. In general, any digital

waveform will have infinite bandwidth. If we attempt to transmit this waveform as a signal over any medium, the transmission system will limit the bandwidth that can be transmitted. Furthermore, for any given medium, the greater the bandwidth transmitted, the greater the cost. Thus, on the one hand, economic and practical reasons dictate that digital information be approximated by a signal of limited bandwidth. On the other hand, limiting the bandwidth creates distortions, which makes the task of interpreting the received signal more difficult. *The more limited the bandwidth, the greater the distortion, and the greater the potential for error by the receiver.*

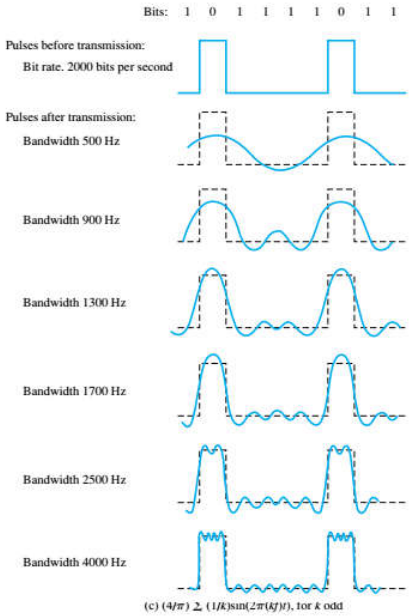

Figure 3.8: Effect of bandwidth on digital signal

One more illustration should serve to reinforce these concepts. Fig. 3.8 shows a digital bit stream with a data rate of 2000 bits per second. With a bandwidth of 2500 Hz, or even 1700 Hz, the representation is quite good. Furthermore, we can generalize these results. If the data rate of the digital signal is W bps, then a very good representation can be achieved with a bandwidth of $2W$ Hz. However, unless noise is very severe, the bit pattern can be recovered with less bandwidth than this. Thus, there is a direct relationship between data rate and bandwidth: *The higher the data rate of a signal, the greater is its required effective bandwidth.* Looked at the other way, *the greater the bandwidth of a transmission system, the higher is the data rate that can be transmitted over that system.* Another observation worth making is this: If we think of the bandwidth of a signal as being centered about some frequency, referred to as the center frequency, then the *higher the center frequency, the higher the potential bandwidth and therefore the higher the potential data rate.* For example, if a signal is centered at 2 MHz, its maximum potential bandwidth is 4 MHz.

3.2 Analog and Digital Data Transmission

The terms analog and digital correspond, roughly, to continuous and discrete, respectively.These two terms are used frequently in data communications in at least three contexts: *data, signaling, and transmission.* Briefly, we define:

- Data: Entities that convey meaning, or information.
- Signals: Electric or electromagnetic representations of data. Signaling is the physical propagation of the signal along a suitable medium.
- Transmission: Communication of data by the propagation and processing of signals.

In what follows, we try to make these abstract concepts clear by discussing the terms analog and digital as applied to data, signals, and transmission.

3.2.1 Analog and Digital Data

The concepts of analog and digital *data* are simple enough. *Analog data* take on *continuous values* in some interval. For example, voice and video are continuously varying patterns of intensity. Most data collected by sensors, such as temperature and pressure, are continuous valued. Digital data take on discrete values; examples are text and integers. The most familiar example of analog data is audio, which, in the form of acoustic sound waves, can be perceived directly by human beings. Frequency components of typical speech may be found between approximately 100 Hz and 7 kHz. Although much of the energy in speech is concentrated at the lower frequencies, tests have shown that frequencies below 600 or 700 Hz add very little to the intelligibility of speech to the human ear. Typical speech has a dynamic range of about 25 dB; i.e., the power produced by the loudest shout may be as much as 300 times greater than the least whisper.

Another common example of analog data is video. Here it is easier to characterize the data in terms of the TV screen (destination) rather than the original scene (source) recorded by the TV camera. To produce a picture on the screen, an electron beam scans across the surface of the screen from left to right and top to bottom. For black-and-white television, the amount of illumination produced (on a scale from black to white) at any point is proportional to the intensity of the beam as it passes that point.Thus at any instant in time the beam takes on an analog value of intensity to produce the desired brightness at that point on the screen. Further, as the beam scans, the analog value changes.Thus the video image can be thought of as a time-varying analog signal.

A familiar example of digital data is text or character strings. While textual data are most convenient for human beings, they cannot, in character form, be easily stored or transmitted by data processing and communications systems. Such systems are designed for binary data. Thus a number of codes have been devised by which characters are represented by a sequence of bits. Perhaps the earliest common example of this is the Morse code. Today, the most commonly used text code is the International Reference Alphabet (IRA). Each character in this code is represented by a unique 7-bit pattern; thus 128 different characters can be represented. This is a larger number than is necessary, and some of the patterns represent invisible control characters. IRA-encoded characters are almost always stored and transmitted using 8 bits per character. The eighth bit is a parity bit used for error detection. This bit is set such that the total number of binary 1s in each octet is always odd (odd parity) or always even (even parity). Thus a transmission error that changes a single bit, or any odd number of bits, can be detected.

3.2.2 Analog and Digital Signals

In a communications system, data are propagated from one point to another by means of electromagnetic signals. An analog signal is a continuously varying electromagnetic wave that may be propagated over a variety of media, depending on spectrum; examples are wire media, such as twisted pair and coaxial cable; fiber optic cable; and unguided media, such as atmosphere or space propagation. A digital signal is a sequence of voltage pulses that may be transmitted over a wire medium; for example, a constant positive voltage level may represent binary 0 and a constant negative voltage level may represent binary 1.

The principal advantages of digital signaling are that it is generally cheaper than analog signaling and is less susceptible to noise interference. The principal disadvantage is that digital signals suffer more from attenuation than do analog signals. Fig. 3.9 shows a sequence of voltage pulses, generated by a source using two voltage levels, and the received voltage some distance down a conducting medium. Because of the attenuation, or reduction, of signal strength at higher frequencies, the pulses become rounded and smaller. It should be clear that this attenuation can

lead rather quickly to the loss of the information contained in the propagated signal.

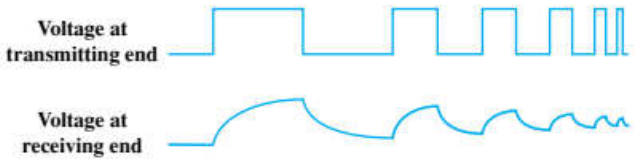

Figure 3.9: Attenuation of Digital Signals

In the foregoing discussion, we have looked at analog signals used to represent analog data and digital signals used to represent digital data. Generally, analog data are a function of time and occupy a limited frequency spectrum; such data can be represented by an electromagnetic signal occupying the same spectrum. Digital data can be represented by digital signals, with a different voltage level for each of the two binary digits.

As Fig.3.10 illustrates, these are not the only possibilities. Digital data can also be represented by analog signals by use of a modem (modulator/demodulator). The modem converts a series of binary (two-valued) voltage pulses into an analog signal by encoding the digital data onto a carrier frequency. The resulting signal occupies a certain spectrum of frequency centered about the carrier and may be propagated across a medium suitable for that carrier. The most common modems represent digital data in the voice spectrum and hence allow those data to be propagated over ordinary voice-grade telephone lines. At the other end of the line, another modem demodulates the signal to recover the original data. In an operation very similar to that performed by a modem, analog data can be represented by digital signals. The device that performs this function for voice data is a codec (coder-decoder). In essence, the codec takes an analog signal that directly represents the voice data and approximates that signal by a bit stream. At the receiving end, the bit stream is used to reconstruct the analog data.

3.2.3 Analog and Digital Transmission

Both analog and digital signals may be transmitted on suitable transmission media. The way these signals are treated is a function of the transmission system. Fig. 3.11 summarizes the methods of data transmission. Analog transmission is a means of transmitting analog signals without regard to their content; the signals may represent analog data (e.g., voice) or digital data (e.g., binary data that pass through a modem). In either case, the analog signal will become weaker (attenuate) after a certain distance. To achieve longer distances, the analog transmission system includes amplifiers that boost the energy in the signal. Unfortunately, the amplifier also boosts the noise components. With amplifiers cascaded to achieve long distances, the signal becomes more and more distorted.

For analog data, such as voice, quite a bit of distortion can be tolerated and the data remain intelligible. However, for digital data, cascaded amplifiers will introduce errors. Digital transmission, in contrast, assumes a binary content to the signal. A digital signal can be transmitted only a limited distance before attenuation, noise, and other impairments endanger the integrity of the data. To achieve greater distances, repeaters are used. A repeater receives the digital signal, recovers the pattern of 1s and 0s, and retransmits a new signal. Thus the attenuation is overcome. The same technique may be used with an analog signal if it is assumed that the signal carries digital data. At appropriately spaced points, the transmission system has repeaters rather than amplifiers. The repeater recovers the digital data from the analog signal and generates a new, clean analog signal. Thus noise is not cumulative. The question naturally arises as to which is the preferred method of transmission. The answer being supplied by the telecommunications industry and its customers is digital. Both long-haul telecommunications facilities and intra-building services have moved to

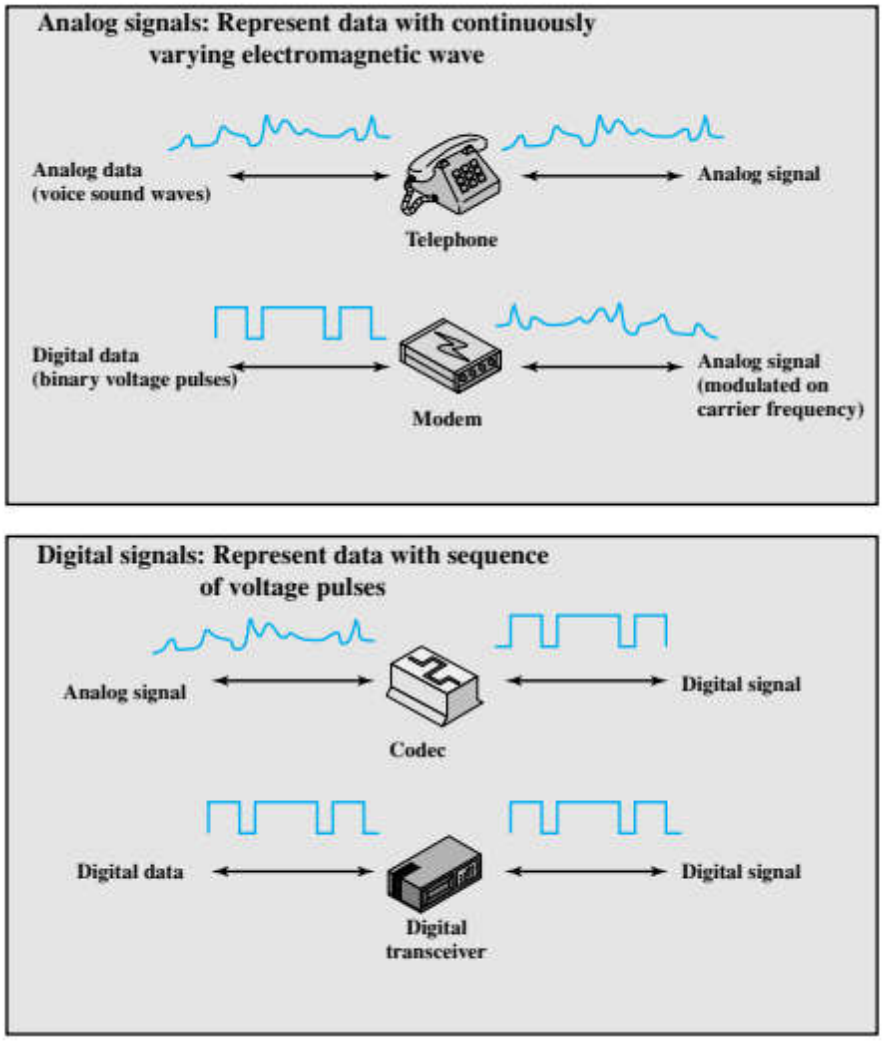

Figure 3.10: Analog and Digital Signaling of Analog and Digital Data

(a) Data and Signals

	Analog Signal	Digital Signal
Analog Data	Two alternatives: (1) signal occupies the same spectrum as the analog data; (2) analog data are encoded to occupy a different portion of spectrum.	Analog data are encoded using a codec to produce a digital bit stream.
Digital Data	Digital data are encoded using a modem to produce analog signal.	Two alternatives: (1) signal consists of two voltage levels to represent the two binary values; (2) digital data are encoded to produce a digital signal with desir ed properties.

(b) Treatment of Signals

	Analog Transmission	Digital Transmission
Analog Signal	Is propagated through amplifiers; same treatment whether signal is used to represent analog data or digital data.	Assumes that the analog signal represents digital data. Signal is propagated through repeaters; at each repeater, digital data are recovered from inbound signal and used to generate a new analog outbound signal.
Digital Signal	Not used	Digital signal represents a stream of 1s and 0s, which may represent digital data or may be an encoding of analog data. Signal is propagated through repeaters; at each repeater, stream of 1s and 0s is recovered from inbound signal and used to generate a new digital outbound signal.

Figure 3.11: Analog and Digital Transmission

digital transmission and, where possible, digital signaling techniques. The most important reasons are as follows:

- Digital technology: The advent of large-scale integration (LSI) and very-large scale integration (VLSI) technology has caused a continuing drop in the cost and size of digital circuitry. Analog equipment has not shown a similar drop.
- Data integrity: With the use of repeaters rather than amplifiers, the effects of noise and other signal impairments are not cumulative. Thus it is possible to transmit data longer distances and over lower quality lines by digital means while maintaining the integrity of the data.
- Capacity utilization: It has become economical to build transmission links of very high bandwidth, including satellite channels and optical fiber. A high degree of multiplexing is needed to utilize such capacity effectively, and this is more easily and cheaply achieved with digital (time division) rather than analog (frequency division) techniques.
- Security and privacy: Encryption techniques can be readily applied to digital data and to analog data that have been digitized.
- Integration: By treating both analog and digital data digitally, all signals have the same form and can be treated similarly. Thus economies of scale and convenience can be achieved by integrating voice, video, and digital data.

3.3 Transmission Impairments

With any communications system, the signal that is received may differ from the signal that is transmitted due to various transmission impairments. For analog signals, these impairments can degrade the signal quality. For digital signals, bit errors may be introduced, such that a binary 1 is transformed into a binary 0 or vice versa. In this section, we examine the various impairments and how they may affect the information-carrying capacity of a communication link. The most significant impairments are

- Attenuation and attenuation distortion: The strength of a signal falls off with distance over any transmission medium. For guided media, this reduction in strength, or attenuation, is generally exponential and thus is typically expressed as a constant number of decibels per unit distance. For unguided media, attenuation is a more complex function of distance and the makeup of the atmosphere. Attenuation introduces three considerations for the transmission engineer. First, a received signal must have sufficient strength so that the electronic circuitry in the receiver can detect the signal. Second, the signal must maintain a level sufficiently higher than noise to be received without error. Third, attenuation varies with frequency. The first and second problems are dealt with by attention to signal strength and the use of amplifiers or repeaters. For a point-to-point link, the signal strength of the transmitter must be strong enough to be received intelligibly, but not so strong as to overload the circuitry of the transmitter or receiver, which would cause distortion. Beyond a certain distance, the attenuation becomes unacceptably great, and repeaters or amplifiers are used to boost the signal at regular intervals.These problems are more complex for multi-point lines where the distance from transmitter to receiver is variable. The third problem is particularly noticeable for analog signals. Because the attenuation varies as a function of frequency, the received signal is distorted, reducing intelligibility. To overcome this problem, techniques are available for equalizing attenuation across a band of frequencies. This is commonly done for voice-grade telephone lines by using loading coils that change the electrical properties of the line; the result is to smooth out attenuation effects. Another approach is to use amplifiers that amplify high frequencies more than lower frequencies.
- Delay distortion: Delay distortion occurs because the velocity of propagation of a signal through a guided medium varies with frequency. For a bandlimited signal, the velocity tends to be highest near the center frequency and fall off toward the two edges of the band.

Thus various frequency components of a signal will arrive at the receiver at different times, resulting in phase shifts between the different frequencies. This effect is referred to as delay distortion because the received signal is distorted due to varying delays experienced at its constituent frequencies. Delay distortion is particularly critical for digital data. Consider that a sequence of bits is being transmitted, using either analog or digital signals. Because of delay distortion, some of the signal components of one bit position will spill over into other bit positions, causing intersymbol interference, which is a major limitation to maximum bit rate over a transmission channel.

- Noise: For any data transmission event, the received signal will consist of the transmitted signal, modified by the various distortions imposed by the transmission system, plus additional unwanted signals that are inserted somewhere between transmission and reception. The latter, undesired signals are referred to as noise. Noise is the major limiting factor in communications system performance. Noise may be divided into four categories:

  - Thermal noise: Thermal noise is due to thermal agitation of electrons. It is present in all electronic devices and transmission media and is a function of temperature. Thermal noise is uniformly distributed across the bandwidths typically used in communications systems and hence is often referred to as white noise. Thermal noise cannot be eliminated and therefore places an upper bound on communications system performance. Because of the weakness of the signal received by satellite earth stations, thermal noise is particularly significant for satellite communication.
  - Intermodulation noise: When signals at different frequencies share the same transmission medium, the result may be intermodulation noise. The effect of intermodulation noise is to produce signals at a frequency that is the sum or difference of the two original frequencies or multiples of those frequencies. Intermodulation noise is produced by non-linearities in the transmitter, receiver, and/or intervening transmission medium. Ideally, these components behave as linear systems; that is, the output is equal to the input times a constant. However, in any real system, the output is a more complex function of the input. Excessive nonlinearity can be caused by component malfunction or overload from excessive signal strength. It is under these circumstances that the sum and difference frequency terms occur.
  - Crosstalk: Crosstalk has been experienced by anyone who, while using the telephone, has been able to hear another conversation; it is an unwanted coupling between signal paths. It can occur by electrical coupling between nearby twisted pairs or, rarely, coax cable lines carrying multiple signals. Crosstalk can also occur when microwave antennas pick up unwanted signals; although highly directional antennas are used, microwave energy does spread during propagation. Typically, crosstalk is of the same order of magnitude as, or less than, thermal noise. All of the types of noise discussed so far have reasonably predictable and relatively constant magnitudes. Thus it is possible to engineer a transmission system to cope with them.
  - Impulse noise: Impulse noise, however, is noncontinuous, consisting of irregular pulses or noise spikes of short duration and of relatively high amplitude. It is generated from a variety of causes, including external electromagnetic disturbances, such as lightning, and faults and flaws in the communications system. Impulse noise is generally only a minor annoyance for analog data. For example, voice transmission may be corrupted by short clicks and crackles with no loss of intelligibility. However, impulse noise is the primary source of error in digital data communication. For example, a sharp spike of energy of 0.01 s duration would not destroy any voice data but would wash out about 560 bits of digital data being transmitted at 56 kbps. Fig. 3.12 is an example of the effect of noise on a digital signal. Here the noise consists of a relatively modest level of

thermal noise plus occasional spikes of impulse noise. The digital data can be recovered from the signal by sampling the received waveform once per bit time.As can be seen, the noise is occasionally sufficient to change a 1 to a 0 or a 0 to a 1.

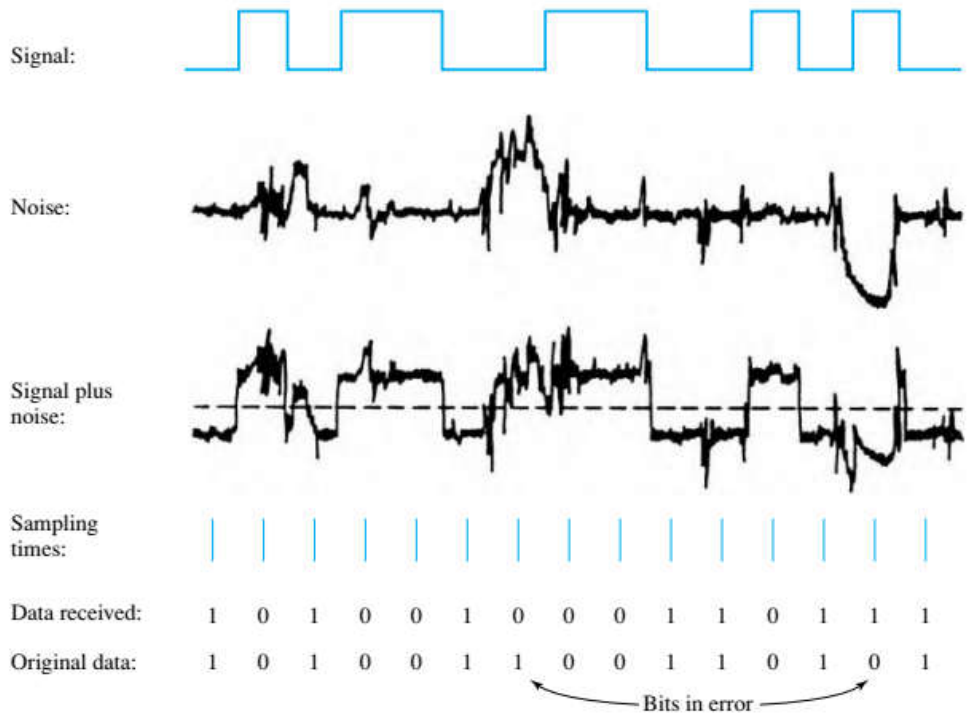

Figure 3.12: Effect of Noise on Digital Signal

3.3.1 Channel Capacity

The maximum rate at which data can be transmitted over a given communication path, or channel, under given conditions, is referred to as the channel capacity. There are four concepts here that we are trying to relate to one another.

- Data rate: The rate, in bits per second (bps), at which data can be communicated
- Bandwidth: The bandwidth of the transmitted signal as constrained by the transmitter and the nature of the transmission medium, expressed in cycles per second, or Hertz
- Noise: The average level of noise over the communications path
- Error rate: The rate at which errors occur, where an error is the reception of a 1 when a 0 was transmitted or the reception of a 0 when a 1 was transmitted

The problem we are addressing is this: Communications facilities are expensive and, in general, the greater the bandwidth of a facility, the greater the cost. Furthermore, all transmission channels of any practical interest are of limited bandwidth. The limitations arise from the physical properties of the transmission medium or from deliberate limitations at the transmitter on the bandwidth to prevent interference from other sources. Accordingly, we would like to make as efficient use as possible of a given bandwidth. For digital data, this means that we would like to get as high a data rate as possible at a particular limit of error rate for a given bandwidth. The main constraint on achieving this efficiency is noise.

Nyquist Bandwidth

To begin, let us consider the case of a channel that is noise free. In this environment, the limitation on data rate is simply the bandwidth of the signal. A formulation of this limitation, due to Nyquist, states that if the rate of signal transmission is $2B$, then a signal with frequencies no greater than B is sufficient to carry the signal rate. The converse is also true: Given a bandwidth of B, the highest signal rate that can be carried is $2B$. This limitation is due to the effect of intersymbol interference, such as is produced by delay distortion. Note that in the preceding paragraph, we referred to signal rate. If the signals to be transmitted are binary (two voltage levels), then the data rate that can be supported by B Hz is $2B$ bps. However, signals with more than two levels can be used; that is, each signal element can represent more than one bit. For example, if four possible voltage levels are used as signals, then each signal element can represent two bits. With multilevel signaling, the Nyquist formulation becomes $C = 2Blog_2M$, where M is the number of discrete signal or voltage levels.

So, for a given bandwidth, the data rate can be increased by increasing the number of different signal elements. However, this places an increased burden on the receiver: Instead of distinguishing one of two possible signal elements during each signal time, it must distinguish one of M possible signal elements. Noise and other impairments on the transmission line will limit the practical value of M.

EXAMPLE: Consider a voice channel being used, via modem, to transmit digital data.Assume a bandwidth of 3100 Hz.Then the Nyquist capacity, C, of the channel is For a value used with some modems, C becomes 18,600 bps for a bandwidth of 3100 Hz.

Shannon Capacity Formula

Nyquist's formula indicates that, all other things being equal, doubling the bandwidth doubles the data rate. Now consider the relationship among data rate, noise, and error rate.The presence of noise can corrupt one or more bits. If the data rate is increased, then the bits become *shorter* so that more bits are affected by a given pattern of noise. Figure 3.16 illustrates this relationship. If the data rate is increased, then more bits will occur during the interval of a noise spike, and hence more errors will occur. All of these concepts can be tied together neatly in a formula developed by the mathematician Claude Shannon. As we have just illustrated, the higher the data rate, the more damage that unwanted noise can do. For a given level of noise, we would expect that a greater signal strength would improve the ability to receive data correctly in the presence of noise. The key parameter involved in this reasoning is the signal-to-noise ratio (SNR, or S/N), which is the ratio of the power in a signal to the power contained in the noise that is present at a particular point in the transmission. Typically, this ratio is measured at a receiver, because it is at this point that an attempt is made to process the signal and recover the data. For convenience, this ratio is often reported in decibels: $SNR_{dB} = 10log_{10}\frac{signalpower}{noisepower}$. This expresses the amount, in decibels, that the intended signal exceeds the noise level. A high SNR will mean a high-quality signal and a low number of required intermediate repeaters. The signal-to-noise ratio is important in the transmission of digital data because it sets the upper bound on the achievable data rate. Shannon's result is that the maximum channel capacity, in bits per second, obeys the equation:

$$C = Blog_2(1+SNR) \tag{3.1}$$

where C is the capacity of the channel in bits per second and B is the bandwidth of the channel in Hertz. The Shannon formula represents the theoretical maximum that can be achieved. In practice, however, only much lower rates are achieved. One reason for this is that the formula assumes white noise (thermal noise). Impulse noise is not accounted for, nor are attenuation

distortion or delay distortion. Even in an ideal white noise environment, present technology still cannot achieve Shannon capacity due to encoding issues, such as coding length and complexity. The capacity indicated in the preceding equation is referred to as the error-free capacity. Shannon proved that if the actual information rate on a channel is less than the error-free capacity, then it is theoretically possible to use a suitable signal code to achieve error-free transmission through the channel. Shannon's theorem unfortunately does not suggest a means for finding such codes, but it does provide a yardstick by which the performance of practical communication schemes may be measured. Several other observations concerning the preceding equation may be instructive. For a given level of noise, it would appear that the data rate could be increased by increasing either signal strength or bandwidth. However,as the signal strength increases, so do the effects of nonlinearities in the system, leading to an increase in inter-modulation noise. Note also that, because noise is assumed to be white, the wider the bandwidth, the more noise is admitted to the system.Thus, as B increases, SNR decreases.

EXAMPLE: Let us consider an example that relates the Nyquist and Shannon formulations. Suppose that the spectrum of a channel is between 3 MHz and 4 MHz and $SNR_{dB} = 24dB$. Then, $B = 4MHz - 3MHz = 1MHz$, $SNR_{dB} = 24dB = 10log_{10}(SNR)$, $SNR = 251$. Using Shannon's formula, $C = 10^6 \times log_2(1+251) \approx 10^6 \times 8 = 8Mbps$

This is a theoretical limit and, as we have said, is unlikely to be reached. But assume we can achieve the limit. Based on Nyquist's formula, how many signaling levels are required? We have $C = 2Blog_2M$, i.e., $8 \times 10^6 = 2 \times (10^6) \times log_2M$, i.e. $4 = log_2M$, i.e. $M = 16$

3.4 Transmission of Signal

A computer network is used for communication of data from one station to another in the network. Analog or digital data traverses through a communication media in the form of a signal from the source to the destination. The channel bridging the transmitter and the receiver may be a guided transmission medium such as a wire or a wave-guide or it can be an unguided atmospheric or space channel. Irrespective of the medium, the signal traversing the channel becomes attenuated and distorted with increasing distance. Hence a process is adopted to match the properties of the transmitted signal to the channel characteristics so as to efficiently communicate over the transmission media. There are two alternatives; the data can be either converted to digital or analog signal. Both the approaches have pros and cons. What to be used depends on the situation and the available bandwidth.

Either form of data can be encoded into either form of signal. For digital signaling, the data source can be either analog or digital, which is encoded into digital signal, using different encoding techniques.The basis of analog signaling is a constant frequency signal known as a carrier signal, which is chosen to be compatible with the transmission media being used, so that it can traverse a long distance with minimum of attenuation and distortion. Data can be transmitted using these carrier signals by a process called modulation, where one or more fundamental parameters of the carrier wave, i.e. amplitude, frequency and phase are modulated by the source data. The resulting signal, called modulated signal, traverses the media, which is demodulated at the receiving end and the original signal is extracted. All the four possibilities are shown in Table 3.1.

3.4.1 Digital signal encoding

The first approach converts digital data to digital signal, known as line coding. Important parameters that affect the characteristics of line coding techniques are:

Table 3.1: Data to Signal Conversion Approaches

Data	Signal	Approach
Digital	Digital	Encoding
Analog	Digital	Encoding
Digital	Analog	Modulation
Analog	Analog	Modulation

- **No of signal levels:** This refers to the number values allowed in a signal, known as signal levels, to represent data. Fig. 3.13(a) shows two signal levels, whereas Fig. 3.13(b) shows three signal levels to represent binary data.

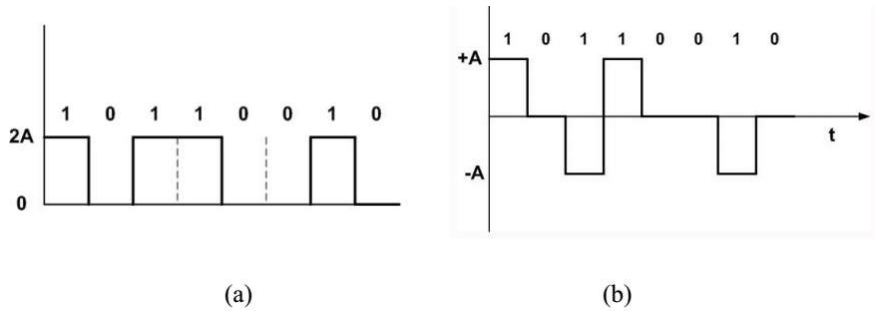

Figure 3.13: (a) Signal with two voltage levels, (b) Signal with three voltage levels

- **Bit rate versus Baud rate:** The bit rate represents the number of bits sent per second, whereas the baud rate defines the number of signal elements per second in the signal. Depending on the encoding technique used, baud rate may be more than or less than the data rate.
- **DC components:** After line coding, the signal may have zero frequency component in the spectrum of the signal, which is known as the direct-current (DC) component. DC component in a signal is not desirable because the DC component does not pass through some components of a communication system such as a transformer. This leads to distortion of the signal and may create error at the output. The DC component also results in unwanted energy loss on the line.
- **Signal Spectrum:** Different encoding of data leads to different spectrum of the signal. It is necessary to use suitable encoding technique to match with the medium so that the signal suffers minimum attenuation and distortion as it is transmitted through a medium.
- **Synchronization:** To interpret the received signal correctly, the bit interval of the receiver should be exactly same or within certain limit of that of the transmitter. Any mismatch between the two may lead wrong interpretation of the received signal. Usually, clock is generated and synchronized from the received signal with the help of a special hardware known as Phase Lock Loop (PLL). However, this can be achieved if the received signal is self-synchronizing having frequent transitions (preferably, a minimum of one transition per bit interval) in the signal.
- **Cost of Implementation:** It is desirable to keep the encoding technique simple enough such that it does not incur high cost of implementation.

Line Coding Techniques

Line coding techniques can be broadly divided into three broad categories:

- **Unipolar:** In unipolar encoding technique, only two voltage levels are used. It uses

only one polarity of voltage level. In this encoding approach, the bit rate same as data rate. Unfortunately, DC component present in the encoded signal and there is loss of synchronization for long sequences of 0's and 1's. It is simple but obsolete.

- **Polar:** Polar encoding technique uses two voltage levels-one positive and the other one negative. Four different encoding schemes come under this category as discussed below:
  - **Non Return to zero (NRZ):** The most common and easiest way to transmit digital signals is to use two different voltage levels for the two binary digits. Usually a negative voltage is used to represent one binary value and a positive voltage to represent the other. The data is encoded as the presence or absence of a signal transition at the beginning of the bit time. In NRZ encoding, the signal level remains same throughout the bit-period. There are two encoding schemes in NRZ: NRZ-L and NRZ-I as shown in Fig. 3.14.

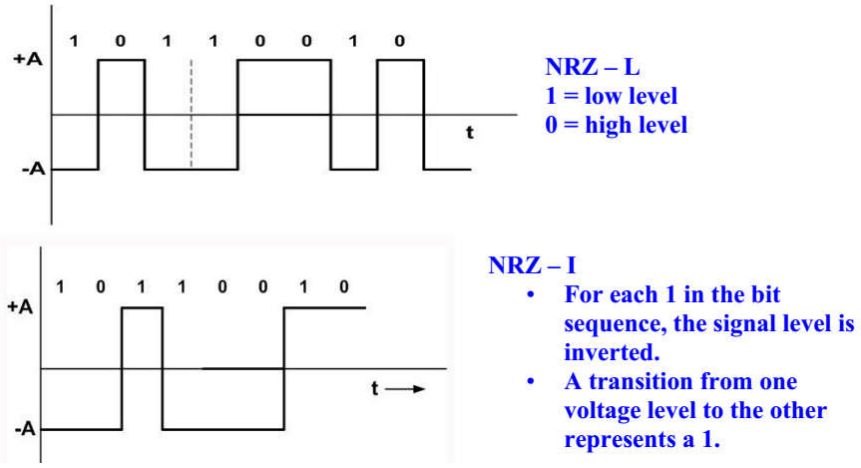

Figure 3.14: NRZ Encoding

The advantages of NRZ coding are:

  - Detecting a transition in presence of noise is more reliable than to compare a value to a threshold.
  - NRZ codes are easy to engineer and it makes efficient use of bandwidth.

The spectrum of the NRZ-L and NRZ-I signals are shown in Fig. 3.15. It may be noted that most of the energy is concentrated between 0 and half the bit rate. The main limitations are the presence of a dc component and the lack of synchronization capability. When there is long sequence of 0's or 1's, the receiving side will fail to regenerate the clock and synchronization between the transmitter and receiver clocks will fail.

  - **Return to Zero (RZ):** To ensure synchronization, there must be a signal transition in each bit as shown in Fig. 3.16.
    Key characteristics of the RZ coding are:
    - Three levels
    - Bit rate is double than that of data rate
    - No dc component
    - Good synchronization
    - Main limitation is the increase in bandwidth
  - **Biphase:** To overcome the limitations of NRZ encoding, biphase encoding techniques can be adopted. Manchester and differential Manchester Coding as shown in Fig. 3.17 are the two common Biphase techniques in use. In Biphase encoding the mid-bit transition serves as a clocking mechanism and also as data.

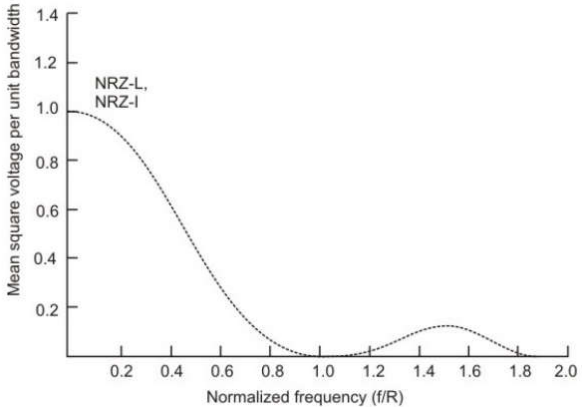

Figure 3.15: NRZ Spectrum

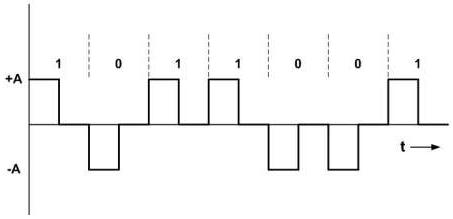

Figure 3.16: RZ Encoding Technique

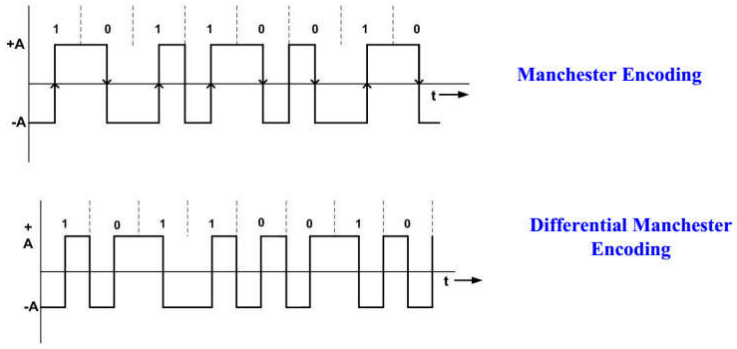

Figure 3.17: Manchester and Differential Manchester Encoding

* **Manchester:** In the standard Manchester coding there is a transition at the middle of each bit period. A binary 1 corresponds to a low-to-high transition and a binary 0 to a high-to-low transition in the middle.
* **Differential Manchester:** In Differential Manchester, inversion in the middle of each bit is used for synchronization. The encoding of a 0 is represented by the presence of a transition both at the beginning and at the middle and 1 is represented by a transition only in the middle of the bit period.

Key characteristics are:

* Two levels
* No DC component
* Good synchronization
* Higher bandwidth due to doubling of bit rate with respect to data rate

The bandwidth required for biphase techniques are greater than that of NRZ techniques, but due to the predictable transition during each bit time, the receiver can synchronize properly on that transition. Biphase encoded signals have no DC components as shown in Fig. 3.18. A Manchester code is now very popular and has been specified for the IEEE 802.3 standard for base band coaxial cables and twisted pair CSMA/CD bus LANs.

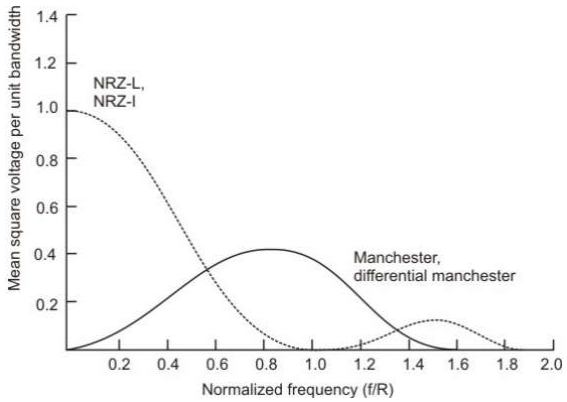

Figure 3.18: Frequency spectrum of the Manchester encoding techniques

- Bipolar:
  - **Bipolar AMI:** Bipolar AMI uses three voltage levels. Unlike RZ, the zero level is used to represent a 0 and a binary 1's are represented by alternating positive and negative voltages, as shown in Fig. 3.19.

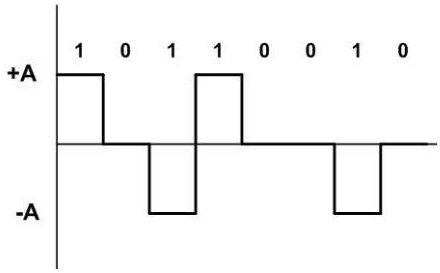

Figure 3.19: Bipolar AMI

  - **Pseudoternary:** This encoding scheme is same as AMI, but alternating positive and negative pulses occur for binary 0 instead of binary 1. Key characteristics are:

* Three Levels
* No DC Component
* Loss of synchronization for long sequences of 0s.
* Lesser bandwidth

Frequency spectrum of different encoding schemes have been compared in Fig. 3.20.

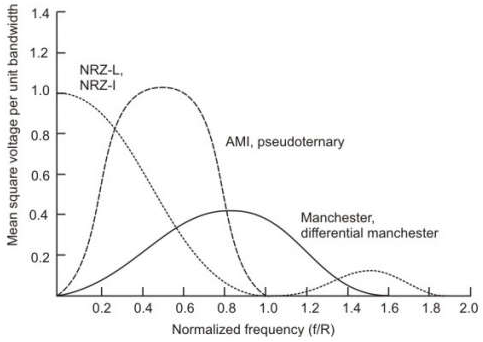

Figure 3.20: Frequency spectrum of different encoding schemes

3.4.2 Scrambling Schemes:

Extension of Bipolar AMI. Used in case of long distance applications.

Goals:

- No dc component
- No long sequences of 0-level line signal
- No increase in bandwidth
- Error detection capability
- Examples:
  - Bipolar with 8-zero substitution(B8ZS): The limitation of bipolar AMI is overcome in B8ZS, which is used in North America. A sequence of eight zero's is replaced by the following encoding:
    A sequence of eight 0's is replaced by 000+-0+-, if the previous pulse was positive. A sequence of eight 0's is replaced by 000-+0+-, if the previous pulse was negative.
  - High Density Bipolar-3 Zeros(HBD3):Another alternative, which is used in Europe and Japan is HDB3. It replaces a sequence of 4 zeros by a code as per the rule given in the following table 3.2. The encoded signals are shown in Fig. 3.21.

Table 3.2: HDB3 Substitution Rule

HDB3 Substitution Rule		
Polarity of the Preceding Pulse	No. of Bipolar Pulses (ones) since last substitution	
	Odd	Even
-	000-	+00+
+	000+	-00-

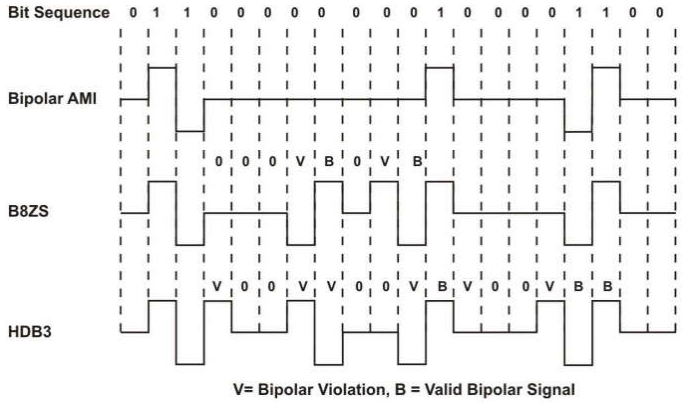

Figure 3.21: B8ZS and HDB3 Encoding Schemes

3.5 Modulation

3.6 Multiplexing

3.7 Assignments