Uttar Pradesh Rajarshi Tandon
Open University

Master in Computer Science

MCS-108

Data Communication and Computer Networks

© UPRTOU, Prayagraj. 2019
ISBN : 978-93-83328-25-3

Printed and Published by Dr. Arun Kumar Gupta Registrar, Uttar Pradesh Rajarshi Tandon Open University, 2020.
Printed By: Chandrakala Universal Pvt. Ltd. 42/7 Jawahar Lal Neharu Road, Prayagraj. 211002.

RIL-089

Uttar Pradesh Rajarshi Tandon
Open University

Master in Computer Science

MCS-108

Data Communication and Computer Networks

Block

1

COMPUTER NETWORK BASICS

Block-I Computer Networks Basics

This is the first block on Data communication & Networks and having detail description of the computer networks. The computer network is very important and wide topic for computer science students. So this block has four following units.

So we will begin with the first unit on introduction of data communication. As all we know the basics of data communication is very important so firstly we have introduced Layered network architecture, we also focused on the review of ISO-OSI Model. After IOS-OSI model we introduced different data communication techniques: like Pulse, Code Modulation, (PCM), Data modems. These techniques are very important in data communication.

Second unit begins with Multiplexing techniques. There are two main multiplexing techniques available in data communication i.e. Frequency-Division known as FDM and Time-Division known as TDM. We also introduce the most important topic Transmission Media. In transmission media we focused detail descriptions of wires, cables, radio, Links, Satellite Links, and Fiber-Optic Links.

In the third unit, we provided another important topic Asynchronous Transfer Mode known as ATM. In ATM we focused on Cells, Header and Cell Formats, different layers in ATM, Class 1,2,3,4. In this unit we also introduced Traffic Random Access Data Networks and Concept of Random Access

In the last, the fourth unit we introduced the concept of pure ALOHA, we also provided the concept of throughput characteristics of slotted ALOHA, throughputs for Finite and Infinite, Population S-ALOHAS. MARKOV Chain Model for S-ALOHA

As you study the material, you will found that figures, tables are properly used and these will help to understand the concept. There are many sections in the units to easily understand the topic. Every unit has summary and review questions in the end of the unit which will help you to review yourself.

In your study, you will found that the every unit has different equal length and your study time will vary for each unit.

We hope you enjoy studying the material and once again wish you all the best for your success.

UNIT-I

Data Communication Introduction

Structure

1.0 Introduction

This is the first unit of this block. In this unit we have described the concept of data communication basics. In this unit there are five sections. In the first section there is an introduction of data communication. In Section 1.1 you will learn about the layered network architecture. In this unit we introduce the ISO-OSI layered architecture of Networks. As you have studied earlier that ISO-OSI has have been divided into 7 layers depending on the complexity of the functionality that each of these layers provide. In Sec. 1.2 you will review about all the layers of OSI model. In the next section i.e. Sec. 1.3 we define the concept of data communication techniques. In the next section you will also learn about pulse code modulation (PCM) and data modems. In Sec. 1.6 and 1.7 you will find summary and review questions respectively.

Objectives

After studying this unit you should be able to:

- Define layered network architecture
- Review ISO-OSI Model
- Describe Data communication techniques like Pulse Code Modulation (PCM) and Data modems.

1.1 Layered network architecture

Network architectures describe the standards and techniques for designing and building communication systems for computers and other network devices. In the previously, vendors of computers developed their specific architectures and required that other sellers follow to this architecture; if they required to develop compatible hardware and software. There are registered network architectures such as IBM's SNA (Systems Network Architecture) and there are open architectures like the OSI (Open Systems Interconnection) model defined by the International Organization for Standardization. The previous strategy, where the computer network is designed with the hardware as the main concern and software is afterthought, no longer works. Network software is now highly structured. To reduce the design complexity, most of the networks are organized as a series of layers or levels, each one build upon one below it. The basic idea of a layered architecture is to divide the design into small pieces. Each layer adds to the services provided by the lower layers in such a manner that the highest layer is provided a full set of services to manage communications and run the applications. The benefits of the layered models are modularity and clear interfaces, i.e. open architecture and comparability between the different providers' components. A basic principle is to ensure independence of layers by defining services provided by each layer to the next higher layer without defining how the services are to be performed. This permits changes in a layer without affecting other layers. Prior to the use of layered protocol architectures, simple changes such as adding one terminal type to the list of those supported by an architecture often required changes to essentially all communications software at a site. The number of layers, functions and contents of each layer differ from network to network. However in all networks, the purpose of each layer is to offer certain services to higher layers, shielding those layers from the details of how the services are actually implemented. The basic elements of a layered model are services, protocols and interfaces. A service is a set of actions that a layer offers to another (higher) layer. Protocol is a set of rules that a layer uses to exchange information with a peer entity. These rules concern both the contents and the order of the messages used. Between the layers service interfaces are defined. The messages from one layer to another are sent through those interfaces.

In an n-layer architecture, layer n on one machine carries on conversation with the layer n on other machine. The rules and conventions used in this conversation are collectively known as the layer-n protocol. Basically, a protocol is an agreement between the communicating parties on how communication is to proceed. Violating the protocol will make communication more difficult, if not impossible. A five-layer architecture is shown in Fig. 1.1, the entities are comprising the corresponding layers on different machines are called peers. In other words, it is the peers that communicate using protocols. In reality, no data is transferred from layer n on one machine to layer n of 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 layer through which actual communication occurs. The peer process abstraction is crucial to all network design. Using it, the un-manageable tasks of designing the complete network can be broken into several smaller, manageable, design problems, namely design of individual layers.

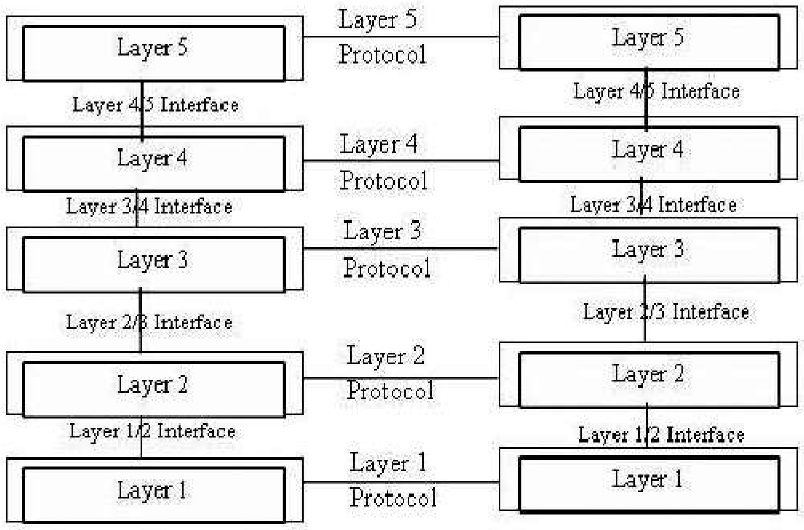


Figure 1.1 Basic Layered architecture.

Between each pair of adjacent layers there is an interface. The interface defines which primitives operations and services the lower layer offers to the upper layer adjacent to it. When network designer decides how many layers to include in the network and what each layer should do, one of the main considerations is defining clean interfaces between adjacent layers. Doing so, in turns requires that each layer should perform well-defined functions. In addition to minimize the amount of information passed between layers, clean-cut interface also makes it simpler to replace the implementation of one layer with a completely different implementation, because all what is required of new implementation is that it offers same set of services to its upstairs neighbor as the old implementation (that is what a layer provides and how to use that service from it is more important than knowing how exactly to implements it).

A set of layers and protocols is known as network architecture. The specification of architecture must contain enough information to allow an implementation to write the program or build the hardware for each layer so that it will correctly obey the appropriate protocol. Neither the details of implementation nor the specification of interface is a part of network architecture because these are hidden away inside machines and not visible from outside. It is not even necessary that the interface on all machines in a network be same, provided that each machine can correctly use all protocols. A list of protocols used by a certain system, one protocol per layer, is called protocol stack.

Check your progress

Q1. What is Layered architecture? Explain.

Q2. Define third layer 3 protocol.

1.2 The ISO/OSI Reference Model

The standard model for networking protocols and distributed applications is the International Standard Organization's Open System Interconnect (ISO/OSI) model. It defines seven network layers.

Open System Interconnection, an ISO standard for worldwide communications that defines a networking framework for implementing protocols in seven layers. Control is passed from one layer to the next, starting at the application layer in one station proceeding to the bottom layer, over the channel to the next station and back up the hierarchy.

At one time, most vendors agreed to support OSI in one form or another, but OSI was too loosely defined and proprietary standards were too entrenched. Except for the OSI-compliant X.400 and X.500 e-mail and directory standards, which are widely used, what was once thought to become the universal communications standard now serves as the teaching model for all other protocols.

Control is passed from one layer to the next, starting at the application layer in one station proceeding to the bottom layer, over the channel to the next station and back up the hierarchy.

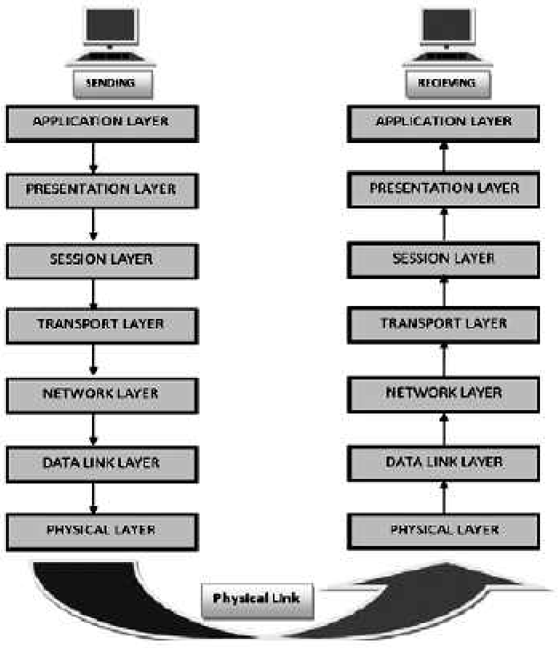


Figure. 1.2 OSI Model

Physical Layer (Layer 1)

The lowest layer of the OSI Reference Model is layer 1, the *physical layer*; it is commonly abbreviated "PHY". The physical layer is special compared to the other layers of the model, because it is the only one where data is physically moved across the network interface. All of the other layers perform useful functions to create messages to be sent, but they must all be transmitted down the protocol stack to the physical layer, where they are actually sent out over the network.

Physical Layer Functions

The following are the main responsibilities of the physical layer in the OSI Reference Model:

- **Definition of Hardware Specifications:** The details of operation of cables, connectors, wireless radio transceivers, network interface cards and other hardware devices are generally a function of the physical layer (although also partially the data link layer; see below).

- **Encoding and Signaling:** The physical layer is responsible for various encoding and signaling functions that transform the data from bits that reside within a computer or other device into signals that can be sent over the network.

- **Data Transmission and Reception:** After encoding the data appropriately, the physical layer actually transmits the data, and of course, receives it. Note that this applies equally to wired and wireless networks, even if there is no tangible cable in a wireless network!

- **Topology and Physical Network Design:** The physical layer is also considered the domain of many hardware-related network design issues, such as LAN and WAN topology.

In general, then, physical layer technologies are ones that are at the very lowest level and deal with the actual ones and zeroes that are sent over the network. For example, when considering network interconnection devices, the simplest ones operate at the physical layer: repeaters, conventional hubs and transceivers. These devices have absolutely no knowledge of the contents of a message. They just take input bits and send them as output. Devices like switches and routers operate at higher layers and look at the data they receive as being more than voltage or light pulses that represent one or zero.

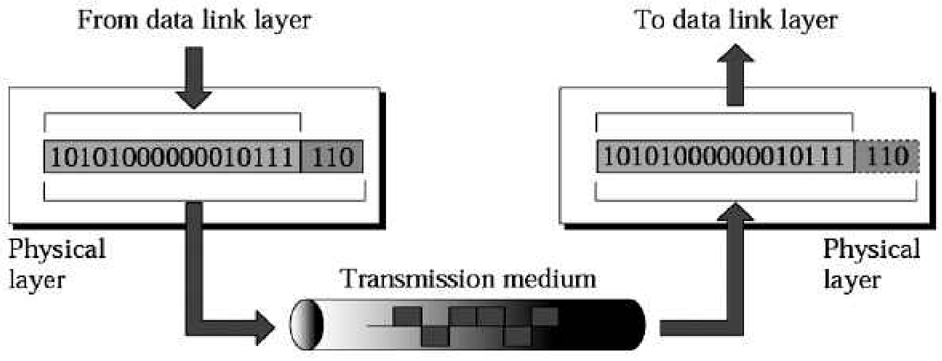


Figure 1.3 Physical Layer

Data Link Layer (Layer 2)

The second-lowest layer (layer 2) in the OSI Reference Model stack is the *data link layer*, often abbreviated "DLL" (though that abbreviation has other meanings as well in the computer world). The data link layer, also sometimes just called the *link layer*, is where many wired and wireless local area networking (LAN) technologies primarily function. For example, Ethernet, Token Ring, FDDI and 802.11 ("wireless Ethernet" or "Wi-Fi') are all sometimes called "data link layer technologies". The set of devices connected at the data link layer is what is commonly considered a simple "network", as opposed to an internetwork.

Data Link Layer Functions

The following are the key tasks performed at the data link layer:

- **Logical Link Control (LLC):** Logical link control refers to the functions required for the establishment and control of logical links between local devices on a network. As mentioned above, this is usually considered a DLL sub layer; it provides services to the network layer above it and hides the rest of the details of the data link layer to allow different technologies to work seamlessly with the higher layers. Most local area networking technologies use the IEEE 802.2 LLC protocol.
- **Media Access Control (MAC):** This refers to the procedures used by devices to control access to the network medium. Since many networks use a shared medium (such as a single network cable, or a series of cables that are electrically connected into a single virtual medium) it is necessary to have rules for managing the medium to avoid conflicts. For example. Ethernet uses the CSMA/CD method of media access control, while Token Ring uses token passing.
- **Data Framing:** The data link layer is responsible for the final encapsulation of higher-level messages into *frames* that are sent over the network at the physical layer.
- **Addressing:** The data link layer is the lowest layer in the OSI model that is concerned with addressing: labeling information with

a particular destination location. Each device on a network has a unique number, usually called a ***hardware address*** or ***MAC address*** that is used by the data link layer protocol to ensure that data intended for a specific machine gets to it properly.

- ❖ **Error Detection and Handling:** The data link layer handles errors that occur at the lower levels of the network stack. For example, a cyclic redundancy check (CRC) field is often employed to allow the station receiving data to detect if it was received correctly.

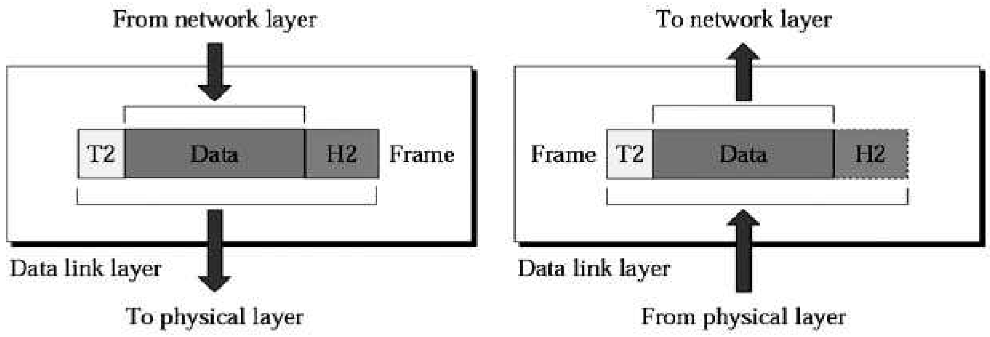


Figure 1.4 Data Link Layer

Network Layer (Layer 3)

The third-lowest layer of the OSI Reference Model is the ***network layer***. If the data link layer is the one that basically defines the boundaries of what is considered a network, the network layer is the one that defines how ***internetworks*** (interconnected networks) function. The network layer is the lowest one in the OSI model that is concerned with actually getting data from one computer to another even if it is on a remote network; in contrast, the data link layer only deals with devices that are local to each other.

While all of layers 2 to 6 in the OSI Reference Model serve to act as "fences" between the layers below them and the layers above them, the network layer is particularly important in this regard. It is at this layer that the transition really begins from the more abstract functions of the higher layers—which don't concern themselves as much with data delivery—into the specific tasks required to get data to its destination. The transport layer, which is related to the network layer in a number of ways, continues this "abstraction transition" as you go up the OSI protocol stack.

Network Layer Functions

Some of the specific jobs normally performed by the network layer include:

- ❖ **Logical Addressing:** Every device that communicates over a network has associated with it a logical address, sometimes called a ***layer three*** address. For example, on the Internet, the Internet Protocol (IP) is the network layer protocol and every machine has an IP address. Note that addressing is done at the data link layer as well, but those addresses refer to local physical devices. In contrast, logical addresses are independent of particular hardware and must be unique across an entire internetwork.

- **Routing:** Moving data across a series of interconnected networks is probably the defining function of the network layer. It is the job of the devices and software routines that function at the network layer to handle incoming packets from various sources, determine their final destination, and then figure out where they need to be sent to get them where they are supposed to go.
- **Datagram Encapsulation:** The network layer normally encapsulates messages received from higher layers by placing them into *datagrams* (also called *packets*) with a network layer header.
- **Fragmentation and Reassembly:** The network layer must send messages down to the data link layer for transmission. Some data link layer technologies have limits on the length of any message that can be sent. If the packet that the network layer wants to send is too large, the network layer must split the packet up, send each piece to the data link layer, and then have pieces reassembled once they arrive at the network layer on the destination machine.
- **Error Handling and Diagnostics:** Special protocols are used at the network layer to allow devices that are logically connected, or that are trying to route traffic, to exchange information about the status of hosts on the network or the devices themselves.

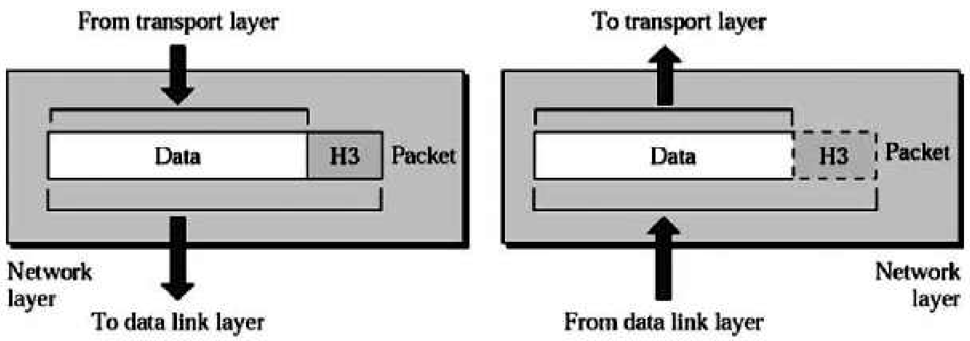


Figure 1.5 Network Layer

Transport Layer (Layer 4)

The fourth and "middle" layer of the OSI Reference Model protocol stack is the *transport layer*. I consider the transport layer in some ways to be part of both the lower and upper "groups" of layers in the OSI model. It is more often associated with the lower layers, because it concerns itself with the *transport* of data, but its functions are also somewhat high-level, resulting in the layer having a fair bit in common with layers 5 through 7 as well.

Recall that layers 1, 2 and 3 are concerned with the actual packaging, addressing, routing and delivery of data; the physical layer handles the bits; the data link layer deals with local networks and the network layer handles routing between networks. The transport layer, in contrast, is sufficiently conceptual that it no longer concerns itself with these "nuts and bolts" matters. It relies on the lower layers to handle the process of moving data between devices.

The transport layer really acts as a "liaison" of sorts between the abstract world of applications at the higher layers, and the concrete functions of layers one to three. Due to this role, the transport layer's overall job is to provide the necessary functions to enable communication between software application processes on different computers. This encompasses a number of different but related duties

Modern computers are multitasking, and at any given time may have many different software applications all trying to send and receive data. The transport layer is charged with providing a means by which these applications can all send and receive data using the same lower-layer protocol implementation. Thus, the transport layer is sometimes said to be responsible for ***end-to-end*** or ***host-to-host*** transport (in fact, the equivalent layer in the TCP/IP model is called the "host-to-host transport layer").

Transport Layer Functions

Let's look at the specific functions often performed at the transport layer in more detail:

- **Process-Level Addressing:** Addressing at layer two deals with hardware devices on a local network, and layer three addressing identifies devices on a logical internetwork. Addressing is also performed at the transport layer, where it is used to differentiate between software programs. This is part of what enables many different software programs to use a network layer protocol simultaneously, as mentioned above. The best example of transport-layer process-level addressing is the TCP and UDP port mechanism used in TCP/IP, which allows applications to be individually referenced on any TCP/IP device.

- **Multiplexing and De-multiplexing:** Using the addresses I just mentioned, transport layer protocols on a sending device ***multiplex*** the data received from many application programs for transport, combining them into a single stream of data to be sent. The same protocols receive data and then ***demultiplex*** it from the incoming stream of datagrams, and direct each package of data to the appropriate recipient application processes.

- **Segmentation, Packaging and Reassembly:** The transport layer segments the large amounts of data it sends over the network into smaller pieces on the source machine, and then reassemble them on the destination machine. This function is similar conceptually to the fragmentation function of the network layer; just as the network layer fragments messages to fit the limits of the data link layer, the transport layer segments messages to suit the requirements of the underlying network layer.

- **Connection Establishment, Management and Termination:** Transport layer connection-oriented protocols are responsible for the series of communications required to establish a connection, maintain it as data is sent over it, and then terminate the connection when it is no longer required.
- **Acknowledgments and Retransmissions:** As mentioned above, the transport layer is where many protocols are implemented that guarantee reliable delivery of data. This is done using a variety of techniques, most commonly the combination of *acknowledgments* and *retransmission timers*. Each time data is sent a timer is started; if it is received, the recipient sends back an acknowledgment to the transmitter to indicate successful transmission. If no acknowledgment comes back before the timer expires, the data is retransmitted. Other algorithms and techniques are usually required to support this basic process.
- **Flow Control:** Transport layer protocols that offer reliable delivery also often implement *flow control* features. These features allow one device in a communication to specify to another that it must "throttle back" the rate at which it is sending data, to avoid bogging down the receiver with data. These allow mismatches in speed between sender and receiver to be detected and dealt with.

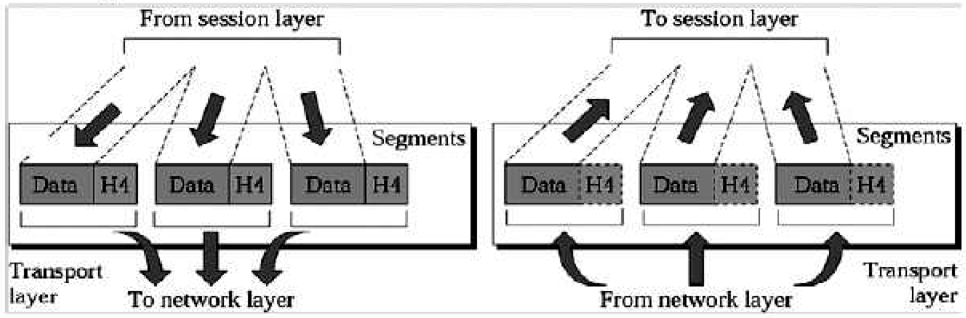


Figure 1.6 Transport Layer

Check your progress

Q1. Define Physical and data link layer.

Q2. Explain network and transport layer.

Session Layer (Layer 5)

The fifth layer in the OSI Reference Model is the *session layer*. As we proceed up the OSI layer stack from the bottom, the session layer is the first one where pretty much all practical matters related to the addressing, packaging and delivery of data are left behind—they are functions of layers four and below. It is the lowest of the three upper layers, which collectively are concerned mainly with software application issues and not with the details of network and internet implementation.

The name of this layer tells you much about what it is designed to do: to allow devices to establish and manage *sessions*. In general terms, a session is a persistent logical linking of two software application processes, to allow them to exchange data over a prolonged period of time. In some discussions, these sessions are called *dialogs*; they are roughly analogous to a telephone call made between two people.

Session Layer Functions

As we have mentioned in a few places in this unit, the boundaries between layers start to get very fuzzy once you get to the session layer, which makes it hard to categorize what exactly belongs at layer 5. Some technologies really span layers 5 through 7, and especially in the world of TCP/IP, it is not common to identify protocols that are specific to the OSI session layer.

The term "session" is somewhat vague, and this means that there is sometimes disagreement on the specific functions that belong at the session layer, or even whether certain protocols belong at the session layer or not. To add to this potential confusion, there is the matter of differentiating between a "connection" and a "session". Connections are normally the province of layer four and layer three, yet a Transmission Control Protocol (TCP) connection, for example, can persist for a long time. The longevity of TCP connections makes them hard to distinguish from "sessions" (and in fact there are some people who feel that the TCP/IP host-to-host transport layer really straddles OSI layers four and five).

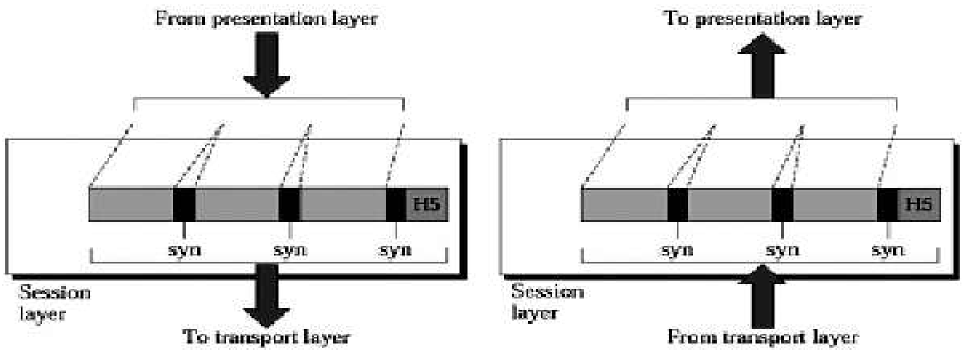


Figure 1.7 Session Layer

Presentation Layer (Layer 6)

The *presentation layer* is the sixth layer of the OSI Reference Model protocol stack, and second from the top. It is different from the other layers in two key respects. First, it has a much more limited and specific function than the other layers; it's actually somewhat easy to describe, hurray! Second, it is used much less often than the other layers; in many types of connections it is not required.

The name of this layer suggests its main function as well: it deals with the *presentation* of data. More specifically, the presentation layer is charged with taking care of any issues that might arise where data sent from one system needs to be viewed in a different way by the other system. It also takes care of any special processing that must be done to data from the time an application tries to send it until the time it is sent over the network.

Presentation Layer Functions

Here are some of the specific types of data handling issues that the presentation layer handles:

- ❖ **Translation:** Networks can connect very different types of computers together: PCs, Macintoshes, UNIX systems, AS/400 servers and mainframes can all exist on the same network. These systems have many distinct characteristics and represent data in different ways; they may use different character sets for example. The presentation layer handles the job of hiding these differences between machines.
- ❖ **Compression:** Compression (and decompression) may be done at the presentation layer to improve the throughput of data. (There are some who believe this is not, strictly speaking a function of the presentation layer.)
- ❖ **Encryption:** Some types of encryption (and decryption) are performed at the presentation layer. This ensures the security of the data as it travels down the protocol stack. For example, one of the most popular encryption schemes that is usually associated with the presentation layer is the Secure Sockets Layer (SSL) protocol. Not all encryption is done at layer 6, however; some encryption is often done at lower layers in the protocol stack, in technologies such as IPsec.

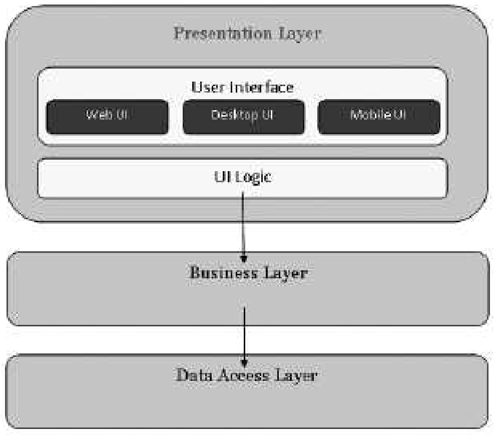


Figure 1.8 Presentation Layer

Application Layer (Layer 7)

At the very top of the OSI Reference Model stack of layers, we find layer 7, the *application layer*. Continuing the trend that we saw in layers 5 and 6, this one too is named very appropriately: the application layer is the one that is used by network applications. These programs are what actually implement the functions performed by users to accomplish various tasks over the network.

It's important to understand that what the OSI model calls an "application" is not exactly the same as what we normally think of as an "application". In the OSI model, the application layer provides services for user applications to employ. For example, when you use your Web browser, that actual software is an application running on your PC. It doesn't really "reside" at the application layer. Rather, it makes use of the services offered by a protocol that operates at the application layer, which is called the Hypertext Transfer Protocol (HTTP). The distinction between the browser and HTTP is subtle, but important.

The reason for pointing this out is because not all user applications use the application layer of the network in the same way. Sure, your Web browser does, and so does your e-mail client and your Usenet news reader. But if you use a text editor to open a file on another machine on your network, that editor is not using the application layer. In fact, it has no clue that the file you are using is on the network: it just sees a file addressed with a name that has been mapped to a network somewhere else. The operating system takes care of *redirecting* what the editor does, over the network.

Similarly, not all uses of the application layer are by applications. The operating system itself can (and does) use services directly at the application layer.

That caveat aside, under normal circumstances, whenever you interact with a program on your computer that is designed specifically for use on a network, you are dealing directly with the application layer. For example, sending an e-mail, firing up a Web browser, or using an IRC chat program—all of these involve protocols that reside at the application layer.

There are dozens of different application layer protocols that enable various functions at this layer. Some of the most popular ones include HTTP, FTP, SMTP, DHCP, NFS, Telnet, SNMP, POP3, NNTP and IRC.

As the "top of the stack" layer, the application layer is the only one that does not provide any services to the layer above it in the stack—there isn't one! Instead, it provides services to programs that want to use the network, and to you, the user. So the responsibilities at this layer are simply to implement the functions that are needed by users of the network. And, of course, to issue the appropriate commands to make use of the services provided by the lower layers.

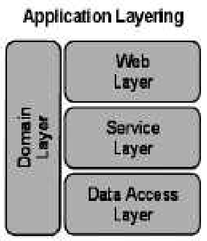


Figure 1.9 Application Layer

Check your progress

Q1. Define Session Layer.

Q2. What is the working of presentation and application layers?

1.3 Data communication techniques

For two devices linked by a transmission medium to exchange data, a high degree of co-operation is required. Typically data is transmitted one bit at a time. The timing (rate, duration, spacing) of these bits must be same for transmitter and receiver. There are two options for transmission of bits.

1. **Parallel** All bits of a byte are transferred simultaneously on separate parallel wires. Synchronization between multiple bits is required which becomes difficult over large distance. Gives large band width but expensive. Practical only for devices close to each other.
2. **Serial** Bits transferred serially one after other. Gives less bandwidth but cheaper. Suitable for transmission over long distances.

Transmission Techniques:

1. **Asynchronous**

   Small blocks of bits (generally bytes) are sent at a time without any time relation between consecutive bytes .when no transmission occurs a default state is maintained corresponding to bit 1. Due to arbitrary delay between consecutive bytes, the time occurrences of the clock pulses at the receiving end need to be synchronized for each byte. This is achieved by providing 2 extra bits start and stop.

   Start bit: It is prefixed to each byte and equals 0. Thus it ensures a transition from 1 to 0 at onset of transmission of byte. The leading edge of start bit is used as a reference for generating clock pulses at required sampling instants. Thus each onset of a byte results in resynchronization of receiver clock.

   Stop bit: To ensure that transition from 1 to 0 is always present at beginning of a byte it is necessary that default state be 1. But there may be two bytes one immediately following the other and if last bit of first byte is 0, transition from 1 to 0 will not occur. Therefore a stop bit is suffixed to each byte equaling 1. Its duration is usually 1, 1.5,2 bits.

   Asynchronous transmission is simple and cheap but requires an overhead of 3 bits i.e. for 7 bit code 2 (start, stop bits) +1 parity bit implying 30% overhead. However % can be reduced by sending larger blocks of data but then timing errors between receiver and sender cannot be tolerated beyond [50/no. of bits in block] % (assuming sampling is done at middle of bit interval). It will not only result in incorrect sampling but also misaligned bit count i.e. a data bit can be mistaken for stop bit if receiver's clock is faster.

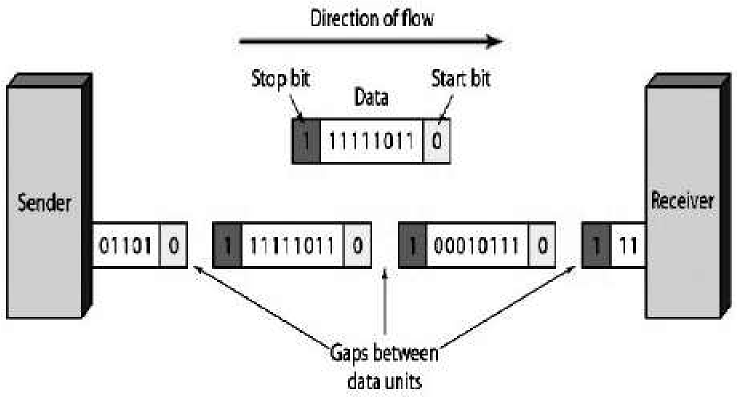


Figure-1.10 Asynchronous Transmission

Synchronous

Larger blocks of bits are successfully transmitted. Blocks of data are either treated as sequence of bits or bytes. To prevent timing drift clocks at two ends need to be synchronized. This can done in two ways:

1. Provide a separate clock line between receiver and transmitter. OR
2. Clocking information is embedded in data signal i.e. biphasic coding for digital signals.

Still another level of synchronization is required so that receiver determines beginning or end of block of data. Hence each block begins with a start code and ends with a stop code. These are in general same known as flag that is unique sequence of fixed no. of bits. In addition some control characters encompass data within these flags. **Data + control information** is called a frame. Since any arbitrary bit pattern can be transmitted there is no assurance that bit pattern for flag will not appear inside the frame thus destroying frame level synchronization, so to avoid this we use bit stuffing

Bit Stuffing: Suppose our flag bits are 01111110 (six 1's). So the transmitter will always insert an extra 0 bit after each occurrence of five 1's (except for flags). After detecting a starting flag the receiver monitors the bit stream. If pattern of five 1's appear, the sixth is examined and if it is 0 it is deleted else if it is 1 and next is 0 the combination is accepted as a flag. Similarly byte stuffing is used for byte oriented transmission. Here we use an escape sequence to prefix a byte similar to flag and 2 escape sequences if byte is itself an escape sequence.

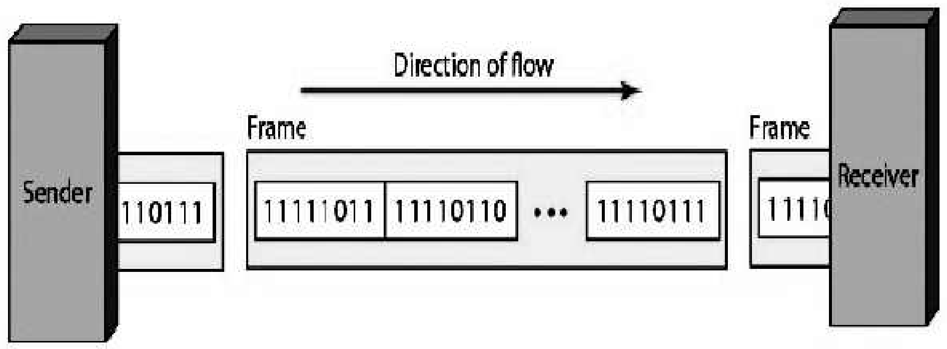


Figure-1.11 Synchronous Transmission

1.4 Pulse Code Modulation

Pulse code Modulation: The pulse code modulator technique samples the input signal x(t) at a sampling frequency. This sampled variable amplitude pulse is then digitalized by the analog to digital converter. Figure. 1.13 shows the PCM generator.

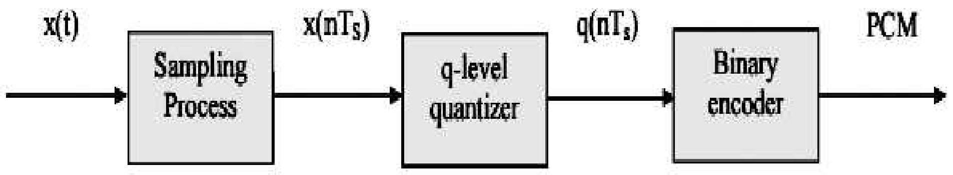


Figure 1.12 PCM modulator

In the PCM generator, the signal is first passed through sampler which is sampled at a rate of (f_s) where:

$$\boxed{f_s \geq 2f_m} \qquad (1)$$

The output of the sampler $x(nT_s)$ which is discrete in time is fed to a q-level quantizer. The quantizer compares the input $x(nT_s)$ with it's fixed levels. It assigns any one of the digital level to $x(nT_s)$ that results in minimum distortion or error. The error is called quantization error, thus the output of the quantizer is a digital level called $q(nT_s)$. The quantized signal level $q(nT_s)$ is binary encode. The encoder converts the input signal to ν digits binary word.

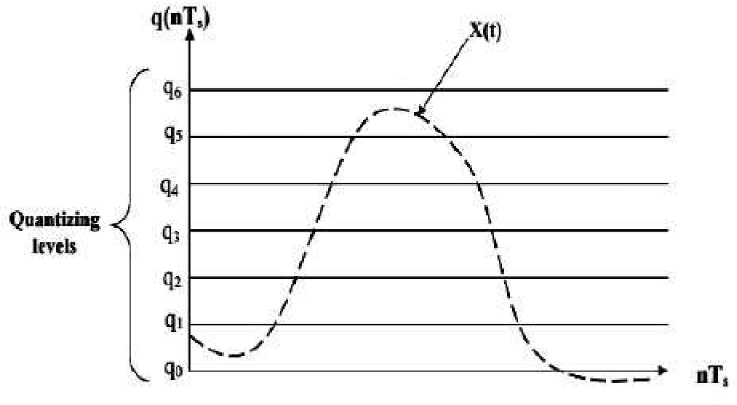


Figure-1.13 A sampled signal and the quantized levels

Figure-1.15 shows the block diagram of the PCM receiver. The receiver starts by reshaping the received pulses, removes the noise and then converts the binary bits to analog. The received samples are then filtered by a low pass filter; the cut off frequency is at f_c.

$$\boxed{f_c = f_m} \qquad (2)$$

Where f_m: is the highest frequency component in the original signal.

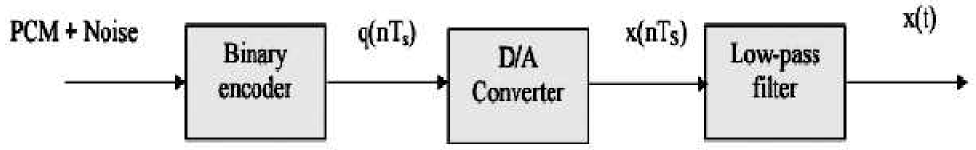


Figure-1.14 PCM demodulator

It is impossible to reconstruct the original signal x(t) because of the permanent quantization error introduced during quantization at the transmitter. The quantization error can be reduced by the increasing quantization levels. This corresponds to the increase of bits per sample (more information). But increasing bits (*v*) increases the signaling rate and requires a large transmission bandwidth. The choice of the parameter for the number of quantization levels must be acceptable with the quantization noise (quantization error).

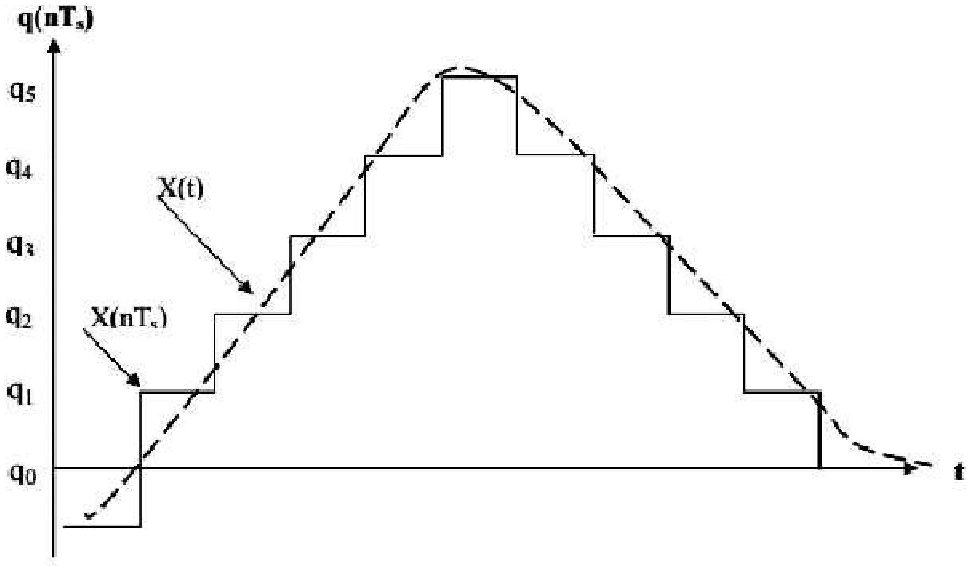


Figure-1.15.The reconstructed signal

Signaling Rate in PCM

Let the quantizer use 'v' number of binary digits to represent each level. Then the number of levels that can be represented by ν digits will be:

$$\boxed{q=2^{\nu}} \qquad (3)$$

The number of bits per second is given by:

(Number of bits per second) = (Number of bits per samples) x (number of samples per second)= ν (bits per sample) x f_s (samples per second)

The number of bits per second is also called signaling rate of PCM and is denoted by 'r':

$$\boxed{\text{Signaling rate} = \nu f_s} \qquad (4)$$

Where

$$f_s \geq f_m$$

Advantages of PCM

1. Effect of noise is reduced.
2. PCM permits the use of pulse regeneration.
3. Multiplexing of various PCM signals is possible.

1.5 Data modems

Modem, (from "***mo**dulator/**dem**odulator*"), any of a class of electronic devices that convert digital data signals into modulated analog signals suitable for transmission over analog telecommunications circuits. A modem also receives modulated signals and demodulates them, recovering the digital signal for use by the data equipment. Modems thus make it possible for established telecommunications media to support a wide variety of data communication, such as e-mail between personal computers, facsimile transmission between fax machines, or the downloading of audio-video files from a World Wide Webserver to a home computer.

Most modems are "voice band"; i.e., they enable digital terminal equipment to communicate over telephone channels, which are designed around the narrow bandwidth requirements of the human voice. Cable modems, on the other hand, support the transmission of data over hybrid fiber-coaxial channels, which were originally designed to provide high-bandwidth television service. Both voice band and cable modems are marketed as freestanding, book-sized modules that plug into a telephone or cable outlet and a port on a personal computer. In addition, voice band modems are installed as circuit boards directly into computers and fax machines. They are also available as small card-sized units that plug into laptop computers.

Data signals consist of multiple alternations between two values, represented by the binary digits, or bits, 0 and 1. Analog signals, on the other hand, consist of time-varying, wavelike fluctuations in value, much like the tones of the human voice. In order to represent binary data, the fluctuating values of the analog wave (i.e., its frequency, amplitude, and phase) must be modified, or modulated, in such a manner as to represent the sequences of bits that make up the data signal.

Each modified element of the modulated carrier wave (for instance, a shift from one frequency to another or a shift between two phases) is known as a baud. In early voice band modems beginning in the early 1960s, one baud represented one bit, so that a modem operating, for instance, at 300 bauds per second (or, more simply, 300 baud) transmitted data at 300 bits per second. In modern modems a baud can represent many bits, so that the more accurate measure of transmission rate is bits or kilobits (thousand bits) per second. During the course of their development, modems have risen in throughput from 300 bits per second (bps) to 56 kilobits per second (Kbps) and beyond. Cable modems achieve a throughput of several megabits per second (Mbps; million bits per second). At the highest bit rates, channel-encoding schemes must be employed in order to reduce transmission errors. In addition, various source-encoding schemes can be used to "compress" the data into fewer bits, increasing the rate of information transmission without raising the bit rate.

Types of Modems

Standard Modems

Most standard modems today are either internal or external.

An internal modem is used inside of the computer and connects directly to the I/O BUS. The internal modem does not require a separate power supply as it gets its power from the computer's internal BUS nor does an internal modem require a serial port or connecting cables to that port. An internal modem will contain a 16550A UART or equivalent circuitry, which will aid in fast data throughput to the computer. Internal modems are usually cheaper than external modems as well. Internal modems are a little more difficult to install than external modems and an available ISA slot must be present to install it into the computer.

On an external modem it is helpful that the modem be connected to a 16550A UART serial port to be assured of the maximum data throughput and of course, an available serial port is necessary. An external modem connects to a serial port on the PC or TI and requires the use of a connecting cable between the PC (or TI) and modem, a power connection and of course a place to put the modem itself. The advantage of an external modem is that external modems can be easily moved from one computer to another and the lights on the modem itself can aid in the diagnosis of any problems. Furthermore, an external modem can easily be reset by turning it off whereas an internal modem cannot just be turned off without turning off the system.

Fax Modems

These allow you to send and receive faxes. The fax part of the modem sends/receives data that is interpreted as a picture. One can create documents on a computer (resumes, papers, thesis proposals) and send them directly from the application to a fax machine or a computer with a fax modem. Instead of sending the print job to the printer you're sending it to the fax modem. The information is sent over telephone lines to a remote fax machine or fax modem. That is, any file in your computer can be sent to a fax machine or another computer with a fax modem. One can fax résumés, thesis proposals, etc. to a fax machine in a matter of minutes.

Intelligent Modems

Intelligent modems are more expensive modems that contain internal read only memory (ROM) coding and microprocessor chips to provide sophisticated communications protocols and diagnostic checking within the modem itself. Some of these intelligent modems not only perform digital-to-analog conversation but also operate as multiplexers, security restrictor devices, encryption devices, error detection and retransmission devices.

Short Haul Modems

Short Haul modems provide transmission of data at medium to high rate in either direction (full duplex). They are also used in the control of remote devices in their network. In a short haul modem, you use your own wire pair cable to transmit direct electrical signals. This type of modem is also called line driver. This device consists of a box approximately the size of a cigarette pack. Because short haul modems do not require an external power source, getting their power through the serial port, they sometimes are referred as modem eliminators. These modems are generally used in business and not the private user. The farther the signal has to travel; the amount of data is reduced. Generally short hauls are synchronous, full duplex.

For Example:

Telebyte M 431: V.35 Interface, full duplex on single twisted pair, up to 128 Kbps, built-in diagnostics, bi-directional handshake, 9600 bps at 3.6 miles, and cost: $575.

CIM M 74: V.35 Interface, full duplex on two twisted pair, up to 19.2 Kbps, bi-directional handshake, .6 Kbps at 10 miles, and cost: $155.

Wireless Modems

Wireless modems transmit the data signals through the air instead of by using a cable. They sometimes are called radio frequency modem. This type of modem is designed to work with cellular technology, and wireless local area networks. Wireless modems use two types of transmission to transfer their data; radio transceivers and infrared (IR). Radio transceiver modems have three ways of transmitting data; transceiver-transceiver, transceiver-satellite-transceiver, and cellular phone. Radio transceiver-transceiver can be used as point-point or point-multipoint operation and generally transmit at the frequency of 900 MHz. Radio transceiver modems have advantages and disadvantages when compared with a wired modem.

Advantages are: no costly wiring to maintain, no costly wiring to install, no down time waiting on a connection, can connect through a hub to communicate with a wired modem, and no fees to pay for a leased line.

Disadvantages are: higher initial cost of equipment ($500-$899), with satellite and cellular modems higher operating cost, security harder to maintain, short transmission range; generally 20 miles unless you use repeaters, data transmission speed goes down as range increases, and loss of data.

RIL-089

As you can see there are good and bad reasons for transceiver modems. IR modems can be used as point-point or with the use of a conversion box, can be connected through a telephone line. IR is very limited in transmission range, 35ft or less. The biggest advantage that IR has is with its use by the person who travels and needs to communicate with their home base.

Cable Modems

When cable was first introduced, it was designed for rural areas with poor reception. It is now an enhanced network offering over 500 site of video, music entertainment, and interactive services (web TV). Due to its huge bandwidth, cable is able to deliver a greater amount of data at much faster speeds than analog or Integrated Services Digital Network (ISDN) device. With these kind of speeds and the data size, we have new high-end applications available, such as real-time video, this technology is not quite fully born yet and is still in the development stage.

Lacks of standards are hindering development of an acceptable system. Most cable systems are a one-way system and were not setup for two-way communication needed for modems. According to the research done by the Yankee Group, more than 90% of the cable systems are one-way, making the system impossible for these systems to use Radio Frequency (RF) return path.

In order for one-way systems to take part in this data capability, the smart solution would be to create a hybrid system that used the cable bandwidth for downstream delivery and an analog or ISDN connection for the upstream. This is just what US Robotics and Zenith has done. By working together and using Zenith's "HomeWorks Universal" cable modem and US Robotics "Total Control Enterprise Network Hub" at the cable headend. This system will use US Robotics Total Control Enterprise Network hub to send upstream request through the use of ISDN or standard analog connections. The high-speed downstream data will be supplied by Zenith's Metro Access cable modem system.

The Home Works Universal modem uses an RS-232 cable connecting the cable modem to the US Robotics or legacy modem of ISDN terminal adapter in the subscriber's home. The cable modem sends standard AT commands to the analog device or the ISDN adapter. This system will be software upgradable as the needs arise and will be optimized by the use of US Robotics Quick Connect.

Zenith's Home Works Universal cable modem will accomplish the following tasks:

- Confirm the attached analog modem or ISDN adapter is working.
- Will identify modem or terminal adapter type.
- Decide what speed can be supported with the devices, up to 128 Kbps.
- Will initialize the modem or terminal adapter.
- Download the dial string that is programmed in the cable modem.

- Disconnect the call when the session is complete.

Network management is available, using standard SNMP commands to the modem. Security is provided in several levels in the cable system. The cable modem is assigned a password. This password is required for the cable modem to pass data up the dial-up link. Invalid users will be denied access to the network. By filtering in the cable modem passes only data packets that are addressed to that computer. This prevents users from snooping the network and viewing others network packets.

This is a promising and exciting system available for data transfer. This hybrid system will allow one-way lines to be used until more two-way lines are available. This new telco return path system is combining the strengths of the Zenith cable modem and US Robotics dial-up technology to become an industry standard and exposing many subscribers to a faster data transfer and internet access.

1.6 Summary

In this unit you have learnt about layered network architecture, Review of ISO-OSI Model. Data communication techniques: Pulse Code Modulation (PCM) and Data modems.

- In layered architecture of Network Model, one whole network process is divided into small tasks. Each small task is then assigned to a particular layer which works dedicatedly to process the task only. Every layer does only specific work.
- ISO stands for **International organization of Standardization**. This is called a model for **Open System Interconnection** (OSI) and is commonly known as OSI model. The ISO-OSI model is a seven layer architecture. It defines seven layers or levels in a complete communication system. These layers are Physical Layer, Data Link Layer, Network Layer, Transport Layer, Session Layer, Presentation Layer and Application Layer.
- In data communication techniques there are two types of transmission i.e. Asynchronous and Synchronous Transmission.
- A combined device for modulation and demodulation, for example, between the digital data of a computer and the analogue signal of a telephone line.

1.7 Review Questions

Q1. What do you mean by layered architecture? Explain in detail.

Q2. Define ISO-OSI model in depth.

Q3. What is the difference between Asynchronous and Synchronous Transmission?

Q4. What is a modem? Define its types.

RIL-089

UNIT-II

Multiplexing Techniques & Transmission Media

Structure

2.0 Introduction

In this unit we will focus on Multiplexing Techniques & Transmission Media. In this unit there are four sections. In the first section we will tell about multiplexing techniques. As you have read earlier about multiplexing techniques that **Multiplexing** is a technique in which several message signals are combined into a composite signal for transmission over a common channel.

These signals to be transmitted over the common channel must be kept apart so that they do not interfere with each other, and hence they can be separated easily at the receiver end. Basically, multiplexing is of two types such as Frequency division multiplexing (FDM) and Time division multiplexing (TDM). In Sec 2.2 you will learn about transmission media. In this section you will learn about wires, cables, radio links, satellite links and fiber-optic links. In Sec. 2.3 and 2.4 you will find summary and review questions respectively.

Objectives

After studying this unit you should be able to:

- ❖ Define multiplexing
- ❖ Describe transmission media and its types

2.1 Multiplexing Techniques

Multiplexer

A Multiplexer is a device that allows one of several analog or digital input signals which are to be selected and transmits the input that is selected into a single medium. Multiplexer is also known as Data Selector. A multiplexer of 2n inputs has n select lines that will be used to select input line to send to the output. Multiplexer is abbreviated as Mux. MUX sends digital or analog signals at higher speed on a single line in one shared device. It recovers the separate signals at the receiving end. The Multiplexer boosts or amplifies the information that later transferred over network within a particular bandwidth and time. This article gives an overview of what is multiplexer and types of multiplexer.

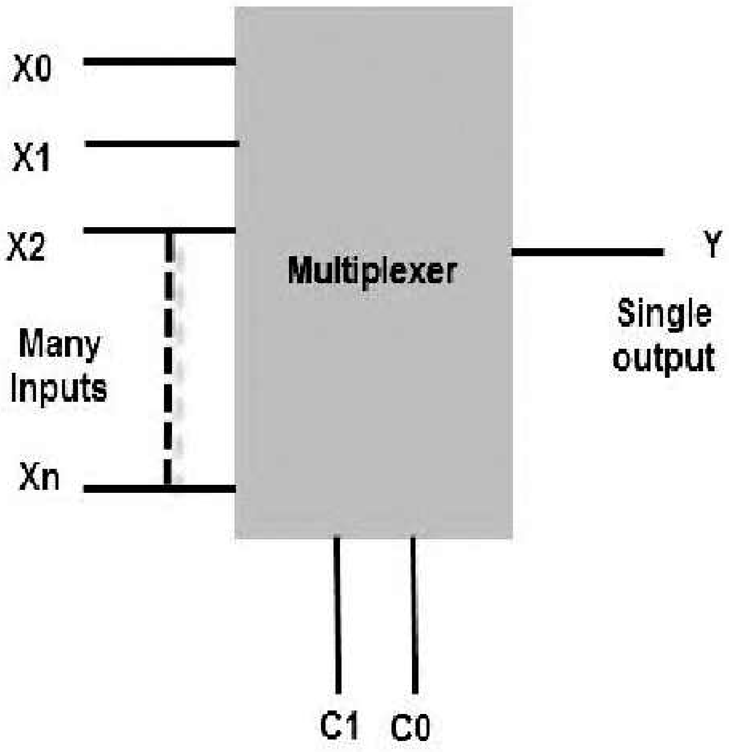


Figure 2.0 Multiplexer

The Multiplexer acts as a multiple-input and single-output switch. Multiple signals share one device or transmission conductor such as a copper wire or fiber optic cable. In telecommunications, the analog or digital signals transmitted on several communication channels by a multiplex method. These signals are single-output higher-speed signals. A 4-to-1 multiplexer contains four input signals and 2-to-1 multiplexer has two input signals and one output signal.

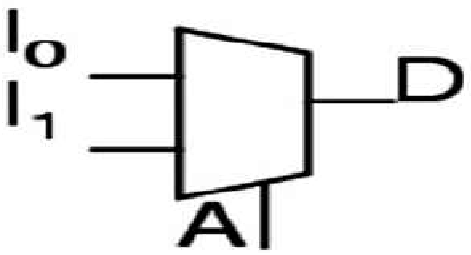
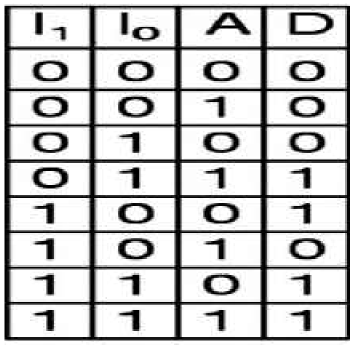


Figure 2.1 Multiplexer Input/ Output

Schematic Symbol for Multiplexer



I_1	I_0	A	D
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Figure 2.2 Truth Table for 2 to 1 Multiplexer

Multiplexers are also extended with same name conventions as DE multiplexers. A 4 to 1 multiplexer circuit is as below

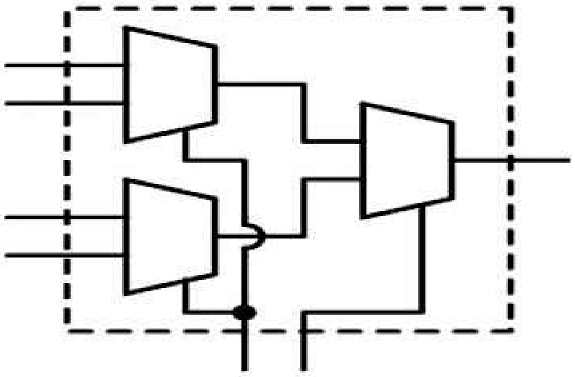


Figure 2.3- 4 to 1 Multiplexer Circuit

Multiplexing

The technique of transmitting multiple signals over a single medium is defined as Multiplexing. It is a technique showed at physical layer of OSI model.

The different types of multiplexing technologies are as below

- ❖ Frequency Division Multiplexing (FDM)
- ❖ Time Division Multiplexing (TDM)
- ❖ Synchronous TDM

❖ Asynchronous TDM

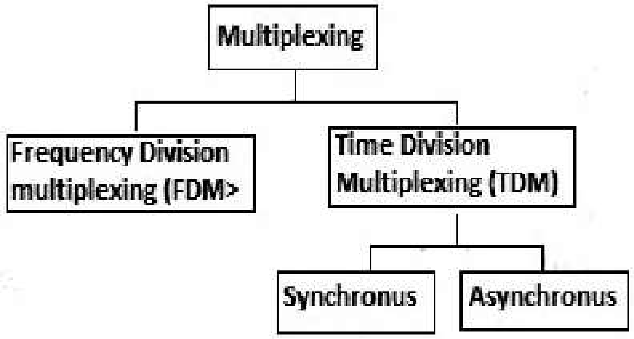

Multiplexing
Frequency Division multiplexing (FDM>
Time Division Multiplexing (TDM)
Synchronus
Asynchronus

Figure 2.4- types of multiplexing

2.1.1 Frequency Division Multiplexing

In FDM, signals generated by each sending device modulate different carrier frequencies. These modulated signals are then combined into a single composite signal that can be transported by the link. The carrier frequencies have to be different enough to accommodate the modulation and demodulation signals. The figure No.2.5 illustrates the FDM multiplexing process. The multiplexing process starts by applying amplitude modulation into each signal by using different carrier frequencies as/i and/j. Then both signals are combined

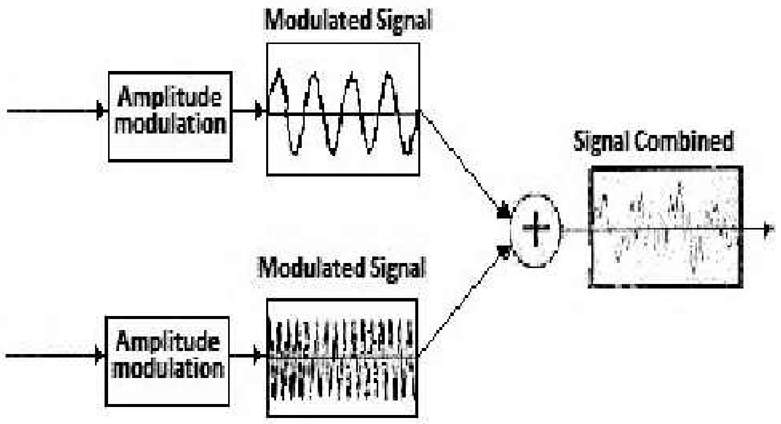


Figure 2.5 FDM

In de-multiplexing process, we use filters to decompose the multiplexed signal into its constituent component signals. Then each signal is passed to an amplitude demodulation process to separate the carrier signal from the message signal. Then, the message signal is sent to the waiting receiver. The process of de-multiplexing is shown in the figure 2.6.

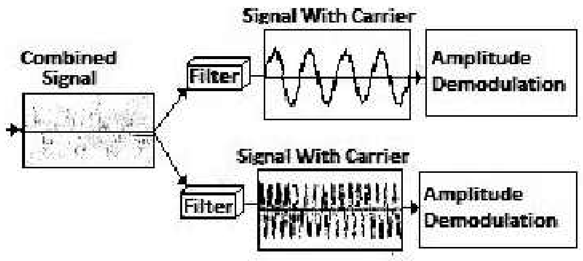


Figure2.6- De-multiplexing process

TIME-DIVISION MULTIPLEXING (TDM)

In the time-division multiplexing, multiple transmissions can occupy a single link by subdividing them and interleaving the portions. We say that TDM is a round robin use of a frequency. TDM can be implemented in two ways: synchronous TDM and asynchronous TDM.

Synchronous TDM

The multiplexer allocates exactly the same time slot to each device at all times, whether or not a device has anything to transmit. Time slot 1, for example, is assigned to device 1 alone and cannot be used by any other device as shown in the figure 2.7

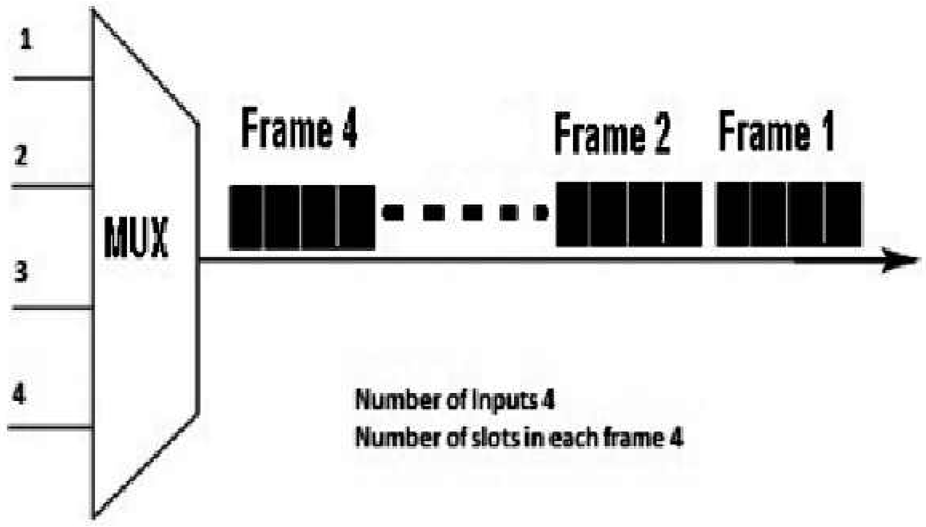


Figure 2.7- Synchronous TDM

Frames: In synchronous TDM, a frame consists of one complete cycle of time slots. Thus the number of slots in frame is equal to the number of inputs. Figures 2.8 (a & b) below are an example of how/the synchronous TDM works.

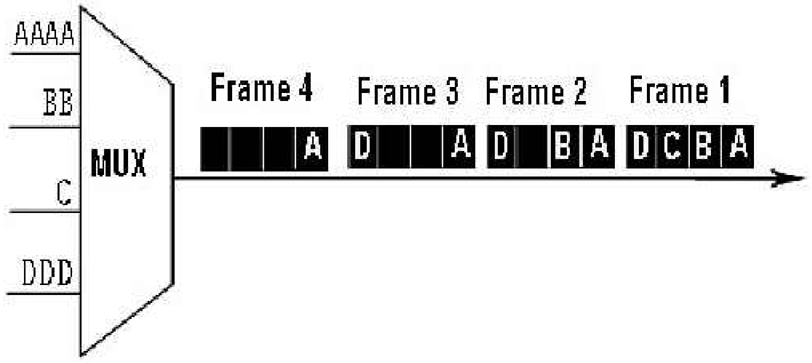


Figure 2.8- (a) Working of Synchronous TDM

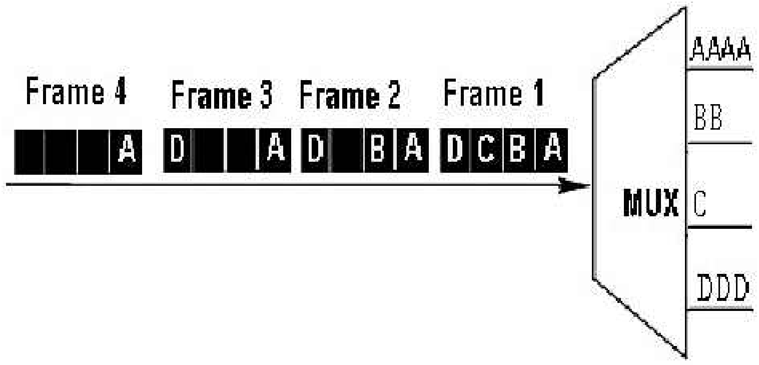


Figure 2.8 (b) - Working of Synchronous TDM

Asynchronous TDM

In asynchronous TDM, each slot in a frame is not dedicated to the fix device. Each slot contains an index of the device to be sent to and a message. Thus, the number of slots in a frame is not necessary to be equal to the number of input devices. More than one slot in a frame can be allocated for an input device Asynchronous TDM allows maximization the link. It allows a number of lower speed input lines to be multiplexed to a single higher speed line. As shown in the figure 2.9

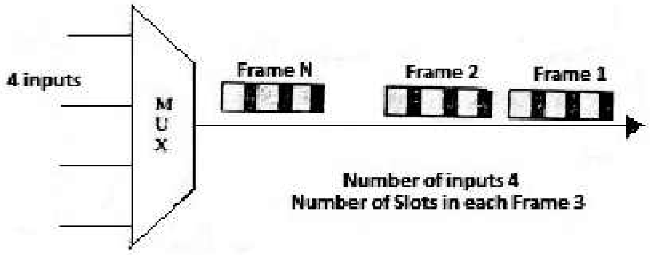


Figure-2.9 Asynchronous TDM

In Asynchronous TDM, a frame contains a fix number of time slots. Each slot has an index of which device to receive.

Figure 2.10 (a & b) are examples of how Asynchronous TDM works

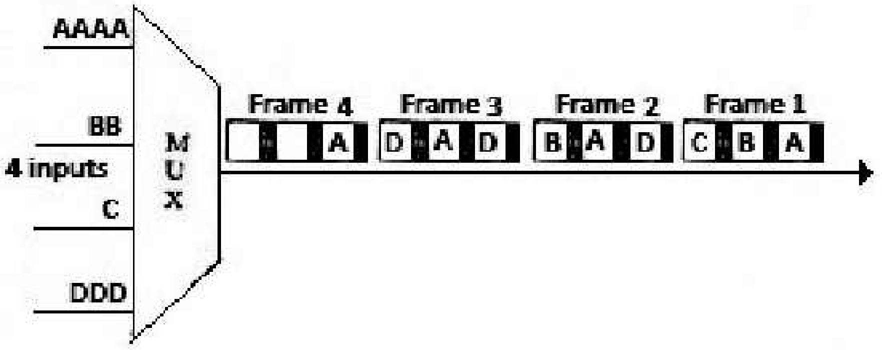


Figure 2.10 (a) - Working of Asynchronous TDM

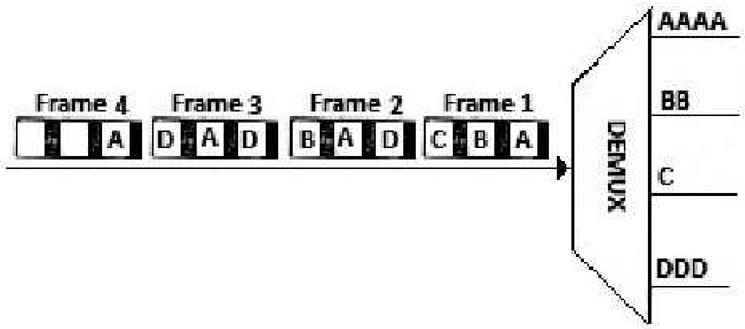


Figure 2.10 (b) - Working of Asynchronous TDM

Check your progress

Q1. Define FDM and TDM.

Q2. Define the types of TDM.

2.2 Transmission Media

Transmission media is a pathway that carries the information from sender to receiver. We use different types of cables or waves to transmit data. Data is transmitted normally through electrical or electromagnetic signals.

An electrical signal is in the form of current. An electromagnetic signal is series of electromagnetic energy pulses at various frequencies. These signals can be transmitted through copper wires, optical fibers, atmosphere, water and vacuum Different Medias have different properties like bandwidth, delay, cost and ease of installation and maintenance. Transmission media is also called Communication channel.

Types of Transmission Media

Transmission media is broadly classified into two groups.

1. Wired or Guided Media or Bound Transmission Media
2. Wireless or Unguided Media or Unbound Transmission Media

Wired or Guided Media or Bound Transmission Media: Bound transmission media are the cables that are tangible or have physical existence and are limited by the physical geography. Popular bound transmission media in use are twisted pair cable, co-axial cable and fiber optical cable. Each of them has its own characteristics like transmission speed, effect of noise, physical appearance, cost etc.

Wireless or Unguided Media or Unbound Transmission Media: Unbound transmission media are the ways of transmitting data without using any cables. These media are not bounded by physical geography. This type of transmission is called Wireless **communication.** Nowadays wireless communication is becoming popular. Wireless LANs are being installed in office and college campuses. This transmission uses Microwave, Radio wave, Infra-red are some of popular unbound transmission media.

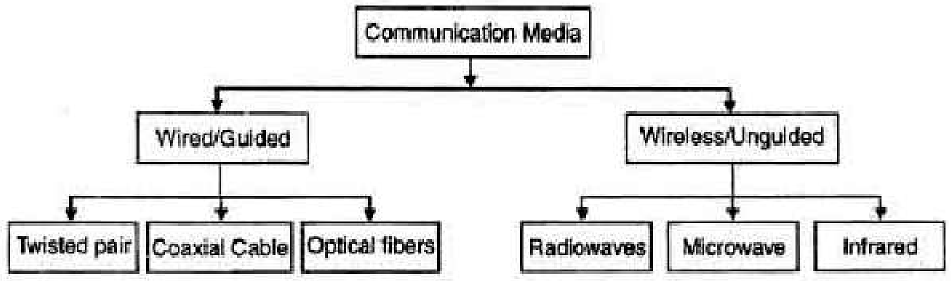


Figure 2.11 Types of Communication Media

The data transmission capabilities of various Medias vary differently depending upon the various factors. These factors are:

1. **Bandwidth**. It refers to the data carrying capacity of a channel or medium. Higher bandwidth communication channels support higher data rates.

2. **Radiation**. It refers to the leakage of signal from the medium due to undesirable electrical characteristics of the medium.

3. **Noise Absorption**. It refers to the susceptibility of the media to external electrical noise that can cause distortion of data signal.

4. **Attenuation.** It refers to loss of energy as signal propagates outwards. The amount of energy lost depends on frequency. Radiations and physical characteristics of media contribute to attenuation.

2.2.1 Wires

Twisted Pair Cable

This cable is the most commonly used and is cheaper than others. It is lightweight, cheap, can be installed easily, and they support many different types of network. Some important points:

- Its frequency range is 0 to 3.5 kHz.
- Typical attenuation is 0.2 dB/Km @ 1kHz.
- Typical delay is 50 µs/km.
- Repeater spacing is 2km.

Twisted Pair is of two types:

- **Unshielded Twisted Pair (UTP)**
- **Shielded Twisted Pair (STP)**

Unshielded Twisted Pair Cable

It is the most common type of telecommunication when compared with Shielded Twisted Pair Cable which consists of two conductors usually copper, each with its own colour plastic insulator. Identification is the reason behind coloured plastic insulation.

UTP cables consist of 2 or 4 pairs of twisted cable. Cable with 2 pair use **RJ-11** connector and 4 pair cable use**RJ-45** connector.

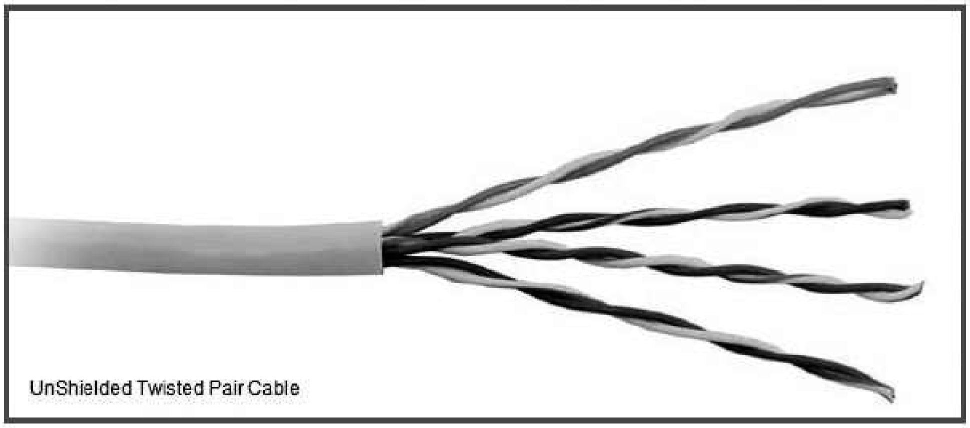


Figure 2.12- Unshielded Twisted Pair Cable

Advantages

- Installation is easy
- Flexible
- Cheap
- It has high speed capacity,

- 100 meter limit
- Higher grades of UTP are used in LAN technologies like Ethernet.

It consists of two insulating copper wires (1mm thick). The wires are twisted together in a helical form to reduce electrical interference from similar pair.

Disadvantages

- Bandwidth is low when compared with Coaxial Cable
- Provides less protection from interference.

Shielded Twisted Pair Cable

This cable has a metal foil or braided-mesh covering which encases each pair of insulated conductors. Electromagnetic noise penetration is prevented by metal casing. Shielding also eliminates crosstalk It has same attenuation as unshielded twisted pair. It is faster the unshielded and coaxial cable. It is more expensive than coaxial and unshielded twisted pair.

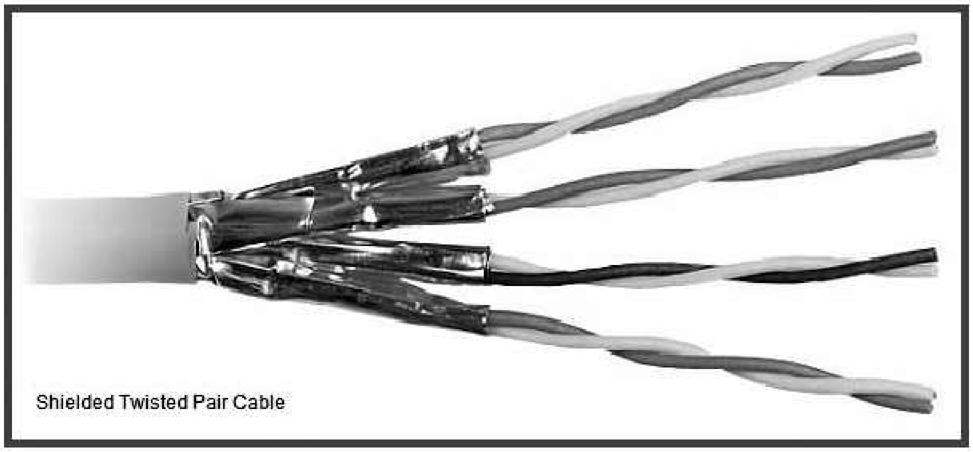


Figure 2.13- Shielded Twisted Pair Cable

Advantages

- Easy to install
- Performance is adequate
- Can be used for Analog or Digital transmission
- Increases the signaling rate
- Higher capacity than unshielded twisted pair
- Eliminates crosstalk

Disadvantages

- ❖ Difficult to manufacture
- ❖ Heavy

2.2.2 Cables

Coaxial Cable

Coaxial is called by this name because it contains two conductors that are parallel to each other. Copper is used in this as center conductor which can be a solid wire or a standard one. It is surrounded by PVC installation, a sheath which is encased in an outer conductor of metal foil, barid or both.

Outer metallic wrapping is used as a shield against noise and as the second conductor which completes the circuit. The outer conductor is also encased in an insulating sheath. The outermost part is the plastic cover which protects the whole cable.

Here the most common coaxial standards.

- ❖ 50-Ohm RG-7 or RG-11 : used with thick Ethernet.
- ❖ 50-Ohm RG-58 : used with thin Ethernet
- ❖ 75-Ohm RG-59 : used with cable television
- ❖ 93-Ohm RG-62 : used with ARCNET.

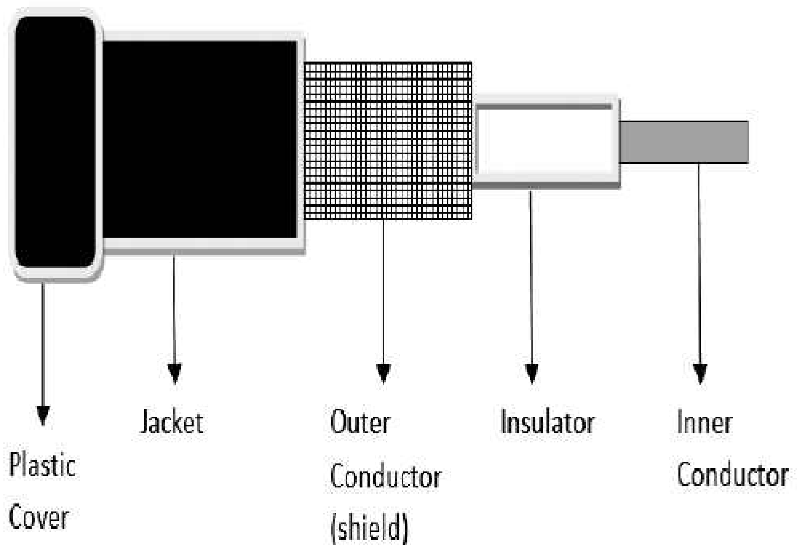


Figure 2.14- Coaxial Cable

There are two types of Coaxial cables:

BaseBand

This is a 50 ohm (Ω) coaxial cable which is used for digital transmission. It is mostly used for LAN's. Baseband transmits a single signal at a time with very high speed. The major drawback is that it needs amplification after every 1000 feet.

BroadBand

This uses analog transmission on standard cable television cabling. It transmits several simultaneous signal using different frequencies. It covers large area when compared with Baseband Coaxial Cable.

Advantages:

- Bandwidth is high
- Used in long distance telephone lines.
- Transmits digital signals at a very high rate of 10Mbps.
- Much higher noise immunity
- Data transmission without distortion.
- The can span to longer distance at higher speeds as they have better shielding when compared to twisted pair cable

Disadvantages:

- Single cable failure can fail the entire network.
- Difficult to install and expensive when compared with twisted pair.
- If the shield is imperfect, it can lead to grounded loop.

Fiber Optic Cable

These are similar to coaxial cable. It uses electric signals to transmit data. At the center is the glass core through which light propagates.

In multimode fibers, the core is 50microns, and In single mode fibres, the thickness is 8 to 10 microns.

The core in fiber optic cable is surrounded by glass cladding with lower index of refraction as compared to core to keep all the light in core. This is covered with a thin plastic jacket to protect the cladding. The fibers are grouped together in bundles protected by an outer shield.

Fiber optic cable has bandwidth more than **2 gbps (Gigabytes per Second)**

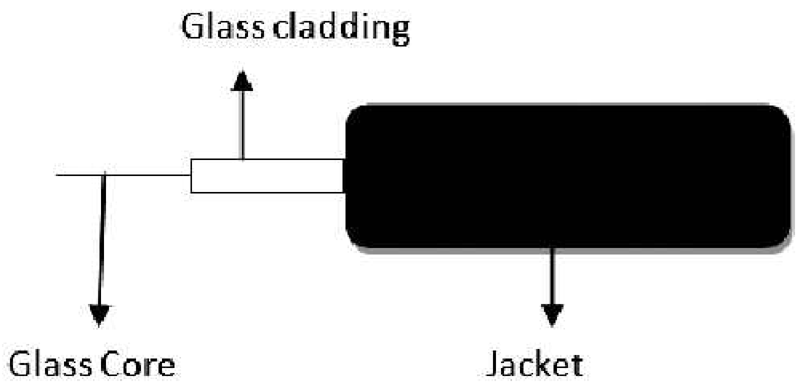


Figure 2.15- Fiber Optic Cable

Advantages:

- Provides high quality transmission of signals at very high speed.
- These are not affected by electromagnetic interference, so noise and distortion is very less.
- Used for both analog and digital signals.

Disadvantages:

- It is expensive
- Difficult to install.
- Maintenance is expensive and difficult.
- Do not allow complete routing of light signals.

2.2.3 Radio Transmission

Its frequency is between 10 kHz to 1GHz. It is simple to install and has high attenuation. These waves are used for multicast communications.

Types of Propogation

Radio Transmission utilizes different types of propogation:

- **Troposphere:** The lowest portion of earth's atmosphere extending outward approximately 30 miles from the earth's surface. Clouds, jet planes, wind is found here.
- **Ionosphere:** The layer of the atmosphere above troposphere, but below space. Contains electrically charged particles.

2.2.4 Satellite Links

This is a microwave relay station which is placed in outer space. The satellites are launched either by rockets or space shuttles carry them.

These are positioned 3600KM above the equator with an orbit speed that exactly matches the rotation speed of the earth. As the satellite is positioned in a geo-synchronous orbit, it is stationery relative to earth and always stays over the same point on the ground. This is usually done to allow ground stations to aim antenna at a fixed point in the sky.

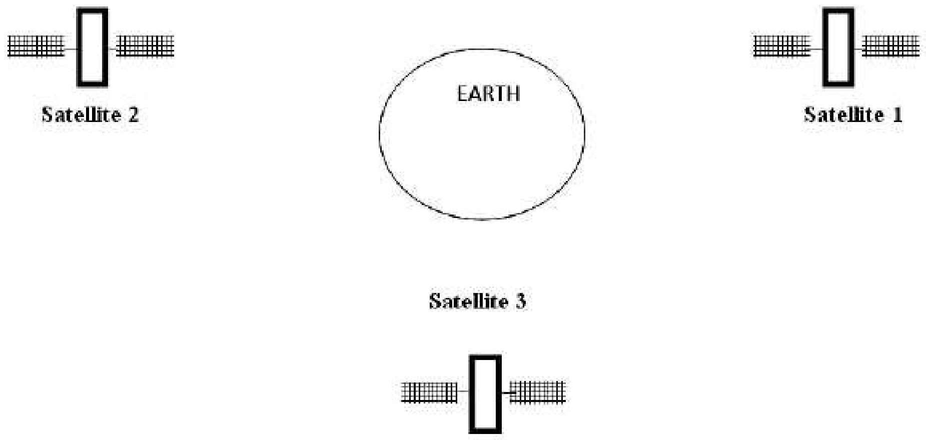


Figure 2.16- Satellite Links

Features of Satellite Microwave:

- Bandwidth capacity depends on the frequency used.
- Satellite microwave deployment for orbiting satellite is difficult.

Advantages of Satellite Microwave:

- Transmitting station can receive back its own transmission and check whether the satellite has transmitted information correctly.
- A single microwave relay station which is visible from any point.

Disadvantages of Satellite Microwave:

- Satellite manufacturing cost is very high
- Cost of launching satellite is very expensive
- Transmission highly depends on whether conditions, it can go down in bad weather

2.2.5 Fiber Optic Links

A fiber-optic link (or *fiber channel*) is a part of an optical fiber communications system which provides a data connection between two points (*point-to-point connection*). It essentially consists of a data transmitter, a transmission fiber (possibly with built-in fiber amplifiers), and a receiver. These components, which are mostly based on fiber optics, are explained in the following, beginning with a simple single-channel system.

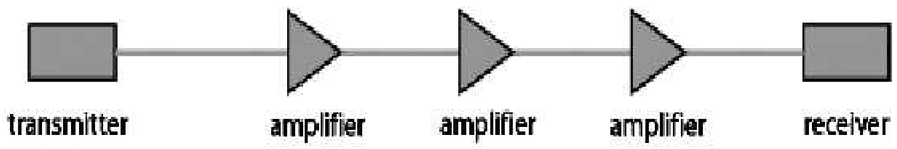


Figure-2.17 Optical Fiber Links

Transmission Formats

In most cases, the data transmission is digital, making the system very versatile and relatively insensitive, e.g. to nonlinear distortions. There are various different *modulation formats*, i.e., different methods for encoding the information. For example, a simple non-return-to-zero (NRZ) format transmits subsequent bits by sending a high or low optical power value, with no gaps between the bits, and extra means for synchronization. In contrast, a return-to-zero (RZ) format is easily self-synchronizing by returning to a rest state after each bit, but it requires a higher optical transmission bandwidth for the same data rate. Apart from details of the equipment and the optical bandwidth required (related to the *modulation efficiency*), different transmission formats also differ in terms of their sensitivity e.g. to noise influences and cross-talk.

Transmitter

The transmitter converts the electronic input signal into a modulated light beam. The information may be encoded e.g. via the optical power (intensity), optical phase or polarization; intensity modulation is most common. The optical wavelength is typically in one of the so-called *telecom windows*

A typical transmitter is based on a single-mode laser diode (normally a VCSEL or a DFB laser), which may either be directly modulated via its drive current (DML = *directly modulated laser*), or with an external optical modulator (e.g. an electro absorption or Mach–Zehnder modulator). Direct modulation is the simpler option, and can work at data rates of 10 Gbit/s or even higher. However, the varying carrier density in the laser diode then leads to a varying instantaneous frequency and thus to signal distortions in the form of a chirp. Particularly for long transmission distances, this makes the signal more sensitive to the influence of chromatic dispersion. Therefore, external modulation is usually preferred for the combination of high data rates (e.g. 10 or 40 Gbit/s) with long transmission distances (many kilometers). The laser can then operate in continuous-wave mode, and signal distortions are minimized.

For even higher single-channel data rates, time division multiplexing may be employed, where e.g. four channels with 40 Gbit/s are temporally interleaved to obtain a total rate of 160 Gbit/s. For high data rates with return-to-zero formats, it can be advantageous to use a pulsed source (e.g. a mode-locked laser emitting soliton pulses) combined with an intensity modulator. This reduces the bandwidth demands on the modulator, as it does not matter how the modulator's transmittivity evolves between the pulses.

For high data rates, the transmitter needs to meet a number of requirements. In particular, it is important to achieve a high extinction ratio (low pedestal pulses), a low timing jitter, low intensity noise, and a precisely controlled clock rate. Of course, a data transmitter should operate stably and reliably with minimum operator intervention.

In simple cases, a light-emitting diode (LED) is used in the transmitter, but due to the poor spatial coherence this requires the use of multimode. The transmission rate or distance is then restricted due to intermodal dispersion; for a longer product, single are required. For short distances, several hundred Mbit/s are possible.

Transmission Fiber

The transmission fiber is usually a single-mode fiber in the case of medium or long-distance transmission, but can also be a multimode fiber for short distances. In the latter case, intermodal dispersion can limit the transmission distance or bit rate.

Long-range broadband fiber channels can contain fiber amplifiers at certain points (*lumped amplifiers*) to prevent the power level from dropping to too low a level. Alternatively, it is possible to use a distributed amplifier, realized with the transmission fiber itself, by injecting an additional powerful pump beam (typically from the receiver end) which generates Raman gain. In addition, means for dispersion compensation (counteracting the effects of chromatic dispersion of the fiber) and for *signal regeneration* may be employed. The latter means that not only the power level but also the signal quality (e.g. pulse width and timing) is restored. This can be achieved either with purely optical signal processing, or by detecting the signal electronically, applying some optical signal processing, and resending the signal.

Receiver

The receiver contains some type of fast photo detector, normally a photodiode, and suitable high-speed electronics for amplifying the weak signal (e.g. with a Trans impedance amplifier) and extracting the digital (or sometimes analog) data. For high data rates, circuitry for electronic dispersion compensation may be included.

Avalanche photodiodes can be used for particularly high sensitivity. The sensitivity of the receiver is limited by noise, normally of electronic origin. Note, however, that the optical signal itself is accompanied by optical noise, such as amplifier noise. Such optical noise introduces limitations which cannot be removed with any receiver design. Noise effects are discussed below in more detail.

Bidirectional Transmission

So-called *full duplex links* provide a data connection in both directions. These may simply be based on separate optical fibers, or work with a single fiber. The latter can be realized e.g. by using fiber-optic beam splitters at each end to connect a transmitter and a receiver. However, the need for bidirectional operation introduces various trade-offs, which in some cases (e.g. for very high data rates) make a system with two separate fibers preferable.

Multiplexing

A typical single-channel system for long-haul transmission has a transmission capacity of e.g. 2.5 or 10 Gbit/s; higher data rates of 40 Gbit/s or even 160 Gbit/s may be used in the future. For higher data rates, several data channels can be multiplexed (combined), transmitted through the fiber, and separated again for detection.

The most common technique is *wavelength division multiplexing* (WDM). Here, different center wavelengths are assigned to different data channels. It is possible to combine even hundreds of channels in that way (*DWDM = dense WDM*), but *coarse WDM* with a moderate number of channels is often preferred in order to keep the system simpler. The main challenges are to suppress channel cross-talk via nonlinearities, to balance the channel powers (e.g. with gain-flattened fiber amplifiers), and to simplify the systems.

Another approach is *time division multiplexing*, where several input channels are combined by nesting in the time domain, and solitons are often used to ensure that the sent ultra short pulses stay cleanly separated even at small pulse-to-pulse spacing.

For short distances, for example for connections within data centers, it can be simpler to use ribbon fiber cables with multiple fibers and corresponding numbers of transmitters and receivers. However, a disadvantage of that approach is that the cables become bulkier.

Active Optical Cables

For short transmission distances, so-called *active optical cables* (AOC) can be used, where a transmitter and a receiver (together with corresponding electronics) are rigidly attached to the ends of an optical fiber cable. Common electrical interfaces such as USB or HDMI ports are available, so that the use of such an active optical cable is essentially the same as that of an electrical cable, while offering advantages like reduced diameter and weight and also a larger possible transmission distance.

2.3 Summary

In this unit you have learnt about different Multiplexing techniques like Frequency-Division, Time-Division. You also learnt about Transmission

Media and its types like Wires, Cables, Radio Links, Satellite Links and Fiber-Optic Links.

- ❖ **Multiplexing** is a method by which multiple analog or digital signals are combined into one signal over a shared medium.
- ❖ FDM is inherently an analog technology. FDM achieves the combining of several signals into one medium by sending signals in several distinct frequency ranges over a single medium.
- ❖ TDM) is a digital (or in rare cases, analog) technology which uses time, instead of space or frequency, to separate the different data streams.
- ❖ **Transmission media** is a pathway that carries the information from sender to receiver. We use different types of cables or waves to transmit data. Data is transmitted normally through electrical or electromagnetic signals.

2.4 Review Questions

Q1. Explain the term multiplexing. How many types of multiplexing techniques available in computer network?

Q2. What is transmission media? Why we use transmission media?

Q3. Explain following:

(a) Wires
(b) Cables
(c) Radio Links
(d) Satellite Links

UNIT-III

Asynchronous Transfer Mode (ATM)

Structure

3.0 Introduction

In this unit we will focus on Asynchronous Transfer Mode (ATM). There are five sections in this unit. In the first section you will learn about Asynchronous Transfer Mode (ATM). In this section you will also learn cells of ATM and header and cell formats of ATM. In the second section i.e. Sec. 3.2 you will know about layers in ATM. There are three main layers of ATM i.e. AAL, ATM and Physical layer. There are four sub-layer in ATM like CS, SAR, TC, and PMD. You will find all layers in Sec. 3.2. In the next section i.e. 3.3 you will learn about different classes of ATM like Class 1, 2, 3 and 4. In Sec. 3.3 and 3.4 you will find summary and review questions respectively.

Objectives

After studying this unit you should be able to:

- Define ATM
- Describe cells and header of ATM
- Express different layers of ATM
- Define different classes of ATM

3.1 Asynchronous Transfer Mode (ATM)

Asynchronous transfer mode (ATM), also known as cell relay, is similar in concept to frame relay. Both frame relay and ATM take advantage of the reliability and fidelity of modern digital facilities to provide faster packet switching than X.25. ATM is even more streamlined than frame relay in its functionality, and can support data rates several orders of magnitude greater than frame relay. The "asynchronous" in ATM means ATM devices do not send and receive information at fixed speeds or using a timer, but instead negotiate transmission speeds based on hardware and information flow reliability. The "transfer mode" in ATM refers to the fixed-size cell structure used for packaging information.

Through the efforts of the ATM Forum (jointly founded in 1991 by Cisco Systems, NET/ADAPTIVE, Northern Telecom, and Sprint), ATM is capable of transferring voice, video, and data through private networks and across public networks. ATM continues to evolve today as the various standards groups finalize specifications that allow interoperability among the equipment produced by vendors in the public and private networking industries.

Evolution

Asynchronous Transfer Mode (ATM) represents a relatively recently developed communications technology designed to overcome the constraints associated with traditional, and for the most part separate, voice and data networks. ATM has its roots in the work of a CCITT (now known as ITU-T) study group formed to develop broadband ISDN standards during the mid-1980s. In 1988, a cell switching technology was chosen as the foundation for broadband ISDN, and in 1991, the ATM Forum was founded.

The ATM Forum represents an international consortium of public and private equipment vendors, data communications and telecommunications service providers, consultants, and end users established to promote the implementation of ATM. _To accomplish this goal, the ATM Forum develops standards with the ITU and other standards organizations.

The first ATM Forum standard was released in 1992. Various ATM Forum working groups are busy defining additional standards required to enable ATM to provide a communications capability for the wide range of LAN and WAN transmission schemes it is designed to support. This standardization effort will probably remain in effect for a considerable period due to the comprehensive design goal of the technology, which was developed to support voice, data, and video on both local and wide area networks.

The Rationale and Underlying Technology

ATM can be considered to represent a unifying technology because it was designed to transport voice, data, and video (including graphics images) on both local and wide area networks. Until the development of ATM, networks were normally developed based on the type of data to be transported. Thus, circuit-switched networks, which included the public switched telephone network and high-speed digital transmission facilities, were primarily used to transport delay-sensitive information, such as voice and video. In comparison, on packet-based networks, such as X.25 and

Frame Relay, information can tolerate a degree of delay. Network users can select a networking technology to satisfy a specific communications application, but most organizations support a mixture of applications. Thus, most organizations are forced to operate multiple networks, resulting in a degree of inefficiency and escalating communications costs. By combining the features from both technologies, ATM enables a single network to support voice, data, and video.

ATM is designed to be scalable, enabling its 53-byte cell to be transported from LAN to LAN via WAN, as well as for use on public and private wide area networks at a range of operating rates. On LANs, ATM support is currently offered at 25 and 155Mbps, whereas access to WAN-based ATM carrier networks can occur at T1 (1.544Mbps), at T3 (45Mbps), or via different SONET facilities at data rates up to 622Gbps, all based on the transportation of 53-byte cells. A key to ATM's ubiquitous transmission capability is its fixed 53-byte cell length, which remains static regardless of changes in media, operating rates, or framing.

The use of a fixed-length cell enables low-cost hardware to be developed to perform required cell switching based on the contents of the cell header, without requiring more complex and costly software. Thus, ATM can be considered to represent a unifying technology that will eventually become very economical to implement when its development expenses are amortized over the growing production cycle of ATM communications equipment.

Although many organizations merged voice and data through the use of multiplexers onto a common circuit, this type of merger is typically not end-to-end. For example, traffic from a router connected to a LAN might be fed into a port on a high-speed multiplexer with another connection to the multiplexer from the company PBX. Although this type of multiplexing enables a common WAN circuit to be used for voice and data, it represents an interim and partial solution to the expense associated with operating separate voice and data networks. In addition, the emergence of multimedia applications requiring the transmission of video can wreak havoc with existing LANs and WANs due to their requirement for high bandwidth for short periods. ATM represents an emerging technology designed to provide support for bandwidth-on-demand applications, such as video, as well as voice and data. A comparison of the key features associated with each technology can give you an appreciation for ATM technology in comparison to conventional data communications- and telecommunications-based technology. Table 3.1 compares nine features of data communications and telecommunications networks with those of an ATM network.

In a data communications environment, the network can range in scope from a token-ring LAN to an X.25 or Frame Relay WAN. Thus, although some features are common to both LAN and WAN environments, there is also some variability. In general, a data communications network transports data by using variable-length packets. Although many WAN protocols are connection-oriented, some are connectionless. Similarly, many LAN protocols are connectionless, whereas others are connection-oriented. Because data communications networks were designed to transport files, records, and screens of data, transmission delay or latency, if small, does not adversely affect users. In comparison, in a telecommunications network, a similar amount of latency that is acceptable on a data network could wreak havoc with a telephone conversation. Recognizing the differences among voice, video, and data transportation, ATM was designed to adapt to the time sensitivity of different applications. It includes different classes of service that enable the technology to match delivery to the time sensitivity of the information it transports.

Table 3-1 Comparing Network Features

Feature	Data Communications	Telecommunications	ATM
Traffic support	Data	Voice	Data, voice, video
Transmission unit	Packet	Frame	Cell
Transmission length	Variable	Fixed	Fixed
Switching type	Packet	Circuit	Cell
Connection type	Connectionless or Connection-oriented	Connection-oriented	Connection-oriented
Time sensitivity	None to some	All	Adaptive
Delivery	Best effort	Guaranteed	Defined class or guaranteed
Media and operating rate	Defined by protocol	Defined by class	Scalable
Media access	Shared or dedicated	Dedicated	Dedicated

Architecture

ATM is based on the switching of 53-byte cells, in which each cell consists of a 5-byte header and a payload of 48 bytes of information. Figure 3.1 illustrates the format of the ATM cell, including the explosion of its 5-byte header to indicate the fields carried in the header.

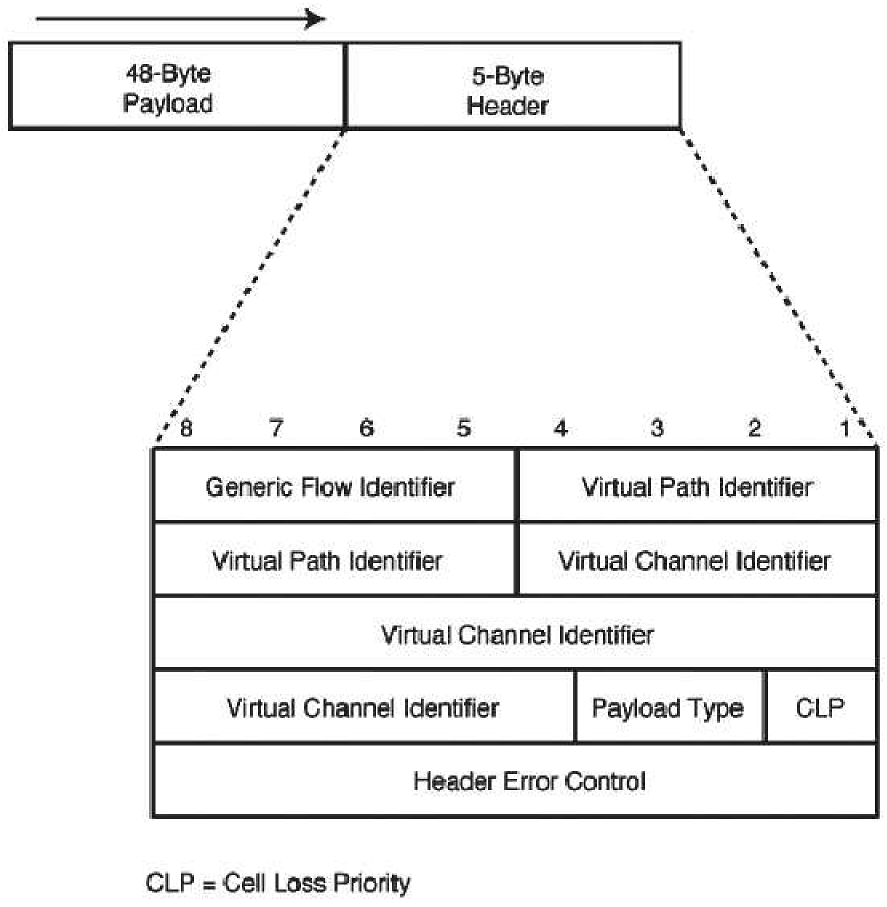


Figure 3.1: The 53-byte ATM cell.

The 4-bit Generic Flow Control (GFC) field is used as a mechanism to regulate the flow of traffic in an ATM network between the network and the user. The use of this field is currently under development. As we will shortly note, ATM supports two major types of interfaces: Network-to-User (UNI) and Network-to-Network (NNI). When a cell flows from the user to the network or from the network to the user, it will carry a GFC bit value. However, when it flows within a network or between networks, the GFC field is not used. Instead of being wasted, its space can be used to expand the length of the Virtual Path Identifier field.

The 8-bit Virtual Path Identifier (VPI) field represents one half of a two-part connection identifier used by ATM. This field identifies a virtual path that can represent a group of virtual circuits transported along the same route. Although the VPI is eight bits long in a UNI cell, the field expands to 12-bit positions to fill the Generic Flow Control field in an NNI cell.

The Virtual Channel Identifier (VCI) is the second half of the two-part connection identifier carried in the ATM header. The 16-bit VCI field identifies a connection between two ATM stations communicating with one another for a specific type of application. Multiple virtual channels (VCs) can be transported within one virtual path. For example, one VC could be used to transport a disk backup operation, while a second VC is used to transport a TCP/IP-based application. The virtual channel represents a one-way cell transport facility. Thus, for each of the previously described operations, another series of VCIs is established from the opposite direction. You can view a virtual channel as an individual one-way end-to-end circuit, whereas a virtual path that can represent a collection of virtual channels can be viewed as a network trunk line. After data is within an ATM network, the VPI is used to route a common group of virtual channels between switches by enabling ATM switches to simply examine the value of the VPI. Later in this chapter, you will examine the use of the VCI.

The Payload Type Identifier (PTI) field indicates the type of information carried in the 48-byte data portion of the ATM cell. Currently, this 3-bit field indicates whether payload data represents management information or user data. Additional PTI field designators have been reserved for future use.

The 1-bit Cell Loss Priority (CLP) field indicates the relative importance of the cell. If this field bit is set to 1, the cell can be discarded by a switch experiencing congestion. If the cell cannot be discarded, the CLP field bit is set to 0.

The last field in the ATM cell header is the 8-bit Header Error Control field. This field represents the result of an 8-bit Cyclic Redundancy Check (CRC) code, computed only over the ATM cell header. This field provides the capability for detecting all single-bit errors and certain multiple-bit errors that occur in the 40-bit ATM cell header.

3.1.1 Cell

ATM uses very large-scale integration (VLSI) technology to segment data (for example, frames from the data link layer of the OSI reference model) at high speeds into units called cells. Each cell consists of 5 octets of header information and 48 octets of payload data, as shown in Figure 3.2

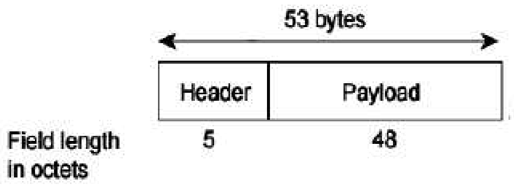


Figure 3.2- ATM Cell Format

Cells transit ATM networks by passing through devices known as ATM switches, which analyze information in the header to switch the cell to the output interface that connects the switch to the next appropriate switch as the cell works its way to its destination. ATM is a cell-switching and multiplexing technology that combines the benefits of circuit switching (constant transmission delay and guaranteed capacity) with those of packet switching (flexibility and efficiency for intermittent traffic). Like X.25 and Frame Relay, ATM defines the interface between the user equipment (such was workstations and routers) and the network (referred to as the User-Network Interface, or UNI). This definition supports the use of ATM switches (and ATM switching techniques) within both public and private networks. Because it is an asynchronous mechanism, ATM differs from synchronous transfer mode methods, where time-division multiplexing (TDM) techniques are employed to pre-assign users to time slots. ATM time slots are made available on demand, with information identifying the source of the transmission contained in the header of each ATM cell. TDM is inefficient relative to ATM because if a station has nothing to transmit when its time slot comes up, that time slot is wasted. The converse situation, where one station has lots of information to transmit, is also less efficient. In this case, that station can only transmit when its turn comes up, even though all the other time slots are empty. With ATM, a station can send cells whenever necessary.

3.1.2 Header and Cell Formats

The ATM standards groups have defined two header formats. The UNI header format is defined by the UNI specification, and the Network-Node Interface (NNI) header format is defined by the NNI specification. The UNI specification defines communications between ATM end-stations (such as workstations and routers) and ATM switches in private ATM networks. The format of the UNI cell header is shown in Figure 3.3

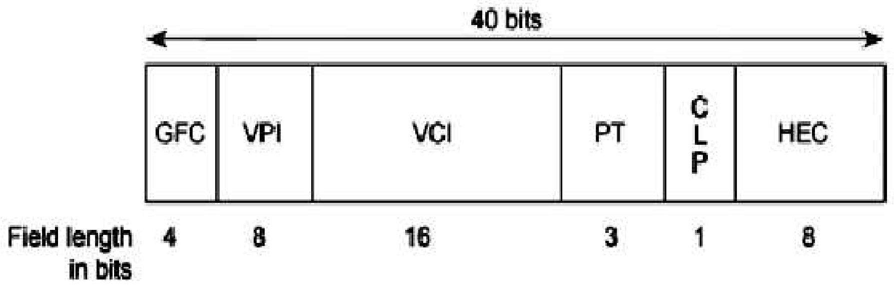


Figure-3.3- UNI Header Format

The UNI header consists of the following fields:

- GFC—4 bits of generic flow control that can be used to provide local functions, such as identifying multiple stations that share a single ATM interface. The GFC field is typically not used and is set to a default value.

- VPI—8 bits of virtual path identifier, which is used, in conjunction with the VCI, to identify the next destination of a cell as it passes through a series of ATM switches on its way to its destination.
- VCI—16 bits of virtual channel identifier, which is used, in conjunction with the VPI, to identify the next destination of a cell as it passes through a series of ATM switches on its way to its destination.
- PT—3 bits of payload type. The first bit indicates whether the cell contains user data or control data. If the cell contains user data, the second bit indicates congestion, and the third bit indicates whether the cell is the last in a series of cells that represent a single AAL5 frame.
- CLP—1 bit of congestion loss priority, which indicates whether the cell should be discarded if it encounters extreme congestion as it moves through the network.
- HEC—8 bits of header error control, which is a checksum calculated only on the header itself. The NNI specification defines communications between ATM switches. The format of the NNI header is shown in Figure 3.4

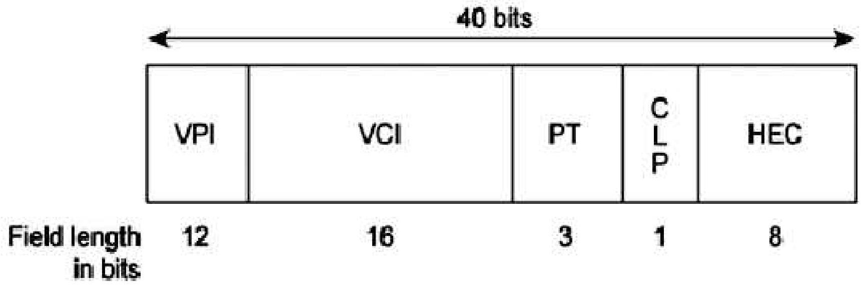


Figure 3.4- NNI Header Format

The GFC field is not present in the format of the NNI header. Instead, the VPI field occupies the first 12 bits, which allows ATM switches to assign larger VPI values. With that exception, the format of the NNI header is identical to the format of the UNI header.

Cells in practice

ATM supports different types of services via AALs. Standardized AALs include AAL1, AAL2, and AAL5, and the rarely used AAL3 and AAL4. AAL1 is used for constant bit rate (CBR) services and circuit emulation. Synchronization is also maintained at AAL1. AAL2 through AAL4 are used for variable bitrate (VBR) services, and AAL5 for data. Which AAL is in use for a given cell is not encoded in the cell. Instead, it is negotiated by or configured at the endpoints on a per-virtual-connection basis.

Following the initial design of ATM, networks have become much faster. A 1500 byte (12000-bit) full-size Ethernet frame takes only 1.2 µs to transmit on a 10 Gbit/s network, reducing the need for small cells to reduce jitter due to contention. Some consider that this makes a case for replacing ATM with Ethernet in the network backbone. However, it should be noted that the increased link speeds by themselves do not alleviate jitter due to queuing. Additionally, the hardware for implementing the service adaptation for IP packets is expensive at very high speeds. Specifically, at speeds of OC-3 and above, the cost of segmentation and reassembly (SAR) hardware makes ATM less competitive for IP than Packet Over SONET (POS); because of its fixed 48-byte cell payload, ATM is not suitable as a data link layer *directly* underlying IP (without the need for SAR at the data link level) since the OSI layer on which IP operates must provide a maximum transmission unit (MTU) of at least 576 bytes. SAR performance limits mean that the fastest IP router ATM interfaces are STM16 - STM64 which actually compares, while as of 2004 POS can operate at OC-192 (STM64) with higher speeds expected in the future.

On slower or congested links (622 Mbit/s and below), ATM does make sense, and for this reason most asymmetric digital subscriber line (ADSL) systems use ATM as an intermediate layer between the physical link layer and a Layer 2 protocol like PPP or Ethernet.

At these lower speeds, ATM provides a useful ability to carry multiple logical circuits on a single physical or virtual medium, although other techniques exist, such as Multi-link PPP and Ethernet VLANs, which are optional in VDSL implementations. DSL can be used as an access method for an ATM network, allowing a DSL termination point in a telephone central office to connect to many internet service providers across a wide-area ATM network. In the United States, at least, this has allowed DSL providers to provide DSL access to the customers of many internet service providers. Since one DSL termination point can support multiple ISPs, the economic feasibility of DSL is substantially improved.

Check your progress

Q1. Define ATM and its Cell format.

Q2. Differentiate between UNI and NNI header format of ATM

3.2 Layers in ATM

Figure 3.5 is a reference model that illustrates the organization of ATM functionality and the interrelationships between the layers of functionality.

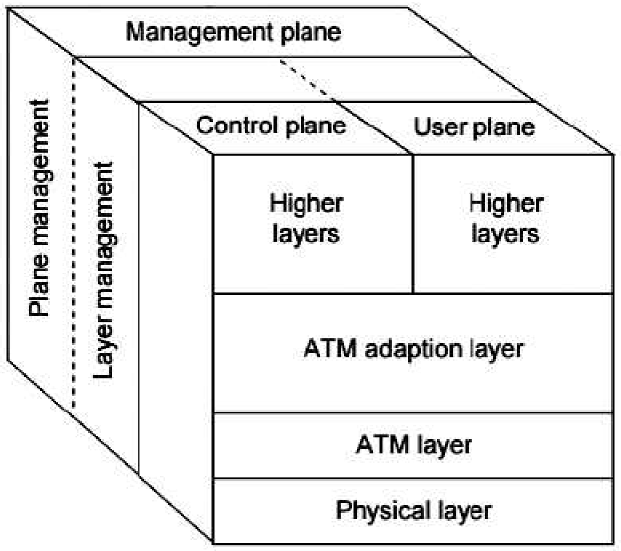


Figure-3.5 ATM Reference Model

In the ATM reference model, the ATM layer and the ATM adaptation layers are roughly analogous parts of the data link layer of the Open System Interconnection (OSI) reference model, and the ATM physical layer is analogous to the physical layer of the OSI reference model. The control plane is responsible for generating and managing signaling requests. The user plane is responsible for managing the transfer of data. Above the ATM adaptation layer are higher-layer protocols representing traditional transports and applications.

Physical Layer

The ATM physical layer controls transmission and receipt of bits on the physical medium. It also keeps track of ATM cell boundaries and packages cells into the appropriate type of frame for the physical medium being used.

The ATM physical layer is divided into two parts: the physical medium sublayer and the transmission convergence sublayer. The physical medium sublayer is responsible for sending and receiving a continuous flow of bits with associated timing information to synchronize transmission and reception. Because it includes only physical-medium-dependent functions, its specification depends on the physical medium used.

ATM can use any physical medium capable of carrying ATM cells. Some existing standards that can carry ATM cells are SONET (Synchronous Optical Network)/SDH, DS-3/E3, 100-Mbps local fiber (Fiber Distributed

Data Interface [FDDI] physical layer), and 155-Mbps local fiber (Fiber Channel physical layer).

Various proposals for use over twisted-pair wire are also under consideration. The transmission convergence sublayer is responsible for the following:

- Cell delineation—Maintains ATM cell boundaries.
- Header error control sequence generation and verification—Generates and checks the header error control code to ensure valid data.
- Cell rate decoupling—Inserts or suppresses idle (unassigned) ATM cells to adapt the rate of valid ATM cells to the payload capacity of the transmission system.
- Transmission frame adaptation—Packages ATM cells into frames acceptable to the particular physical-layer implementation.
- Transmission frame generation and recovery—Generates and maintains the appropriate physical-layer frame structure.

ATM Layer

The ATM layer is responsible for establishing connections and passing cells through the ATM network. To do this, it uses the information contained in the header of each ATM cell. ATM Adaptation Layer the ATM adaptation layer (AAL) translates between the larger service data units (SDUs) (for example, video streams, and data packets) of upper-layer processes and ATM cells. Specifically, the ATM adaptation layer (AAL) receives packets from upper-level protocols (such as AppleTalk, Internet Protocols [IP], and NetWare) and breaks them into the 48-byte segments that form the payload field of an ATM cell. Several ATM adaptation layers are currently specified.

Table 3-2 summarizes the characteristics of each AAL.

Characteristics	**AAL1**	**AAL3/4**	**AAL4**	**AAL5**
Requires timing between source and destination	Yes	No	No	No
Data rate	Constant	Variable	Variable	Variable
Connection mode	Connection-oriented	Connection-oriented	Connection-less	Connection-oriented
Traffic types	Voice and circuit emulation	Data	Data	Data

AAL1

Figure 3-6 shows how AAL1 prepares a cell for transmission. The payload data consists of a synchronous sample—for example, 1 byte of data generated at a sampling rate of 125 microseconds.

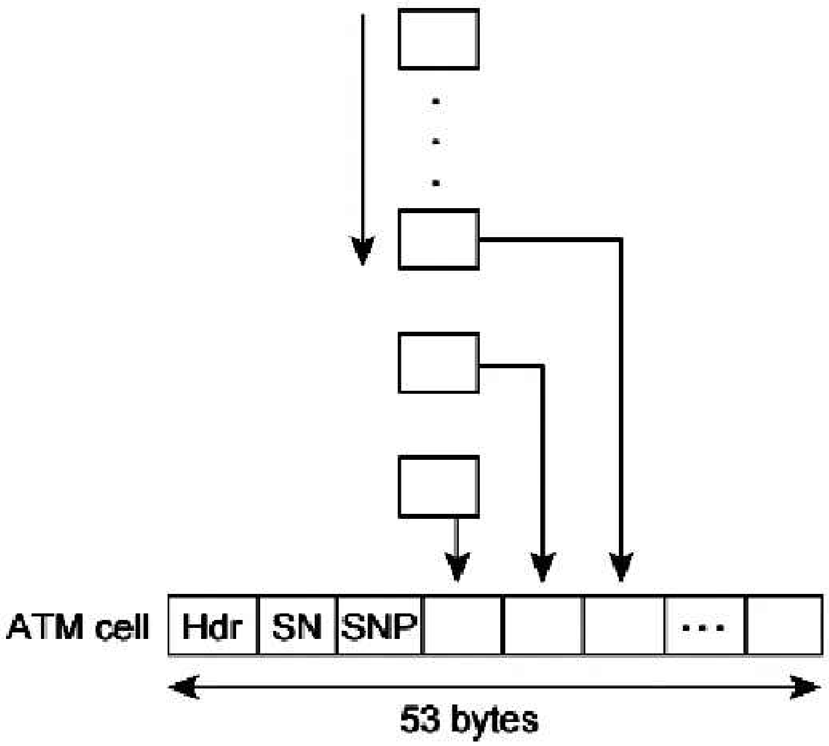


Figure 3-6 AAL1 Cell Preparation

The sequence number field (SN) and sequence number protection (SNP) fields provide the information that the receiving AAL1 needs to verify that it has received the cells in the correct order. The remainder of the payload field is filled with enough single bytes to equal 48 bytes.

AAL1 is appropriate for transporting telephone traffic and uncompressed video traffic. It requires timing synchronization between the source and destination and, for that reason, depends on a media that supports clocking, such as SONET. The standards for supporting clock recovery are currently being defined.

AAL3/4

AAL3/4 was designed for network service providers and is closely aligned with Switched Multimegabit Data Service (SMDS). AAL3/4 will be used to transmit SMDS packets over an ATM network.

As shown in Figure 3-7, the convergence sublayer (CS) creates a protocol data unit (PDU) by prepending a Beginning/End Tag header to the frame and appending a length field as a trailer.

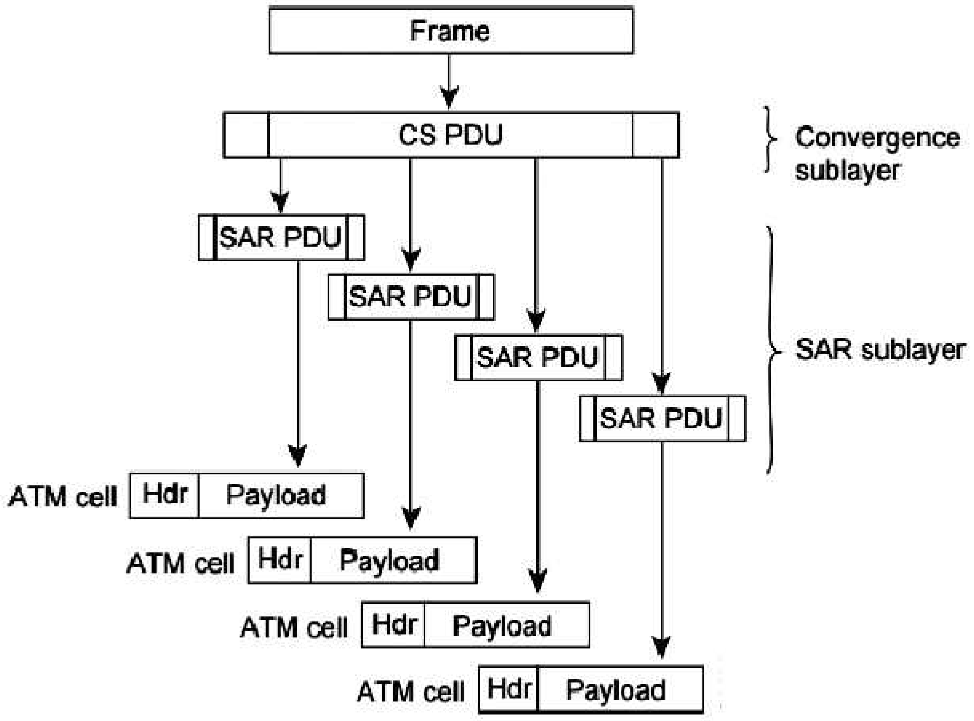


Figure 3-7 AAL3/4 Cell Preparation

The segmentation and reassembly (SAR) sublayer fragments the PDU and prepends to each PDU fragment a header consisting of the following fields:

- ❖ Type—identifies whether the cell is the beginning of a message, continuation of a message, or end of a message.
- ❖ Sequence number—identifies the order in which cells should be reassembled.
- ❖ Multiplexing identifier—Identifies cells from different traffic sources interleaved on the same virtual circuit connection (VCC) so that the correct cells are reassembled at the destination.

The SAR sublayer also appends a CRC-10 trailer to each PDU fragment. The completed SAR PDU becomes the payload field of an ATM cell to which the ATM layer prepends the standard ATM header

AAL5

Figure 3-8 shows how AAL5 prepares a cell for transmission. First, the convergence sublayer of AAL5 appends a variable-length pad and a 8-byte trailer to a frame. The pad is long enough to ensure that the resulting PDU falls on the 48-byte boundary of the ATM cell. The trailer includes the length of the frame and a 32-bit CRC computed across the entire PDU, which allows AAL5 at the destination to detect bit errors and lost or cells that are out of sequence.

Next, the segmentation and reassembly segments the CS PDU into 48-byte blocks. Then the ATM layer places each block into the payload field of an ATM cell. For all cells except the last cell, a bit in the PT field is set to zero to indicate that the cell is not the last cell in a series that represent a single frame. For the last cell, the bit in the PT field is set to one.

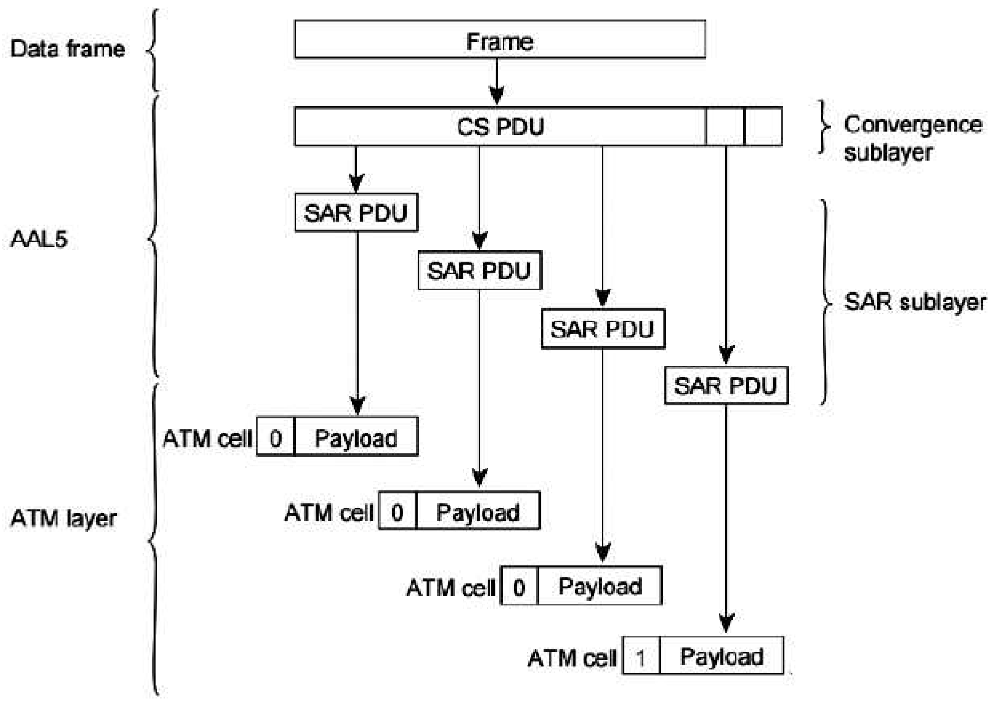


Figure 3-8 AAL5 Cell Preparation

When the cell arrives at its destination, the ATM layer extracts the payload field from the cell; the SAR sublayer reassembles the CS PDU; and the CS uses the CRC and the length field to verify that the frame has been transmitted and reassembled correctly.

AAL5 is the adaptation layer used to transfer most non-SMDS data, such as classical IP over ATM and local-area network (LAN) emulation.

Check your progress

Q1. Define physical layer of ATM

Q2. Explain AAL1, AAL 3/4 and AAL 5 layers in ATM

3.3 Class 1,2,3,4

The AAL adapts the protocols to an ATM intermediate format. It uses socalled 'classes' to do so. AAL type 3 and 4 handle transmissions of connectionless data, AAL type 5 is intended for connection-oriented services.

ATM relies on different classes of service to accommodate different applications (voice, video and data). They define the bits and bytes that are actually transmitted, as well as the required bandwidth, allowable error rates, and so forth. Class A and B, have timing compensation, for applications that cannot tolerate variable delays. Class C and D, no timing compensation, for data applications like LAN interconnect. Class D also simulates connectionless communications, commonly found on LANs.

Table 3-3 summarizes the characteristics of each AAL according to Class

Class	A	B	C	D
Timing	yes	yes	no	No
Bit rate	constant	variable	variable	Variable
Mode	Connection-oriented, circuit emulation	Connection-oriented, variable bit-rate video	Connection-oriented, connection-oriented data	Connectionless, connectionless data
AAL	Type 1	Type 2	Type 3/4	Type ¾
			Type 5	

AAL 1:

For isochronous, constant bit-rate services, such as audio and video. This adaption layer corresponds to fractional and full T1 and T3, but with a greater range of choices for data rates.

AAL 2:

For isochronous variable bit-rate services, such as compressed video.

AAL 3/4:

For variable bi-rate data, such as LAN applications. Originally designed as two different layers, one for connection-oriented services (like frame relay) and one for connectionless services (like SMDS). Both can be done by the same AAL though.

AAL 5:

For variable bit-rate data that must be formatted into 53-byte cells. Similar to AAL 3/4, easier to implement, less features.

The service-specific convergence sublayer (SSCS) maps (converts) the data to the ATM layer. The convergence sublayer (CS) than compensates for the various interfaces (copper and fiber) that may be used on an ATM network. The ATM network can use Sonet, T1, E1, T3, E3, E4, FDDI, pure cells, Sonet SDH, block-encoded fiber, etc.

3.4 Summary

In this unit you have learnt about Asynchronous Transfer Mode (ATM), its cells and Header and Cell Formats. You also learnt about different layers in ATM. We also defined Class 1,2,3,4 in ATM.

- ❖ Asynchronous transfer mode (ATM), also known as cell relay, is similar in concept to frame relay.
- ❖ ATM is even more streamlined than frame relay in its functionality, and can support data rates several orders of magnitude greater than frame relay.
- ❖ The "asynchronous" in ATM means ATM devices do not send and receive information at fixed speeds or using a timer, but instead negotiate transmission speeds based on hardware and information flow reliability.
- ❖ ATM uses very large-scale integration (VLSI) technology to segment data (for example, frames from the data link layer of the OSI reference model) at high speeds into units called cells.

3.5 Review Questions

Q1. What do you understand by ATM in computer networks?

Q2. Define its cell, Header and Cell formats in detail.

Q3. Explain all the layers of ATM.

UNIT-IV
ALOHA

Structure

4.0 Introduction

This unit focused on the topic ALOHA in your course. In this unit there are five sections. In the first section i.e. Sec. 4.1. You will learn about random access data networks. How random access works? In the next section i.e. Sec. 4.2 you will know about ALOHA. As you know **ALOHAnet**, also known as the **ALOHA System**, or simply **ALOHA**, was a pioneering computer networking system developed at the University of Hawaii. ALOHAnet became operational in June, 1971, providing the first public demonstration of a wireless packet data network. ALOHA originally stood for Additive Links On-line Hawaii Area.

The ALOHAnet used a new method of medium access (ALOHA random access) and experimental ultrahigh frequency (UHF) for its operation, since frequency assignments for communications to and from a computer were not available for commercial applications in the 1970s. But even before such frequencies were assigned there were two other media available for the application of an ALOHA channel – cables and satellites. In the 1970s ALOHA random access was employed in the widely used Ethernet cable based network and then in the Marisat (now Inmarsat) satellite network. In next section we define slotted ALOHA. You will learn about it later in this unit. In Sec. 4.4 and 4.5 you will find summary and review questions respectively.

Objectives

After studying this unit you should be able to:

- Random Access Data Networks
- Pure ALOHA and its Throughput, Characteristics
- Slotted ALOHA and its Throughputs for Finite and Infinite Population S-ALOHAS and MARKOV Chain Model for S-ALOHA

4.1 Random Access Data Networks

Random Access, which is to issue a completely random time, relies on the Aloha method. The latter takes its name from an experiment performed on a network connecting the various islands of the Hawaiian Archipelago early 1970. In this method, when a coupler has information to transmit, it sends it without worry about other users. If there is a collision, that is to say superposition of two signals or more users, the signals become indecipherable and are lost. They are subsequently transmitted, as shown in Figure 4.1, in which the couplers 1, 2 and 3 collide. The coupler 1 transmits its field first because he shot the smallest timer. Then, the module 2 emits, and its signals collide with the coupler 1. Both derive a random time of retransmission. The coupler 3 is listening while the couplers 1 and 2 are silent, so that the frame of the coupler 3 passes successfully. Technical aloha is the origin of all the random access methods.

ALOHA

ALOHA is a system for coordinating and arbitrating access to a shared communication Networks channel. It was developed in the 1970s by Norman Abramson and his colleagues at the University of Hawaii. The original system used for ground based radio broadcasting, but the system has been implemented in satellite communication systems.

A shared communication system like ALOHA requires a method of handling collisions that occur when two or more systems attempt to transmit on the channel at the same time. In the ALOHA system, a node transmits whenever data is available to send. If another node transmits at the same time, a collision occurs, and the frames that were transmitted are lost. However, a node can listen to broadcasts on the medium, even its own, and determine whether the frames were transmitted.

Aloha means "Hello". Aloha is a multiple access protocol at the data link layer and proposes how multiple terminals access the medium without interference or collision. In 1972 Roberts developed a protocol that would increase the capacity of aloha two fold. The Slotted Aloha protocol involves dividing the time interval into discrete slots and each slot interval corresponds to the time period of one frame. This method requires synchronization between the sending nodes to prevent collisions.

There are two different versions/types of ALOHA:

- ❖ Pure ALOHA
- ❖ Slotted ALOHA

The ALOHA Principle

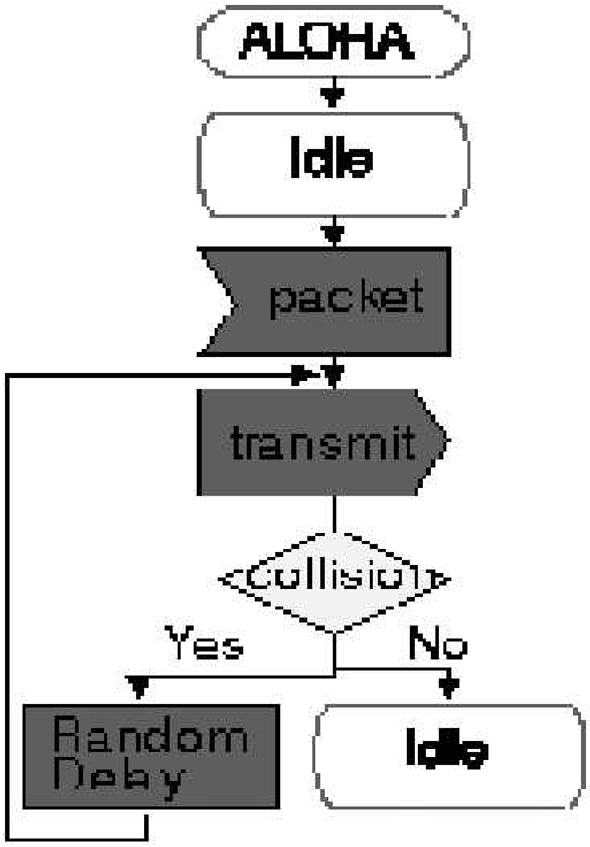


Figure 4-1: Description of terminal behavior in ALOHA random access network

In unslotted ALOHA, a transmission may start at any time. In slotted ALOHA the time axis is divided to slots. All terminals are assumed to know the times at which a new slot begins. Packets may only be transmitted at the beginning of a new slot. Slotted ALOHA has significantly better throughput than unslotted ALOHA.

Example: GSM call set-up

If the transmitter-to-receiver propagation time is large and unknown, the slot time must be equal to the packet length plus a sufficiently large guard time. In the call set-up of GSM, random access packets are substantially shorter than normal telephone speech blocks: During call set-up the propagation time is still unknown because the subscriber can be anywhere in the cell. During the call, the propagation times are measured and the terminal transmitter will compensate for it, by sending all blocks a bit in advance. A closed-loop control circuit, sending adaptive timing advance/delay feedback information, is used to ensure that the timing remains correct even if the subscribers moves in the cell.

ALOHA facts

- ❖ Developed early 70s at University of Hawaii
- ❖ Basic idea is very simple, but many modifications exist
- ❖ Any terminal is allowed to transmit without considering whether channel is idle or busy
- ❖ If packet is received correctly, the base station transmits an acknowledgement.
- ❖ If no acknowledgement is received by the mobile,
  - o it assumes the packet to be lost
  - o it retransmits the packet after waiting a random time
- ❖ Unslotted ALOHA: transmission may start anytime
- ❖ Slotted ALOHA: packets are transmitted in time slots

4.2 Pure ALOHA

In pure ALOHA, the stations transmit frames whenever they have data to send.

- ▪ When two or more stations transmit simultaneously, there is collision and the frames are destroyed.
- ▪ In pure ALOHA, whenever any station transmits a frame, it expects the acknowledgement from the receiver.
- ▪ If acknowledgement is not received within specified time, the station assumes that the frame (or acknowledgement) has been destroyed.
- ▪ If the frame is destroyed because of collision the station waits for a random amount of time and sends it again. This waiting time must be random otherwise same frames will collide again and again.
- ▪ Therefore pure ALOHA dictates that when time-out period passes, each station must wait for a random amount of time before

resending its frame. This randomness will help avoid more collisions.

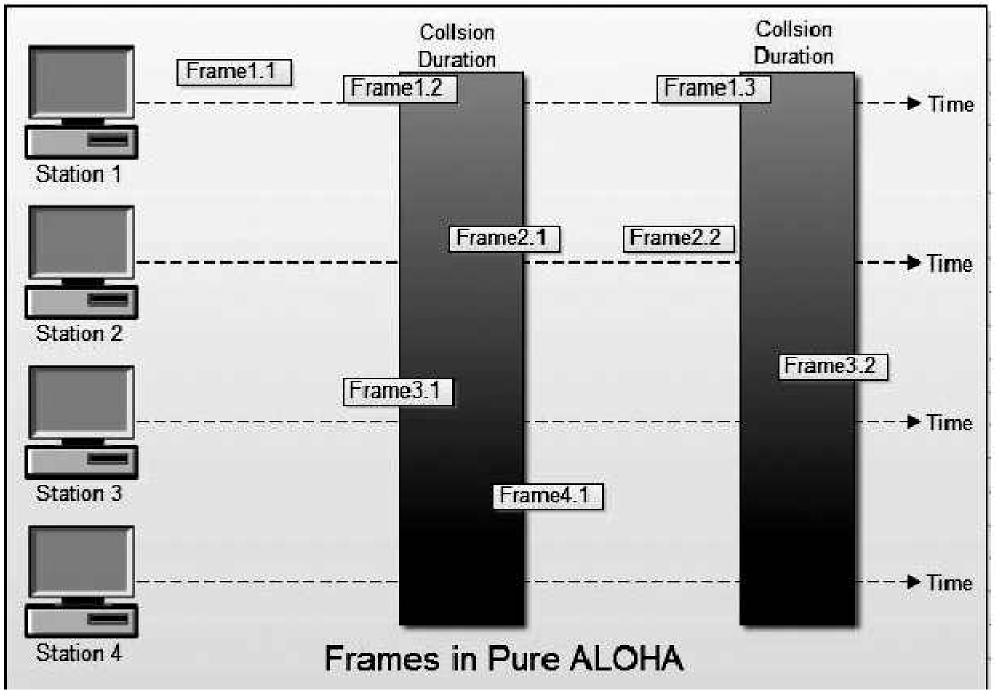


Figure 4.2- Frames collision in Pure Aloha

In figure 4.2 there are four stations that contended with one another for access to shared channel. All these stations are transmitting frames. Some of these frames collide because multiple frames are in contention for the shared channel. Only two frames, frame 1.1 and frame 2.2 survive. All other frames are destroyed.

Whenever two frames try to occupy the channel at the same time, there will be a collision and both will be damaged. If first bit of a new frame overlaps with just the last bit of a frame almost finished, both frames will be totally destroyed and both will have to be retransmitted.

4.2.1 Throughput Characteristics

To express the throughput of the ALOHA random access scheme, it is often assumed that message transmission attempts occur according to a Poisson process with rate *G* attempts per slot. For channels in which a transmission is successful if and and only if in that slot only a single packet transmission is present, the throughput of successful messages is equal to

- The probability of having just one message: $S = G \exp\{-G\}$, or equivalently,
- The attempted traffic *G* multiplied by the probability $\exp\{-G\}$ that no interfering message is present

Both arguments yield the well-known result for the throughput of slotted ALOHA:

$$S = G \exp\{-G\}$$

For unslotted ALOHA without capture, a test packet is destroyed by any overlapping transmission starting in the time window that

- ❖ starts one packet time before the transmission of the test packet and
- ❖ Closes at the end of the transmission of the test packet.

Hence, packets transmitted over an unslotted ALOHA channel see on average twice as many interfering packets as in slotted ALOHA. In fact

$$S = G \exp\{-2\,G\}$$

Both unslotted and slotted ALOHA exhibit the typical behaviour that

- ❖ At low traffic (small G), S is approximately equal to G
- ❖ At high traffic loads (large G), S decreases to zero. Almost all packets are lost in collisions.
- ❖ One throughput value S corresponds to two values of G. The curves misleading suggest that one G would be stable while the other is unstable. For systems without capture it turns out, however, that the ALOHA system with a fixed retransmission procedure, independent of the history of the network, is always unstable.

ALOHA in Mobile Radio Nets

In a radio channel, packets may be received successfully despite interference from competing terminals. This is called `receiver capture'. The larger the differences in received signal power, the more likely it is that one signal is sufficiently strong to capture the receiver

The throughput becomes G times the probability that a particular (a priori chosen) packet is sufficiently stronger than the sum of all interfering packets.

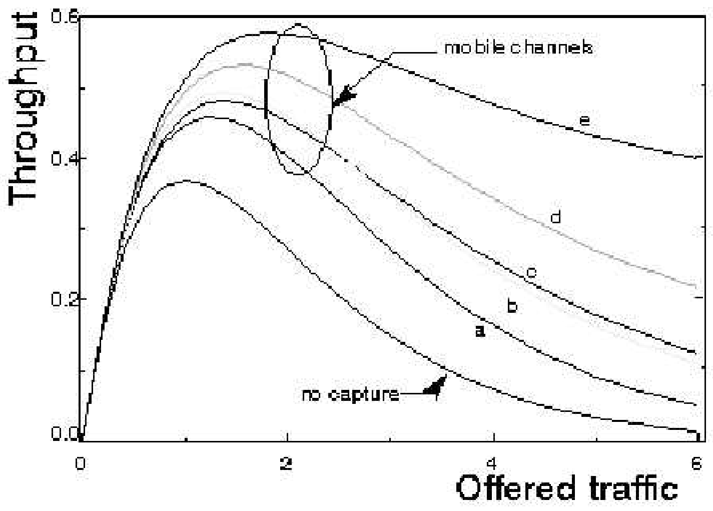


Figure 4.3: Throughput *S* of Slotted ALOHA network (in packet per time slot) versus the attempted traffic *G*.

- Without capture; any collision destroys all packets involved
- With Capture (receiver threshold is 4)
    - a: Rayleigh fading only
    - c: Shadowing and Rayleigh fading
    - e: Near far effect and Rayleigh fading, Uniform distribution of terminal is a circular cell around the receiver.
    - b,d: Some other capture models

Note that some capture models do not predict that *S* reduces to zero for large *G*.

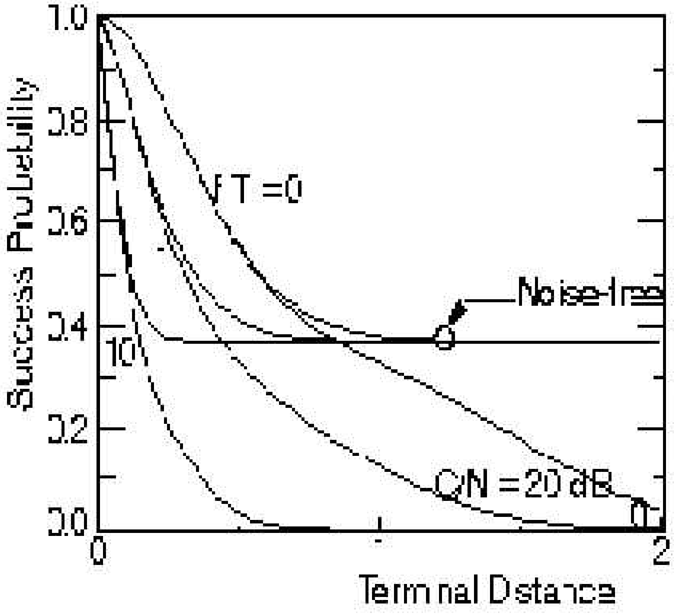


Figure 4.4: Probability that an access attempt is successful versus the distance between terminal and base station.

- Average message arrival and retransmission rate: 1 packet per unit of time.
- Terminals quasi-uniformly distributed over cell, with cell boundary approximately at unity distance.
- Receiver threshold 6 dB.
- Plane earth loss (40 log d) and narrowband Rayleigh fading.
- Various packet durations *T*, normalized to the Doppler spread.
- Median signal-to-noise ratio (for user at distance 0.734) is 20 dB.

- New arrivals and retransmissions for a stationary Poisson process. The network is stable. Theoretically, this condition can only be satisfied if the retransmission backoff delay is infinitely large.

Computation of Throughput

The throughput S is found as the offered traffic, multiplied by the probability that a particular, a priori chosen **test** packet captures the receiver. Thus

$$S = G \sum_{i=0}^{\inf} \text{Prob(capture } | i \text{ interferers) Prob (} i \text{ interferers)}$$

Where the probability on i interferers is Poissonian with value G.

The capture probability equals one minus the outage probability. In most analyses, the probability of capture is first expressed for a given local mean power of the test signal. In a Rayleigh-fading channel, the probability of receiving a signal with local-mean power $\bar{p}\_0$ can be expressed in terms of Laplace Transforms of the probability density functions of the power p_i of interferer i. That is,

$$\text{P(capt} | i) = \text{Prob(capt}|i{=}1)^{\wedge i} = \text{Laplace}^{\wedge i} \{p\_i \; ; \; \frac{z}{\bar{p}\_0}\}$$

Where z is the capture ratio or receiver threshold. This observation leads to the mathematically convenient definition of the interference-vulnerability weight function.

Throughput for various propagation and traffic distribution models

One can use the above expression to find the throughput for various distributions of terminals over the "service" area.

- Ring model.

  For mathematical simplicity, initially the throughput of ALOHA network was computed for the special case that all user signals are received with the same mean power. This model is also appropriate for DS-CDMA packet networks with power control. Delay and throughput performance of packet can be optimized choosing the appropriate spread factor and retransmission strategy.

- Uniform distribution of packet transmissions
  - Uniform within a circular cell
  - Uniform with infinite extension

- . This model is appropriate for
- Collecting telemetric data in ISM bands.
- Collecting road-traffic data from probe vehicles. Floating Car Data.

- Cellular ALOHA networks with frequency reuse; Virtual Cellular Networks (VCN) Random access network with stack collision resolution and DS-CDMA
- Uniform throughput.

  This implies that users at remote locations have to perform more retransmission attempts. It results in **non**-uniform attempted traffic.
- ALOHA messages generated by (probe) vehicles participating in road traffic, considering a specific road infrastructure.

Check your progress

Q1. Define Random access data networks.

Q2. Explain pure ALOHA in detail

4.3 Slotted ALOHA

The Aloha network was developed by the University of Hawaii around 1970. Hawaii consists of several islands and the university has its campuses in each of these islands. So, the university decided to build wireless connections between the computer network of the main campus and those of the remote campuses. Whenever a remote campus has a packet that needs to be sent to the main campus, it just transmits it through the air (cheap!). If another remote campus sends its own packet while the first packet is being transmitted, collision takes place and none of the packets will be received by the main campus. Therefore, a protocol was needed for scheduling the retransmission attempts of backlogged packets. The difficulty here is that the individual nodes do not know whether (a) other nodes have a new packet to be sent, (b) how many nodes were involved in a collision, and (c) when other nodes will attempt to transmit their backlogged packets. Aloha is a protocol that addresses these issues. There are two versions of Aloha: unslotted Aloha and slotted Aloha. The difference between the two versions is that packet transmission can start at any time in an unslotted Aloha system, while packets can only be transmitted during synchronized time slots in a slotted Aloha system. We will focus on slotted Aloha systems.

Assumptions

For the analysis, we make the following assumptions.

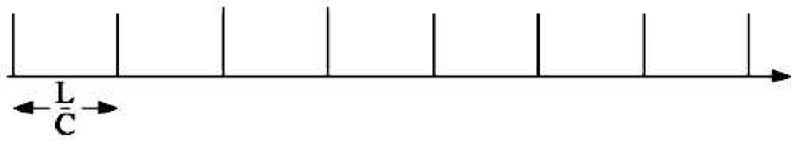


Figure: 4.5- Time is divided into unit slot.

- Slotted system: Each packet has the same length L. The channel capacity is C. Time is divided into slots and each packet requires one slot for transmission. We re-scale time and set 1 unit time equal to L C seconds (the transmission delay of one packet).
- Poisson arrivals: Each host generates new packets according to a Poisson process. The overall arrival rate of new packets to the system is λ.
- Collisions or perfect reception: If more than one host transmit a packet in the same time slot, there is a collision and those packets are lost. If only one packet is transmitted in a time slot, that packet is received correctly.
- Immediate feedback: At the end of each slot, each host can determine whether the slot was idle, one packet was successfully transmitted, or a collision occurred.
- Retransmission of backlogged packets: Packets that are involved in collision must be retransmitted in some later slot until it is finally received. Nodes that have packets that need to be retransmitted are said to be backlogged. Here, we assume that each backlogged node retransmits with a fixed probability $q_r > 0$ in each successive slot until successful.
- Infinite number of hosts: The system has an infinite set of hosts and each host has at most one packet to transmit.

When a collision occurs, each host will discover the collision at the end of the slot, and retransmit packets that collided after waiting for some random time. The average waiting time until the next retransmission attempt is equal to 1 q_r time units. One question that we want to answer is how nodes should choose the retransmission probability q_r. Note that when q_r is very small, then the (expected) idle time between retransmission attempts becomes very large. On the other hand, choosing q_r equal to 1 will lead to a system deadlock.

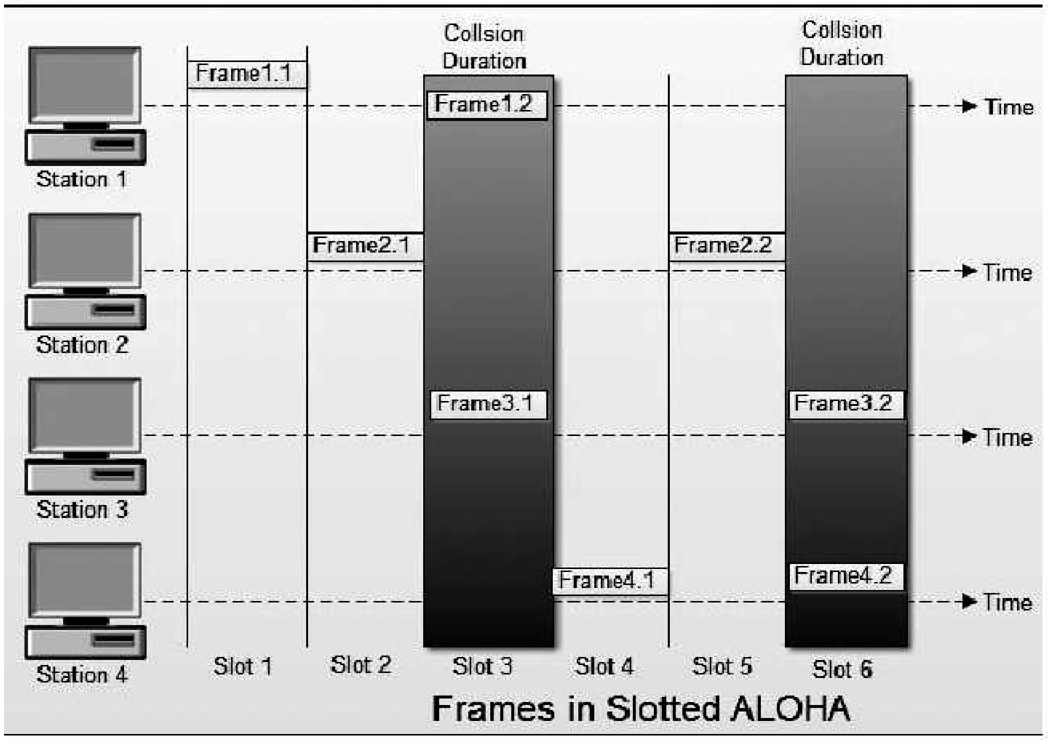


Figure 4-6: Frames in Slotted ALOHA

- ❖ Slotted ALOHA was invented to improve the efficiency of pure ALOHA as chances of collision in pure ALOHA are very high.
- ❖ In slotted ALOHA, the time of the shared channel is divided into discrete intervals called slots.
- ❖ The stations can send a frame only at the beginning of the slot and only one frame is sent in each slot.
- ❖ In slotted ALOHA, if any station is not able to place the frame onto the channel at the beginning of the slot *i.e.* it misses the time slot then the station has to wait until the beginning of the next time slot.
- ❖ In slotted ALOHA, there is still a possibility of collision if two stations try to send at the beginning of the same time slot as shown in figure 4-6
- ❖ Slotted ALOHA still has an edge over pure ALOHA as chances of collision are reduced to one-half.

4.3.1 Throughputs for Finite and Infinite Population S-ALOHAS

In this section, we carry out a (simplified) analysis of the slotted Aloha system. We denote with *n* the number of nodes that have a packet to send. During one time slot, events occur with the following probabilities.

- Probability of a idle slot: $(1 - q_r)^n$
- Probability of a successful transmission: $nq_r(1 - q_r)^{n-1}$
- Probability of a collision: $1 - (1 - q_r)^n - nq_r(1 - q_r)^{n-1}$

Let P_{succ} be the probability of a successful transmission during a time slot. We then have

$$P_{succ} = nq_r(1 - q_r)^{n-1}$$
$$= nq_r /1 - q_r\ (1 - q_r)^n$$

When q_r is small, we can use the following approximations,

$$(1 - q_r)^n \approx e^{-nq_r} \text{ and } nq_r /1 - q_r \approx nq_r,$$

And we obtain

$P_{succ} \approx nq_r e^{-nq_r}$,

$= G(n)e^{-G(n)}$, (1)

Where

$$G(n) = nq_r.$$

As we rescaled time such that the length of each time slot is equal to 1 time unit, when the attempted transmission rate is equal to G then the throughput γ of the system is equal to Ge^{-G} packets per unit time.

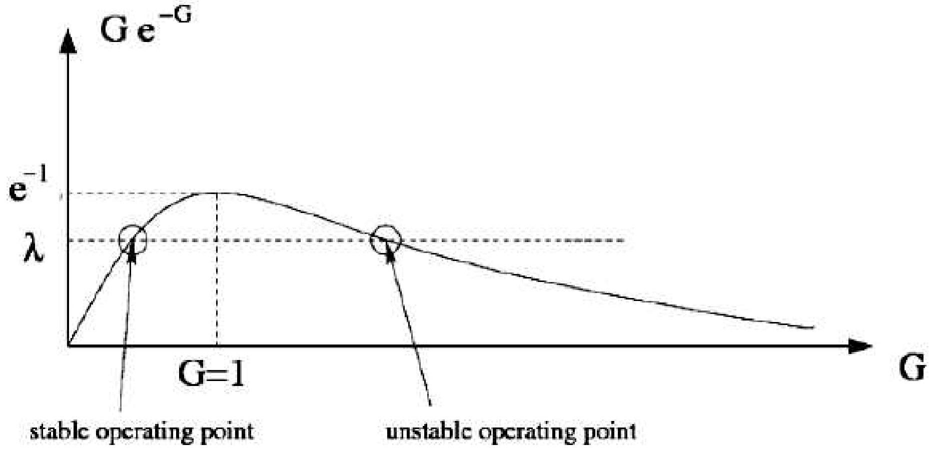


Figure: 4.7- Slotted Aloha: Throughput γ as a function of attempted transmission rate G

Figure 4.7 shows the relationship between the attempted transmission rate G and the throughput γ. We make the following observations. The highest possible throughput is equal to $e^{-1} \approx 0.368$, and is achieved for the attempted transmission rate $G^* = 1$. When the attempted transmission rate is very small (the system is lightly loaded) then there will be few collision; but the channel will be idle most of the time and the throughput γ will be small. On the other hand, when G is large and the system is heavily loaded, then in almost every time-slot a collision will occur and the throughput will be very small. The stable operating point of the system is when the throughput γ is equal to λ which is the arrival rate of new packets. When $\lambda > \gamma$, then (on average) more packets will arrive than depart, and the number of backlogged nodes, and the attempted transmission rate G, tend to increase. When $\lambda < \gamma$, then (on average) more packets depart than arrive, and the number of backlogged nodes, and the attempted transmission rate, tends to decrease. In the above figure, we observe that for any $\lambda < 1/e$, there are two values of G for which we have $\gamma = \lambda$. Note that the operating point on the left is a stable operating point as for small deviations the throughput will drift back to the operating point. However, the operating point on the right is unstable as for small deviations the throughput will drift away from the operating point. In particular, an additional collision will increase the number of backlogged nodes, and therefore the attempted transmission rate, causing the throughput of the system to drift towards 0, and the number of backlogged nodes will drift towards ∞.

One approach to avoid this instable behavior of the system is to dynamically change q_r as the number of backlogged nodes n changes. Naturally, we would like to choose the retransmission probability to maximize the throughput, and therefore reduce the number of backlogged nodes as fast as possible. As the maximal throughput is obtained for G = 1, we should choose q_r such that $nq_r = 1$. Unfortunately, the individual nodes do not know the exact number n of backlogged nodes, but can only be estimated by observing the channel. There are many strategies for estimating n and choosing the retransmission probability q_r. In essence, all of them increase q_r when an idle slot occurs, and decrease q_r when a collision occurs.

Analysis

Let say transmission of a packet takes Psecs. The vulnerable period is 2P, as shown:

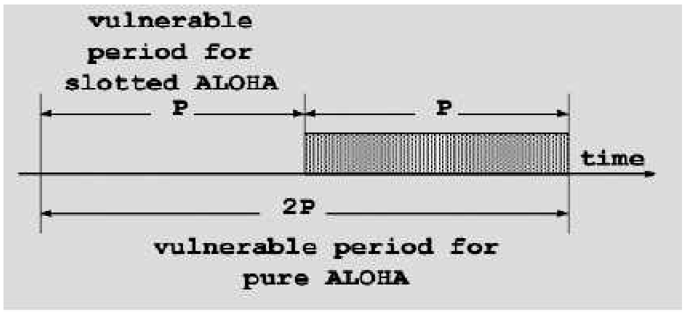


Figure: 4.8- Vulnerable period for Pure and slotted ALOHA

Define:

S denote the throughput of the channel (e.g., average number of successful transmission per transmission period P).

G denote the average channel traffic (e.g., number of packet transmission attempted per P sec).

Time is "slotted" and all users are synchronized to these slot intervals. When a packet arrives within a slot, user delays the transmission until next time slot. Therefore, the vulnerable period is P only. Using similar assumption that the total traffic is a Poisson process:

$$S = Ge^{-G}. \tag{2}$$

To obtain the maximum value of S, use similar technique, the maximum occurs at G = 1, we have

$$S* = e^{-1} = 0.368. \tag{4}$$

Consider the throughput vs. offered traffic

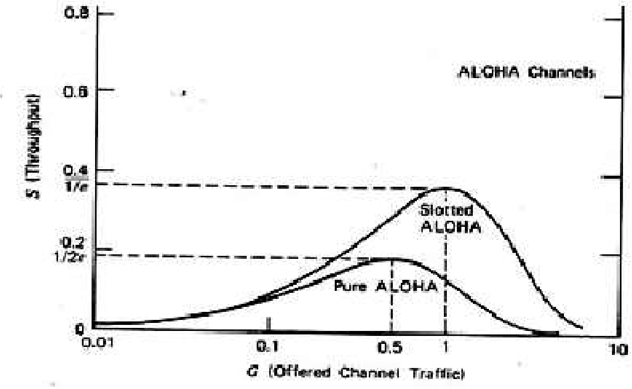


Figure: 4.9- Throughput for pure and slotted ALOHA

Consider a finite population model with M independent users in a slotted ALOHA. User's packet transmission (old or new packet) as a sequence of independent Bernoulli trials, or G_m be the probability the mth user transmits a packet in any give slot, where m = 1, . . . , M. G_m can be viewed as the average traffic (per slot) for the mth user. So the average offered load is

$$G = \sum_{i=1}^{M} G_m.$$

Let S_m be the probability that the mth user's packet is successfully transmitted. Similarly,

$$S = \sum_{i=1}^{m}$$

S_m is the system throughput (per slot). We have:

$$S_m = G_m \prod_{i \neq m} (1 - G_i) \quad m = 1, 2, \ldots, M.$$

The set of M equations has a solution set, $\{S_m\}$, which defines the allowable mixtures of source rates which this channel can support.

4.3.2 MARKOV Chain Model for S-ALOHA

Assume that each backlogged node retransmits with some fixed probability qr in each successive slot until a successful transmission occurs. In other words, the number of slots from a collision until a given node involved in collision retransmits in a geometric random variable having value i ≥1 with probability $q_r\ (1-q_r)^{i-1}$.

The behavior of slotted Aloha can now be described as a discrete-time Markov chain. Let n be the number of backlogged nodes at the beginning of a given slot. Each of these nodes will transmit a packet in the given slot, independently of each other, with probability q_r. Each of the m-n other nodes will transmit a packet in the given slot if one (or more) such packets arrived during the previous slot. Since such arrivals are Poisson distributed with mean λ/m, the probability of no arrivals is $e^{-\lambda/m}$; thus, the probability that an unbacklogged node transmits a packet in the given slot is qa =1- $e^{-\lambda/m}$.

Let

$$Q_a(i,n) = \text{prob } i \text{ unbacklogged nodes transmitted}$$
$$= \binom{m-n}{i}(1-q_a)^{m-n-i} q_a^i$$
$$Q_r(i,n) = \text{prob } i \text{ backlogged nodes transmitted}$$
$$= \binom{n}{i} q_r^i (1-q_r)^{n-i}$$

Then we obtain

$$P_{n,n+i} = \begin{cases} Q_a(i,n) & 2 \le i \le (m-n) \\ Q_a(1,n)[1-Q_r(0,n)] & i=1 \\ Q_a(1,n)Q_r(0,n)+Q_a(0,n)[1-Q_r(1,n)] & i=0 \\ Q_a(0,n)Q_r(1,n) & i=-1 \end{cases}$$

To solve this Markov chain, as in Fig.4.3, we have

$$P_0[\sum_{j=2}^{m} P_{0j}] = P_0P_{00} + P_1P_{10} \quad \text{slove } P_1 \text{ for in term of } P_0$$

$$P_1[\sum_{j=2}^{m} P_{1j}] = P_1P_{11} + P_2P_{21} \quad \text{slove } P_2 \text{ for in term of } P_1(P_0)$$

$$\vdots$$

$$\text{and } \sum_{j=0}^{m} P_j = 1$$

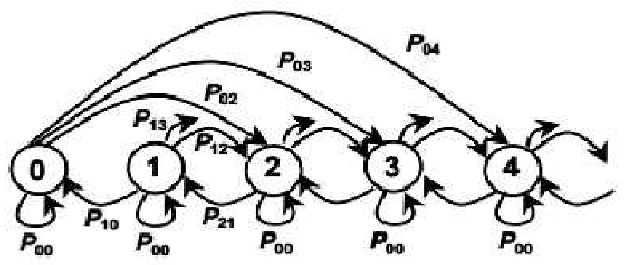


Figure: 4.10-Markov chain for Slotted ALOHA

Check your progress

Q1. Explain Slotted Aloha.

Q2. Differentiate between pure Aloha and Slotted Aloha.

Q3. Define Throughputs for Finite and Infinite Population S-ALOHAS.

4.4 Summary

In this unit you have learnt about Random Access Data Networks, Pure ALOHA and its Throughput and Characteristics, Slotted ALOHA and its Throughputs for Finite and Infinite Population S-ALOHAS and MARKOV Chain Model for S-ALOHA.

❖ **Aloha random access** is a widely used technique for coordinating the access of large numbers of intermittent transmitters in a single shared communication channel.

❖ In an ALOHA channel each transmitter sharing the channel transmits data packets at random times. In most ALOHA channels the transmitters then rely on some protocol (such as repetition) to handle the case of packets lost due to interference by other packets.

❖ An ALOHA channel may also just provide a best effort delivery mechanism and leave it to the receiver to deal with lost packets.

4.5 Review Questions

Q1. Define Random access data network. Why we need random access data network?

Q2. Explain pure ALOHA and its throughput and characteristics.

Q3. Why slotted ALOHA needed? Explain.

Q4. Differentiate between pure and slotted aloha.

Q5. How do we develop MARKOV Chain Model for S-ALOHA? Elaborate your answer.

BIBLIOGRAPHY

Behrouz A. Forouzan: Data Communications and Networking, Fourth Edition. McGraw-Hill Professional, 2007

Douglas E. Comer. *Computer Networks and Internets, Sixth Edition.* Prentice-Hall, 2015.

Nader F. Mir. *Computer and Communication Networks, Second Edition.* Prentice-Hall, 2015.

James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach, Sixth Edition.* Addison-Wesley, 2013.

Jeffrey S. Beasley and Piyasat Nilkaew. *Networking Essentials, Third Edition.* Pearson Education, 2012.

T. H. Bonn, "A Standard for Computer Networks, "Computer, vol. 4, p. 10-14, May/June, 1971.

http://electronicspost.com/pulse-code-modulation-pcm-system/

http://einstein.informatik.uni-oldenburg.de/rechnernetze/pcm.htm

E.W. Biersack, "Performance evaluation of forward error correction in ATM networks", Proc. ACM Sigcomm Conference, (Baltimore, MD 1992), pp. 248-257.

J. Byers, M. Luby, M. Mitzenmacher, A Rege, "A digital fountain approach to reliable distribution of bulk data," Proc. ACM Sigcomm Conference, (Vancouver, 1998), pp. 56-67

William Stallings. *Data and Computer Communications, Eighth Edition.* Prentice Hall, 2006.

Andrew S. Tanenbaum. *Computer Networks, Fourth Edition.* Prentice Hall PTR, 2003.

Alan Dennis. *Networking in the Internet Age.* John Wiley & Sons, 2002.

Charles E. Perkins, editor. *Ad Hoc Networking.* Addison-Wesley, 2001.

Radia Perlman. *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols, Second Edition.* Addison-Wesley, 2000.

Dimitri Bertsekas and Robert Gallager. *Data Networks, Second Edition.* Prentice Hall, 1992.

https://labprograms.files.wordpress.com/2009/09/modems.pdf

http://www.computerhope.com/jargon/m/modem.htm

https://www.cpe.ku.ac.th/~nguan/presentations/datacom/modem.pdf

http://electronicspost.com/what-is-multiplexing-describe-different-types-of-multiplexing-techniques/

http://electronicspost.com/what-is-multiplexing-frequency-division-multiplexing-fdm-and-time-division-multiplexing-tdm/

RIL-089

Uttar Pradesh Rajarshi Tandon
Open University

Master in Computer Science

MCS-108

Data Communication and
Computer Networks

Block

2

DATA LINK LAYER

© UPRTOU, Prayagraj. 2019
ISBN : 978-93-83328-25-3

Block-II Data Link Layer

This is the second block on Data communication & Networks and having detail description of data link layer. The data link layer or layer 2 is the second layer of the seven-layer OSI model of computer networking. This layer is the protocol layer that transfers data between adjacent network nodes in a wide area network (WAN) or between nodes on the same local area network (LAN) segment. We will detailed study of data link layer in this block.

So we will begin the first unit on LANs. As all we know the basics of LAN. Local area network is very important so firstly we introduce IEEE 802.4 and 802.5 protocols. In this unit we also introduce the performance of Ethernet and token ring protocols.

Second unit begins with Protocols. In this unit we define all the protocols like FDDI Protocol, Distributed Queue Dual Bus (DQDB) protocol. Network Layer Protocols. In this unit we also covers the topic design issues of network layer protocols, virtual circuits and datagrams.

In the third unit, we provide another important topic i.e. Error detection and correction. In this unit you will learn how a network handle the error and how this error correct. In this unit we also explain Parity Check Codes, Cyclic Redundancy Codes.

In the last, the fourth unit introduced the concept of Data link protocols. In this unit we describe Stop and Wait Protocols: Noise Free and Noisy Channels Performance and Efficiency, Verification of Protocols using Finite State Machine, HDLC Data Link Protocol.

As you study the material, you will find that figures, tables are properly used and these will help to understand the concept. There are many sections in the units to easily understand the topic. Every unit has summary and review questions in the end of the unit which will help you to review yourself.

In your study, you will find that the every unit has different equal length and your study time will vary for each unit.

We hope you enjoy studying the material and once again wish you all the best for your success.

UNIT-V

LANs

Structure

5.0 Introduction

This unit has contained detailed description about local area networks. There are five sections in this unit. In the first section i.e. Sec. 1.1 you will learn about LANs. As you have studied earlier the IEEE 802 LAN/MAN Standards Committee develops and maintains networking standards and recommended practices for local, metropolitan, and other area networks, using an open and accredited process, and advocates them on a global basis. The most widely used standards are for Ethernet, Bridging and Virtual Bridged LANs Wireless LAN, Wireless PAN, Wireless MAN, Wireless Coexistence, Media Independent Handover Services, and Wireless RAN. An individual Working Group provides the focus for each area. In the next section you will learn about IEEE 802.4 and 802.5 protocols. In the protocol 802.4 it is use in token bus and protocol 802.5 use by token ring. The original token-passing standard for twisted-pair, shielded copper cables. Supports copper and fiber cabling from 4 Mbps to 100 Mbps. Often called "IBM Token-Ring." In the Sec 1.3 you will learn about the performance of Ethernet and Token ring protocols. In Sec. 1.4 and 1.5 you will find summary and review questions respectively.

Objectives

After studying this unit you should be able to:

- ❖ Define Local Area Networks (LANs)
- ❖ Describe IEEE 802.4 and 802.5 Protocols
- ❖ Express performance of Ethernet and Token ring protocols.

5.1 Local Area Networks (LANs)

A network is any collection of independent computers that exchange information with each other over a shared communication medium. Local Area Networks or LANs are usually confined to a limited geographic area, such as a single building or a college campus. LANs can be small, linking as few as three computers, but can often link hundreds of computers used by thousands of people. The development of standard networking protocols and media has resulted in worldwide proliferation of LANs throughout business and educational organizations.

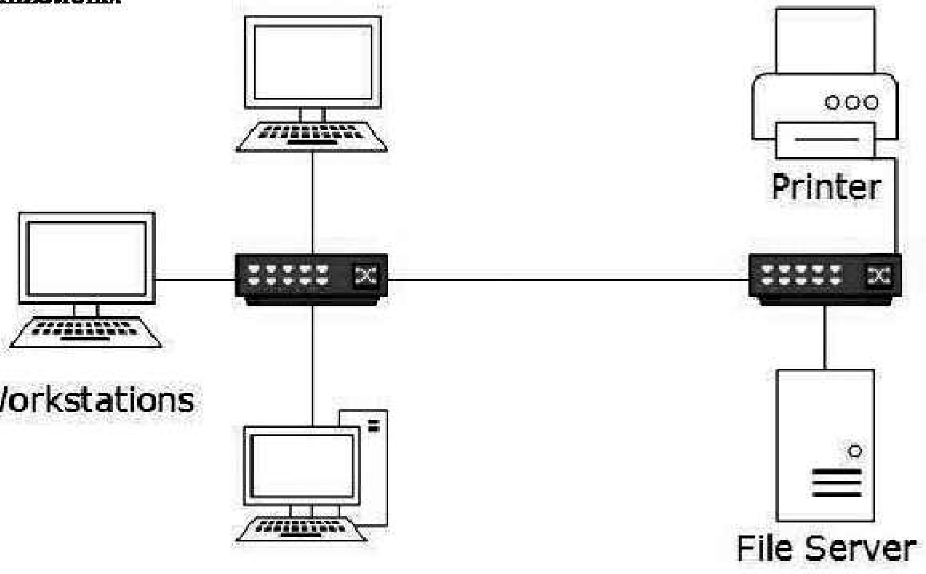


Figure 5.1- LAN

Types of LAN Technology

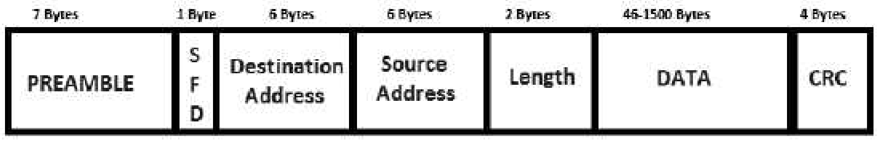
Ethernet

Ethernet is the most popular physical layer LAN technology in use today. It defines the number of conductors that are required for a connection, the performance thresholds that can be expected, and provides the framework for data transmission. A standard Ethernet network can transmit data at a rate up to 10 Megabits per second (10 Mbps). Other LAN types include Token Ring, Fast Ethernet, Gigabit Ethernet, 10 Gigabit Ethernet, Fiber Distributed Data Interface (FDDI), Asynchronous Transfer Mode (ATM) and LocalTalk.

Ethernet is popular because it strikes a good balance between speed, cost and ease of installation. These benefits, combined with wide acceptance in the computer marketplace and the ability to support virtually all popular network protocols, make Ethernet an ideal networking technology for most computer users today.

The Institute for Electrical and Electronic Engineers developed an Ethernet standard known as IEEE Standard 802.3. This standard defines rules for configuring an Ethernet network and also specifies how the elements in an Ethernet network interact with one another. By adhering to the IEEE standard, network equipment and network protocols can communicate efficiently.

Ethernet (IEEE 802.3) Frame Format –



IEEE 802.3 ETHERNET Frame Format

Figure 5.2- IEEE 802.3 Ethernet Frame Format

- **PREAMBLE** – Ethernet frame starts with 7-Bytes Preamble. This is pattern of alternative 0's and 1's which indicates starting of the frame and allow sender and receiver to establish bit synchronization. Initially, PRE (Preamble) was introduced to allow for the loss of few bits due to signal delays. But todays high-speed Ethernet don't need Preamble to protect the frame bits.

  - PRE (Preamble) indicates the receiver that frame is coming and allow the receiver to lock onto the data stream before the actual frame begins.

- **Start of frame delimiter (SFD)** – This is a 1-Byte field which is always set to 10101011. SFD indicates that upcoming bits are starting of frame, which is destination address. Sometimes SFD is considered the part of PRE, this is the reason Preamble is described as 8 Bytes in many places.
- **Destination Address** – This is 6-Byte field which contains the MAC address of machine for which data is destined.
- **Source Address** – This is a 6-Byte field which contains the MAC address of source machine. As Source Address is always an individual address (Unicast), the least significant bit of first byte is always 0.
- **Length** – Length is a 2-Byte field, which indicates the length of entire Ethernet frame. This 16-bit field can hold the length value between 0-65534, but length cannot be larger than 1500 because of some own limitations of Ethernet.
- **Data** – This is the place where actual data is inserted, also known as **Payload**. Both IP header and data will be inserted here, if Internet Protocol is used over Ethernet. The maximum data present may be as long as 1500 Bytes. In case data length is less than minimum length i.e. 46 bytes, then padding 0's is added to meet the minimum possible length.
- **Cyclic Redundancy Check (CRC)** – CRC is 4 Byte field. This field contains 32-bits hash code of data, which is generated over Destination Address, Source Address, Length and Data field. If the checksum

computed by destination is not same as sent checksum value, data received is corrupted.

Note – Size of frame of Ethernet IEEE 802.3 varies 64 bytes to 1518 bytes including data length (46 to 1500 bytes).

Fast Ethernet

The Fast Ethernet standard (IEEE 802.3u) has been established for Ethernet networks that need higher transmission speeds. This standard raises the Ethernet speed limit from 10 Mbps to 100 Mbps with only minimal changes to the existing cable structure. Fast Ethernet provides faster throughput for video, multimedia, graphics, Internet surfing and stronger error detection and correction.

There are three types of Fast Ethernet: 100BASE-TX for use with level 5 UTP cable; 100BASE-FX for use with fiber-optic cable; and 100BASE-T4 which utilizes an extra two wires for use with level 3 UTP cable. The 100BASE-TX standard has become the most popular due to its close compatibility with the 10BASE-T Ethernet standard.

Network managers who want to incorporate Fast Ethernet into an existing configuration are required to make many decisions. The number of users in each site on the network that need the higher throughput must be determined; which segments of the backbone need to be reconfigured specifically for 100BASE-T; plus what hardware is necessary in order to connect the 100BASE-T segments with existing 10BASE-T segments. Gigabit Ethernet is a future technology that promises a migration path beyond Fast Ethernet so the next generation of networks will support even higher data transfer speeds.

Gigabit Ethernet

Gigabit Ethernet was developed to meet the need for faster communication networks with applications such as multimedia and Voice over IP (VoIP). Also known as "gigabit-Ethernet-over-copper" or 1000Base-T, GigE is a version of Ethernet that runs at speeds 10 times faster than 100Base-T. It is defined in the IEEE 802.3 standard and is currently used as an enterprise backbone. Existing Ethernet LANs with 10 and 100 Mbps cards can feed into a Gigabit Ethernet backbone to interconnect high performance switches, routers and servers.

From the data link layer of the OSI model upward, the look and implementation of Gigabit Ethernet is identical to that of Ethernet. The most important differences between Gigabit Ethernet and Fast Ethernet include the additional support of full duplex operation in the MAC layer and the data rates.

10 Gigabit Ethernet

10 Gigabit Ethernet is the fastest and most recent of the Ethernet standards. IEEE 802.3ae defines a version of Ethernet with a nominal rate of 10Gbits/s that makes it 10 times faster than Gigabit Ethernet.

Unlike other Ethernet systems, 10 Gigabit Ethernet is based entirely on the use of optical fiber connections. This developing standard is moving away from a LAN design that broadcasts to all nodes, toward a system which includes some elements of wide area routing. As it is still very new, which of the standards will gain commercial acceptance has yet to be determined.

Virtual LAN

LAN uses Ethernet which in turn works on shared media. Shared media in Ethernet create one single Broadcast domain and one single Collision domain. Introduction of switches to Ethernet has removed single collision domain issue and each device connected to switch works in its separate collision domain. But even Switches cannot divide a network into separate Broadcast domains.

Virtual LAN is a solution to divide a single Broadcast domain into multiple Broadcast domains. Host in one VLAN cannot speak to a host in another. By default, all hosts are placed into the same VLAN.

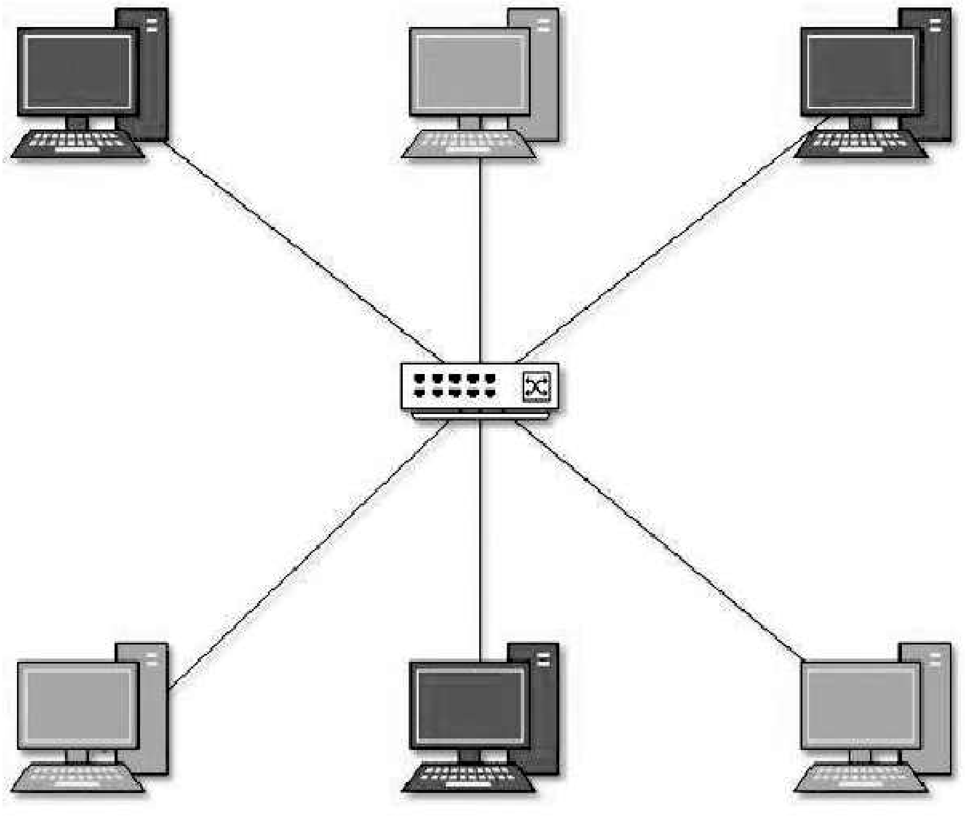


Figure 5.3- Virtual LAN

In this diagram, different VLANs are depicted in different color codes. Hosts in one VLAN, even if connected on the same Switch cannot see or speak to other hosts in different VLANs. VLAN is Layer-2 technology which works closely on Ethernet. To route packets between two different VLANs a Layer-3 device such as Router is required.

Asynchronous Transfer Mode (ATM)

ATM is a cell-based fast-packet communication technique that can support data-transfer rates from sub-T1 speeds to 10 Gbps. ATM achieves its high speeds in part by transmitting data in fixed-size cells and dispensing with error-correction protocols. It relies on the inherent integrity of digital lines to ensure data integrity.

ATM can be integrated into an existing network as needed without having to update the entire network. Its fixed-length cell-relay operation is the signaling technology of the future and offers more predictable performance than variable length frames. Networks are extremely versatile and an ATM network can connect points in a building, or across the country, and still be treated as a single network.

Power over Ethernet (PoE)

PoE is a solution in which an electrical current is run to networking hardware over the Ethernet Category 5 cable or higher. This solution does not require an extra AC power cord at the product location. This minimizes the amount of cable needed as well as eliminates the difficulties and cost of installing extra outlets.

LAN Technology Specifications

Name	IEEE Standard	Data Rate	Media Type	Maximum Distance
Ethernet	802.3	10Mbps	10Base-T	100 Meters
Fast Ethernet/100Base-T	802.3u	100Mbps	100Base-TX	100 Meters
			100Base-FX	2000 Meters
Gigabit Ethernet/GigE	802.3z	1000Mbps	1000Base-T	100 Meters
			1000Base-SX	275/550 Meters
			1000Base-LX	550/5000 Meters
10 Gigabit Ethernet	IEEE 802.3ae	10Gbps	10GBase-SR	300 Meters
			10GBase-LX4	300m MMF/10kmSMF
			10GBase-LR/ER	10km/40km
			10GBase-SW/LW/EW	300m/10km/40km

Table-5.1 LAN Technologies Specification

Token Ring

Token Ring is another form of network configuration. It differs from Ethernet in that all messages are transferred in one direction along the ring at all times. Token Ring networks sequentially pass a "token" to each connected device. When the token arrives at a particular computer (or device), the recipient is allowed to transmit data onto the network. Since only one device may be transmitting at any given time, no data collisions occur. Access to the network is guaranteed, and time-sensitive applications can be supported. However, these benefits come at a price. Component costs are usually higher, and the networks themselves are considered to be more complex and difficult to implement. Various PC vendors have been proponents of Token Ring networks.

5.2 IEEE 802.4 and 802.5 Protocols

802.4 Protocol

The 802.4 IEEE standard defines the Token Bus protocol for a token-passing access method on a bus topology. In a token-passing access method, a special packet called a token is passed from station to station and only the token holder is permitted to transmit packets onto the LAN. Thus, no collisions can occur with this protocol. When a station is done transmitting its packets, it passes the token to the "next" station. The next station does not need to be physically closest to this one on the bus, just the next logical station. A station can hold the token for only a certain amount of time before it must pass it on -even if it has not completed transmitting all of its data. This assures access to all stations on the bus within a specified period of time.

IEEE 802.4 Token Bus

In token bus Computer network station must have possession of a token before it can transmit on the computer network. The IEEE 802.4 Committee has defined **token bus** standards as broadband computer networks, as opposed to Ethernet's baseband transmission technique. Physically, the token bus is a linear or tree-shape cable to which the stations are attached

The topology of the computer network can include groups of workstations connected by long trunk cables. Logically, the stations are organized into a ring. These workstations branch from hubs in a star configuration, so the network has both a bus and star topology. Token bus topology is well suited to groups of users that are separated by some distance. **IEEE 802.4 token bus networks are constructed with 75-ohm coaxial cable using a bus topology**. The broadband characteristics of the 802.4 standard support transmission over several different channels simultaneously.

When the logical ring is initialized, the highest numbered station may send the first frame. The token and frames of data are passed from one station to another following the numeric sequence of the station addresses. Thus, the token follows a logical ring rather than a physical ring. The last station in numeric order passes the token back to the first station. The token does not follow the physical ordering of workstation attachment to the cable. Station 1 might be at one end of the cable and station 2 might be at the other, with station 3 in the middle.

In such a case, there is no collision as only one station possesses a token at any given time. In token bus, each station receives each frame; the station whose address is specified in the frame processes it and the other stations discard the frame

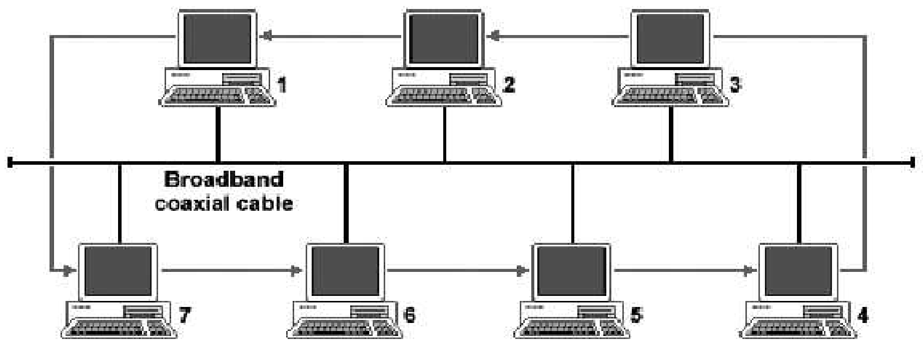


Figure 5.4- A Token Bus

MAC Sub layer Function

When the ring is initialized, stations are inserted into it in order of station address, from highest to lowest.

- ❖ Token passing is done from high to low address.
- ❖ Whenever a station acquires the token, it can transmit frames for a specific amount of time.
- ❖ If a station has no data, it passes the token immediately upon receiving it.
- ❖ The token bus defines four priority classes, 0, 2, 4, and 6 for traffic, with 0 the lowest and 6 the highest.
- ❖ Each station is internally divided into four substations, one at each priority level *i.e.* 0, 2, 4 and 6.
- ❖ As input comes in to the MAC sub layer from above, the data are checked for priority and routed to one of the four substations.
- ❖ Thus each station maintains its own queue of frames to be transmitted.
- ❖ When a token comes into the station over the cable, it is passed internally to the priority 6 substation, which can begin transmitting its frames, if it has any.
- ❖ When it is done or when its time expires, the token is passed to the priority 4 substation, which can then transmit frames until its timer

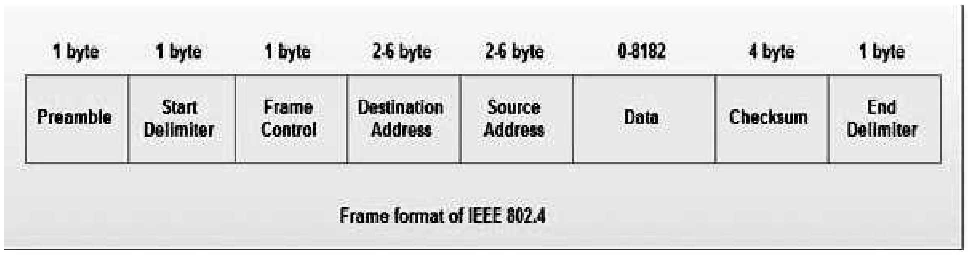
expires. After this the token is then passed internally to priority 2 substation.

- ❖ This process continues until either the priority 0 substation has sent all its frames or its time expires.
- ❖ After this the token is passed to the next station in the ring.

Frame format of Token Bus

The various fields present in the frame format are

1. **Preamble:** This. Field is at least 1 byte long. It is used for bit synchronization.



1 byte	1 byte	1 byte	2-6 byte	2-6 byte	0-8182	4 byte	1 byte
Preamble	Start Delimiter	Frame Control	Destination Address	Source Address	Data	Checksum	End Delimiter

Frame format of IEEE 802.4

Figure 5.5: Frame format of IEEE 802.4

2. **Start Delimiter:** This one byte field marks the beginning of frame.
3. **Frame Control:** This one byte field specifies the type of frame. It distinguishes data frame from control frames. For data frames it carries frame's priority. For control frames, it specifies the frame type. The control frame types include. Token passing and various ring maintenance frames, including the mechanism for letting new station enter the ring, the mechanism for allowing stations to leave the ring.
4. **Destination address:** It specifies 2 to 6 bytes destination address.
5. **Source address:** It specifies 2 to 6 bytes source address.
6. **Data:** This field may be up to 8182 bytes long when 2 bytes addresses are used & up to 8174 bytes long when 6 bytes address is used.
7. **Checksum:** This 4 byte field detects transmission errors.
8. **End Delimiter:** This one byte field marks the end of frame.

The various control frames used in token bus are:

Farme Control Field	Name	Meaning
00000000	Claim_token	Claim token during ring initialization
00000001	Solicit successor_1	Allow station to enter the ring
00000010	Solicit successor_2	Allow stations to enter the ring
00000011	Who_follows	Recover from lost token.
00000100	Resolve_contention	Used when multiple stations want to enter.
00001000	Token	Pass the token
00001100	Set successor	Allow station to leave the ring.

Figure 5.6: Control Frames in Token Bus

Periodically, a station will transmit a ***SOLIT_SUCCESSOR*** frame to solicit bids from stations wishing to join the ring. The frame includes the sender's address, and that of its current successor in the ring. Only stations with an address falling between these two addresses may bid to enter the ring (in order to maintain the logical order of station addresses on the ring). If no station bids to enter within a slot time, the response window is closed, and the token holder returns to its normal business. If only one station bids to enter, it is inserted into the ring and becomes the token holder's successor. If two or more stations bid to enter, their frames will collide and be garbled. The token holder then runs an arbitration process that begins with the broadcast of a ***RESOLVE_CONTENTION*** frame. The algorithm is a variation of *binary countdown*, using two bits at a time.

All station interfaces maintain two random bits which are used to delay all bids by 0, 1, 2 or 3 slot times to further reduce contention. Two stations will only collide on a bid, therefore, if the current two address bits being used are the same and they happen to have the same two random bits. To prevent stations that must wait 3 slot times from being at a permanent disadvantage, the random bits are regenerated either every time they are used, or every 50 msec.

The solicitation of new stations is not allowed to interfere with the guaranteed worst case for token rotation. Each station has a timer that is reset whenever it acquires the token. When the token arrives, the existing value of this timer (i.e. the previous token rotation time) is inspected before the timer is reset. If a pre-determined threshold value has been exceeded, recent levels of traffic have been considered to be too high, and no bids may be solicited this time round. In any case, only one station can enter the ring during each solicitation, to limit the amount of time that can be used for ring maintenance. There is no guaranteed time limit set on how long a station has to wait to enter the ring when traffic is heavy, but in practice it is not normally longer than a few seconds.

To leave the ring, a station ***X*** with a predecessor ***P*** and a successor ***S*** simply sends a ***SET_SUCCESSOR*** frame to ***P*** telling it that from now on its successor is ***S***. Station ***X*** then just stops transmitting.

Ring initialization is a special case of adding new stations. When the first station comes on line, it registers the fact that there is no traffic for a specified period. It then broadcasts a ***CLAIM_TOKEN*** frame. Not receiving a reply, it creates a token and sets up a ring consisting of just itself, and periodically solicits bids for new stations. As new stations are powered on, they will respond and join the ring, if necessary using the contention algorithm described above. If the first two stations are powered on simultaneously, they are allowed to bid for the token using the standard modified binary countdown algorithm and the two random bits.

Problems sometimes arise with the token or the logical ring due to transmission errors (for example a station tries to pass a token to a station which has been taken offline). After passing the token, therefore, a station monitors the network to determine whether its successor has either transmitted a frame or passed the token. If neither of these events occurs, it generates a second token. If that also fails to produce the required outcome, the station transmits a ***WHO_FOLLOWS*** frame specifying the address of its successor. When the failed station's successor sees a ***WHO_FOLLOWS*** frame naming its predecessor, it responds with a ***SET_SUCCESSOR*** frame, naming itself as the new successor. The failed station is then removed from the ring.

If two consecutive stations go offline, the ***WHO_FOLLOWS*** frame will fail to elicit a response. In this situation, the station that originally passed the token sends a ***SOLICIT_SUCCESSOR_2*** frame to see if any other stations are still active. The standard connection protocol is run once again, with all active stations bidding for a place until the ring is re-established. A problem can also occur if the token holder goes down and takes the frame with it. In this case, the ring initialization algorithm is used to re-establish the ring.

Multiple tokens on the ring are another problem, and if a station currently holding a token notices a transmission from another station, it discards its token. If multiple tokens are present on the network at the same time, this process is repeated until all but one of the tokens are discarded. If all of the tokens are discarded by accident, the lack of activity will cause one or more of the stations to try and claim the token.

802.5 Protocols

The 802.5 IEEE standard defines the Token Ring protocol which, like Token Bus, is another token-passing access method, but for a ring topology. A ring topology consists of a series of individual point-to-point links that form a circle.

A token is passed from station to station in one direction around the ring, and only the station holding the token can transmit packets onto the ring. Data packets travel in only one direction around the ring. When a station receives a packet addressed to it, it copies the packet and puts it back on the ring. When the originating station receives the packet, it removes the packet.

IEEE 802.5 Token Ring

Token ring is the IEEE 802.5 standard for a token-passing ring in Communication networks. A ring consists of a collection of ring interfaces connected by point-to-point lines *i.e.* ring interface of one station is connected to the ring interfaces of its left station as well as right station. Internally, signals travel around the Communication network from one station to the next in a ring.

These point-to-point links can be created with twisted pair, coaxial cable or fiber optics. Each bit arriving at an interface is copied into a 1-bit buffer. In this buffer the bit is checked and may be modified and is then copied out to the ring again. This copying of bit in the buffer introduces a 1-bit delay at each interface.

Token Ring is a LAN protocol defined in the IEEE 802.5 where all stations are connected in a ring and each station can directly hear transmissions only from its immediate neighbor. Permission to transmit is granted by a message (token) that circulates around the ring. A token is a special bit pattern (3 bytes long). There is only one token in the network.

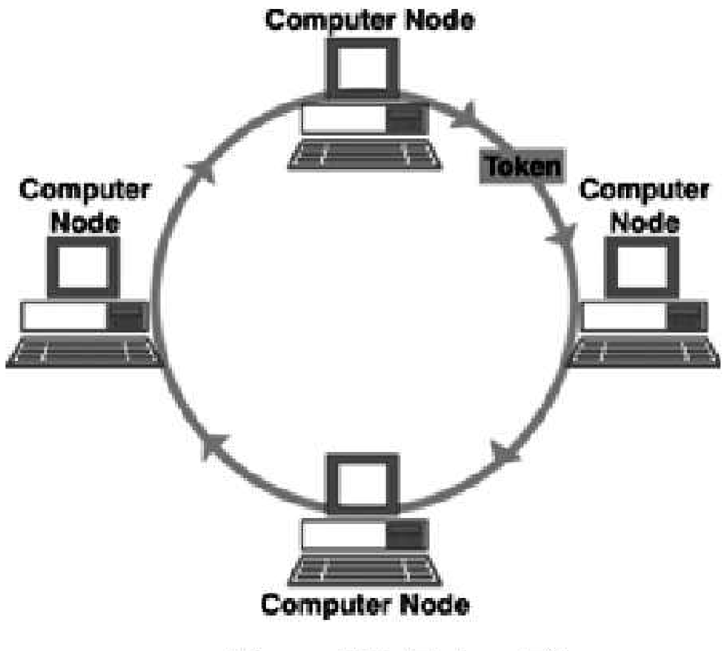


Figure 5.7: Token Ring

Token-passing networks move a small frame, called a token, around the network. Possession of the token grants the right to transmit. If a node receiving the token in order to transmit data, it seizes the token, alters 1 bit of the token (which turns the token into a start-of-frame sequence), appends the information that it wants to transmit, and sends this information to the next station on the ring. Since only one station can possess the token and transmit data at any given time, there are no collisions.

There are two operating modes of ring interfaces. They are listen and transmit. In listen mode, the input bits are simply copied to output with a delay of 1- bit time. In transmit mode the connection between input and output is broken by the interface so that it can insert its own data. The station comes in transmit mode when it captures the token.

The frames are acknowledged by the destination in a very simple manner. The sender sends frames to receiver with ACK bit 0. The receiver on receiving frames, copies data into its buffer, verifies the checksum and set the ACK bit to 1. The verified frames come back to sender, where they are removed from the ring.

The information frame circulates the ring until it reaches the intended destination station, which copies the information for further processing. The information frame continues to circle the ring and is finally removed when it reaches the sending station. The sending station can check the returning frame to see whether the frame was seen and subsequently copied by the destination.

A station can hold a token for a specific duration of time. During this time, it has to complete its transmission and regenerates the token in ring. Whenever a station finishes its transmissions, the other station grabs the token and starts its own transmission.

Handling cable breakage in ring networks

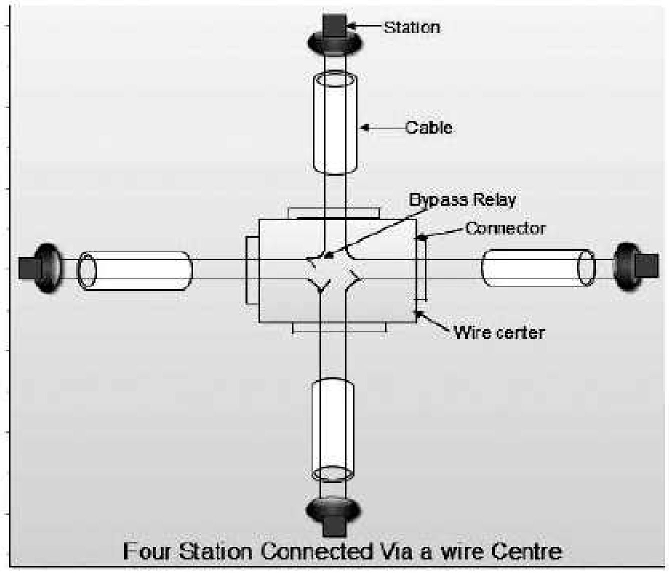


Figure 5.8: Handling Cable Breakage

- If the cable breaks, the entire ring network goes down. This can completely stop the propagation of token in the ring.
- This problem can be solved by using wire center as shown in Fig. 1.8.
- This wire center bypasses the terminal that has gone down in following manner:
    - Each station is connected to wire center by a cable containing two twisted pairs, one for data to station and one for data from the station.
    - Inside the wire center are bypass relays that are energized by the current from the stations.
    - If the ring breaks or a station goes down loss of drive current will release the relay and bypass the station.

Frame format

Two basic frame types are used - tokens, and data/command frames. The token is three bytes long and consists of a *start delimiter*, an *access control byte*, and an *end delimiter*. The format of the token is shown below.

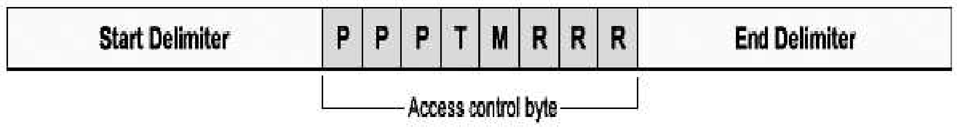
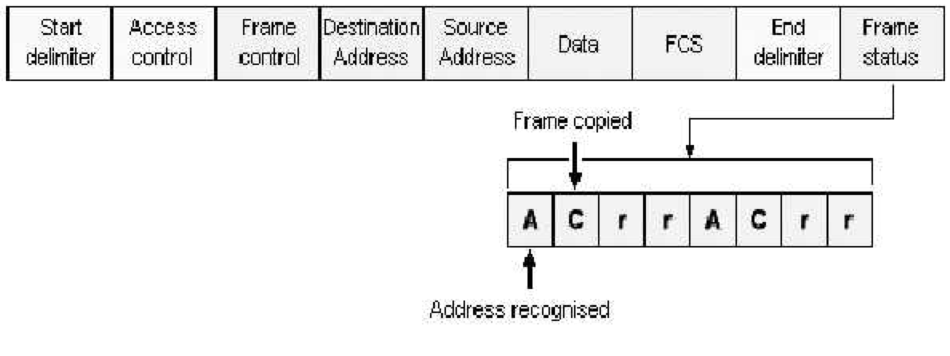


Figure 5.9: The Token Ring token

A data/command frame has the same fields as the token, plus several additional fields. The format of the data/command frame is shown below.

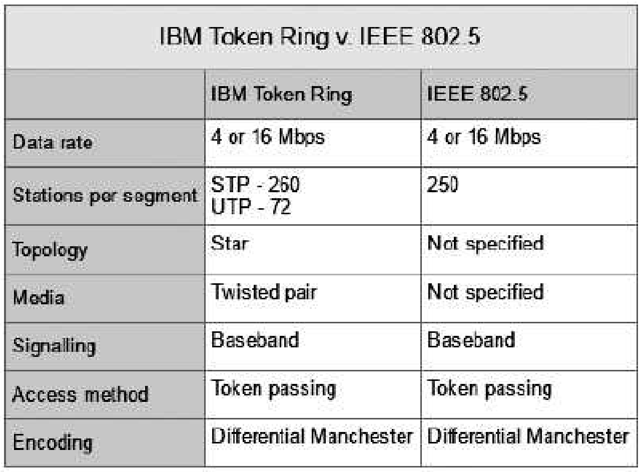


- **Figure 5.10: The Token Ring frame format**

- ***Start delimiter*** - alerts each station of the arrival of a token or frame.
- ***Access control byte*** - contains the *priority field*, the *reservation field*, the *token bit* and a *monitor bit*.
- ***Frame control byte*** - indicates whether the frame contains data or control information. In a control frame, this byte specifies the type of control information carried.
- ***Destination and source addresses*** - two six-byte fields that identify the destination and source station MAC addresses.
- ***Data*** - the maximum length is limited by the ring *token holding time*, which defines the maximum time a station can hold the token
- ***Frame check sequence (FCS)*** - filled by the source station with a calculated value dependent on the frame contents. The destination station recalculates the value to determine whether the frame was damaged in transit. If so, the frame is discarded.
- ***End delimiter*** - signals the end of the token or frame, and contains bits that may be used to indicate a damaged frame, and to identify the last frame in a logical sequence.
- ***Frame status*** - a one-byte field that terminates a frame, and includes the one-bit *address-recognized* and *frame-copied* fields. These one-bit fields, if set, provide confirmation that the frame has been delivered to the source address and the data read. Both fields are duplicated within the frame status byte.

If the network is quiet and none of the stations has any data to transmit, the token simply circulates around the ring continuously. When a station has data to transmit, it waits until it receives the token, marks it as "busy" by setting the *token bit*, adds the data and/or control information to create a data or command frame, and transmits the frame to the next station. Each station that receives the frame will re-transmit the frame to the next station until it reaches the destination station. This station reads the data, sets the *address recognized* and *frame copied* bits in the *frame status* field, and transmits the frame to the next node. When the frame arrives back at its point of origin, the originating station generates a new token, which it transmits to the next station, even if it has further data to send. In this way, each station network has an equal number of opportunities to transmit data. Because only one token is allowed to exist on the network, only one station can transmit at any one time, and collisions cannot occur. Although the IEEE 802.5 specification reflects IBM's Token Ring technology, the specifications differ slightly. IBM specifies a star topology, with all end stations star-wired to a *multi-station access unit* (MSAU), whereas IEEE 802.5 does not specify a topology (although virtually all IEEE 802.5

implementations were based on a star). In addition, IEEE 802.5 does not specify a media type, while IBM originally specified shielded twisted pair cable. The table below summarizes the IBM and IEEE 802.5 specifications.



IBM Token Ring v. IEEE 802.5		
	IBM Token Ring	IEEE 802.5
Data rate	4 or 16 Mbps	4 or 16 Mbps
Stations per segment	STP - 260 UTP - 72	250
Topology	Star	Not specified
Media	Twisted pair	Not specified
Signalling	Baseband	Baseband
Access method	Token passing	Token passing
Encoding	Differential Manchester	Differential Manchester

Table 5.2: IBM Token Ring vs. IEEE 802.5

Priority System

Token Ring networks provide a user-configurable priority system that allows stations that are designated as having a high-priority to use the network more frequently. Token Ring frames have two fields that control priority - the *priority field*, and the *reservation field*. Only stations with a priority equal to, or higher than, the value contained in a token's priority field can acquire the token. Once the token is in use, only stations with a priority value higher than that of the transmitting station can reserve the token for the next pass around the network. When the next token is generated, it is set to the priority of the reserving station. Any station that raises the token's priority level must restore it to the previous level after use.

Fault Management

One station (it can be any station on the network) is selected to be the *active monitor*. The active monitor acts as a central source of timing information for the other stations on the network, and performs various maintenance functions, including making sure that there is always a token available on the network. The active monitor also sets the *monitor bit* on any data or command frame it encounters on the ring so that, in the event that a sending device fails after transmitting a frame, the frame can be prevented from circling the ring endlessly and thereby denying access to the network for other stations. If the active monitor receives a frame with the monitor bit already set, it removes the frame from the ring and generates a new token.

The use of a multi station access unit (or *wiring center*) in a star topology contributes to network reliability, since these devices can be configured to check for problems and remove faulty stations from the ring if necessary. A Token Ring algorithm called *beaconing* can be used to detect certain types of network fault. When a station detects a serious problem on the network (a cable break, for example), it transmits a *beacon frame* which initiates an *auto-reconfiguration* process. Stations that receive a beacon frame perform diagnostic procedures and attempt to reconfigure the network around the failed areas. Much of this reconfiguration process can be handled internally by the MSAU. The MSAU contains relays that switch a computer into the ring when it is turned on, or out of the ring when the computer is powered off. A MSAU has a number of ports to which network devices can be connected, a *ring-out* port allowing the unit to be connected to another MSAU, and a *ring-in port* that can accept an incoming connection from another MSAU. A number of MSAUs can thus be connected together in daisy-chain fashion to create a larger network. The ring-out port of the last MSAU in the chain must be connected back to the ring-in port of the first MSAU.

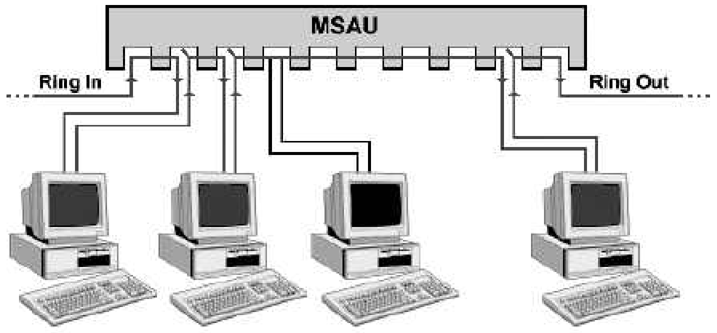


Figure 5.11: Connections in a multi-station access unit

Check your progress

Q1. Define LAN in detail?

Q2. Explain IEEE 802.4 and IEEE 802.5 protocols.

5.3 Performance of Ethernet and Token ring protocols.

Token Ring networks are deterministic in nature -nodes may only transmit at certain well defined times. Result is high bandwidth efficiency. Up to 90% in Token Ring, 40% in Ethernet.

Guaranteed sequential access to network eliminates fluctuating response times experienced on other network topologies.

Token Ring performance does not deteriorate to the same extent as Ethernet when network traffic increases. This means that at high loads, the presence of collisions of data frames on Ethernet networks becomes a major problem and can seriously affect the throughput.

By its nature Token Ring has a higher reliability, the ring can continue normal operation in most cases despite any single fault.

Ethernet has an advantage over Token Ring in that the cost of network equipment is lower for Ethernet. Token Ring networks tend to be more expensive to set up and maintain than Ethernet, although hardware costs for Token Ring are decreasing.

Advances in Ethernet technology have tended to be much more rapid than Token Ring. Gigabit Ethernet being an example of this. Token Ring technologies are being developed however to allow data transfer rates of 100Mbps using technologies such as Token Ring Switching.

Sr. No.	Parameters	Descriptions
1	Cost	Ethernet is generally less expensive and easier to install than Token Ring.
2	Stability	Token Ring is generally more secure and more stable than Ethernet.
3	Scalability	It is usually more difficult to add more computers on a Token Ring LAN than it is to an Ethernet LAN. However, as additional computers are added, performance degradation will be less pronounced on the Token Ring LAN than it will be on the Ethernet LAN.
4	QoS	Ethernet uses CSMA/CD media access control and Token Ring uses token passing. This makes Ethernet better suited in a situation where there are a large number of computers sending fewer, larger data frames. Token Ring is better suited for small to medium size LANs sending many, smaller data frames.

Table 3: Performance Comparison Ethernet vs. Token Ring

5.4 Summary

In this unit you have learnt about Local Area Networks (LANs), IEEE 802.4 and 802.5 Protocols and performance of Ethernet and Token ring protocols.

❖ A local area network (LAN) is a group of computers and associated devices that share a common communications line or wireless link to a server. Typically, a LAN encompasses computers and peripherals connected to a server within a distinct geographic area such as an office or a commercial establishment.

❖ Token bus is a network implementing the token ring protocol over a "virtual ring" on a coaxial cable. A token is passed around the network nodes and only the node possessing the token may transmit. If a node doesn't have anything to send, the token is passed on to the next node on the virtual ring. Each node must know the address of its neighbor in the ring, so a special protocol is needed to notify the other nodes of connections to, and disconnections from, the ring.

❖ Token ring local area network (LAN) technology is a communications protocol for local area networks. It uses a special three-byte frame called a "token" that travels around a logical "ring" of workstations or servers. This token passing is a channel access method providing fair access for all stations, and eliminating the collisions of contention-based access methods.

5.5 Review Questions

Q1. Define LAN in depth.

Q2. What is token bus? Explain its working. Also explain the advantages of token bus.

Q3. What is token ring? Why do we need token ring? Elaborate your answer.

Q4. Compare the performance of Ethernet and token ring.

UNIT-VI

Protocols: FDDI, DQDB, Network Layers

Structure

6.0 Introduction

In this unit we focused on different protocols like FDDI, DQDB and network layers. This unit divided into six sections. In the section 6.1 you will learn about FDDI protocol. FDDI (Fiber Distributed Data Interface) is a set of ANSI and ISO standards for data transmission on fiber optic lines in a local area network (LAN) that can extend in range up to 200 km (124 miles). The FDDI protocol is based on the token ring protocol. In addition to being large geographically, an FDDI local area network can support thousands of users. FDDI is frequently used on the backbone for a wide area network (WAN). In the Sec 6.2 you will know about Distributed Queue Dual Bus (DQDB) protocol. Distributed Queue Dual Bus (DQDB) is a Data-link layer communication protocol for Metropolitan Area Networks (MANs), specified in the IEEE 802.6 standard and designed for use in MANs. DQDB is designed for data as well as voice and video transmission and is based on cell switching technology (similar to ATM). DQDB, which permits multiple systems to interconnect using two unidirectional logical buses, is an open standard that is designed for compatibility with carrier transmission standards such as SMDS. In next section we described about network layer protocols its design issues. In Sec. 6.4 we explained virtual circuits and datagrams. In Sec. 6.5 and 6.6 you will find summary and review questions respectively.

Objectives

After studying this unit you should be able to:

- ❖ Define FDDI Protocol.
- ❖ Describe Distributed Queue Dual Bus (DQDB) protocol.
- ❖ Express Network Layer Protocols, its Design issues and Virtual Circuits and Datagrams.

6.1 FDDI Protocol

A high-speed network technology, conforming to the Open Systems Interconnection (OSI) reference model for networking and the American National Standards Institute (ANSI) standard X3T9, which runs at 100 Mbps over fiber-optic cabling; often used for network backbones in a local area network (LAN) or metropolitan area network (MAN).

The full form of FDDI is Fiber Distributed Data Interface. It is a data link layer protocol that is used for long distance networks provides communication with fiber optic lines up to 200 kilometers at a speed of 100 megabit per second (Mbps). FDDI is frequently used as high-speed backbone technology because of its support for high bandwidth and greater distances than copper. FDDI networks are typically used as backbones for wide-area networks. It can be used to interconnect LANs using other protocols. FDDI-II is a version of FDDI that adds the capability to add circuit-switched service to the network so that voice signals can also be handled. There is one more thing, FDDI uses a dual-ring architecture with traffic on each ring flowing in opposite directions. The dual-rings consist of a primary and a secondary ring. During normal operation, the primary ring is used for data transmission, and the secondary ring remains idle.

FDDI Specifications

FDDI is defined by four separate specifications (see Figure 6.1):

- ❖ Media Access Control (MAC)-Defines how the medium is accessed, including frame format, token handling, addressing, algorithm for calculating a cyclic redundancy check value, and error recovery mechanisms.
- ❖ Physical Layer Protocol (PHY)-Defines data encoding/decoding procedures, clocking requirements, framing, and other functions.
- ❖ Physical Layer Medium (PMD)-Defines the characteristics of the transmission medium, including the fiber-optic link, power levels, bit error rates, optical components, and connectors.
- ❖ Station Management (SMT)-Defines the FDDI station configuration, ring configuration, and ring control features, including station insertion and removal, initialization, fault isolation and recovery, scheduling, and collection of statistics.

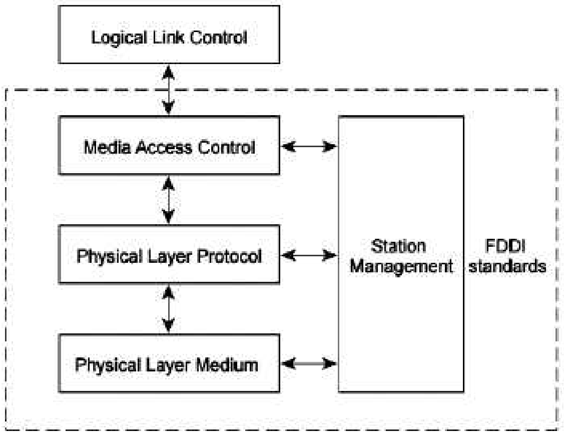


Figure 6.1 FDDI Standards

Physical Connections

FDDI specifies the use of dual rings. Traffic on these rings travels in opposite directions. Physically, the rings consist of two or more point-to-point connections between adjacent stations. One of the two FDDI rings is called the primary ring; the other is called the secondary ring. The primary ring is used for data transmission, while the secondary ring is generally used as a backup.

Class B or single-attachment stations (SAS) attach to one ring; Class A or dual-attachment stations (DAS) attach to both rings. SASs are attached to the primary ring through a concentrator, which provides connections for multiple SASs. The concentrator ensures that failure or power down of any given SAS does not interrupt the ring. This is particularly useful when PCs, or similar devices that frequently power on and off, connect to the ring.

A typical FDDI configuration with both DASs and SASs is shown in Figure 6.2.

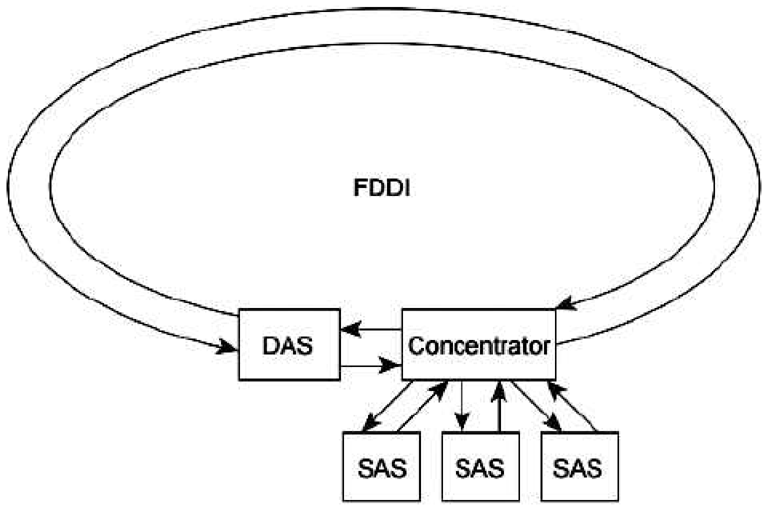


Figure 6.2 FDDI Nodes DAS, SAS and Concentrator.

Each FDDI DAS has two ports, designated A and B. These ports connect the station to the dual FDDI ring. Therefore, each port provides a connection for both the primary and the secondary ring, as shown in Figure 2-3.

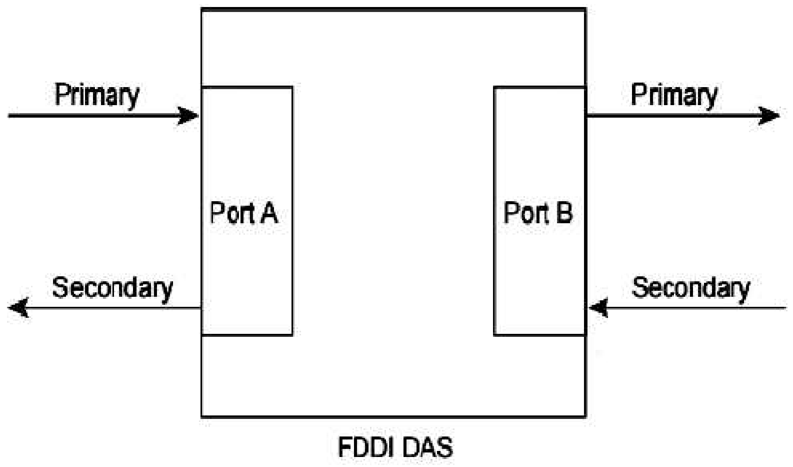


Figure 6.3 FDDI DAS Ports

Traffic Types

FDDI supports real-time allocation of network bandwidth, making it ideal for a variety of different application types. FDDI provides this support by defining two types of traffic: synchronous and asynchronous. Synchronous traffic can consume a portion of the 100-Mbps total bandwidth of an FDDI network, while asynchronous traffic can consume the rest. Synchronous bandwidth is allocated to those stations requiring continuous transmission capability. Such capability is useful for transmitting voice and video information, for example. Other stations use the remaining bandwidth asynchronously. The FDDI SMT specification defines a distributed bidding scheme to allocate FDDI bandwidth. Asynchronous bandwidth is allocated using an eight-level priority scheme. Each station is assigned an asynchronous priority level. FDDI also permits extended dialogues, where stations may temporarily use all asynchronous bandwidth. The FDDI priority mechanism can essentially lock out stations that cannot use synchronous bandwidth and have too low an asynchronous priority.

Frame Format

FDDI frame formats (shown in Figure -2.4) are similar to those of Token Ring.

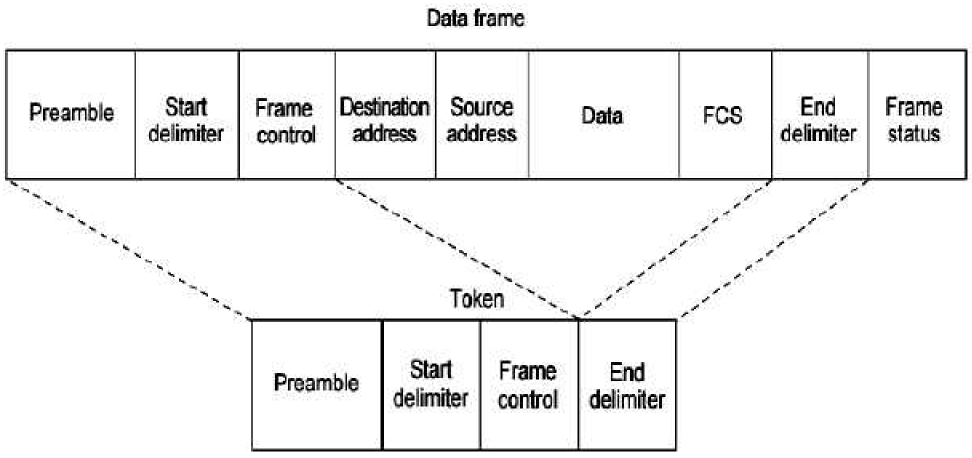


Figure 2.4 FDDI Frame Format

The fields of an FDDI frame are as follows:

- ❖ Preamble-Prepares each station for the upcoming frame.
- ❖ Start delimiter-Indicates the beginning of the frame. It consists of signaling patterns that differentiate it from the rest of the frame.
- ❖ Frame control-Indicates the size of the address fields, whether the frame contains asynchronous or synchronous data, and other control information.
- ❖ Destination address-Contains a unicast (singular), multicast (group), or broadcast (every station) address. As with Ethernet and Token Ring, FDDI destination addresses are 6 bytes.
- ❖ Source address-Identifies the single station that sent the frame. As with Ethernet and Token Ring, FDDI source addresses are 6bytes.
- ❖ Data-Contains either information destined for an upper-layer protocol or control information.
- ❖ Frame check sequence (FCS)-Filled by the source station with a calculated cyclic redundancy check (CRC) value dependent on the frame contents (as with Token Ring and Ethernet). The destination station recalculates the value to determine whether the frame may have been damaged in transit. If so, the frame is discarded.
- ❖ End delimiter-Contains non-data symbols that indicate the end of the frame.
- ❖ Frame status-Allows the source station to determine if an error occurred and if the frame was recognized and copied by a receiving station.

How FDDI Works

Fiber Distributed Data Interface (FDDI) is usually implemented as a dual token-passing ring within a ring topology (for campus networks) or star topology (within a building). The dual ring consists of a primary and secondary ring. The primary ring carries data. The counter-rotating secondary ring can carry data in the opposite direction, but is more commonly reserved as a backup in case the primary ring goes down. This provides FDDI with the degree of fault tolerance necessary for network backbones. In the event of a failure on the primary ring, FDDI automatically reconfigures itself to use the secondary ring as shown in the figure 6.5. Faults can be located and repaired using a fault isolation technique called beaconing. However, the secondary ring can also be configured for carrying data, extending the maximum potential bandwidth to 200 Mbps.

Stations connect to one (or both) rings using a media interface connector (MIC). Its two fiber ports can be either male or female, depending on the implementation. There are two different FDDI implementations, depending on whether stations are attached to one or both rings:

❖ **Single-attached stations (Class B stations):**

Connect to either the primary or secondary ring using M ports. Single-attached FDDI uses only the primary ring and is not as commonly deployed for network backbones as dual-attached FDDI. Single-attached stations are used primarily to connect Ethernet LANs or individual servers to FDDI backbones.

Dual-attached stations (Class A stations):

Connect to both rings. The A port is the point at which the primary ring enters and the secondary ring leaves; the B port is the reverse. M ports provide attachment points for single-attached stations. Dual-attached FDDI uses both rings, with the secondary ring serving as a backup for the primary. Dual-attached FDDI is used primarily for network backbones that require fault tolerance. Single-attached stations can be connected to dual-attached FDDI backbones using a dual-attached device called a concentrator or multiplexer.

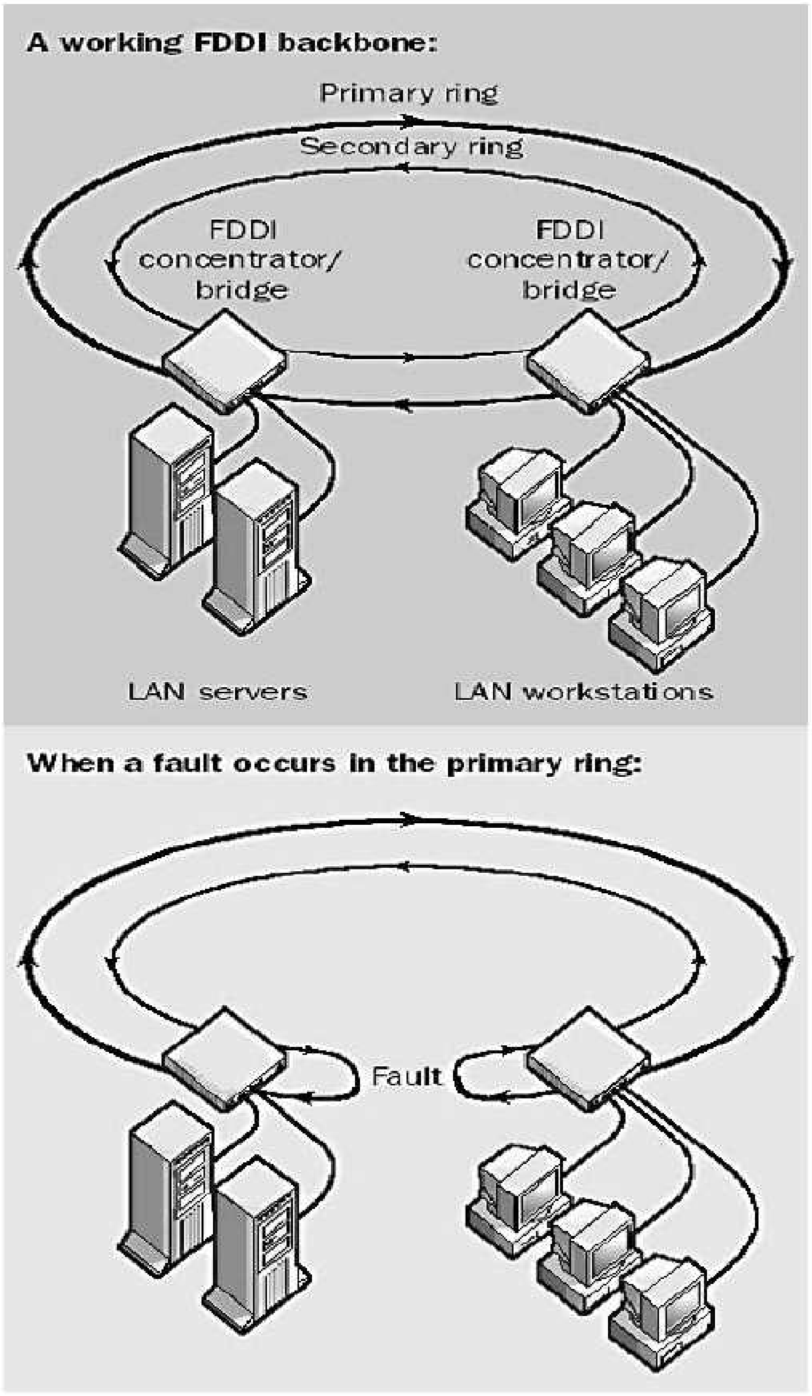

A working FDDI backbone:
Primary ring
Secondary ring
FDDI concentrator/ bridge
FDDI concentrator/ bridge
LAN servers
LAN workstations
When a fault occurs in the primary ring:
Fault

Figure 6.5 Working of FDDI

FDDI uses a timed token-passing technology similar to that of token ring networks as defined in the IEEE 802.5 standard. FDDI stations generate a token that controls the sequence in which other stations will gain access to the wire. The token passes around the ring, moving from one node to the next. When a station wants to transmit information, it captures the token, transmits as many frames of information as it wants (within the specified access period), and then releases the token. This feature of transmitting multiple data frames per token capture is known as a capacity allocation scheme, in contrast to the priority mechanism used in the IEEE 802.5 token ring standard. Every node on the ring checks the frames. The recipient station then reads the information from the frames, and when the frames return to the originating station, they are stripped from the ring.

There can be up to 500 stations on a dual-ring FDDI network. The maximum circumference for an FDDI ring is 100 kilometers (or 200 kilometers for both rings combined), and there must be a repeater every 2 kilometers or less. Bridges or routers are used to connect the FDDI backbone network to Ethernet or token ring departmental LANs. For these reasons, FDDI is not often used as a wide area network (WAN) solution, but is more often implemented in campus-wide networks as a network backbone.

FDDI makes a great network backbone for an Ethernet or Token Ring network. Put your servers directly on the FDDI ring to increase server performance. When bridging between Ethernet LANs and FDDI backbones, be aware that there are two different types of bridges:

- **Encapsulating bridges:**
  - Encapsulate Ethernet frames into FDDI frames
- **Translating bridges:**

Translate source and destination MAC addresses into FDDI addresses

These two FDDI bridging technologies can cause incompatibilities. For example, while Cisco FDDI bridges can generally interoperate with translating bridges from other vendors, their encapsulation method is proprietary and usually won't work with encapsulating bridges from other vendors. Both types of bridging methods are commonly used in FDDI networks.

6.2 Distributed Queue Dual Bus (DQDB) protocol.

Distributed Queue Dual Bus

Distributed Queue Dual Bus (DQDB) is a Data-link layer communication protocol for Metropolitan Area Networks (MANs), specified in the IEEE 802.6 standard and designed for use in MANs. DQDB is designed for data as well as voice and video transmission and is based on cell switching technology (similar to ATM). DQDB, which permits multiple systems to interconnect using two unidirectional logical buses, is an open standard that is designed for compatibility with carrier transmission standards such as SMDS.

For a MAN to be effective it requires a system that can function across long, "city-wide" distances of several miles, have a low susceptibility to error, adapt to the number of nodes attached and have variable bandwidth distribution. Using DQDB, networks can be thirty miles long and function in the range of 34 Mbps to 155 Mbps. The data rate fluctuates due to many hosts sharing a dual bus, as well as to the location of a single host in relation to the frame generator, but there are schemes to compensate for this problem making DQDB function reliably and fairly for all hosts.

The DQDB is composed of two bus lines with stations attached to both and a frame generator at the end of each bus. The buses run in parallel in such a fashion as to allow the frames generated to travel across the stations in opposite directions. Below is a picture of the basic DQDB architecture.

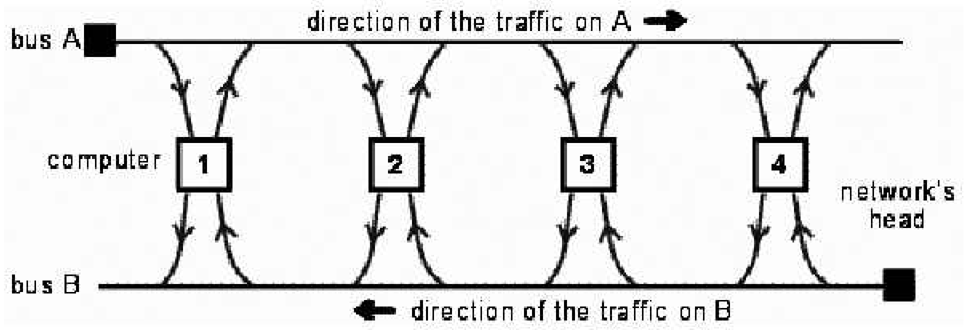


Figure 6.6: DQDB Architecture

DQDB Architecture

- Each bus supports traffic in only one direction
- Beginning of bus is denoted by a square and end by a triangle
- Bus B traffic moves from right to left and Bus A traffic from left to right
- Each bus connects to stations directly through input and output ports
- The DQDB is composed of a two bus lines with stations attached to both and a cell (Empty slots) generator at the start of each bus.
- The buses run in parallel in such a fashion as to allow the cells generated to travel across the stations in opposite directions.
- The cell generator (head-end) is constantly producing empty cells consisting of fifty-three bytes (a five byte header and a forty-eight byte payload).

Upstream & Downstream

- **As Bus A is configured**
  - **Stations 2 & 3 are considered to be upstream w.r.t station 1**
  - **Stations 1 & 2 are considered to be downstream w.r.t. station 3**
- **As Bus B is configured**

  - ❖ Station 2 & 3 are considered to be downstream w.r.t. station 1

- ❖ Stations 1 & 2 are considered to be upstream w.r.t. station 3
- ❖ **DQDB Working**
- ❖ Head-ends generate fixed size cells in both directions (cell generators)
- ❖ To transmit, a host must know whether the destination is to its right or its left
  - ❖ If right, the host must send on left bus
  - ❖ If left, the host must send on the right bus
- ❖ A "Distributed Queue" is used to make sure that cells are transmitted on a first-come first-serve basis
- ❖ **Technical Facts of DQDB**
- ❖ Distance up to 200 KM
- ❖ Medium: Copper or Fiber
- ❖ At distance up to 160 KM approx. speed is 44.73 Mbps (Copper)
- ❖ At distance up to 100 KM approx. speed is 150 Mbps (Fiber)
- ❖ Transmission Rate: 34 Mbps to 150 Mbps
- ❖ **DQDB Features**
- ❖ DQDB is a DLL communication protocol for MAN
- ❖ Unlike FDDI, DQDB is an IEEE standard: 802.6
- ❖ Designed for both voice & video
- ❖ Topology used: Dual Bus - uses 2 unidirectional logical buses
- ❖ Extend up to 30 miles at 34-55 Mbps
- ❖ Uses optical fiber links
- ❖ Queued-packet distributed switch (QPSX) algorithm
- ❖ Works on Data-link layer (especially in MAC sub-layer)
- ❖ Used in data, voice and video transmissions
- ❖ Used in data over cable services
- ❖ Based on Cell Relay Technology (like ATM)
- ❖ Provides connection-oriented, connection less services & asynchronous services

Check your progress

Q1. Define FDDI protocol

Q2. Explain DQDB protocol.

6.3 Network Layer Protocols

Every computer in a network has an IP address by which it can be uniquely identified and addressed. An IP address is Layer-3 (Network Layer) logical address. This address may change every time a computer restarts. A computer can have one IP at one instance of time and another IP at some different time.

Address Resolution Protocol (ARP)

The Address Resolution Protocol (ARP) was developed to enable communications on an internetwork and is defined by RFC 826. Layer 3 devices need ARP to map IP network addresses to MAC hardware addresses so that IP packets can be sent across networks. Before a device sends a datagram to another device, it looks in its ARP cache to see if there is a MAC address and corresponding IP address for the destination device. If there is no entry, the source device sends a broadcast message to every device on the network. Each device compares the IP address to its own. Only the device with the matching IP address replies to the sending device with a packet containing the MAC address for the device (except in the case of "proxy ARP"). The source device adds the destination device MAC address to its ARP table for future reference, creates a data-link header and trailer that encapsulates the packet, and proceeds to transfer the data. The figure below illustrates the ARP broadcast and response process.

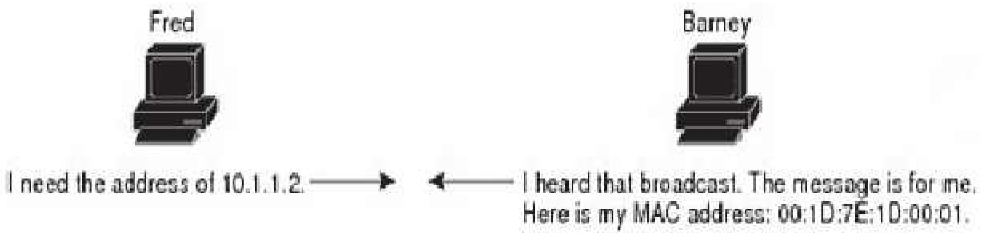

Fred
Barney
I need the address of 10.1.1.2.
I heard that broadcast. The message is for me.
Here is my MAC address: 00:1D:7E:1D:00:01.

Figure 6.7: ARP Process

When the destination device lies on a remote network, one beyond another Layer 3 device, the process is the same except that the sending device sends an ARP request for the MAC address of the default gateway. After the address is resolved and the default gateway receives the packet, the default gateway broadcasts the destination IP address over the networks connected to it. The Layer 3 device on the destination device network uses

ARP to obtain the MAC address of the destination device and delivers the packet.

Encapsulation of IP datagrams and ARP requests and replies on IEEE 802 networks other than Ethernet use Sub-network Access Protocol (SNAP).

The ARP request message has the following fields:

- ❖ HLN--Hardware address length. Specifies how long the hardware addresses are in the message. For IEEE 802 MAC addresses (Ethernet) the value is 6.
- ❖ PLN--Protocol address length. Specifies how long the protocol (Layer 3) addresses are in the message. For IPv4, the value is 4.
- ❖ OP—Op-code. Specifies the nature of the message by code:
    - ❖ 1--ARP request.
    - ❖ 2--ARP reply.
    - ❖ 3 through 9--RARP and Inverse ARP requests and replies.
- ❖ SHA--Sender hardware address. Specifies the Layer 2 hardware address of the device sending the message.
- ❖ SPA--Sender protocol address. Specifies the IP address of the sending device.
- ❖ THA--Target hardware address. Specifies the Layer 2 hardware address of the receiving device.
- ❖ TPA--Target protocol address. Specifies the IP address of the receiving device.

ARP Caching

Because the mapping of IP addresses to media access control (MAC) addresses occurs at each hop (Layer 3 device) on the network for every datagram sent over an internetwork, performance of the network could be compromised. To minimize broadcasts and limit wasteful use of network resources, Address Resolution Protocol (ARP) caching was implemented.

ARP caching is the method of storing network addresses and the associated data-link addresses in memory for a period of time as the addresses are learned. This minimizes the use of valuable network resources to broadcast for the same address each time a datagram is sent. The cache entries must be maintained because the information could become outdated, so it is critical that the cache entries are set to expire periodically. Every device on a network updates its tables as addresses are broadcast.

There are static ARP cache entries and dynamic ARP cache entries. Static entries are manually configured and kept in the cache table on a permanent basis. Static entries are best for devices that have to communicate with other devices usually in the same network on a regular basis. Dynamic entries are added by Cisco software, kept for a period of time, and then removed.

Static and Dynamic Entries in the ARP Cache

Static routing requires an administrator to manually enter IP addresses, subnet masks, gateways, and corresponding media access control (MAC) addresses for each interface of each device into a table. Static routing enables more control but requires more work to maintain the table. The table must be updated each time routes are added or changed.

Dynamic routing uses protocols that enable the devices in a network to exchange routing table information with each other. The table is built and changed automatically. No administrative tasks are needed unless a time limit is added, so dynamic routing is more efficient than static routing. The default time limit is 4 hours. If the network has a great many routes that are added and deleted from the cache, the time limit should be adjusted.

The routing protocols that dynamic routing uses to learn routes, such as distance-vector and link-state, is beyond the scope of this document.

Devices That Do Not Use ARP

When a network is divided into two segments, a bridge joins the segments and filters traffic to each segment based on Media Access Control (MAC) addresses. The bridge builds its own address table, which uses MAC addresses only, as opposed to a router, which has an Address Resolution Protocol (ARP) cache that contains both IP addresses and the corresponding MAC addresses.

Passive hubs are central-connection devices that physically connect other devices in a network. They send messages out all ports to the devices and operate at Layer 1, but they do not maintain an address table.

Layer 2 switches determine which port is connected to a device to which the message is addressed and send the message only to that port, unlike a hub, which sends the message out all its ports. However, Layer 3 switches are routers that build an ARP cache (table).

Inverse ARP

Inverse ARP, which is enabled by default in ATM networks, builds an ATM map entry and is necessary to send unicast packets to a server (or relay agent) on the other end of a connection. Inverse ARP is supported only for the **aal5snap**encapsulation type.

For multipoint interfaces, an IP address can be acquired using other encapsulation types because broadcast packets are used. However, unicast packets to the other end will fail because there is no ATM map entry and thus DHCP renewals and releases also fail.

For more information about Inverse ARP and ATM networks, see the "Configuring ATM" feature module in the *Asynchronous Transfer Mode Configuration Guide.*

Reverse ARP

Reverse ARP (RARP) as defined by RFC 903 works the same way as the Address Resolution Protocol (ARP), except that the RARP request packet requests an IP address instead of a media access control (MAC) address. RARP often is used by diskless workstations because this type of device has no way to store IP addresses to use when they boot. The only address that is known is the MAC address because it is burned in to the hardware.

RARP requires a RARP server on the same network segment as the device interface. The figure below illustrates how RARP works.

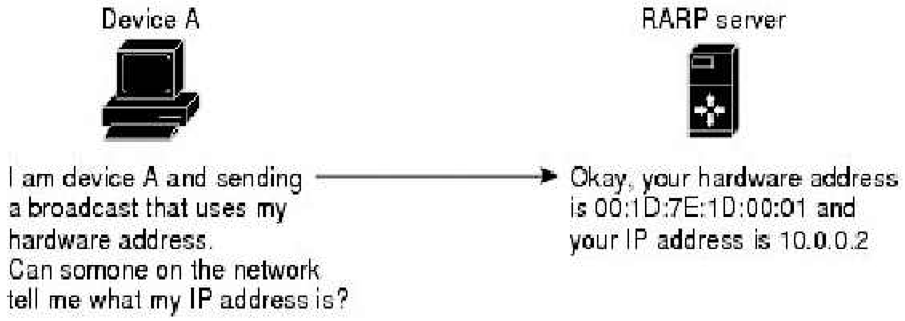

Device A
RARP server
I am device A and sending a broadcast that uses my hardware address. Can somone on the network tell me what my IP address is?
Okay, your hardware address is 00:1D:7E:1D:00:01 and your IP address is 10.0.0.2

Figure 6.8: Reverse ARP

Because of the limitations with RARP, most businesses use Dynamic Host Configuration Protocol (DHCP) to assign IP addresses dynamically. DHCP is cost-effective and requires less maintenance than RARP. The most important limitations with RARP are as follows:

❖ Because RARP uses hardware addresses, if the internetwork is large with many physical networks, a RARP server must be on every segment with an additional server for redundancy. Maintaining two servers for every segment is costly.

❖ Each server must be configured with a table of static mappings between the hardware addresses and the IP addresses. Maintenance of the IP addresses is difficult.

❖ RARP only provides IP addresses of the hosts but not subnet masks or default gateways.

Cisco software attempts to use RARP if it does not know the IP address of an interface at startup to respond to RARP requests that it is able to answer. The Auto Install feature of the software automates the configuration of Cisco devices.

Auto Install supports RARP and enables a network manager to connect a new device to a network, turn it on, and automatically load a pre-existing configuration file. The process begins when no valid configuration file is found in NVRAM. For more information about Auto Install, see the *Configuration Fundamentals Configuration Guide*.

Proxy ARP

Proxy Address Resolution Protocol, as defined in RFC 1027, was implemented to enable devices that are separated into physical network segments connected by a router in the same IP network or sub-network to resolve IP-to-MAC addresses. When devices are not in the same data link layer network but are in the same IP network, they try to transmit data to each other as if they were on the local network. However, the router that separates the devices will not send a broadcast message because routers do not pass hardware-layer broadcasts. Therefore, the addresses cannot be resolved.

Proxy ARP is enabled by default so the "proxy router" that resides between the local networks responds with its MAC address as if it were the router to which the broadcast is addressed. When the sending device receives the MAC address of the proxy router, it sends the datagram to the proxy router, which in turns sends the datagram to the designated device.

Proxy ARP is invoked by the following conditions:

- ❖ The target IP address is not on the same physical network (LAN) on which the request is received.
- ❖ The networking device has one or more routes to the target IP address.
- ❖ All of the routes to the target IP address go through interfaces other than the one on which the request is received.

When proxy ARP is disabled, a device responds to ARP requests received on its interface only if the target IP address is the same as its IP address or if the target IP address in the ARP request has a statically configured ARP alias.

Serial Line Address Resolution Protocol

Serial Line ARP (SLARP) is used for serial interfaces that use High-Level Data Link Control (HDLC) encapsulation. A SLARP server, intermediate (staging) device, and another device providing a SLARP service might be required in addition to a TFTP server. If an interface is not directly connected to a server, the staging device is required to forward the address-resolution requests to the server. Otherwise, a directly connected device with SLARP service is required. Cisco software attempts to use SLARP if it does not know the IP address of an interface at startup to respond to SLARP requests that software is able to answer.

Cisco software automates the configuration of Cisco devices with the Auto Install feature. Auto Install supports SLARP and enables a network manager to connect a new device to a network, turn it on, and automatically load a pre-existing configuration file. The process begins when no valid configuration file is found in NVRAM.

Internet Control Message Protocol (ICMP)

The operation of the Internet is closely monitored by the routers. Although the IP layer provides a best-effort unreliable delivery system, each router provides an Internet Control Message Protocol (ICMP) error message to the original sender whose IP address is encapsulated in the IP datagram. The ICMP message allows the router to send error or control messages to the sending host. These ICMP messages travel across the internet in the data portion of IP datagram, but are, however, considered a part of the IP protocol suite. An exception is made to the error handling procedure if an IP datagram carrying an ICMP message causes an error. This is established to avoid the problem of having error messages. Technically, ICMP is an error reporting mechanism. Whenever a datagram causes an error, ICMP can only report the error condition back to the original source of the datagram; the source must accordingly relate the error to an individual application program or take appropriate action to correct the problem. For example, a datagram is supposed to follow a path through a sequence of routers $R_1, \ldots, R_{k-1}, R_k$. If R_{k-1} has incorrect routing information and mistakenly routes the datagram to router RE, then RE uses an ICMP to report the problem to router R1 and not R_{k-1}. This is because the IP datagram only contains the source IP address of router R_1. It is now the responsibility of router R_1 to remedy the situation.

1. **Destination unreachable:** This message is used when the router cannot locate the destination since it was down, or the destination address is invalid. It can also occur if an IP packet with the don't fragment bit set cannot be delivered since a network with a small MTU stands in the way.

2. **Time exceeded:** This message is sent when an IP datagram with the TTL field equal to zero is obtained at a router. This event is a symptom that packets are looping (due the mistakes in the routing tables), that there is enormous congestion, or that the TTL values are being set too low.

Message Type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	TTL field hit 0
Parameter Problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router some geography
Echo	Ask a machine if it is alive (ping)
Echo reply	Yes, I am alive (ping response)
Timestamp request	Same as Echo request, but with timestamp
Timestamp reply	Same as Echo reply, but with timestamp

Table 1: The principal ICMP message types

3. **Parameter Problem**: This indicates that some of the parameters in the header field are corrupted. This can be computed using the header checksum. This could be due to a bug in the sending host's IP software or possibly in the software of a router encountered along the way.
4. **Source quench:** This was formerly used to throttle hosts that were sending too many packets. It is rarely used anymore because when congestion occurs, these ICMP source quench packets only lead to more traffic on the network, adding more fuel to the fire!.
5. **Redirect message:** This message is used when an intermediate router notices that a packet is being routed wrongly. The router then informs the sending host about a shorter path that exists between the source and the destination.
6. **Echo request and reply:** This is the ping command used to see if the destination is reachable and alive. Upon receiving the echo request, the destination host is expected to send an echo reply message back. The timestamp request and reply are similar, except that the arrival of the message and the departure time of the reply are recorded. This facility is used to measure network performance.

6.3.1 Network Layer Design Issues

1. Store-and-Forward Packet Switching

The major components of the system are the carrier's equipment (routers connected by transmission lines), shown inside the shaded oval, and the customers' equipment, shown outside the oval.

Host H1 is directly connected to one of the carrier's routers, A, by a leased line. In contrast, H2 is on a LAN with a router, F, owned and operated by the customer. This router also has a leased line to the carrier's equipment.

We have shown F as being outside the oval because it does not belong to the carrier, but in terms of construction, software, and protocols, it is probably no different from the carrier's routers.

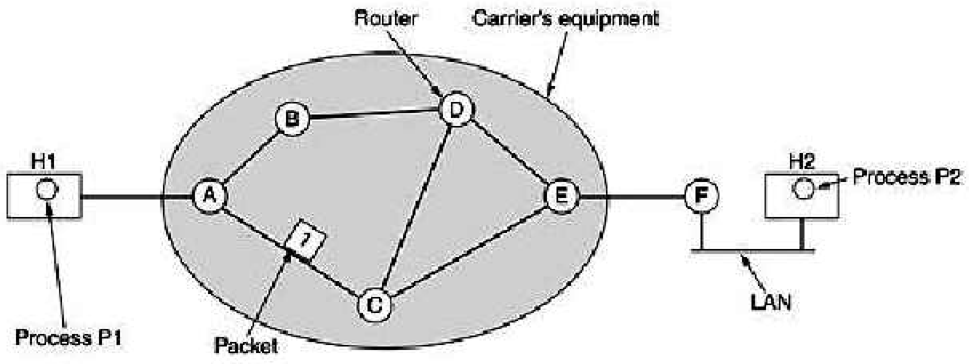


Figure 6.9. The environment of the network layer protocols.

This equipment is used as follows. A host with a packet transmits it to the nearest router, either on its own LAN or over a point-to-point link to the carrier. The packet is stored there until it has fully arrived so the checksum can be verified.

Then it is forwarded to the next router along the path until it reaches the destination host, where it is delivered. This mechanism is store-and-forward packet switching.

2. Services Provided to the Transport Layer

The network layer provides services to the transport layer at the network layer/transport layer interface. An important question is what kind of services the network layer provides to the transport layer.

The network layer services have been designed with the following goals in mind.

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

Given these goals, the designers of the network layer have a lot of freedom in writing detailed specifications of the services to be offered to the transport layer. This freedom often degenerates into a raging battle between two warring fractions.

The other camp argues that the subnet should provide a reliable, connection-oriented service. They claim that 100 years of successful experience with the worldwide telephone system is an excellent guide. In this view, quality of service is the dominant factor, and without connections in the subnet, quality of service is very difficult to achieve, especially for real-time traffic such as voice and video.

These two camps are best exemplified by the Internet and ATM. The Internet offers connectionless network-layer service; ATM networks offer connection-oriented network-layer service. However, it is interesting to note that as quality-of-service guarantees are becoming more and more important, the Internet is evolving.

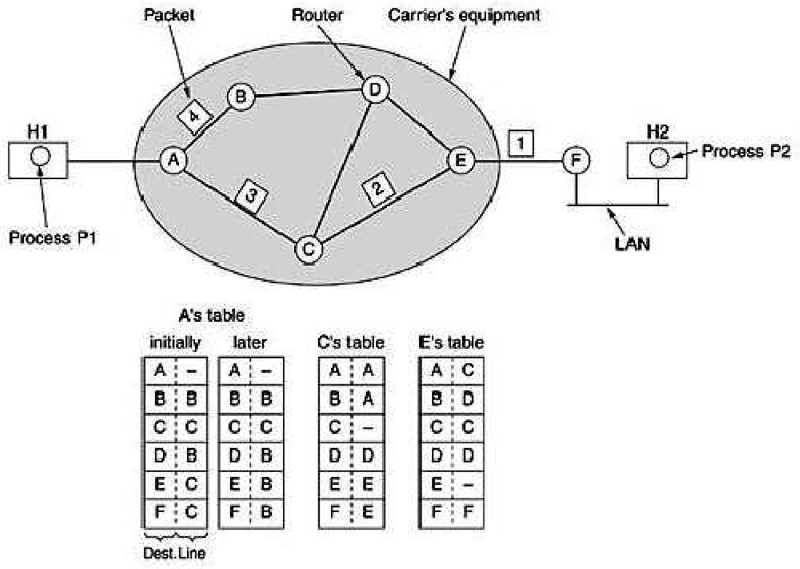
3. Implementation of Connectionless Service

Two different organizations are possible, depending on the type of service offered. If connectionless service is offered, packets are injected into the subnet individually and routed independently of each other. No advance setup is needed.
In this context, the packets are frequently called datagrams (in analogy with telegrams) and the subnet is called a datagram subnet. If connection-oriented service is used, a path from the source router to the destination router must be established before any data packets can be sent.

This connection is called a VC (virtual circuit), in analogy with the physical circuits set up by the telephone system, and the subnet is called a virtual-circuit subnet. In this section we will examine datagram subnets; in the next one we will examine virtual-circuit subnets.

Let us now see how a datagram subnet works. Suppose that the process P1 in Fig. 2-10 has a long message for P2. It sends the message to the transport layer with instructions to deliver it to process P2 on host H2.

The transport layer code runs on H1, typically within the operating system. It prepends a transport header to the front of the message and sends the result to the network layer, probably just another procedure within the operating system.



Dest.	A's table initially	A's table later	C's table	E's table
A	–	–	A	C
B	B	B	A	D
C	C	C	–	C
D	B	B	D	D
E	C	B	E	–
F	C	B	E	F

Figure 6.10. Routing within a datagram subnet.

Let us assume that the message is four times longer than the maximum packet size, so the network layer has to break it into four packets, 1, 2, 3, and 4 and sends each of them in turn to router A using some point-to-point protocol, for example, PPP.

At this point the carrier takes over. Every router has an internal table telling it where to send packets for each possible destination. Each table entry is a pair consisting of a destination and the outgoing line to use for that destination.

Only directly-connected lines can be used. For example, in Fig. 6.10, A has only two outgoing lines—to B and C—so every incoming packet must be sent to one of these routers, even if the ultimate destination is some other router. A's initial routing table is shown in the figure under the label "initially."

However, something different happened to packet 4. When it got to A it was sent to router B, even though it is also destined for F. For some reason, A decided to send packet 4 via a different route than that of the first three.

Perhaps it learned of a traffic jam somewhere along the ACE path and updated its routing table, as shown under the label "later." The algorithm that manages the tables and makes the routing decisions is called the routing algorithm.

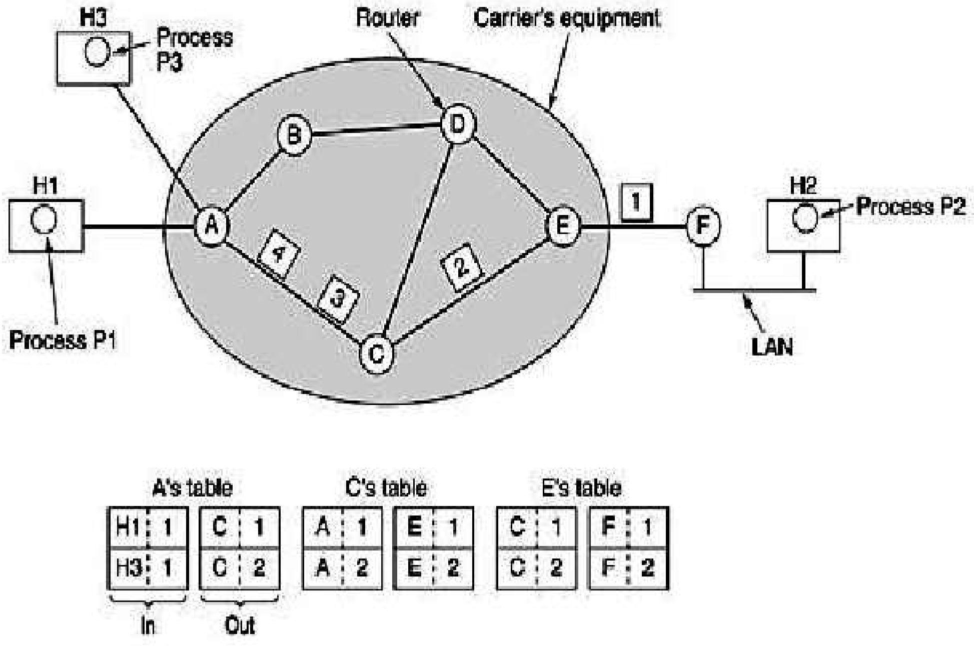
4. Implementation of Connection-Oriented Service

For connection-oriented service, we need a virtual-circuit subnet. The idea behind virtual circuits is to avoid having to choose a new route for every packet sent, as in Fig. 6.10.

Instead, when a connection is established, a route from the source machine to the destination machine is chosen as part of the connection setup and stored in tables inside the routers. That route is used for all traffic flowing over the connection, exactly the same way that the telephone system works.

When the connection is released, the virtual circuit is also terminated. With connection-oriented service, each packet carries an identifier telling which virtual circuit it belongs to. As an example, consider the situation of Fig. 6.11. Here, host H1 has established connection 1 with host H2.

It is remembered as the first entry in each of the routing tables. The first line of A's table says that if a packet bearing connection identifier 1 comes in from H1, it is to be sent to router C and given connection identifier 1. Similarly, the first entry at C routes the packet to E, also with connection identifier 1.



A's table In	A's table Out	C's table In	C's table Out	E's table In	E's table Out
H1 : 1	C : 1	A : 1	E : 1	C : 1	F : 1
H3 : 1	C : 2	A : 2	E : 2	C : 2	F : 2

Figure 6.11. Routing within a virtual-circuit subnet.

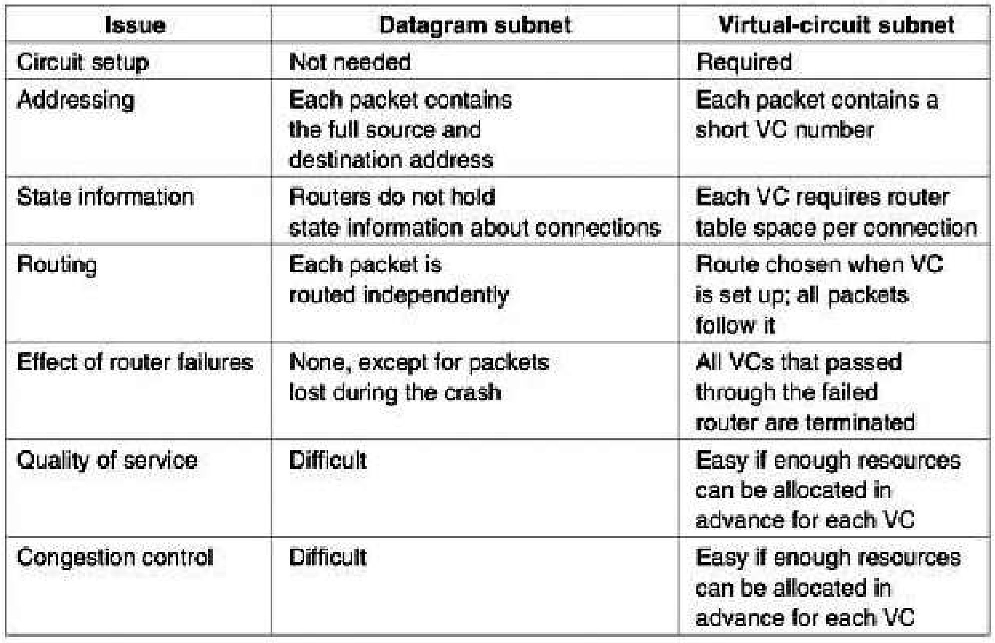
Now let us consider what happens if H3 also wants to establish a connection to H2. It chooses connection identifier 1 and tells the subnet to establish the virtual circuit. This leads to the second row in the tables.

Note that we have a conflict here because although A can easily distinguish connection 1 packets from H1 from connection 1 packets from H3, C cannot do this. For this reason, A assigns a different connection identifier to the outgoing traffic for the second connection.

Avoiding conflicts of this kind is why routers need the ability to replace connection identifiers in outgoing packets. In some contexts, this is called label switching.

5. Comparison of Virtual-Circuit and Datagram Subnets

Both virtual circuits and datagrams have their supporters and their detractors. We will now attempt to summarize the arguments both ways. The major issues are listed in Fig. 6.11, although purists could probably find a counterexample for everything in the figure.



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

Figure 6.11. Comparison of datagram and virtual-circuit subnets.

Inside the subnet, several trade-offs exist between virtual circuits and datagrams. One trade-off is between router memory space and bandwidth. Virtual circuits allow packets to contain circuit numbers instead of full destination addresses.

If the packets tend to be fairly short, a full destination address in every packet may represent a significant amount of overhead and hence, wasted bandwidth. The price paid for using virtual circuits internally is the table space within the routers.

Depending upon the relative cost of communication circuits versus router memory, one or the other may be cheaper. Another trade-off is setup time versus address parsing time. Using virtual circuits requires a setup phase, which takes time and consumes resources.

However, figuring out what to do with a data packet in a virtual-circuit subnet is easy: the router just uses the circuit number to index into a table to find out where the packet goes. In a datagram subnet, a more complicated lookup procedure is required to locate the entry for the destination.

For transaction processing systems (e.g., stores calling up to verify credit card purchases), the overhead required to set up and clear a virtual circuit may easily dwarf the use of the circuit. If the majority of the traffic is expected to be of this kind, the use of virtual circuits inside the subnet makes little sense.

On the other hand, permanent virtual circuits, which are set up manually and last for months or years, may be useful here. Virtual circuits also have a vulnerability problem. If a router crashes and loses its memory, even if it comes back up a second later, all the virtual circuits passing through it will have to be aborted.

In contrast, if a datagram router goes down, only those users whose packets were queued in the router at the time will suffer, and maybe not even all those, depending upon whether they have already been acknowledged.

The loss of a communication line is fatal to virtual circuits using it but can be easily compensated for if datagrams are used. Datagrams also allow the routers to balance the traffic throughout the subnet, since routes can be changed partway through a long sequence of packet transmissions.

6.4 Virtual Circuits and Datagrams.

Conceived in the 1960's, *packet switching* is a more recent technology than circuit switching which addresses a disadvantage of circuit switching: the need to allocate resources for a circuit, thus incurring link capacity wastes when no data flows on a circuit.

Packet switching introduces the idea of cutting data on a flow into packets which are transmitted over a network without any resource being allocated. If no data is available at the sender at some point during a communication, then no packet is transmitted over the network and no resources are wasted.

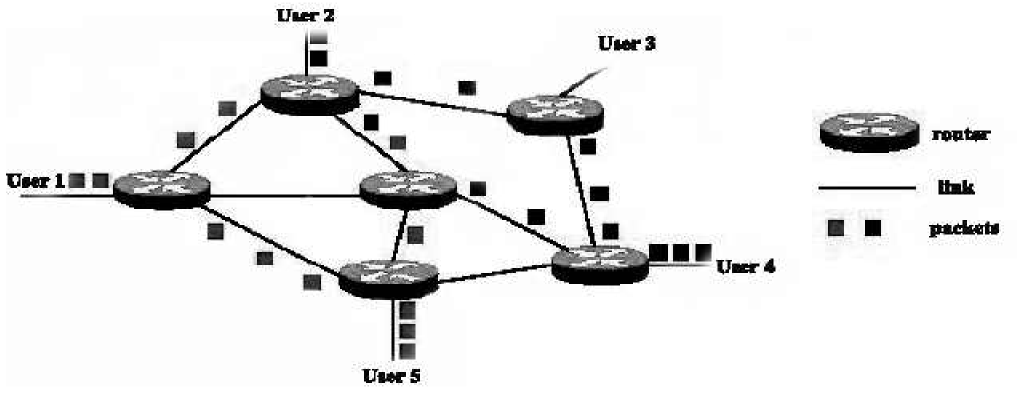


Figure 6.12: Datagram Packet Switching. Packets from a given flow are independent and a router can forward two packets from the same flow on two different links.

Since each packet is processed individually by a router, all packets sent by a host to another host are not guaranteed to use the same physical links. If the routing algorithm decides to change the routing tables of the network between the instants two packets are sent, then these packets will take different paths and can even arrive out of order.

In this Figure for instance, packets use two different paths to go from User 1 to User 5. Second, on a network topology change such as a link failure, the routing protocol will automatically recomputed routing tables so as to take the new topology into account and avoid the failed link. As opposed to circuit switching, no additional traffic engineering algorithm is required to reroute traffic.

Virtual circuit packet switching

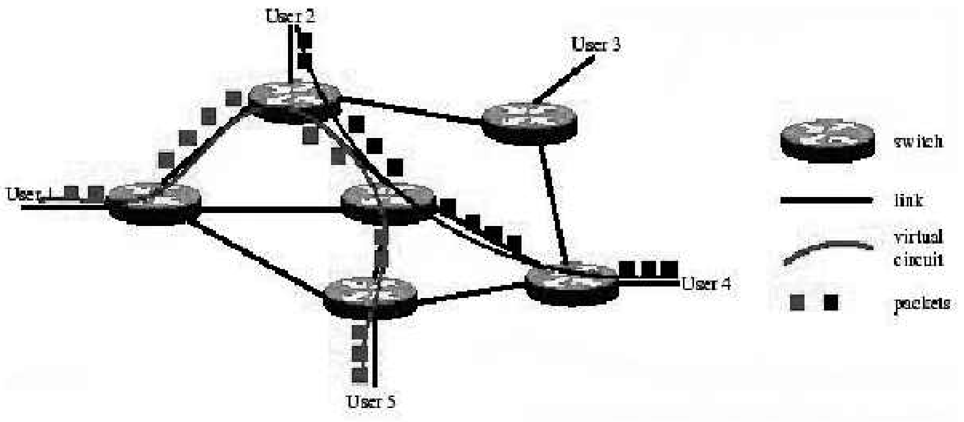


Figure 6.13: Virtual circuit packet switching. All packets from the same flow use the same virtual circuit.

Virtual circuit packet switching (VC-switching) is a packet switching technique which merges datagram packet switching and circuit switching to extract both of their advantages. VC switching is a variation of datagram packet switching where packets flow on so-called logical circuits for which no physical resources like frequencies or time slots are allocated (see Figure 6.13).

Each packet carries a circuit identifier, which is local to a link and updated by each switch on the path of the packet from its source to its destination. A virtual circuit is defined by the sequence of the mappings between a link taken by packets and the circuit identifier packets carry on this link. In VC-switching, routing is performed at circuit establishment time to keep packet forwarding fast.

Other advantages of VC-switching include the traffic engineering capability of circuit switching, and the resources usage efficiency of datagram packet switching. Nevertheless, a main issue of VC-Switched networks is the behavior on a topology change.

As opposed to Datagram Packet Switched networks which automatically recomputed routing tables on a topology change like a link failure, in VC-switching all virtual circuits that pass through a failed link are interrupted. Hence, rerouting in VC-switching relies on traffic engineering techniques.

There are a number of important differences between virtual circuit and datagram networks. The choice strongly impacts complexity of the different types of node. Use of datagrams between intermediate nodes allows relatively simple protocols at this level, but at the expense of making the end (user) nodes more complex when end-to-end virtual circuit service is desired.

The Internet transmits datagrams between intermediate nodes using IP. Most Internet users need additional functions such as end-to-end error and sequence control to give a reliable service (equivalent to that provided by virtual circuits). This reliability may be provided by the Transmission Control Protocol (TCP), which is used end-to-end across the Internet, or by applications such as the trivial file transfer protocol (TFTP) running on top of the User Datagram Protocol (UDP)

Check your progress

Q1. Define network layer protocols.

Q2. Explain Virtual circuits and datagrams

6.5 Summary

In this unit you have learnt FDDI Protocol, Distributed Queue Dual Bus (DQDB) protocol, Network Layer Protocols, Design issues of network layer protocols and Virtual Circuits and Datagrams.

- ❖ Fiber Distributed Data Interface (FDDI) is a standard for data transmission in a local area network. It uses optical fiber as its standard underlying physical medium, although it was also later specified to use copper cable, in which case it may be called CDDI (Copper Distributed Data Interface), standardized as TP-PMD (Twisted-Pair Physical Medium-Dependent), and also referred to as TP-DDI (Twisted-Pair Distributed Data Interface).
- ❖ A distributed-queue dual-bus network (DQDB) is a distributed multi-access network that (a) supports integrated communications using a dual bus and distributed queuing, (b) provides access to local or metropolitan area networks, and (c) supports connectionless data transfer, connection-oriented data transfer, and isochronous communications, such as voice communications.
- ❖ Network layers protocols are: ICMP (Internet Control Message Protocol), ARP Address Resolution Protocol, RARP Reverse Address Resolution Protocol

6.6 Review Questions

Q1. What do you mean by FDDI protocols? Define working of FDDI protocols.

Q2. Why we need DQDB protocol? Elaborate your answer.

Q3. Explain ARP, RARP and ICMP protocols.

Q4. Explain design issues of network layer.

Q5. What is the difference between virtual circuit and datagrams?

RIL-089

UNIT-VII

Error Detection & Correction

Structure

7.0 Introduction

In this unit we focused on Error detection and correction. In this unit there are five sections. In the first section i.e. Sec. 7.1 you will learn about error. You will also learn about error detecting codes. In this unit you will also know about how error detect in a network and how we can correct that error. In the next section i.e. Sec. 7.2 we define parity check codes. Error detection & correction takes place with the help of the parity check codes. We use parity bits for error detection and corrections. In the section 7.3 you will know about CRC i.e. Cyclic Redundancy Codes. A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. Blocks of data entering these systems get a short check value attached, based on the remainder of a polynomial division of their contents. In Sec. 7.4 and 7.5 you will find summary and review questions respectively.

Objectives

After studying this unit you should be able to:

- ❖ Define error in network
- ❖ Describe error detection and correction
- ❖ Express parity check codes
- ❖ Define cyclic redundancy check

7.1 Error Detection & Correction

Error

Error is a condition when the output information does not match with the input information. During transmission, digital signals suffer from noise that can introduce errors in the binary bits travelling from one system to other. That means a 0 bit may change to 1 or a 1 bit may change to 0.

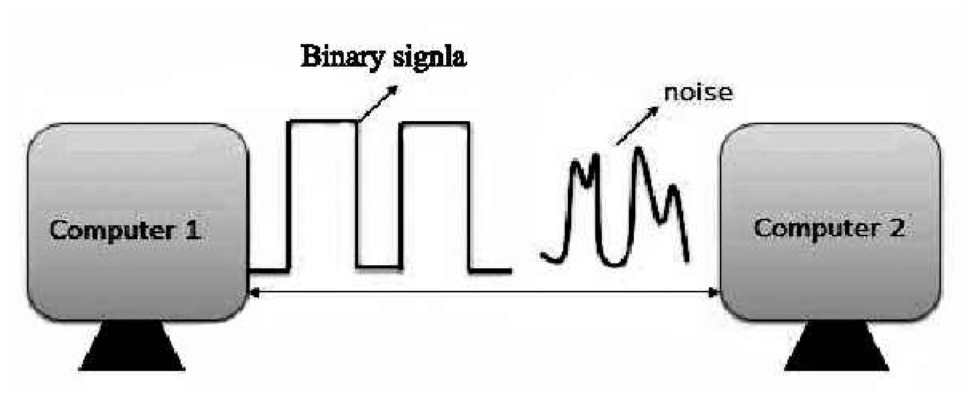


Figure 7.1: Error

There are many reasons such as noise, cross-talk etc., which may help data to get corrupted during transmission. The upper layers work on some generalized view of network architecture and are not aware of actual hardware data processing. Hence, the upper layers expect error-free transmission between the systems. Most of the applications would not function expectedly if they receive erroneous data. Applications such as voice and video may not be that affected and with some errors they may still function well.

Data-link layer uses some error control mechanism to ensure that frames (data bit streams) are transmitted with certain level of accuracy. But to understand how errors is controlled, it is essential to know what types of errors may occur.

Types of Errors

There may be three types of errors:

- Single bit error
- Multiple bit error
- Burst error

Single bit error

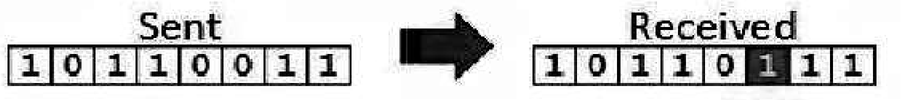


Figure 7.2: Single bit error

Multiple bits error

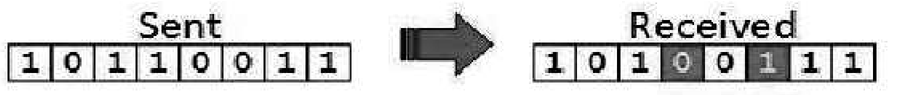


Figure 7.3: Multiple bits error

Frame is received with more than one bits in corrupted state.

Burst error

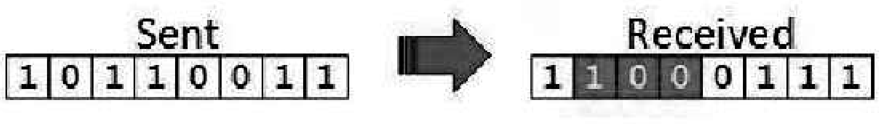


Figure 7.4: Burst error Frame

contains more than1 consecutive bits corrupted. Error

control mechanism may involve two possible ways:

- Error detection
- Error correction

In this section, we'll examine a few of the simplest techniques that can be used to detect and, in some cases, correct such bit errors. Our goal here is to develop an intuitive feel for the capabilities that error detection and correction techniques provide, and to see how a few simple techniques work and are used in practice in the data link layer.

Figure 3.5 illustrates the setting for our study. At the sending node, data, *D*, to be "protected" against bit errors is augmented with error detection and correction bits, *EDC*. Typically, the data to be protected includes not only the datagram passed down from the network layer for transmission across the link, but also link-level addressing information, sequence numbers, and other fields in the data link frame header. Both *D* and *EDC* are sent to the receiving node in a link-level frame. At the receiving node, a sequence of bits, *D'* and *EDC'* are received. Note that *D'* and *EDC'* may differ from the original *D* and *EDC* as a result of in-transit bit flips.

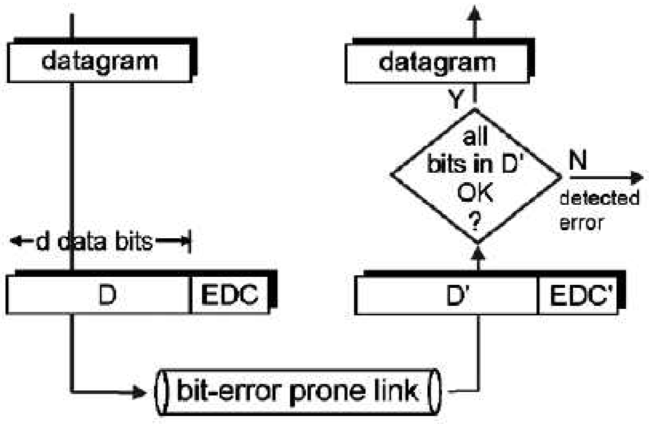


Figure 7.5: Error detection and correction scenario

The receiver's challenge is to determine whether or not *D'* is the same as the original *D*, given that it has only received *D'* and *EDC'*. The exact wording of the receiver's decision in Figure 3.5 (we ask whether an error is detected, not whether an error has occurred!) is important. Error detection and correction techniques allow the receiver to sometimes, *but not always*, detect that bit errors have occurred. That is, even with the use of error detection bits there will still be a possibility that **undetected bit errors** will occur, i.e., that the receiver will be unaware that the received information contains bit errors. As a consequence, the receiver might deliver a corrupted datagram to the network layer, or be unaware that the contents of some other field in the frame's header have been corrupted. We thus want to choose an error detection scheme so that the probability of such occurrences is small. Generally, more sophisticated error detection and correction techniques (i.e., those that have a smaller probability of allowing undetected bit errors) incur a larger overhead - more computation is need to compute and transmit a larger number of error detection and correction bits.

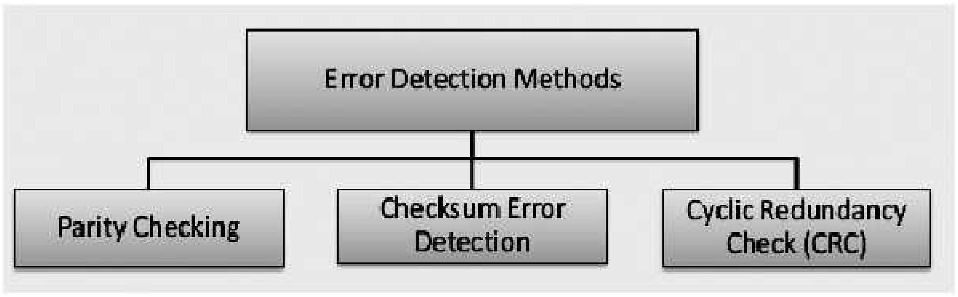


Figure 7.6: Types of Error Detection

Error-Detecting codes

Whenever a message is transmitted, it may get scrambled by noise or data may get corrupted. To avoid this, we use error-detecting codes which are additional data added to a given digital message to help us detect if an error occurred during transmission of the message. A simple example of error-detecting code is **parity check**.

Error-Correcting codes

Along with error-detecting code, we can also pass some data to figure out the original message from the corrupt message that we received. This type of code is called an error-correcting code. Error-correcting codes also deploy the same strategy as error-detecting codes but additionally, such codes also detect the exact location of the corrupt bit.

In error-correcting codes, parity check has a simple way to detect errors along with a sophisticated mechanism to determine the corrupt bit location. Once the corrupt bit is located, its value is reverted (from 0 to 1 or 1 to 0) to get the original message.

How to Detect and Correct Errors?

To detect and correct the errors, additional bits are added to the data bits at the time of transmission.

- The additional bits are called **parity bits**. They allow detection or correction of the errors.

- The data bits along with the parity bits form a **code word.**

7.2 Parity Check Codes

Perhaps the simplest form of error detection is the use of a single **parity bit.** Suppose that the information to be sent, *D* in Figure 3.6, has *d* bits. In an even parity scheme, the sender simply includes one additional bit and chooses its value such that the total number of 1's in the *d+1* bits (the original information plus a parity bit) is even. For odd parity schemes, the parity bit value is chosen such that there are an odd number of 1's. Figure 7.7 illustrates an even parity scheme, with the single parity bit being stored in a separate field.

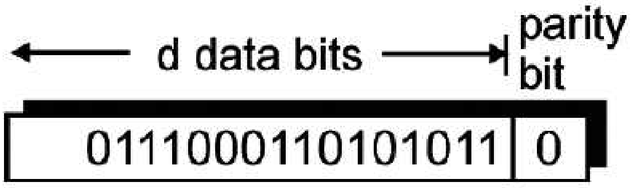


Figure 7.7: One-bit even parity

Receiver operation is also simple with a single parity bit. The receiver need only count the number of 1's in the received *d+1* bits. If an odd number of 1-valued bits are found with an even parity scheme, the receiver knows that at least one bit error has occurred. More precisely, it knows that some *odd* number of bit errors have occurred.

But what happens if an even number of bit errors occur? You should convince yourself that this would result in an undetected error. If the probability of bit errors is small and errors can be assumed to occur independently from one bit to the next, the probability of multiple bit errors in a packet would be extremely small. In this case, a single parity bit might suffice. However, measurements have shown that rather than occurring independently, errors are often clustered together in ``bursts.'' Under burst error conditions, the probability of undetected errors in a frame protected by single-bit-parity can approach 50 percent. Clearly, a more robust error detection scheme is needed (and, fortunately, is used in practice!). But before examining error detection schemes that are used in practice, let's consider a simple generalization of one-bit parity that will provide us with insight into error correction techniques.

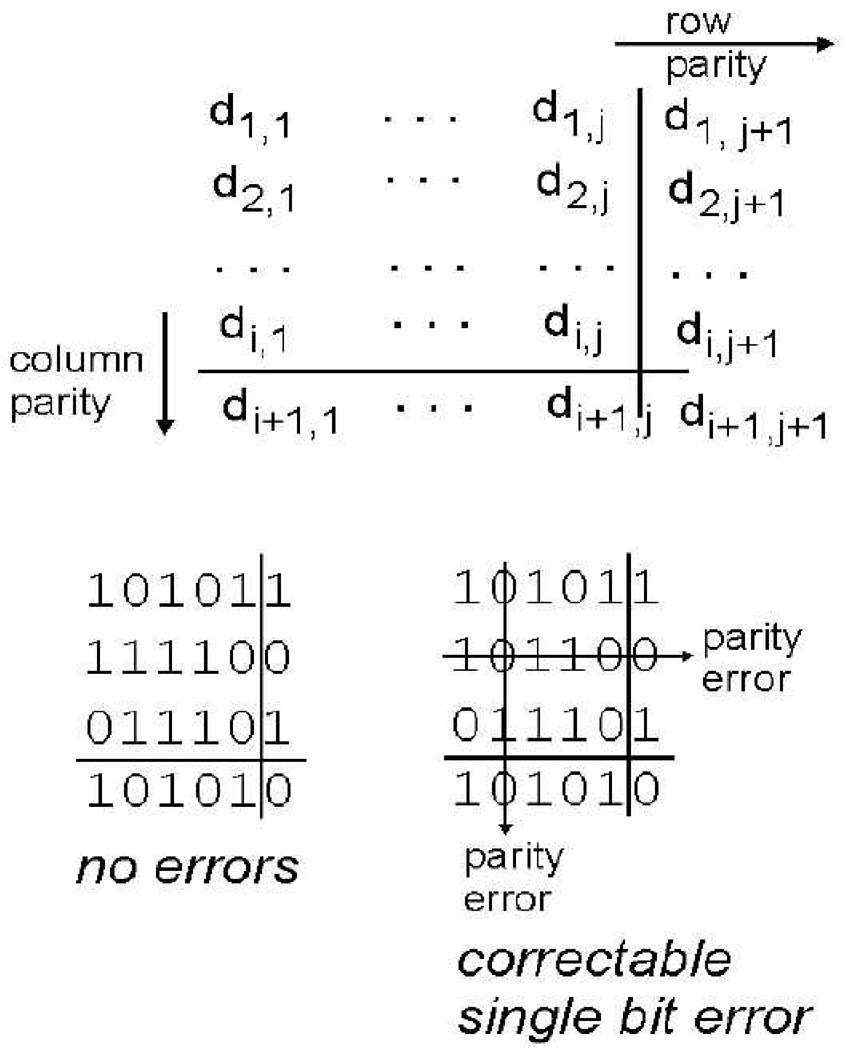

row parity
$d_{1,1}$ ... $d_{1,j}$ $d_{1,j+1}$
$d_{2,1}$ ... $d_{2,j}$ $d_{2,j+1}$
$d_{i,1}$ ... $d_{i,j}$ $d_{i,j+1}$
column parity
$d_{i+1,1}$ ... $d_{i+1,j}$ $d_{i+1,j+1}$
101011
111100
011101
101010
no errors
101011
101100
011101
101010
parity error
parity error
correctable single bit error

Figure 7.8: Two-dimensional even parity

Figure 7.8 shows a two-dimensional generalization of the single-bit parity scheme. Here, the *d* bits in *D* are divided into *i* rows and *j* columns. A parity value is computed for each row and for each column. The resulting *i+j+1* parity bits are the data link frame's error detection bits.

Suppose now that a single bit error occurs in the original *d* bits of information. With this **two-dimensional parity** scheme, the parity of both the column and the row containing the flipped bit will be in error. The receiver can thus not only *detect* the fact that a single bit error has occurred, but can use the column and row indices of the column and row with parity errors to actually identify the bit that was corrupted and *correct* that error! Figure 3.8 shows an example in which the 0-valued bit in position (1, 1) is corrupted and switched to a 1 -- an error that is both detectable and correctable at the receiver. Although our discussion has focused on the original *d* bits of information, a single error in the parity bits themselves is also detectable and correctable. Two dimensional parity can also detect (but not correct!) any combination of two errors in a packet. The ability of the receiver to both detect and correct errors is known as **forward error correction (FEC).** These techniques are commonly used in audio storage and playback devices such as audio CD's. In a network setting, FEC techniques can be used by themselves, or in conjunction with the ARQ techniques. FEC techniques are valuable because they can decrease the number of sender retransmissions required. Perhaps more importantly, they allow for immediate correction of errors at the receiver. This avoids having to wait the round-trip propagation delay needed for the sender to receive a NAK packet and for the retransmitted packet to propagate back to the receiver -- a potentially important advantage for real-time network applications

Checksumming Methods

In checksumming techniques, the *d* bits of data in Figure 3.5 are treated as a sequence of *k*-bit integers. One simple checksumming method is to simply sum these *k*-bit integers and use the resulting sum as the error detection bits. The so-called **Internet checksum** is based on this approach - bytes of data are treated as 16-bit integers and their ones-complement sum forms the Internet checksum. A receiver calculates the checksum it calculates over the received data and checks whether it matches the checksum carried in the received packet. In the TCP/IP protocols, the Internet checksum is computed over all fields (header and data fields included). In other protocols, e.g., XTP, one checksum is computed over the header, with another checksum computed over the entire packet.

Working of parity check

Parity is a means of detecting errors in binary transmission streams. It is based on sending redundant data to verify the integrity of the received data. It is not foolproof.

To use a parity check, at the transmission end the data are divided into groups of bits (typically 7 or 8 bits per group). For each group a parity bit is generated and sent along with the data group. At the receiving end, another parity bit is generated based on the received data and compared with the parity bit sent by the transmitter. If the parity bits match, the data are considered (probably) valid, but if the parity bits do not match, an error has occurred during transmission.

There are two "types" of parity checks that can be used; even parity and odd parity. The transmitter and receiver must use the same method for the check to work properly.

With even parity, the total number of 1's in the data plus parity bit must be an even number. Thus, if there are already an even number of 1's in the data itself the parity bit generated is 0, but if there are an odd number of 1's in the data itself, the parity bit generated is 1 to make the total even. For example (using 7 data bits, P is the parity bit generated):

EVEN PARITY

Data	#1's in data	P	Total # 1's (data and P)
-----	---------	-	--------------
0110110	4 (Even)	0	4 (Even)
0011111	5 (Odd)	1	6 (Even)
0000000	0 (Even)	0	0 (Even)
1010100	3 (Odd)	1	4 (Even)
1111111	7 (Odd)	1	8 (Even)

With odd parity, the total number of 1's in the data plus parity bit must be an odd number. Thus, if there are already an odd number of 1's in the data itself the parity bit generated is 0, but if there are an even number of 1's in the data itself, the parity bit generated is 1 to make the total odd. For example (using 7 data bits, P is the parity bit generated):

ODD PARITY

Data	#1's in data	P	Total # 1's (data and P)
-----	---------	-	--------------
0110110	4 (Even)	1	5 (Odd)
0011111	5 (Odd)	0	5 (Odd)
0000000	0 (Even)	1	1 (Odd)
1010100	3 (Odd)	0	3 (Odd)
1111111	7 (Odd)	0	7 (Odd)

If a single bit is switched during transmission, a parity check will catch it, and we know the data are bad. We can't tell, however, which bit was switched. For example, suppose 0110110 is sent using odd parity, and the second bit from the left is switched:

Data sent: 0110110

#1's in data: 4 (Even)

Parity bit: 1

Tl. # 1's: 5 (Odd)

Data received: 0010110 (Note the changed bit)

#1's in data: 3 (Odd)

Parity bit: 0

Tl. # 1's: 3 (Odd)

Comparing the two parity bits, we can see that there was an error. However, we don't know which bit was switched. Any of the following data could have produced the received result with only a single bit change:

1010110

0110110

0000110

0011110

0010010

0010100

0010111

We have no way of knowing which of the above the original data were.

As mentioned at the beginning, parity is not foolproof. For example, suppose 0110110 is sent using odd parity, and the second and fourth bits from the left are both switched:

Data sent: 0110110

#1's in data: 4 (Even)

Parity bit: 1

Tl. # 1's: 5 (Odd)

Data received: 0011110   (Note the changed bits)

#1's in data: 4 (Even)

Parity bit: 1

Tl. # 1's: 5 (Odd)

Here, the parity bit generated by the receiver matches the parity bit sent by the transmitter, so the data are assumed to be valid even though an error has occurred. In fact, any time two bits (or any even number of bits) are switched, the parity check is fooled.

There are other error detection (and even correction) schemes, many of which are more reliable than a simple parity check. Some examples are vertical parity, HVP (horizontal and vertical parity), checksums, and CRC (circular redundancy check.) Many of these, however, are built on the principles of parity.

Check your progress

Q1. What is an error? Define its types.

Q2. How to detect an error? Elaborate your answer.

Other techniques for computing a checksum are to form the exclusive or of all the bytes in the message, or to compute a sum with end-around carry of all the bytes. In the later method the carry from each 8-bit sum is added into the least significant bit of the accumulator. It is believed that this is more likely to detect errors than the simple exclusive or, or the sum of the bytes with carry discarded. A technique that is believed to be quite good in terms of error detection, and which is easy to implement in hardware, is the cyclic redundancy check. This is another way to compute a checksum, usually eight, 16, or 32 bits in length that is appended to the message. We will briefly review the theory and then give some algorithms for computing in software a commonly used 32-bit CRC checksum.

An error detection technique used widely in today's computer networks is based on **cyclic redundancy check (CRC) codes**. CRC codes are also known as **polynomial codes**, since it is possible to view the bit string to be sent as a polynomial whose coefficients are the 0 and 1 values in the bit string, with operations on the bit string interpreted as polynomial arithmetic.

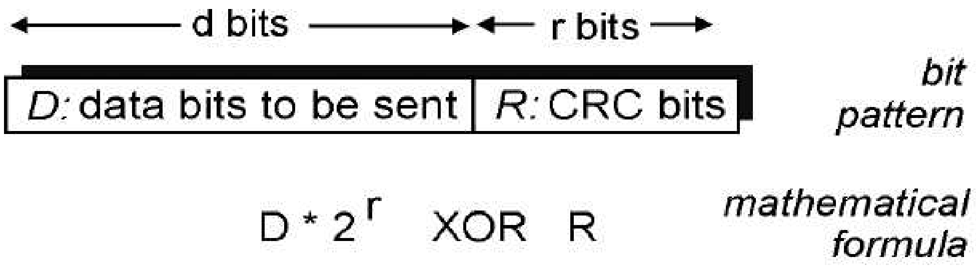


Figure 7.10: CRC codes

CRC codes operate as follows. Consider the *d-bit* piece of data, *D*, that the sending node wants to send to the receiving node. The sender and receiver must first agree on a $_{r+1}$ bit pattern, known as a **generator**, which we will denote as G. We will require that the most significant (leftmost) bit of *G* be a 1. The key idea behind CRC codes is shown in Figure 3.10. For a given piece of data, *D*, the sender will choose *r* additional bits, *R*, and append them to *D* such that the resulting *d+r* bit pattern (interpreted as a binary number) is exactly divisible by *G* using modulo 2 arithmetic. The process of error checking with CRC's is thus simple: the receiver divides the *d+r* received bits by *G*. If the remainder is non-zero, the receiver knows that an error has occurred; otherwise the data is accepted as being correct.

All CRC calculations are done in modulo 2 arithmetic without carries in addition or borrows in subtraction. This means that addition and subtraction are identical, and both are equivalent to the bitwise exclusive-or (XOR) of the operands. Thus, for example,

1011 XOR 0101 = 1110
1001 XOR 1101 = 0100

Also, we similarly have

1011 - 0101 = 1110

1001 - 1101 = 0100

Multiplication and division are the same as in base 2 arithmetic, except that any required addition or subtraction is done without carries or borrows. As in regular binary arithmetic, multiplication by 2^k left shifts a bit pattern by *k* places. Thus, given *D* and *R*, the quantity *D*2r XOR R* yields the *d+r* bit pattern shown in Figure 3.9. We'll use this algebraic characterization of the *d+r* bit pattern from Figure 3.9 in our discussion below.

Let us now turn to the crucial question of how the sender computes *R*. Recall that we want to find *R* such that there is an *n* such that

*D*2r* XOR *R* = *nG*

That is, we want to choose *R* such that *G* divides into *D*2r*XOR *R* without remainder. If we exclusive-or (i.e., add modulo 2, without carry) *R* to both sides of the above equation, we get

*D*2r* = *nG* XOR *R*

This equation tells us that if we divide $D*2^r$ by *G*, the value of the remainder is precisely *R*. In other words, we can calculate *R* as

R = remainder (*D*2r / G)*

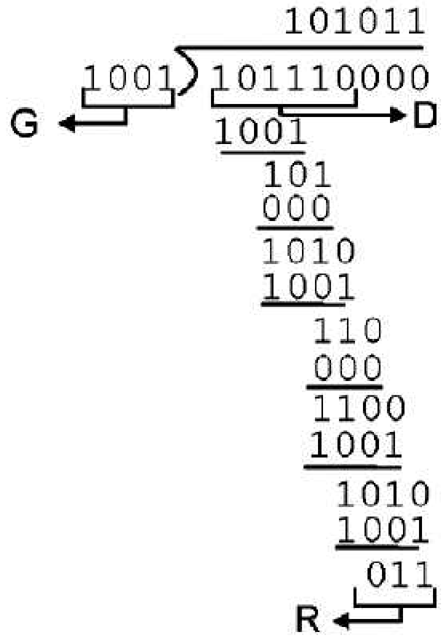


Figure 7.11: An example CRC calculation

Figure 3.11 illustrates this calculation for the case of *D* = 101110, *d* = 6 and *G* = 1001, *r*=3. The nine bits transmitted in this case are 101110 011. You should check these calculations for yourself and also check that indeed

$$D2^r = 101011 * G \text{ XOR } R.$$

International standards have been defined for 8-, 12-, 16- and 32-bit generators, *G*. An 8-bit CRC is used to protect the 5-byte header in ATM cells. The CRC-32 32-bit standard, which has been adopted in a number of link-level IEEE protocols, uses a generator of

$$G_{CRC\text{-}32} = 100000100110000010001110110110111$$

Each of the CRC standards can detect burst errors of less than $r+1$ bits and any odd number of bit errors. Furthermore, under appropriate assumptions, a burst of length greater than $r+1$ bits is detected with probability $1 - 0.5^r$.

CRC Example

Example: Suppose that the message consisted of the two bytes (6,23) as in the previous example. These can be considered to be the hexadecimal number 0617 which can be considered to be the binary number 0000-0110-0001-0111. Suppose that we use a checksum register one-byte wide and use a constant divisor of 1001, then the checksum is the remainder after 0000-0110-0001-0111 is divided by 1001. While in this case, this calculation could obviously be performed using common garden variety 32-bit registers, in the general case this is messy. So instead, we'll do the division using good-'ol long division which you learned in school (remember?). Except this time, it's in binary:

```
          ...0000010101101 = 00AD =  173 = QUOTIENT
         _____-____-____-____-
9= 1001 ) 0000011000010111 = 0617 = 1559 = DIVIDEND
DIVISOR   0000.,,.---,.,,,
          ----.,,.---,.,,,
           0000,,.---,.,,,
           0000,,.---,.,,,
           ----,,.---,.,,,
            0001,.---,.,,,
            0000,.---,.,,,
            ----,.---,.,,,
             0011.---,.,,,
             0000.---,.,,,
             ----.---,.,,,
              0110.--,.,,,
              0000.--,.,,,
              ----.--,.,,,
               1100.-,.,,,
               1001.-,.,,,
               ====.-,.,,,
                0110.,.,,,
                0000.,.,,,
                ----.,.,,,
                 1100,.,,,
                 1001,.,,,
                 ====,.,,,
                  0111.,,,
                  0000.,,,
                  ----.,,,
                   1110,,,
                   1001,,,
                   ====,,,
                    1011,,
                    1001,,
                    ====,,
                     0101,
                     0000,
                     ----
                      1011
                      1001
                      ====
                      0010 = 02 = 2 = REMAINDER
```

In decimal this is "1559 divided by 9 is 173 with a remainder of 2".

Although the effect of each bit of the input message on the quotient is not all that significant, the 4-bit remainder gets kicked about quite a lot during the calculation, and if more bytes were added to the message (dividend) it's value could change radically again very quickly. This is why division works where addition doesn't.

In case you're wondering, using this 4-bit checksum the transmitted message would look like this (in hexadecimal): 06172 (where the 0617 is the message and the 2 is the checksum). The receiver would divide 0617 by 9 and see whether the remainder was 2.

Practice

Table shows the generator polynomials used by some common CRC standards. The "Hex" column shows the hexadecimal representation of the generator polynomial; the most significant bit is omitted, as it is always 1.

TABLE 7.1. GENERATOR POLYNOMIALS OF SOME CRC CODES

Common Name	r	Generator	
		Polynomial	Hex
CRC-12	12	$x^{12}+x^{11}+x^3+x^2+x+1$	80F
CRC-16	16	$x^{16}+x^{15}+x^2+1$	8005
CRC-CCITT	16	$x^{16}+x^{12}+x^5+1$	1021
CRC-32	32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+$ $x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$	04C11DB7

The CRC standards differ in ways other than the choice of generating polynomial. Most initialize by assuming that the message has been preceded by certain nonzero bits, others do no such initialization. Most transmit the bits within a byte least significant bit first, some most significant bit first. Most append the checksum least significant byte first, others most significant byte first. Some complement the checksum. CRC-12 is used for transmission of 6-bit character streams, and the others are for 8-bit characters, or 8-bit bytes of arbitrary data. CRC-16 is used in IBM's BISYNCH communication standard. The CRC-CCITT polynomial, also known as ITU-TSS, is used in communication protocols such as XMODEM, X.25, IBM's SDLC, and ISO's HDLC [Tanen]. CRC-32 is also known as AUTODIN-II and ITU-TSS (ITU-TSS has defined both 16- and a 32-bit polynomials). It is used in PKZip, Ethernet, AAL5 (ATM Adaptation Layer 5), FDDI (Fiber Distributed Data Interface), the IEEE-802 LAN/MAN standard, and in some DOD applications. It is the one for which software algorithms are given here. The first three polynomials in Table 3–1 have as a factor. The last (CRC-32) does not. To detect the error of erroneous insertion or deletion of leading 0's, some protocols prepend one or more nonzero bits to the message. These don't actually get transmitted, they are simply used to initialize the key register (described below) used in the CRC calculation. A value of r 1-bits seems to be universally used. The receiver initializes its register in the same way. The problem of trailing 0's is a little more difficult. There would be no problem if the receiver operated by comparing the remainder based on just the message bits to the checksum received. But, it seems to be simpler for the receiver to calculate the remainder for all bits received (message and checksum) plus r appended 0-bits. The remainder should be 0. But, with a 0 remainder, if the message has trailing 0-bits inserted or deleted, the remainder will still be 0, so this error goes undetected. The usual solution to this problem is for the sender to complement the checksum before appending it. Because this makes the remainder calculated by the receiver nonzero (usually), the remainder will change if trailing 0's are inserted or deleted. How then does the receiver recognize an error-free transmission?

Using the "mod" notation for remainder, we know that

$$(Mx^r + R) \bmod G = 0.$$

Denoting the "complement" of the polynomial R by R' we have

$$\begin{aligned}(Mx^r + R') \bmod G &= (Mx^r + (x^{r-1} + x^{r-2} + \ldots + 1 - R)) \bmod G\\ &= ((Mx^r + R) + x^{r-1} + x^{r-2} + \ldots + 1) \bmod G\\ &= (x^{r-1} + x^{r-2} + \ldots + 1) \bmod G.\end{aligned}$$

Thus the checksum calculated by the receiver for an error-free transmission should be

$$(x^{r-1} + x^{r-2} + \ldots + 1) \bmod G.$$

This is a constant (for a given G). For CRC-32 this polynomial, called the *residual* or *residue*, is

$$x^{31} + x^{30} + x^{26} + x^{25} + x^{24} + x^{18} + x^{15} + x^{14} + x^{12} +$$
$$x^{11} + x^{10} + x^{8} + x^{6} + x^{5} + x^{4} + x^{3} + x\ 1,$$

or hex C704DD7B

7.4 Summary

In this unit you have learnt about error, types of error, Error Detection & Correction, Parity Check Codes, and Cyclic Redundancy Codes.

- ❖ There are many reasons such as noise, cross-talk etc., which may help data to get corrupted during transmission. The upper layers work on some generalized view of network architecture and are not aware of actual hardware data processing.
- ❖ Hence, the upper layers expect error-free transmission between the systems. Most of the applications would not function expectedly if they receive erroneous data. Applications such as voice and video

may not be that affected and with some errors they may still function well.

- Data-link layer uses some error control mechanism to ensure that frames (data bit streams) are transmitted with certain level of accuracy. But to understand how errors is controlled, it is essential to know what types of errors may occur.
- The movement of digital data from one location to another can result in transmission errors, the receiver not receiving the same signal as transmitted by the transmitter as a result of electrical noise in the transmission process.
- Sometimes a noise pulse may be large enough to alter the logic level of the signal. For example, the transmitted sequence 1001 may be incorrectly received as 1101. In order to detect such errors a *parity bit* is often used.

7.5 Review Questions

Q1. Define single bit, multiple bit, and burst error.

Q2. Define error detection and correction in detail.

Q3. Define parity check codes.

Q4. Explain CRC in error detection.

UNIT-VIII

Data Link Protocols

Structure

8.0 Introduction

In this unit you will learn about data link layer protocols. This unit is divided in to seven sections. In the first section i.e. Sec 4.1 you will know about data link layer. As you have studied earlier that Data-link layer is responsible for implementation of point-to-point flow and error control mechanism. You will learn Stop and Wait Protocols in Sec 4.2. This flow control mechanism forces the sender after transmitting a data frame to stop and wait until the acknowledgement of the data-frame sent is received. In the next section i.e. Sec.4.3 Noise Free and Noisy Channels performance and efficiency is described. Sec. 4.4 describe verification of Protocols using Finite State Machine. Sec. 4.5 define HDLC Data Link Protocol. In Sec. 4.6 and 4.7 you will find summary and review questions respectively.

Objectives

After studying this unit you should be able to:

- Define data Link Protocols.
- Describe Stop and Wait Protocols
- Express Noise Free and Noisy Channels performance and efficiency
- Define Verification of Protocols using Finite State Machine
- Describe HDLC Data Link Protocol.

8.1 Data Link Protocols

The purpose of the data link layer is to transfer blocks of data without error between two adjacent devices. Adjacent devices are physically connected by a communication channel such as telephone lines, coaxial cables, optical fibers, or satellites. The implication of such a physical link is that the data bits are delivered in exactly the same order in which they are sent. The physical link has no inherent storage capacity, therefore the delay involved is the propagation delay over the link.

Transmission of data over the link would be very simple indeed if no error ever occurred. Unfortunately, this is not so in a real physical link for a number of reasons:

- ❖ Natural phenomena such as noises and interference are introduced into the link causing errors in detecting the data.
- ❖ There is a propagation delay in the link.
- ❖ There is a finite data processing time required by the transmitting and receiving stations.

A data link protocol thus has to be designed to ensure an error-free transmission and also to achieve an efficiency of the data transfer as high as possible.

Simplified Model

At layer 2, user's messages are already broken up into segments. Control information (headers and trailers) is added to each segment to make up a frame. We are concerned only with the transmission of frames between two adjacent IMPs.

Functions and requirements of the Data Link Protocols

The basic function of the layer is to transmit frames over a physical communication link. Transmission may be *half duplex* or *full duplex*. To ensure that frames are delivered free of errors to the destination station (IMP) a number of requirements are placed on a data link protocol. The protocol (control mechanism) should be capable of performing:

1. The identification of a frame (i.e. recognize the first and last bits of a frame).
2. The transmission of frames of any length up to a given maximum. Any bit pattern is permitted in a frame.
3. The detection of transmission errors.

4. The retransmission of frames which were damaged by errors.
5. The assurance that no frames were lost.
6. In a multi drop configuration
    - Some mechanism must be used for preventing conflicts caused by simultaneous transmission by many stations.
7. The detection of failure or abnormal situations for control and monitoring purposes.

It should be noted that as far as layer 2 is concerned a host message is pure data, every single bit of which is to be delivered to the other host. The frame header pertains to layer 2 and is never given to the host.

We will first look at three elementary data link protocols of increasing complexity.

8.2 Stop and Wait Protocols

In this protocol we assume that

- Data are transmitted in one direction only
- No errors occur (perfect channel)
- The receiver can only process the received information at a finite rate

These assumptions imply that the transmitter cannot send frames at a rate faster than the receiver can process them.

The problem here is how to prevent the sender from flooding the receiver.

A general solution to this problem is to have the receiver provide some sort of feedback to the sender. The process could be as follows: The receiver send an acknowledge frame back to the sender telling the sender that the last received frame has been processed and passed to the host; permission to send the next frame is granted. The sender, after having sent a frame, must wait for the acknowledge frame from the receiver before sending another frame. This protocol is known as *stop-and-wait*.

The protocol is as follows:

```
/* protocol 2 */
Sender()
{
    forever
    {
        from_host(buffer);
        S.info = buffer;
        sendf(S);
        wait(event);
    }
}
Receiver()
{
    forever
    {
        wait(event);
        getf(R);
        to_host(R.info);
        sendf(S);
    }
}
```

Stop and Wait transmission is the simplest reliability technique and is adequate for a very simple communications protocol. A stop and wait protocol transmits a Protocol Data Unit (PDU) of information and then waits for a response. The receiver receives each PDU and sends an Acknowledgement (ACK) PDU if a data PDU is received correctly, and a Negative Acknowledgement (NACK) PDU if the data was not received. In practice, the receiver may not be able to reliably identify whether a PDU has been received, and the transmitter will usually also need to implement a timer to recover from the condition where the receiver does not respond.

Under normal transmission the sender will receive an ACK for the data and then commence transmission of the next data block. For a long delay link, the sender may have to wait an appreciable time for this response. While it is waiting the sender is said to be in the "idle" state and is unable to send further data.

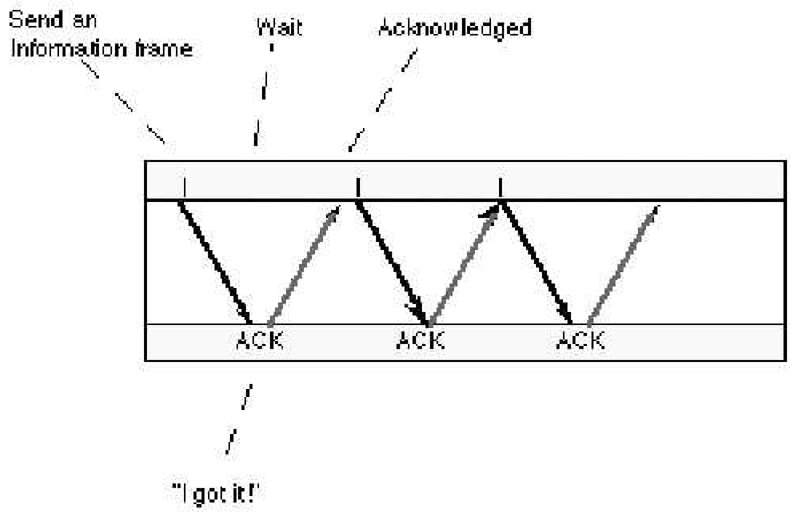


Figure 8.1 Stop and Wait ARQ - Waiting for Acknowledgment (ACK) from the remote node.

The blue arrows show the sequence of data PDUs being sent across the link from the sender (top to the receiver (bottom). A Stop and Wait protocol relies on two way transmission (full duplex or half duplex) to allow the receiver at the remote node to return PDUs acknowledging the successful transmission. The acknowledgements are shown in green in the diagram, and flow back to the original sender. A small processing delay may be introduced between reception of the last byte of a Data PDU and generation of the corresponding ACK.

When PDUs are lost, the receiver will not normally be able to identify the loss (most receivers will not receive anything, not even an indication that something has been corrupted). The transmitter must then rely upon a timer to detect the lack of a response.

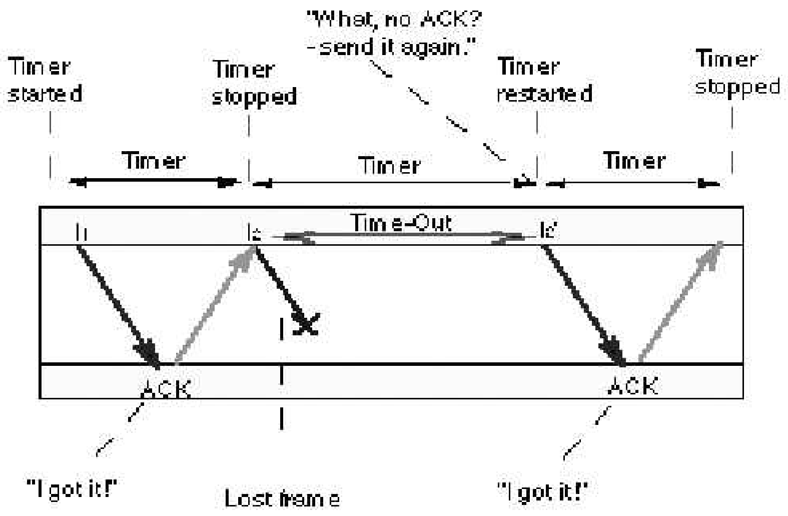


Figure 8.2: Stop and Wait ARQ - Retransmission due to timer expiry

In the diagram, the second PDU of Data is corrupted during transmission. The receiver discards the corrupted data (by noting that it is followed by an invalid data checksum). The sender is unaware of this loss, but starts a timer after sending each PDU. Normally an ACK PDU is received before this the timer expires. In this case no ACK is received, and the timer counts down to zero and triggers retransmission of the same PDU by the sender. The sender always starts a timer following transmission, but in the second transmission receives an ACK PDU before the timer expires, finally indicating that the data has now been received by the remote node.

The state diagram (also showing the operation of NACK) is shown below:

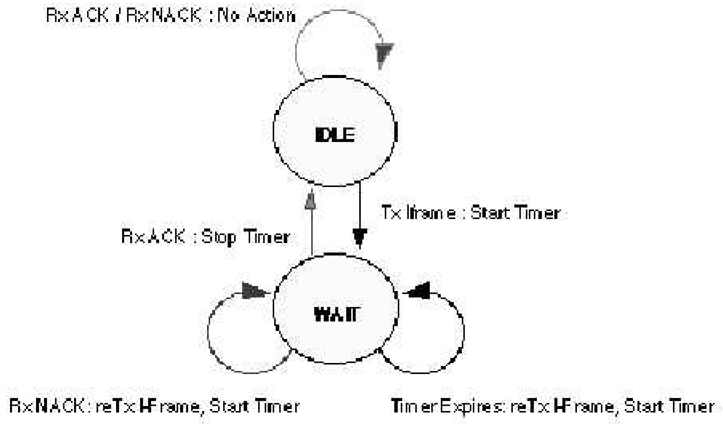

RxACK / RxNACK : No Action
IDLE
Tx Iframe : Start Timer
RxACK : Stop Timer
WAIT
RxNACK: reTx I-Frame, Start Timer
Timer Expires: reTx I-Frame, Start Timer

Figure 8.3: State Diagram for a simple stop and wait protocol

(Green for ACK, Blue for Data, Red for NACK)

Check your progress

Q1. Define data link layer protocol.

Q2. Explain stop and wait protocol.

8.3 Noise Free and Noisy Channels performance and efficiency

In this protocol the unreal "error free" assumption in protocol 2 is dropped. Frames may be either damaged or lost completely. We assume that transmission errors in the frame are detected by the hardware checksum.

One suggestion is that the sender would send a frame, the receiver would send an ACK frame only if the frame is received correctly. If the frame is in error the receiver simply ignores it; the transmitter would time out and would retransmit it.

One fatal flaw with the above scheme is that if the ACK frame is lost or damaged, duplicate frames are accepted at the receiver without the receiver knowing it.

Imagine a situation where the receiver has just sent an ACK frame back to the sender saying that it correctly received and already passed a frame to its host. However, the ACK frame gets lost completely, the sender times out and retransmits the frame. There is no way for the receiver to tell whether this frame is a retransmitted frame or a new frame, so the receiver accepts this duplicate happily and transfers it to the host. The protocol thus fails in this aspect.

To overcome this problem it is required that the receiver be able to distinguish a frame that it is seeing for the first time from a retransmission. One way to achieve this is to have the sender put a sequence number in the header of each frame it sends. The receiver then can check the sequence number of each arriving frame to see if it is a new frame or a duplicate to be discarded.

The receiver needs to distinguish only 2 possibilities: a new frame or a duplicate; a 1-bit sequence number is sufficient. At any instant the receiver expects a particular sequence number. Any wrong sequence numbered frame arriving at the receiver is rejected as a duplicate. A correctly numbered frame arriving at the receiver is accepted, passed to the host, and the expected sequence number is incremented by 1 (modulo 2).

The protocol is depicted below:

```
/* protocol 3 */

Sender()

{

    NFTS = 0;           /* NFTS = Next Frame To Send */

    from_host(buffer);

    forever

    {

        S.seq = NFTS;

        S.info = buffer;

        sendf(S);

        start_timer(S.seq);

        wait(event);

        if(event == frame_arrival)

        {

            from_host(buffer);
```

```
                ++NFTS;  /* modulo 2 operation */
            }
        }
}


Receiver()
{
        FE = 0;             /* FE = Frame Expected */
        forever
        {
            wait(event);
            if(event == frame_arrival)
            {
                getf(R);
                if(R.seq == FE)
                {
                    to_host(R.info);
                    ++FE;  /* modulo 2 operation */
                }
                sendf(S);      /* ACK */
            }
        }
}
```

This protocol can handle lost frames by timing out. The timeout interval has to be long enough to prevent premature timeouts which could cause a "deadlock" situation.

8.4 Verification of Protocols using Finite State Machine

Finite State Machine Protocol Models

- A protocol may be represented by a finite state machine (protocol machine).
- States are chosen when the protocol machine is waiting for the next event (i.e sending or receiving a protocol data unit PDU).
- The state of the complete protocol is the combination of the state of the two protocol machines and the channel.
- The state of the channel depends on its content.
- Each state may have one or more transitions to other states when protocol events occur.
- Incomplete state machine specification.
- Deadlock states.

Real time protocols and programs are very complex. Basic concept in verifying is Finite State Machine. In this technique, every machine which is sender or receiver will be in a specific state at given time. State will contain all the values of the variable which also include program counter.

At the receiver end we could find out from all the possible states two important ones: waiting for frame 0 or waiting for frame 1. All other states can be considered as transient, which mean from those are the states from one main state to another main state. These are the states those instants where system is waiting for next event to occur. At this instant state of system is determined by state of variable at that instant. The number of states is then 2 to the power n, where n is the number of bits needed to represent all the variables combined.

The combination of all the states of the two protocol machines and the Channel is the state of the complete system. From every state, there might be zero or more probable transitions to another state. Transition happens when an event occurs. For protocol machine transition happens when a frame is sent or received or when time frame expires.

A finite state machine model is shown in the figure. This graph corresponds to protocol 3 as described above: each protocol machine has two states and the channel has four states. A total of 16 states exist, not all of them reachable from the initial one. The unreachable ones are not shown in the figure. Checksum errors are also ignored here for simplicity.

State diagram for protocol:

State diagram for protocol:

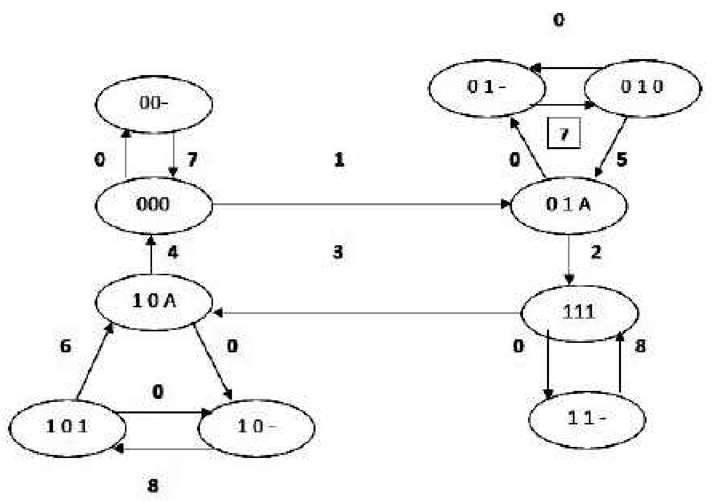


Figure 8.4 State diagram for protocol

TRANSITION	WHO RUNS?	FRAME ACCEPTED	FRAME EMMITED	TO NETWORK LAYER
0	-	FRAME LOST	FRAME LOST	-
1	R	0	A	YES
2	S	A	1	-
3	R	1	A	YES
4	S	A	0	-
5	R	0	A	NO
6	R	1	A	NO
7	S	(timeout)	0	-
8	S	(timeout)	1	-

Table 8.1: Transition table

Every state is named by three characters, SRC, where S is 0 or 1, pertaining to the frame the sender is trying to send; R is also 0 or 1, pertaining to the frame the receiver expects, and C is 0, 1, A, or empty (–), pertaining to the state of the channel. In this example the initial state has been chosen as (000). In other words, the sender has just sent frame 0, the receiver expects frame 0, and frame 0 is currently on the channel.

Nine type of transitions are shown above. Transition 1 consists of the channel correctly delivering packet 0 to the receiver, with the receiver then changing its state to expect frame 1 and emitting an acknowledgement. Transition 0 consists of the channel losing its contents Transition 1 also corresponds to the receiver delivering packet 0 to the network layer. The other transitions are listed in table above. The arrival of a frame with a checksum error has not been shown because it does not change the state.

During usual operation, transitions 1, 2, 3, and 4 are repeated in order over and over. In each cycle, two packets are delivered, bringing the sender back to the initial state of trying to send a new frame with sequence number 0. If the channel loses frame 0, it makes a transition from state (000) to state (00–). Eventually, the sender times out (transition 7) and the system moves back to (000). The loss of an acknowledgement is more complicated, requiring two transitions, 7 and 5, or 8 and 6, to repair the damage.

8.5 HDLC Data Link Protocol.

HDLC is the protocol which is now considered an umbrella under which many Wide Area protocols sit. ITU-T developed HDLC in 1979, and within HDLC there are three types of stations defined:

- **Primary Station** - this completely controls all data link operations issuing commands from secondary stations and has the ability to hold separate sessions with different stations.
- **Secondary Station** - this can only send responses to one primary station. Secondary stations only talk to each other via a Primary station.
- **Combined Station** - this can transmit and receive commands and responses from one other station.

Configuring a channel for use by a station can be done in one of three ways:

- **Unbalanced** - this configuration allows one primary station to talk to a number of secondary stations over half-duplex, full-duplex, switched, un-switched, point-to-point or multipoint paths.
- **Symmetrical** - where commands and responses are multiplexed over one physical channel when two stations with primary and secondary parts have a point-to-point link joining them.
- **Balanced** - where two combined stations communicate over a point-to-point link which can be full/half-duplex or switch/un-switched.

When transferring data, stations are in one of three modes:

- **Normal Response Mode (NRM)** where the secondary station needs permission from the primary station before it can transmit data. Mainly used on multi-point lines.
- **Asynchronous Response Mode (ARM)** where the secondary station can send data without receiving permission from the primary station. This is hardly ever used.
- **Asynchronous Balanced Mode (ABM)** where either station can initiate transmission without permission from the other. This is the most common mode used on point-to-point links.

The following diagram details the HDLC frame format:

RIL-089

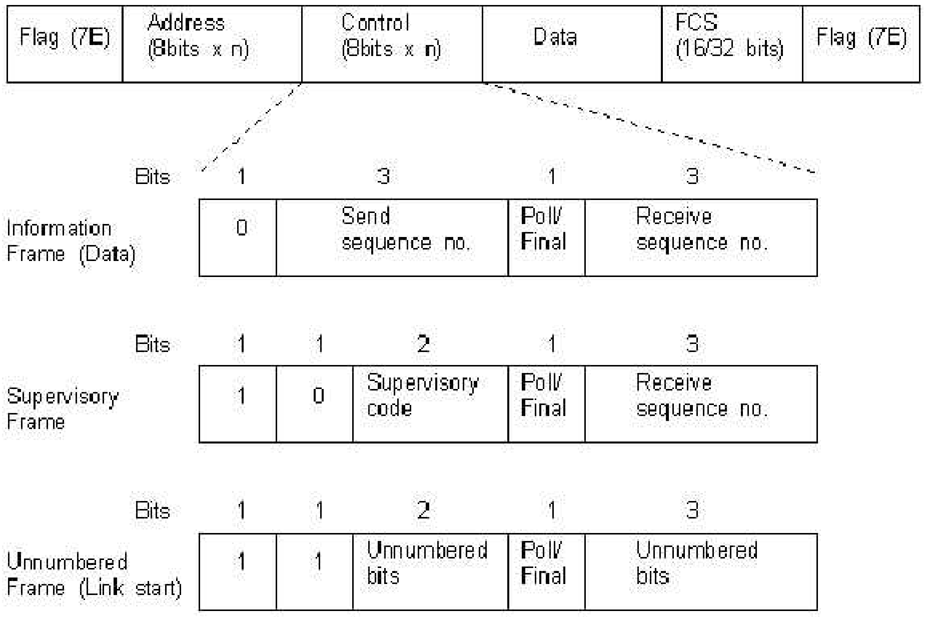


Figure 8.5 HDLC Frame Format

The HDLC frame begins and ends the error checking procedure with 0x7E which is 01111110 in binary.

There are three types of HDLC frame types defined by the control field:

- **Information Frames are used for the data transfer between stations. The send sequence, or next send N(S), and the receive sequence, or next receive N(R), hold the frame sequence numbers. The Poll/Final bit is called Poll when used by the primary station to obtain a response from a secondary station, and Final when used by the secondary station to indicate a response or the end of transmission.**
- **Supervisory Frames are used to acknowledge frames, request for retransmissions or to ask for suspension of transmission. The Supervisory code denotes the type of supervisory frame being sent.**
- **Unnumbered Frames are used for link initialization or link disconnection. The Unnumbered bits indicate the type of unnumbered frame being used.**

Types of Frames in HDLC

HDLC defines three types of frames:

1. Information frames (I-frame)
2. Supervisory frame (S-frame)
3. Unnumbered frame (U-frame)

1. Information frames

- I-frames carry user's data and control information about user's data.
- I-frame carries user data in the information field.
- The I-frame format is shown in diagram.

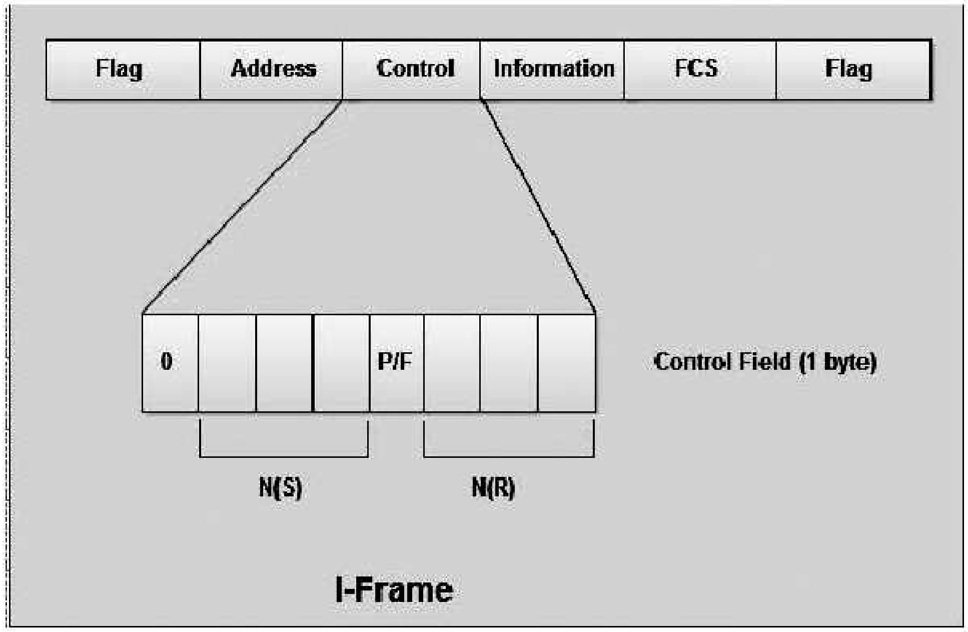


Figure 8.6 I-frame

- The first bit of control field is always zero, *i.e.* the presence of zero at this place indicates that it is I-frame.
- Bit number 2, 3 & 4 in control field is called N(S) that specifies the sequence number of the frame. Thus it specifies the number of the frame that is currently being sent. Since it is a 3.bit field, only eight sequence numbers are possible 0, 1,2,3,4,5,6, 7 (000 to 111).
- Bit number 5 in control field is P/F i.e. Poll/Final and is used for these two purposes. It has, meaning only when it is set i.e. when P/F=1. It can represent the following two cases.
    - It means poll when frame is sent by a primary station to secondary (when address field contains the address of receiver).
    - It means final when frame is sent by secondary to a primary (when the address field contains the address of the sender).

- ❖ Bit number 6, 7, and 8 in control field specifies N(R) i.e. the sequence number of the frame expected in return in two-way communication.
- ❖ If last frame received was error-free then N(R) number will be that of the next frame is sequence. If the last frame was not received correctly, the N(R) number will be the number of the damaged frame, asking for its retransmission.

2. Supervisory frame

- ❖ S-frame carries control information, primarily data link layer flow and error controls.
- ❖ It does not contain information field.
- ❖ The format of S-frame is shown in diagram.

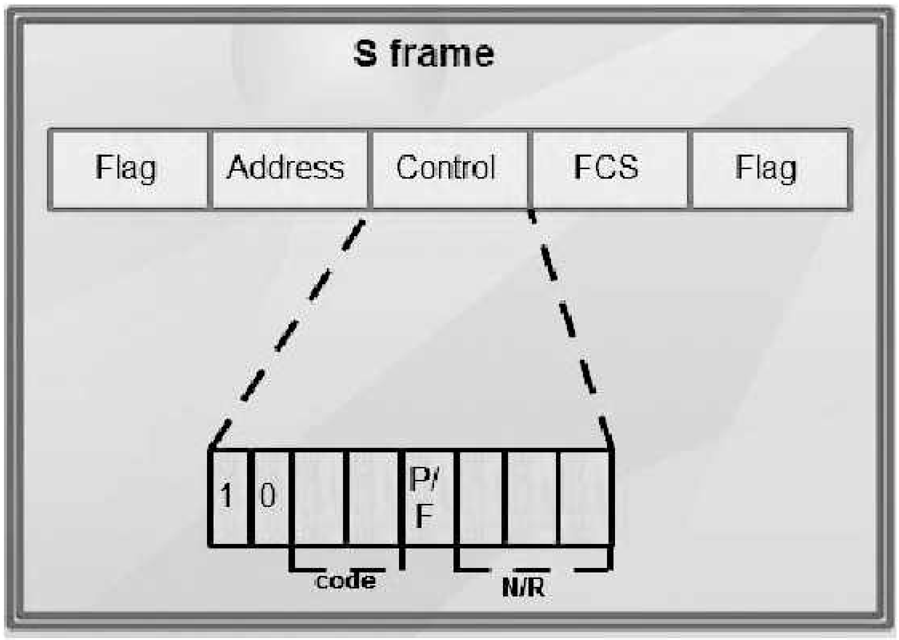


Figure 8.7: S-Frame

- ❖ The first two bits in the control field of S-frame are always 10.
- ❖ Then there is a bit code field that specifies four types of S-frame with combination 00,01, 10, 11 as shown in table :-

Table: Types of S-frame

Code	Command
00	RR Receive Ready
01	REJ Reject
10	RNR Receive Not Ready
11	SREJ Selective Reject

1. RR, Receive Ready-used to acknowledge frames when no I-frames are availab1e to piggyback the acknowledgement.
2. REJ, Reject-used by the receiver to send a NAK when error has occurred.
3. RNR Receive Not Ready-used for flow control.
4. SREJ Selective Reject-indicates to the transmitter that it should retransmit the frame indicated in the N(R) subfield.

- There is no N(S) field in control field of S-frame as S-frames do not transmit data.
- *P/F* bit is the fifth bit and serves the same purpose as discussed earlier.
- Last three bits in control field indicates N(R) *i.e.* they correspond to the ACK or NAK value.

3. Unnumbered frame

- U-frames are reserved for system management and information carried by them is used for managing the link
- U-frames are used to exchange session management and control information between the two connected devices.
- Information field in U-frame does not carry user information rather, it carries system management information.
- The frame format of U-frame is shown in diagram.
- U-frame is identified by the presence of 11 in the first and second bit position in control field.
- These frames do not contain N(S) or N(R) in control field.

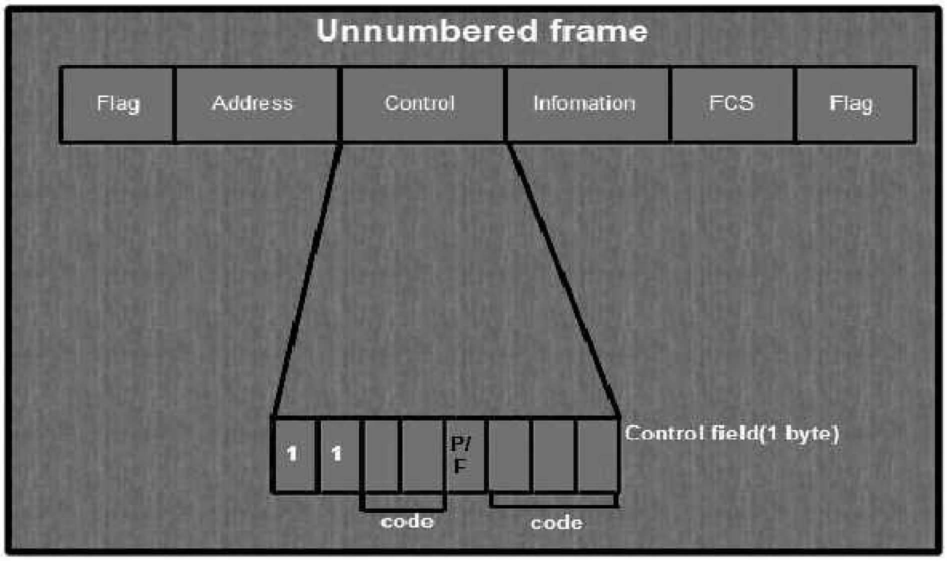


Figure 8-8: U-Frame

- U-frame contains two code fields, one two bit and other three bit.
- These five bits can create upto 32 different U-frames.
- *P/F* bit in control field has same purpose in V-frame as discussed earlier.

Protocol Structure - HDLC: High Level Data Link Control

Flag - The value of the flag is always (0x7E).

Address field - Defines the address of the secondary station which is sending the frame or the destination of the frame sent by the primary station. It contains Service Access Point (6bits), a Command/Response bit to indicate whether the frame relates to information frames (I-frames) being sent from the node or received by the node, and an address extension bit which is usually set to true to indicate that the address is of length one byte. When set to false it indicates an additional byte follows.

Extended address - HDLC provides another type of extension to the basic format. The address field may be extended to more than one byte by agreement between the involved parties.

Control field - Serves to identify the type of the frame. In addition, it includes sequence numbers, control features and error tracking according to the frame type.

FCS - The Frame Check Sequence (FCS) enables a high level of physical error control by allowing the integrity of the transmitted frame data to be checked.

8.6 Summary

In this unit you have learnt about Data Link Protocols, Stop and Wait Protocols, Noise Free and Noisy Channels performance and efficiency, Verification of Protocols using Finite State Machine, and HDLC Data Link Protocol.

- ❖ The **data link** layer or layer 2 is the second layer of the seven-layer OSI model of **computer networking**. This layer is the **protocol** layer that transfers **data** between adjacent **network** nodes in a wide area **network** (WAN) or between nodes on the same local area **network** (LAN) segment.
- ❖ **Stop-and-wait ARQ** also can be referred as alternating bit protocol is a method used in telecommunications to send information between two connected devices. It ensures that information is not lost due to dropped packets and that packets are received in the correct order. It is the simplest kind of automatic repeat-request (ARQ) method.
- ❖ A stop-and-wait ARQ sender sends one frame at a time; it is a special case of the general sliding window protocol with both transmit and receive window sizes equal to 1 and more than one respectively . After sending each frame, the sender doesn't send any further frames until it receives an acknowledgement (ACK) signal.
- ❖ After receiving a good frame, the receiver sends an ACK. If the ACK does not reach the sender before a certain time, known as the timeout, the sender sends the same frame again. Timer is set after each frame transmission. The above behavior is the simplest Stop-and-Wait implementation. However, in a real life implementation there are problems to be addressed.
- ❖ HDLC defines a method for encapsulating or formatting data into frames for synchronous transmission over synchronous serial WAN links to remote sites. HDLC is a bit-stream protocol (bit streams are not broken into individual characters) that uses a 32-bit checksum for error correction and supports full-duplex communication. HDLC frames consist of a flag byte followed by address and control information, data bits, and a CRC byte. A control field at the start of a frame is used for establishing and terminating data link connections.

8.7 Review Questions

Q1. Define functions and requirements of the Data Link Protocols

Q2. Explain Stop and Wait ARQ Retransmission due to timer expiry.

Q3. Define noise free and noisy channels performance and efficiency.

Q4. Explain verification of protocols using finite state machine

Q5. Describe all three types of HDLC frames.

BIBLIOGRAPHY

Behrouz A. Forouzan. *Data Communications and Networking, Fourth Edition.* McGraw-Hill, 2007.

Larry L. Peterson and Bruce S. Davie. *Computer Networks: A Systems Approach, 4th Edition.* Morgan Kaufmann, 2007.

William Stallings. *Data and Computer Communications, Eighth Edition.* Prentice Hall, 2006.

Andrew S. Tanenbaum. *Computer Networks, Fourth Edition.* Prentice Hall PTR, 2003.

Alan Dennis. *Networking in the Internet Age.* John Wiley & Sons, 2002.

Douglas Comer, David Stevens, and Michael Evangelista. *Internetworking with TCP/IP, Volume 2: Design, Implementation, and Internals, Fourth Edition.* Prentice-Hall, 2006.

Douglas Comer and David Stevens. *Internetworking with TCP/IP, Volume 3: Client-Server Programming and Applications, Linux/Posix Sockets Version.* Prentice-Hall, 2001.

Douglas Comer. *Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architecture, Fourth Edition.* Prentice-Hall, 2000.

Ted G. Lewis. *Network Science: Theory and Applications.* John Wiley & Sons, 2011.

Peter M. Higgins. *Nets, Puzzles, and Postmen: An Exploration of Mathematical Connections.* Oxford University Press, 2007.

Internet Engineering Task Force (IETF). http://www.ietf.org

IETF Requests for Comments (RFCs) – Internet standards. http://www.ietf.org/rfc.html

Internet Assigned Numbers Authority (IANA). http://www.iana.org

A. McAuley, "Weighted Sum Codes for Error Detection and Their Comparison with Existing Codes", IEEE/ACM Transactions on Networking, Vol. 2, No. 1 (February 1994), pp. 16-22.

J. Nonnenmacher, E. Biersak, D. Towsley, "Parity-Based Loss Recovery for Reliable Multicast Transmission," IEEE/ACM Transactions on Networking, Vol. 6, No. 4 (Aug. 1998), Pages 349 - 361.

B. Draden, D. Borman, C. Partridge, "Computing the Internet Checksum," RFC 1071, Sept. 1988.

D. Rubenstein, J. Kurose, D. Towsley ``Real-Time Reliable Multicast Using Proactive Forward Error Correction", Proceedings of NOSSDAV '98, (Cambridge, UK, July 1998).

Uttar Pradesh Rajarshi Tandon
Open University

Master in Computer Science

MCS-108

Data Communication and Computer Networks

Block

3

NETWORK & TRANSPORT LAYER

© UPRTOU, Prayagraj. 2019
ISBN : 978-93-83328-25-3

RIL-089

Block-III Network & Transport layer protocols

This is the third block on Data communication & Networks and having detail description of network and transport layer. The network layer or layer 3 is the third layer of the seven-layer OSI model of computer networking. The network layer is responsible for packet forwarding including routing through intermediate routers, since it knows the address of neighboring network nodes, and it also manages quality of service (QoS), and recognizes and forwards local host domain messages to the Transport layer (layer 4). In this block we also covers transport player of ISO-OSI model. The transport layer is the fourth layer of the OSI reference model. It provides transparent transfer of data between end systems using the services of the network layer (e.g. IP) below to move PDUs of data between the two communicating systems. The transport service is said to perform "peer to peer" communication, with the remote (peer) transport entity. The data communicated by the transport layer is encapsulated in a transport layer PDU and sent in a network layer SDU. The network layer nodes (i.e. Intermediate Systems (IS)) transfer the transport PDU intact, without decoding or modifying the content of the PDU. In this way, only the peer transport entities actually communicate using the PDUs of the transport protocol.

So we will begin the first unit on Integrated Services for Digital Network (ISDN). In this unit we also describe ISDN Interfaces, Devices. You will also learn about Channel Structure, Dead Locks and their avoidance, and ATM.

Second unit begins with Internetworking. In this unit you will know all about internetworking like Bridges, Routers and Gateways, Internet Architecture and Addressing.

In the third unit, we provide another important topic i.e. Transport Layer Protocols. This unit also describe Design issues of transport layer, Quality of Services, Primitives Connection Management and Addressing.

In the last, the fourth unit introduced the concept of Connection Establishment and Releases. We also define Use of Timers, Flow Control and Buffering, Multiplexing, Crash Recovery.

As you study the material, you will find that figures, tables are properly used and these will help to understand the concept. There are many sections in the units to easily understand the topic. Every unit has summary and review questions in the end of the unit which will help you to review yourself.

In your study, you will find that the every unit has different equal length and your study time will vary for each unit.

We hope you enjoy studying the material and once again wish you all the best for your success.

UNIT-IX
ISDN

Structure

9.0 Introduction

This unit focused on Integrated Services for Digital Network (ISDN). In this unit there are eight sections. In the first section i.e. Sec.9.1 you will know about ISDN. As you have studied earlier in telecommunications technology, the abbreviation ISDN stands for the technical term "Integrated Services Digital Network" and refers to a digital standard for telephone networks. As the name implies, various types of communication services can be transmitted through this kind of network. In addition to voice calls, ISDN allows packet-switched or circuit-switched data transmission and fix transmission. In some countries, ISDN technology has taken over from analogue telephone networks, integrating all types of services into a common network for the first time. This digitization has also enabled digital ISDN phone lines to be made available to end users. In Sec. 9.2 you will learn about Interfaces. In the next section we define Devices. In section 9.4 you will know about Channel Structure of ISDN. In the next section i.e. Sec. 9.5 Dead Locks and their avoidance. Sec. 9.6 describe ATM. In Sec. 9.7 and 9.8 you will find summary and review questions respectively.

Objectives

After studying this unit you should be able to:

- Define Integrated Services Digital Network.
- Describe Interfaces, Devices and Channel Structure of ISDN
- Express Dead Locks and their avoidance.
- Define ATM

9.1 Integrated Services Digital Network

ISDN, which stands for Integrated Services **Digital Network**, is a system of digital phone connections which has been available for over a decade. This system allows voice and data to be transmitted simultaneously across the world using end-to-end digital connectivity.

With ISDN, voice and data are carried by bearer channels (**B channels**) occupying a bandwidth of 64 kb/s (bits per second). Some switches limit B channels to a capacity of 56 kb/s. A data channel (**D channel**) handles signaling at 16 kb/s or 64 kb/s, depending on the service type. Note that, in ISDN terminology, "**k**" means 1000 (10^3), not 1024 (2^{10}) as in many computer applications (the designator "**K**" is sometimes used to represent this value); therefore, a 64 kb/s channel carries data at a rate of 64000 b/s. A new set of standard prefixes has recently been created to handle this. Under this scheme, "k" (kilo-) means 1000 (10^3), "M" (mega-) means 1000000 (10^6), and so on, and "Ki" (kibi-) means 1024 (2^{10}), "Mi" (mebi-) means 1048576 (2^{20}), and so on.

There are two basic types of ISDN service: **Basic Rate Interface (BRI)** and **Primary Rate Interface (PRI)**. BRI consists of two 64 kb/s B channels and one 16 kb/s D channel for a total of 144 kb/s. This basic service is intended to meet the needs of most individual users.

PRI is intended for users with greater capacity requirements. Typically the channel structure is 23 B channels plus one 64 kb/s D channel for a total of 1536 kb/s. In Europe, PRI consists of 30 B channels plus one 64 kb/s D channel for a total of 1984 kb/s. It is also possible to support multiple PRI lines with one 64 kb/s D channel using **Non-Facility Associated Signaling (NFAS)**.

H channels provide a way to aggregate B channels. They are implemented as:

- H0=384 kb/s (6 B channels)
- H10=1472 kb/s (23 B channels)
- H11=1536 kb/s (24 B channels)
- H12=1920 kb/s (30 B channels) - International (E1) only

To access BRI service, it is necessary to subscribe to an ISDN phone line. Customer must be within 18000 feet (about 3.4 miles or 5.5 km) of the telephone company central office for BRI service; beyond that, expensive repeater devices are required, or ISDN service may not be available at all. Customers will also need special equipment to communicate with the phone company switch and with other ISDN devices. These devices include ISDN **Terminal Adapters** (sometimes called, incorrectly, "ISDN Modems") and **ISDN Routers**.

ISDN History

The early phone network consisted of a pure analog system that connected telephone users directly by a mechanical interconnection of wires. This system was very inefficient, was very prone to breakdown and noise, and did not lend itself easily to long-distance connections. Beginning in the 1960s, the telephone system gradually began converting its internal connections to a packet-based, digital switching system. Today, nearly all voice switching in the U.S. is digital within the telephone network. Still, the final connection from the local central office to the customer equipment was, and still largely is, an analog Plain-Old Telephone Service (POTS) line.

A standards movement was started by the International Telephone and Telegraph Consultative Committee (CCITT), now known as the International Telecommunications Union (ITU). The ITU is a United Nations organization that coordinates and standardizes international telecommunications. Original recommendations of ISDN were in CCITT Recommendation I.120 (1984) which described some initial guidelines for implementing ISDN.

Local phone networks, especially the regional Bell operating companies, have long hailed the system, but they had been criticized for being slow to implement ISDN. One good reason for the delay is the fact that the two major switch manufacturers, Northern Telecom (now known as Nortel Networks), and AT&T (whose switch business is now owned by Lucent Technologies), selected different ways to implement the CCITT standards. These standards didn't always interoperate. This situation has been likened to that of earlier 19th century railroading. "People had different gauges, different tracks... nothing worked well."

In the early 1990s, an industry-wide effort began to establish a specific implementation for ISDN in the U.S. Members of the industry agreed to create the National ISDN 1 (NI-1) standard so that end users would not have to know the brand of switch they are connected to in order to buy equipment and software compatible with it. However, there were problems agreeing on this standard. In fact, many western states would not implement NI-1. Both Southwestern Bell and U.S. West (now Qwest) said that they did not plan to deploy NI-1 software in their central office switches due to incompatibilities with their existing ISDN networks.

Ultimately, all the Regional Bell Operating Companies (RBOCs) did support NI-1. A more comprehensive standardization initiative, National

ISDN 2 (NI-2), was later adopted. Some manufacturers of ISDN communications equipment, such as Motorola and U S Robotics (now owned by 3Com), worked with the RBOCs to develop configuration standards for their equipment. These kinds of actions, along with more competitive pricing, inexpensive ISDN connection equipment, and the desire for people to have relatively low-cost high-bandwidth Internet access have made ISDN more popular in recent years.

Most recently, ISDN service has largely been displaced by broadband internet service, such as xDSL and Cable Modem service. These services are faster, less expensive, and easier to set up and maintain than ISDN. Still, ISDN has its place, as backup to dedicated lines, and in locations where broadband service is not yet available.

Advantages of ISDN

- **Speed**
  The modem was a big breakthrough in computer communications. It allowed computers to communicate by converting their digital information into an analog signal to travel through the public phone network. There is an upper limit to the amount of information that an analog telephone line can hold. Currently, it is about 56 kb/s bi-directionally. Commonly available modems have a maximum speed of 56 kb/s, but are limited by the quality of the analog connection and routinely go about 45-50 kb/s. Some phone lines do not support 56 kb/s connections at all. There were currently 2 competing, incompatible 56 kb/s standards (X2 from U S Robotics (recently bought by 3Com), and K56flex from Rockwell/Lucent). This standards problem was resolved when the ITU released the V.90, and later V.92, standard for 56 kb/s modem communications. ISDN allows multiple digital channels to be operated simultaneously through the same regular phone wiring used for analog lines. The change comes about when the telephone company's switches can support digital connections. Therefore, the same physical wiring can be used, but a digital signal, instead of an analog signal, is transmitted across the line. This scheme permits a much higher data transfer rate than analog lines. BRI ISDN, using a channel aggregation protocol such as BONDING or Multilink-PPP, supports an uncompressed data transfer speed of 128 kb/s, plus bandwidth for overhead and signaling. In addition, the latency, or the amount of time it takes for a communication to begin, on an ISDN line is typically about half that of an analog

line. This improves response for interactive applications, such as games.

- **Multiple Devices**

  Previously, it was necessary to have a separate phone line for each device you wished to use simultaneously. For example, one line each was required for a telephone, fax, computer, bridge/router, and live video conference system. Transferring a file to someone while talking on the phone or seeing their live picture on a video screen would require several potentially expensive phone lines.

  ISDN allows multiple devices to share a single line. It is possible to combine many different digital data sources and have the information routed to the proper destination. Since the line is digital, it is easier to keep the noise and interference out while combining these signals. ISDN technically refers to a specific set of digital services provided through a single, standard interface. Without ISDN, distinct interfaces are required instead.

- **Signaling**

  Instead of the phone company sending a ring voltage signal to ring the bell in your phone ("In-Band signal"), it sends a digital packet on a separate channel ("Out-of-Band signal"). The Out-of-Band signal does not disturb established connections, no bandwidth is taken from the data channels, and call setup time is very fast. For example, a V.90 or V.92 modem typically takes 30-60 seconds to establish a connection; an ISDN call setup usually takes less than 2 seconds.

  The signaling also indicates who is calling, what type of call it is (data/voice), and what number was dialed. Available ISDN phone equipment is then capable of making intelligent decisions on how to direct the call.

9.2 Interfaces

In the U.S., the telephone company provides its BRI customers with a **U interface**. The U interface is a two-wire (single pair) interface from the phone switch, the same physical interface provided for POTS lines. It supports full-duplex data transfer over a single pair of wires, therefore only a single device can be connected to a U interface. This device is called a **Network Termination 1 (NT-1)**. The situation is different elsewhere in the world, where the phone company is allowed to supply the NT-1, and thereby the customer is given an S/T interface.

The NT-1 is a relatively simple device that converts the 2-wire U interface into the 4-wire **S/T interface**. The S/T interface supports multiple devices (up to 7 devices can be placed on the S/T bus) because, while it is still a full-duplex interface, there is now a pair of wires for receive data, and another for transmit data. Today, many devices have NT-1s built into their design. This has the advantage of making the devices less expensive and easier to install, but often reduces flexibility by preventing additional devices from being connected.

Technically, ISDN devices must go through a **Network Termination 2 (NT-2)** device, which converts the T interface into the S interface (Note: the S and T interfaces are electrically equivalent). Virtually all ISDN devices include an NT-2 in their design. The NT-2 communicates with terminal equipment, and handles the Layer 2 and 3 ISDN protocols. Devices most commonly expect either a U interface connection (these have a built-in NT-1), or an S/T interface connection.

Devices that connect to the S/T (or S) interface include ISDN capable telephones and FAX machines, video teleconferencing equipment, bridge/routers, and terminal adapters. All devices that are designed for ISDN are designated **Terminal Equipment 1 (TE1)**. All other communication devices that are *not* ISDN capable, but have a POTS telephone interface (also called the **R interface**), including ordinary analog telephones, FAX machines, and modems, are designated **Terminal Equipment 2 (TE2)**. A **Terminal Adapters (TA)** connects a TE2 to an ISDN S/T bus.

Going one step in the opposite direction takes us inside the telephone switch. Remember that the U interface connects the switch to the customer premises equipment. This local loop connection is called *Line Termination* (LT function). The connection to other switches within the phone network is called *Exchange Termination* (ET function). The LT function and the ET function communicate via the **V interface**.

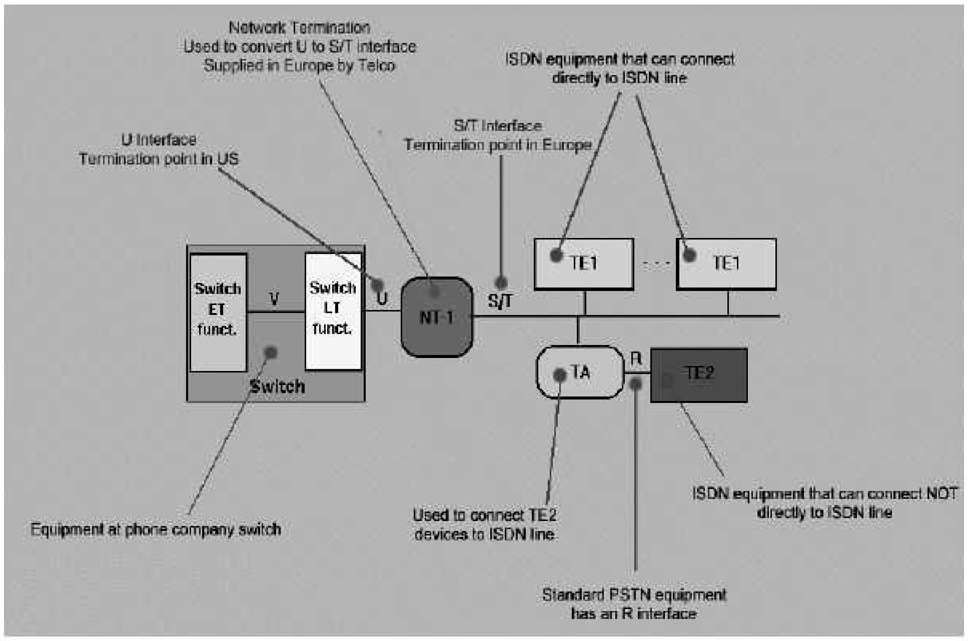

Network Termination
Used to convert U to S/T interface
Supplied in Europe by Telco
ISDN equipment that can connect directly to ISDN line
S/T Interface
Termination point in Europe
U Interface
Termination point in US
TE1
TE1
Switch ET funct.
V
Switch LT funct.
U
NT-1
S/T
TA
R
TE2
Switch
Equipment at phone company switch
Used to connect TE2 devices to ISDN line
ISDN equipment that can connect NOT directly to ISDN line
Standard PSTN equipment has an R interface

Figure 9.1 : ISDN Interface

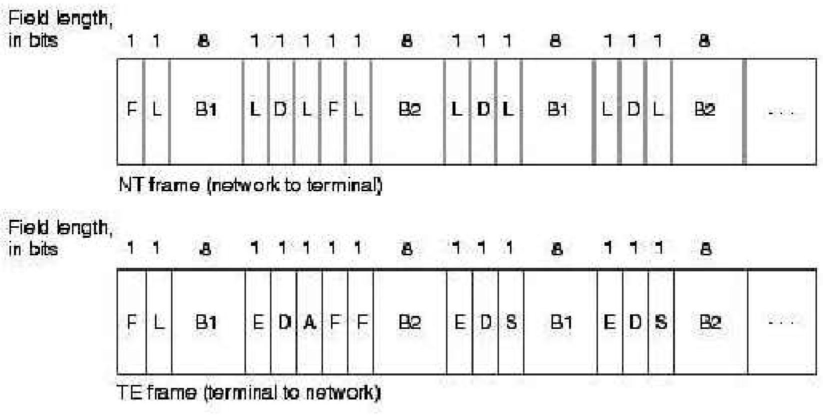
Layer 1 - Physical Layer

The ISDN Physical Layer is specified by the ITU I-series and G-series documents. The U interface provided by the telco for BRI is a 2-wire, 160 kb/s digital connection. Echo cancellation is used to reduce noise, and data encoding schemes (2B1Q in North America, 4B3T in Europe) permit this relatively high data rate over ordinary single-pair local loops.

ISDN physical layer (Layer 1) frame formats differ depending on whether the frame is outbound (from terminal to network) or inbound (from network to terminal). Both physical layer interfaces are shown in Figure 1.2: ISDN Physical Layer Frame Formats Differ Depending on Their Direction.

The frames are 48 bits long, of which 36 bits represent data. The bits of an ISDN physical layer frame are used as follows:

- F - Provides synchronization
- L - Adjusts the average bit value
- E - Ensures contention resolution when several terminals on a passive bus contend for a channel
- A - Activates devices
- S - Is unassigned
- B1, B2, and D - Handle user data



Field length, in bits
1 1 8 1 1 1 1 1 8 1 1 1 8 1 1 1 8
F L B1 L D L F L B2 L D L B1 L D L B2 ...
NT frame (network to terminal)
Field length, in bits
1 1 8 1 1 1 1 1 8 1 1 1 8 1 1 1 8
F L B1 E D A F F B2 E D S B1 E D S B2 ...
TE frame (terminal to network)

A = Activation bit
B1 = B1 channel bits
B2 = B2 channel bits
D = D channel (4 bits x 4000 frames/sec. = 16 kbps)
E = Echo of previous D bit
F = Framing bit
L = Load balancing
S = Spare bit

Figure 9.2 : ISDN Physical Layer Frame Formats Differ Depending on Their Direction

Multiple ISDN user devices can be physically attached to one circuit. In this configuration, collisions can result if two terminals transmit simultaneously. Therefore, ISDN provides features to determine link contention. When an NT receives a D bit from the TE, it echoes back the bit in the next E-bit position. The TE expects the next E bit to be the same as its last transmitted D bit.

Terminals cannot transmit into the D channel unless they first detect a specific number of ones (indicating "no signal") corresponding to a pre-established priority. If the TE detects a bit in the echo (E) channel that is different from its D bits, it must stop transmitting immediately. This simple technique ensures that only one terminal can transmit its D message at one time. After successful D-message transmission, the terminal has its priority reduced by requiring it to detect more continuous ones before transmitting. Terminals cannot raise their priority until all other devices on the same line have had an opportunity to send a D message. Telephone connections have higher priority than all other services, and signaling information has a higher priority than non-signaling information.

2B1Q

2B1Q (2 Binary 1 Quaternary) is the most common signaling method on U interfaces. This protocol is defined in detail in 1988 ANSI spec T1.601. In summary, 2B1Q provides:

- Two bits per baud
- 80 kilobaud (baud = 1 modulation per second)
- Transfer rate of 160 kb/s

Bits	Quaternary Symbol	Voltage Level
00	-3	-2.5
01	-1	-0.833
10	+3	+2.5
11	+1	+0.833

Table-9.1 : Quaternaries Symbol

This means that the input voltage level can be one of 4 distinct levels (note: 0 volts is not a valid voltage under this scheme). These levels are called **Quaternaries**. Each quaternary represents 2 data bits, since there are 4 possible ways to represent 2 bits, as in the table above.

Frame Format

Each U interface frame is 240 bits long. At the prescribed data rate of 160 kb/s, each frame is therefore 1.5 ms long. Each frame consists of:

- Frame overhead - 16 kb/s
- D channel - 16 kb/s
- 2 B channels at 64 kb/s - 128 kb/s

Sync 18 bits	**12 * (B_1 + B_2 + D)** 216 bits	**Maintenance** 6 bits

Figure 9.3: Frame Format

- The Sync field consists of 9 Quaternaries (2 bits each) in the pattern +3 +3 -3 -3 -3 +3 -3 +3 -3.
- (B_1 + B_2 + D) is 18 bits of data consisting of 8 bits from the first B channel, 8 bits from the second B channel, and 2 bits of D channel data.
- The Maintenance field contains CRC information, block error detection flags, and "embedded operator commands" used for loopback testing without disrupting user data.

Data is transmitted in a **super frame** consisting of 8 240-bit frames for a total of 1920 bits (240 octets). The sync field of the first frame in the super frame is inverted (i.e. -3 -3 +3 +3 +3 -3 +3 -3 +3).

Layer 2 - Data Link Layer

Layer 2 of the ISDN signaling protocol is Link Access Procedure, D channel (LAPD). LAPD is similar to High-Level Data Link Control (HDLC) and Link Access Procedure, Balanced (LAPB). As the expansion of the LAPD acronym indicates, this layer is used across the D channel to ensure that control and signaling information flows and is received properly. The LAPD frame format (see Figure 1-4: LAPD Frame Format Is Similar to That of HDLC and LAPB) is very similar to that of HDLC;

like HDLC, LAPD uses supervisory, information, and unnumbered frames. The LAPD protocol is formally specified in ITU-T Q.920 and ITU-T Q.921.

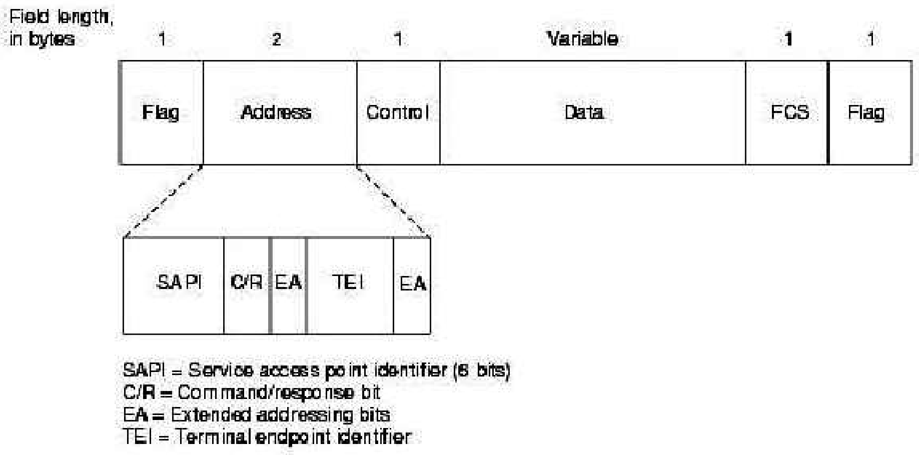

Field length, in bytes
1
2
1
Variable
1
1
Flag
Address
Control
Data
FCS
Flag
SAPI
C/R
EA
TEI
EA
SAPI = Service access point identifier (6 bits)
C/R = Command/response bit
EA = Extended addressing bits
TEI = Terminal endpoint identifier

Figure 9.4: LAPD Frame Format Is Similar to That of HDLC and LAPB

The LAPD Flag and Control fields are identical to those of HDLC. The LAPD Address field can be either 1 or 2 bytes long. If the extended address bit of the first byte is set, the address is 1 byte; if it is not set, the address is 2 bytes. The first Address-field byte contains the service access point identifier (SAPI), which identifies the portal at which LAPD services are provided to Layer 3. The C/R bit indicates whether the frame contains a command or a response. The Terminal Endpoint Identifier (TEI) field identifies either a single terminal or multiple terminals. A TEI of all ones indicates a broadcast.

Address (2 octets)							
1	2	3	4	5	6	7	8
SAPI (6 bits)						C/R	EA0
TEI (7 bits)							EA1

Table 9.2 Address

SAPI (Service access point identifier), 6-bits (see below)
C/R (Command/Response) bit indicates if the frame is a command or a response

EA0 (Address Extension) bit indicates whether this is the final octet of the address or not

TEI (Terminal Endpoint Identifier) 7-bit device identifier.

EA1 (Address Extension) bit, same as EA0

Control (2 octets) - The frame level control field indicates the frame type (Information, Supervisory, or Unnumbered) and sequence numbers (N(r) and N(s)) as required.

Information - Layer 3 protocol information and User data
CRC (2 octets) - Cyclic Redundancy Check is a low-level test for bit errors on the user data.

Flag (1 octet) - This is always $7E_{16}$ ($0111\ 1110_2$)

SAPIs

The Service Access Point Identifier (SAPI) is a 6-bit field that identifies the point where Layer 2 provides a service to Layer 3.

See the following table:

SAPI	**Description**
0	Call control procedures
1	Packet Mode using Q.931 call procedures
16	Packet Mode communications procedures
32-47	Reserved for national use
63	Management Procedures
Others	Reserved for Future Use

Table 9.3 SAPI

TEIs

Terminal Endpoint Identifiers (TEIs) are unique IDs given to each device (TE) on an ISDN S/T bus. This identifier can be dynamic; the value may be assigned statically when the TE is installed, or dynamically when activated.

TEI	**Description**
0-63	Fixed TEI assignments
64-126	Dynamic TEI assignment (assigned by the switch)
127	Broadcast to all devices

Table- 9.4: TEI

Establishing the Link Layer

The Layer 2 establishment process is very similar to the X.25 LAP-B setup, if you are familiar with it.

1. The TE (Terminal Endpoint) and the Network initially exchange Receive Ready (RR) frames, listening for someone to initiate a connection
2. The TE sends an Unnumbered Information (UI) frame with a SAPI of 63 (management procedure, query network) and TEI of 127 (broadcast)
3. The Network assigns an available TEI (in the range 64-126)
4. The TE sends a Set Asynchronous Balanced Mode (SABME) frame with a SAPI of 0 (call control, used to initiate a SETUP) and a TEI of the value assigned by the network
5. The network responds with an Unnumbered Acknowledgement (UA), SAPI=0, TEI=assigned.

At this point, the connection is ready for a Layer 3 setup.

Layer 3 - Network Layer

The ISDN Network Layer is also specified by the ITU Q-series documents Q.930 through Q.939. Layer 3 is used for the establishment, maintenance, and termination of logical network connections between two devices.

SPIDs

Service Profile IDs (SPIDs) are used to identify what services and features the Telco switch provides to the attached ISDN device. SPIDs are optional; when they are used, they are only accessed at device initialization time, before the call is set up. The format of the SPID is defined in a recommendation document, but it is only rarely followed. It is usually the 10-digit phone number of the ISDN line, plus a prefix and a suffix that are sometimes used to identify features on the line, but in reality it can be whatever the telco decides it should be. If an ISDN line requires a SPID, but it is not correctly supplied, then Layer 2 initialization will take place, but Layer 3 will not, and the device will not be able to place or accept calls.

Information Field Structure

The Information Field is a variable length field that contains the Q.931 protocol data.

Information Field							
1	2	3	4	5	6	7	8
Protocol Discriminator							
0	0	0	0	Length of CRV			
Call Reference Value (1 or 2 octets)							
0	Message Type						
Mandatory & Optional Information Elements (variable)							

Table 9.5: Information Field

These are the fields in a Q.931 header:
Protocol Discriminator (1 octet) - identifies the Layer 3 protocol. If this is a Q.931 header, this value is always 08_{16}.
Length (1 octet) - indicates the length of the next field, the CRV.
Call Reference Value (CRV) (1 or 2 octets) - used to uniquely identify each call on the user-network interface. This value is assigned at the beginning of a call, and this value becomes available for another call when the call is cleared.
Message Type (1 octet) - identifies the message type (i.e., SETUP, CONNECT, etc.). This determines what additional information is required and allowed.
Mandatory and Optional Information Elements (variable length) - are options that are set depending on the Message Type.

Layer 3 Call Setup

These are the steps that occurs when an ISDN call is established. In the following example, there are three points where messages are sent and received; 1) the Caller, 2) the ISDN Switch, and 3) the Receiver.

1. Caller sends a SETUP to the Switch.
2. If the SETUP is OK, the switch sends a CALL PROCeeding to the Caller, and then a SETUP to the Receiver.
3. The Receiver gets the SETUP. If it is OK, then it rings the phone and sends an ALERTING message to the Switch.
4. The Switch forwards the ALERTING message to the Caller.
5. When the receiver answers the call, is sends a CONNECT message to the Switch
6. The Switch forwards the CONNECT message to the Caller.
7. The Caller sends a CONNECT ACKnowledge message to the Switch
8. The Switch forwards the CONNECT ACK message to the Receiver.
9. Done. The connection is now up.

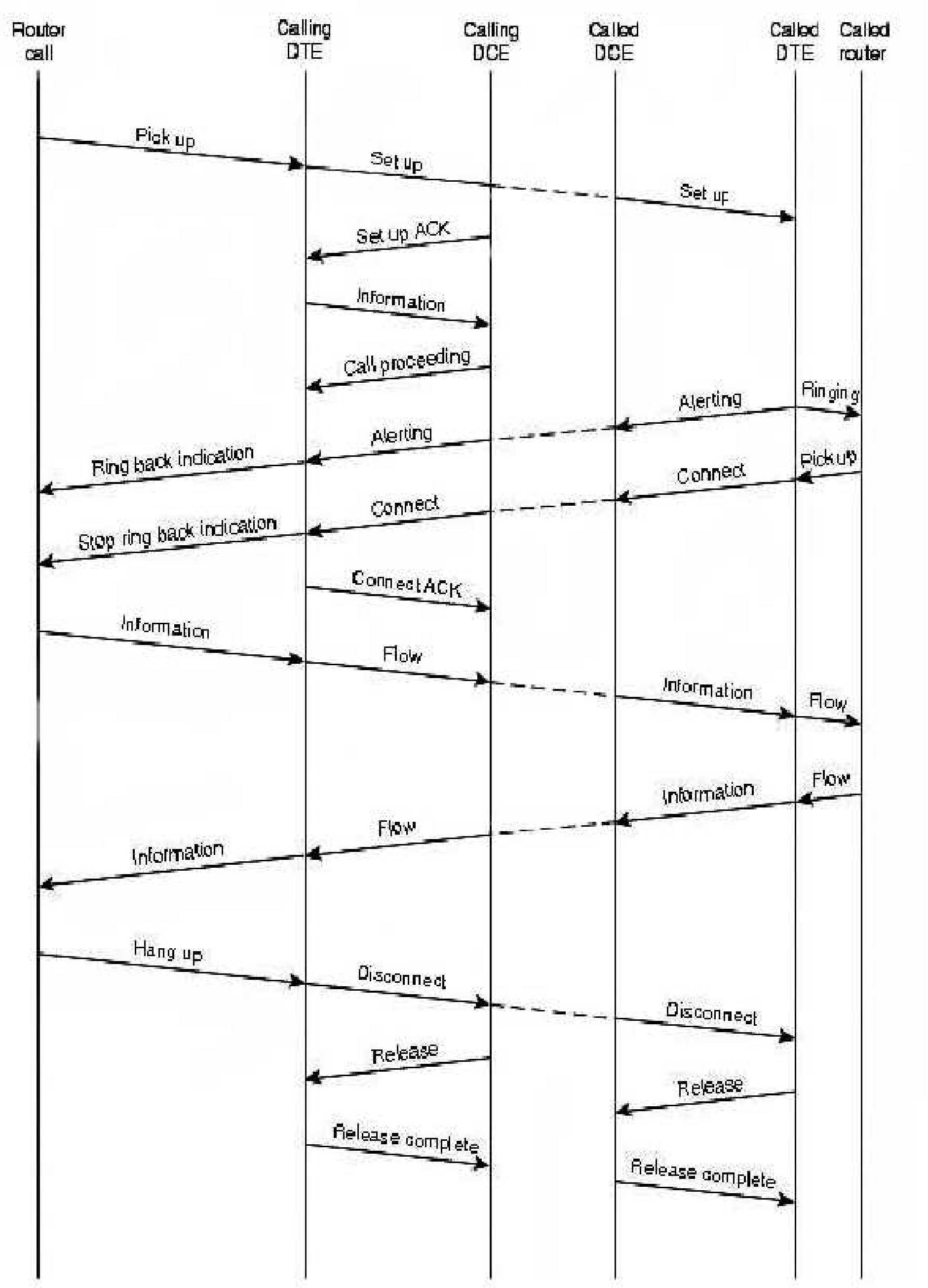


Figure 9-5: An ISDN Circuit-Switched Call Moves Through Various Stages to Its Destination

Check your progress

Q1. What is ISDN? Explain.

Q2. Define the working of ISDN in detail.

9.3 Devices

The phrase ISDN standard refers to the devices that are required to connect the end node to the network. Although some vendors provide devices that include several functions, a separate device defines each function within the standard. The protocols that each device uses are also defined and are associated with a specific letter. Also known as reference points, these letters are R, S, T, and U. ISDN standards also define the device types. They are NT1, NT2, TE1, TE2, and TA.

ISDN Reference Points

Reference points are used to define logical interfaces. They are, in effect, a type of protocol used in communications. The following list contains the reference points:

- **R** Defines the reference point between a TE2 device and a TA device.
- **S** Defines the reference point between TE1 devices and NT1 or NT2 devices.
- **T** Defines the reference point between NT1 and NT2 devices.
- **U** Defines the reference point between NT1 devices and line termination equipment. This is usually the central switch.

Network terminator 1 (NT1) is the device that communicates directly with the central office switch. The NT1 receives a U-interface connection from the telephone company and puts out a T-interface connection for the NT2. NT1 handles the physical layer portions of the connection, such as physical and electrical termination, line monitoring, and multiplexing.

Network terminator 2 (NT2) is placed between an NT1 device and any adapters or terminal equipment. Many devices provide the NT1 and NT2 devices in the same physical hardware. Larger installations generally separate these devices. An example of an NT2 device is a digital private branch exchange (PBX) or ISDN router. An NT2 device provides an S interface and accepts a T interface from an NT1. NT2 usually handles data link and network layer functions, such as contention monitoring and routing, in networks with multiple devices.

Terminal equipment 1 (TE1) is a local device that speaks via an S interface. It can be directly connected to the NT1 or NT2 devices. An ISDN telephone and an ISDN fax are good examples of TE1 devices.

Terminal equipment 2 (TE2) devices are common, everyday devices that can be used for ISDN connectivity. Any telecommunications device that is not in the TE1 category is classified as a TE2 device. A terminal adapter is used to connect these devices to an ISDN and attaches through an R interface. Examples of TE2 devices include standard fax machines, PCs, and regular telephones.

A terminal adapter (TA) connects TE2 devices to an ISDN. A TA connects through the R interface to the TE2 device and through the S interface to the ISDN. The peripheral required for personal computers often includes an NT1 device, better known as ISDN modems.

ISDN modems are used in PCs to connect them to an ISDN network. The term modem is used incorrectly here. ISDN passes data in a digital format, so there is no need to convert the digital data from the computer the way a traditional modem converts the digital data to analog so that it can travel. Conventional modems convert analog to digital and vice versa.

Identifiers

Standard telephone lines use a ten-digit identifier, better known as a telephone number, which is permanently assigned. ISDN uses similar types of identifiers, but they are not as easily used as a telephone number. ISDN uses five separate identifiers to make a connection. When the connection is first set up, the provider assigns two of these: the service profile identifier (SPID) and the directory number (DN). These are the most common numbers used because the other three are dynamically set up each time a connection is made. The three dynamic identifiers are the terminal endpoint identifier (TEI), the service address point identifier (SAPI), and the bearer code (BC).

The directory number (DN) is the ten-digit phone number the telephone company assigns to any analog line. ISDN services enable a greater degree of flexibility in using this number than analog services do. Unlike an analog line, where a one-to-one relationship exists, the DN is only a logical mapping. A single DN can be used for multiple channels or devices. In addition, up to eight DNs can be assigned to one device. Because a single BRI can have up to eight devices, it can support up to 64 directory numbers. This is how offices are able to have multiple numbers assigned to them. Most standard BRI installations include only two directory numbers, one for each B channel.

The service profile identifier (SPID) is the most important number needed when you are using ISDN. The provider statically assigns the SPID when the ISDN service is set up. It usually includes the DN plus a few extra digits. The SPID usually contains between 10 and 14 characters and varies from region to region. SPIDs can be assigned for every ISDN device, for the entire line or for each B channel.

The SPID is unique throughout the entire switch and must be set up correctly. If it is incorrect, the result is like dialing the wrong phone number-you will not be able to contact the person you are trying to reach. When an ISDN device is connected to the network, it sends the SPID to the switch. If the SPID is correct, the switch uses the stored information about your service profile to set up the data link. The ISDN device will not send the SPID again unless the device is disconnected from the network.

A terminal endpoint identifier (TEI) identifies the particular ISDN device to the switch. This identifier changes each time a device is connected to the ISDN. Unlike the SPID or the DN, the TEI is dynamically allocated by the central switch.

The service address point identifier (SAPI) identifies the particular interface on the switch to which your devices are connected. This identifier is used by the switch and is also dynamically updated each time a device connects to the network.

The bearer code (BC) is an identifier made up of the combination of TEI and SAPI. It is used as the call reference and is dynamic, like the two identifiers included within it. It changes each time a connection is established.

9.4 Channel Structure

ISDN provides two basic types of interfaces to users.

1. Basic Rate Interface (BRI)
2. Primary Rate Interface (PRI)

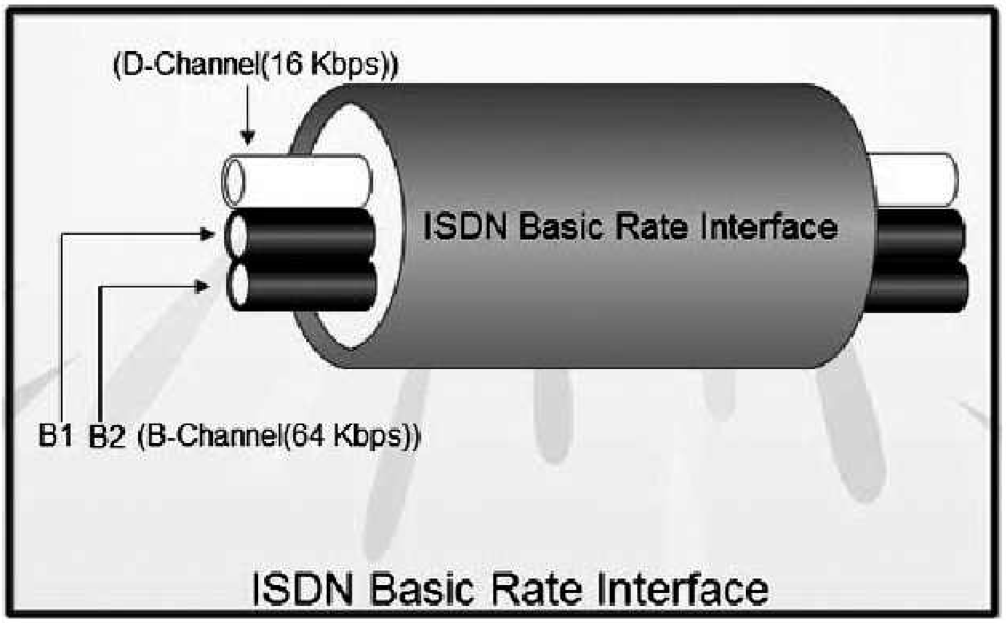

(D-Channel(16 Kbps))
ISDN Basic Rate Interface
B1 B2 (B-Channel(64 Kbps))
ISDN Basic Rate Interface

Figure 9.6: ISDN BRI

Basic Rate Interface (BRI)

Basic Rate Interface specifies a digital pipe, consisting of two 64 Kbps B channels and one 16 Kbps D channel for a total of 144 Kbps (2B+D). Figure shows the structure of Basic Rate Interface. In addition, the BRI service itself requires 48 Kbps operating overhead. BRI therefore requires a digital pipe of 192 Kbps. Conceptually, the BRI service is like a large pipe that contains three smaller pipes, two for the B channels and one for the D channel. The remainder of the space inside the large pipe carries the overhead bits required for its operation. All 192 Kbps can be used to carry a single signal.

The BRI is designed to cater to the needs of home users and small business establishments. In most cases, there is no need to replace the existing local-loop cable. The existing twisted pair local loop can be used to carry digital transmission.

Primary Rate Interface (PRI)

Primary Rate Interface is intended for users with higher data rate requirements, such as large business establishments, offices with a digital PBX or a LAN. Because of differences in the digital-transmission hierarchies used in different countries, it was not possible to reach an agreement on a single data rate.

The United States, Canada and Japan make use of a transmission structure based on 1.544 Mbps; this corresponds to the T-1 transmission facility of AT& T. In Europe, 2.048 Mbps is the standard rate. Both of these data rates are provided as a Primary Rate Interface.

Typically, the channel structure for the 1.544 Mbps rate will be 23 B channels and one 64 Kbps D channel and, for the 2.048 Mbps rate, 30 B channels and one 64 Kbps D channel. (Figure shows the structure of Primary Rate Interface).

It is also possible for a customer with few requirements to employ fewer B channels, in which case, the channel structure is $nB + 0$,where n ranges from 1 to 23 or from 1 to 30 for the two primary services. Also, a customer with high data-rate demands may be provided with more than one primary physical interface. In this case, a single D channel on one of the interfaces may suffice for all signaling needs, and the other interfaces may consist solely of B channels (24B or 31B).

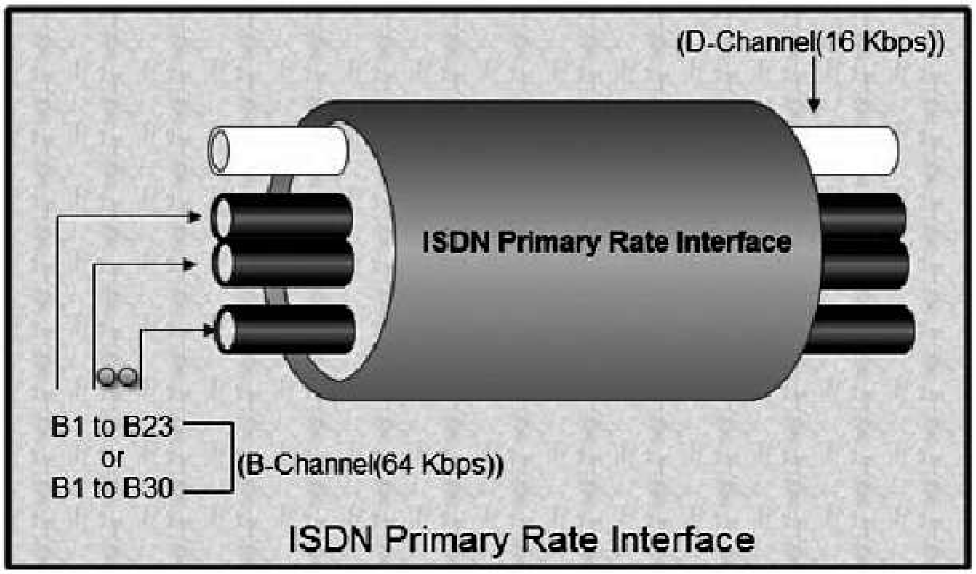


Figure 9.7 ISDN PRI

PRI with H-channels The Primary Rate Interface may also be used to support H channels. Some of these structures include a 64 Kbps D channel for control signaling. When no D channel is present, it is assumed that a D channel on another primary interface at the same subscriber location will provide any required signaling. The following structures are recognized:

Primary Rate Interface (PRI) H0 channel structures This interface supports multiple 384 Kbps HO channels. The structures are 3HO+D and 4 HO for the 1.544 Mbps interface and 5HO +D for the 2.048 Mbps interface.

Primary Rate Interface (PRI) H1 and H12 channel structures The H1 channel structure consists of one 1,536 Kbps H11 channel. The H12 channel structure consists of one 1,920 Kbps H12 channel and one D channel.

Primary Rate Interface (PRI) structures for mixtures of B and H0channels This interface consists of zero or one D channel and any possible combination of Band H0 channels up to the capacity of the physical interface (e.g., 3HO+ 5B + D and 3HO+ 6B for the 1.544 MBps interface).

Check your progress

Q1. Explain the devices of ISDN.

Q2. What is the channel structure of ISDN?

9.5 Dead Locks and their avoidance

Dead Lock

A deadlock is a situation in which two computer programs sharing the same resource are effectively preventing each other from accessing the resource, resulting in both programs ceasing to function.

The earliest computer operating systems ran only one program at a time. All of the resources of the system were available to this one program. Later, operating systems ran multiple programs at once, interleaving them. Programs were required to specify in advance what resources they needed so that they could avoid conflicts with other programs running at the same time. Eventually some operating systems offered dynamic allocation of resources. Programs could request further allocations of resources after they had begun running. This led to the problem of the deadlock. Here is the simplest example:

```
Program 1 requests resource A and receives it. Program
2 requests resource B and receives it. Program 1
requests resource B and is queued up, pending the
release of B. Program 2 requests resource A and is
queued up, pending the release of A.
```

Now neither program can proceed until the other program releases a resource. The operating system cannot know what action to take. At this point the only alternative is to abort (stop) one of the programs.

Learning to deal with deadlocks had a major impact on the development of operating systems and the structure of databases. Data was structured and the order of requests was constrained in order to avoid creating deadlocks.

Deadlock characteristics

- Permanent blocking of a set of processes that either compete for system resources or communicate with each other.
- No efficient solutions
- Involve conflicting needs for resources by two or more processes.

Here is an example of a deadlock situation:

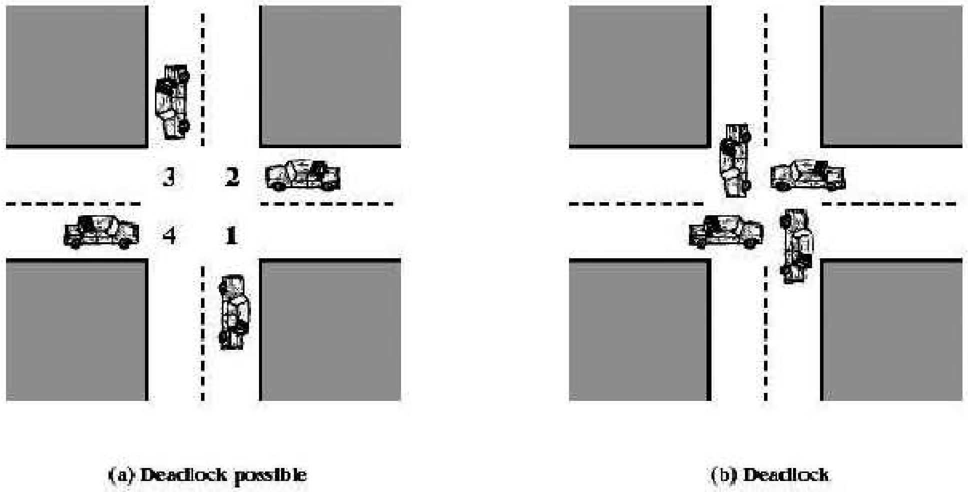


Figure 9.8: Deadlock example

Types of resources

Reusable: Used by only one process at a time and is not depleted by that use.
Consumable: Can be created (produced) and destroyed (consumed).

Conditions for deadlock

1. **Mutual exclusion:** Only one process may use a resource at a time.
2. **Hold and wait:** A process may hold allocated resources while awaiting assignment of others.
3. **No preemption:** No resources can be forcibly removed from a process holding it.
4. **Circular wait:** A closed chain of processes exists, such that each process holds at least one resource needed by the next process in the chain.

Conditions 1 through 3 are necessary but not sufficient the unresolvable circular wait is in fact a definition of deadlock.

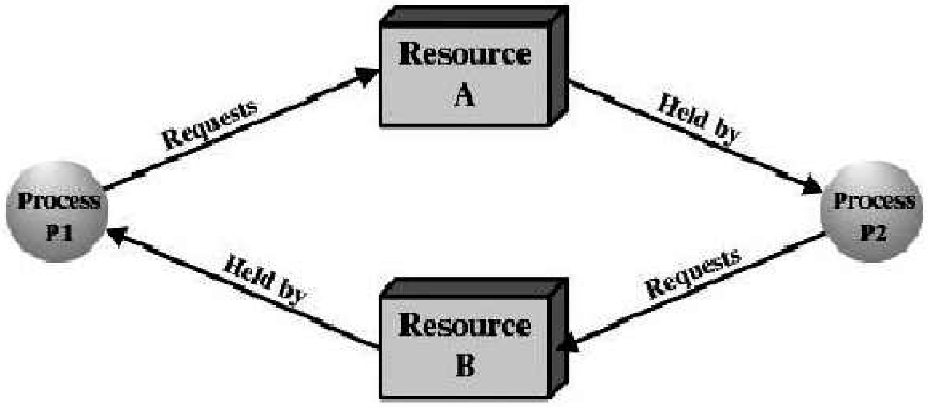


Figure 9.9: Circular wait

Deadlock avoidance

Basic idea

- A decision is made dynamically whether the current resource allocation request will, if granted, potentially lead to a deadlock
- Requires knowledge of future process requests

Approaches to deadlock avoidance:

- **Process initial denial:** Do not start a process if its demands might lead to a deadlock.

A process is started if the maximum needs of all current processes plus the needs of the process can be met.

Not efficient strategy - assumes that all processes will make maximum claims together.

- **Resource allocation denial:** Do not grant an incremental resource request to a process if this allocation might lead to a deadlock.

Banker's algorithm – the strategy of resource allocation denial.

Data needed for the algorithm:

- Claim matrix
- Allocation matrix
- Resource vector
- Available vector

The algorithm determines whether a given state of the system is safe or unsafe. The state of the system is represented by the current allocation of resources to processes. Thus, the state consists of the two vectors, Resource and Available, and the two matrices, Claim and Allocation.

The **state** of the system is represented by the current allocation of resources to processes. Thus, the state consists of the two vectors, Resource and Available, and the two matrices, Claim and Allocation.

A **safe state** is one in which there is at least one sequence of processes that can finish if granted the resources they need. i.e. at least one sequence that does not result in a deadlock.

An **unsafe state** is a state that is not safe.

The algorithms operates as follows:

For all processes do:

Is there a process that can finish with the available resources?

a. YES: assume that the process has finished and add its resources to the available resources. Continue the check for the remaining processes.

b. NO: there is a deadlock situation, the state of the system is unsafe. Stop the check.

If there is at least one sequence of processes that can finish, the state is safe.

In order to determine whether granting a resource would lead to a unsafe state, the data tables are updated as if the resource were granted and then the algorithm is applied for the obtained state of the system. If the algorithm determines that this is a safe state, the resource is actually granted.

Restrictions on use of deadlock avoidance

- The maximum resource requirement for each process must be stated in advance.
- The processes under consideration must be independent; that is, the order in which they execute must be unconstrained by any synchronization requirements.
- There must be a fixed number of resources to allocate.
- No process may exit while holding resources.

9.6 Network layer in ATM

- The ATM layer handles the functions of the network layer:
  - Moving cells from source to destination in order.

    - Routing algorithms within ATM switches, global addressing.
- Connection-oriented without acknowledgments.
- The basic element is the unidirectional virtual circuit or channel with fixed-size cells.
- Two possible interfaces:
    - UNI (User-Network Interface): Boundary between an ATM network and host.
    - NNI (Network-Network Interface): Between two ATM switches (or routers).

9.7 Summary

In this unit you have learnt about Integrated Services Digital Network, Interfaces of ISDN, Devices used in ISDN, Channel Structure of ISDN, Dead Locks and their avoidance and network layer in ATM.

- **Integrated Services Digital Network (ISDN)** is a set of communication standards for simultaneous digital transmission of voice, video, data, and other network services over the traditional circuits of the public switched telephone network.
- ISDN is a circuit-switched telephone network system, which also provides access to packet switched networks, designed to allow digital transmission of voice and data over ordinary telephone copper wires, resulting in potentially better voice quality than an analog phone can provide.
- *Integrated services* refers to ISDN's ability to deliver at minimum two simultaneous connections, in any combination of data, voice, video, and fax, over a single line. Multiple devices can be attached to the line, and used as needed.
- A **deadlock** occurs when two competing actions wait for the other to finish, and thus neither ever does. Deadlock is a common problem in multiprocessing systems, parallel computing, and distributed systems, where software and hardware locks are used to handle shared resources and implement process synchronization.
- In a communications system, deadlocks occur mainly due to lost or corrupt signals rather than resource contention.

9.8 Review Questions

Q1. Why do we need ISDN? Elaborate your answer.

Q2. Explain the interface of ISDN.

Q3. What types of devices do we need in ISDN?

Q4. Explain the channel structure of ISDN.

Q5. What is a deadlock? How we avoid it?

UNIT-X
Internetworking

Structure

10.0 Introduction

This is the second unit of this block. This unit covers internetworking in detail. There are seven sections in this unit. As we know a computer network is a set of different computers connected together using networking devices such as switches and hubs. To enable communication, each individual network node or segment is configured with similar protocol or communication logic, which usually is TCP/IP. When a network communicates with another network having the same or compatible communication procedures, it is known as Internetworking. Internetworking is also implemented using internetworking devices such as routers. These are physical hardware devices which have the ability to connect different networks and ensure error free data communication. They are the core devices enabling internetworking and are the backbone behind the Internet. Sec. 10.1 define the concept of Bridges. In the next section you will learn about routers. In Sec. 10.3 we define gateways. Next section i.e. Sec. 10.4 Internet Architecture has been described. In section 10.5 you will know about Addressing. In Sec. 10.6 and 10.7 you will find summary and review questions respectively.

Objectives

After studying this unit you should be able to:

- ❖ Define Bridges, Routers and Gateways
- ❖ Describe Internet Architecture and Addressing.

10.1 Bridges

A bridge is a computer networking device that builds the connection with the other bridge networks which use the same protocol. It works at the Data Link layer of the OSI Model and connects the different networks together and develops communication between them. It connects two local-area networks; two physical LANs into larger logical LAN or two *segments* of the same LAN that use the same protocol.

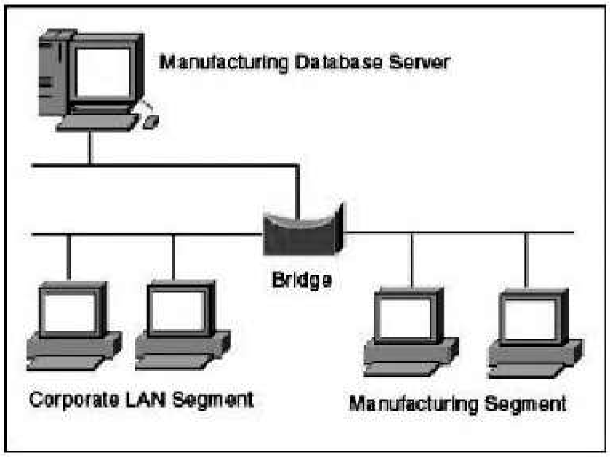


Figure 10.1: Bridge

Apart from building up larger networks, bridges are also used to segment larger networks into *smaller* portions. The bridge does so by placing itself between the two portions of two physical networks and controlling the flow of the data between them. Bridges nominate to forward the data after inspecting into the MAC address of the devices connected to every segment. The forwarding of the data is dependent on the acknowledgement of the fact that the destination address resides on some other interface. It has the capacity to block the incoming flow of data as well. Today **Learning bridges** have been introduced that build a list of the MAC addresses on the interface by observing the traffic on the network. This is a leap in the development field of manually recording of MAC addresses.

Network Bridge makes it inexpensive and easy to connect LAN segments. A LAN segment is a section of network media that connects computers. Frequently, a network has more than one LAN segment. Prior to Windows XP; Windows Server 2003, Standard Edition; and Windows Server 2003, Enterprise Edition, if you wanted to have a network with more than one LAN segment, then you had two options: IP routing and hardware bridging. IP routing required you to buy hardware routers or configure computers to act like routers, configure IP addressing for each computer on each network segment, and configure each network segment as a separate subnet. Hardware bridging did not require difficult configurations, but it did require you to purchase hardware bridges. Additionally, if you were using different types of network media, you needed to create a separate subnet for each type of media.

In contrast, the Network Bridge feature that is available with Windows XP; Windows Server 2003, Standard Edition; and Windows Server 2003, Enterprise Edition allows you to connect LAN segments simply by clicking the **Bridge Connections** menu command. No configuration is required, and you do not need to purchase additional hardware, such as routers or bridges. Network Bridge automates the configuration that is required to route traffic between multi-segment networks that consist of a single type of media or mixed media.

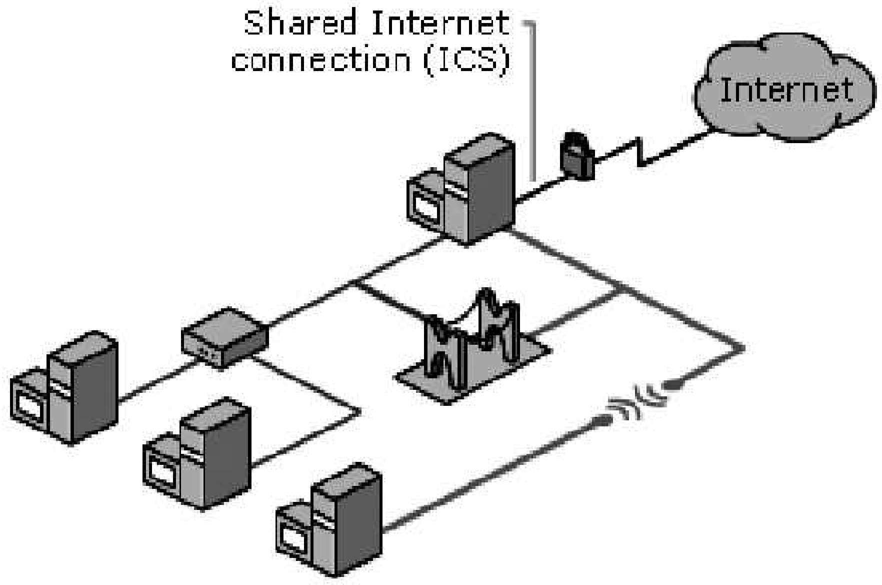


Figure 10.2: Network Bridge

Network Bridge example

Suppose you have a small office network with four computers (PC1, PC2, PC3, and PC4) and one Ethernet hub. The four computers are running Windows XP; Windows Server 2003, Standard Edition; or Windows Server 2003, Enterprise Edition and have the following hardware installed:

- PC1 has an adapter connecting it to the Internet, an Ethernet network adapter, an HPNA network adapter, and a wireless adapter.
- PC2 has an Ethernet network adapter.
- PC3 has an HPNA network adapter.
- PC4 has a wireless network adapter.

The Ethernet adapters on PC1 and PC2 are connected to a common Ethernet hub to form the first LAN segment. PC1 is connected to PC3 with the HPNA adapter to form a second LAN segment, and PC1 is connected to PC4 with the wireless adapter to form a third LAN segment.

You can use Network Bridge to connect the Ethernet network adapter, the HPNA network adapter, and the wireless network adapter on PC1.

Network Bridge can forward traffic from one LAN segment to another and enable all of your computers to communicate with each other.

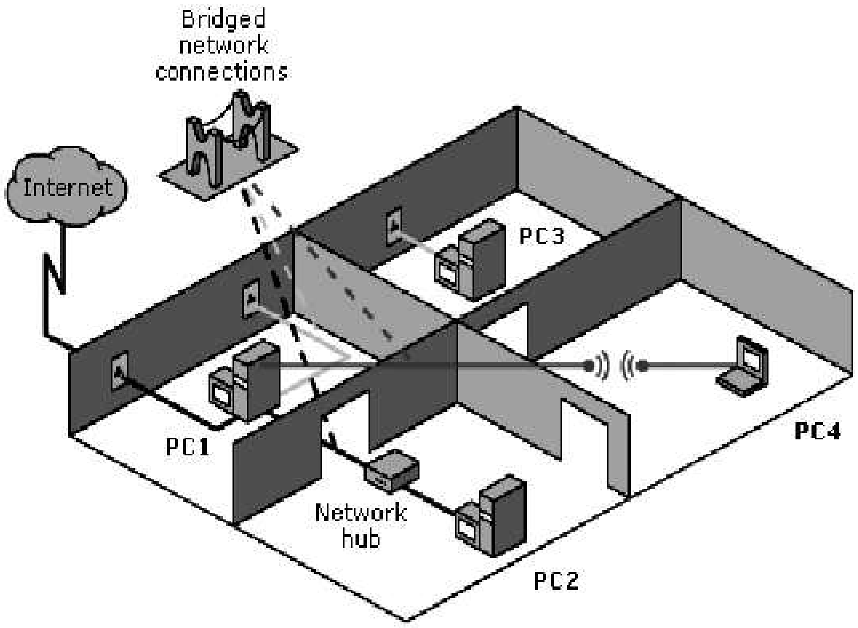


Figure 10.3- Network Bridge Example

Types of Bridges:

There are mainly three types in which bridges can be characterized:

- **Transparent Bridge:** As the name signifies, it appears to be transparent for the other devices on the network. The other devices are ignorant of its existence. It only blocks or forwards the data as per the MAC address.
- **Source Route Bridge:** It derives its name from the fact that the path which packet takes through the network is implanted within the packet. It is mainly used in Token ring networks.
- **Translational Bridge:** The process of conversion takes place via Translational Bridge. It converts the data format of one networking to another. For instance Token ring to Ethernet and vice versa.

Switches superseding Bridges:

Ethernet switches are seen to be gaining trend as compared to bridges. They are succeeding on the account of provision of logical divisions and segments in the networking field. In fact switches are being referred to as **multi-port bridges** because of their advanced functionality.

10.2 Routers

Routers are network layer devices and are particularly identified as Layer-3 devices of the OSI Model. They process *logical* addressing information in the Network header of a packet such as IP Addresses. Router is used to create larger complex networks by complex traffic routing. It has the ability to connect dissimilar LANs on the same protocol. It also has the ability to limit the flow of broadcasts. A router primarily comprises of a hardware device or a system of the computer which has more than one network interface and routing software.

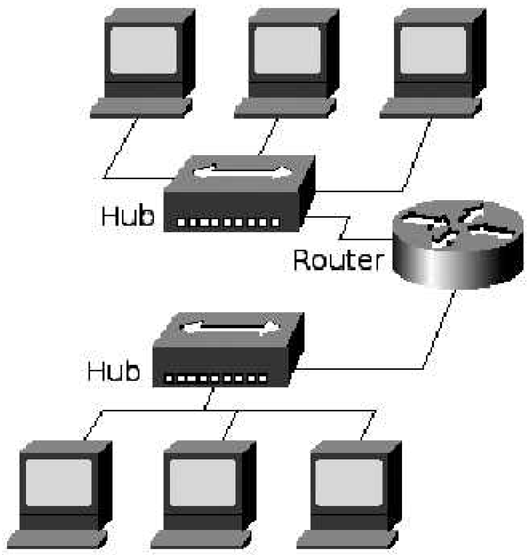


Figure 10.4: Router

Functionality:

When a router receives the data, it determines the destination address by reading the header of the packet. Once the address is determined, it searches in its **routing table** to get know how to reach the destination and then forwards the packet to the higher hop on the route. The hop could be the final destination or another router.

Routing tables play a very pivotal role in letting the router makes a decision. Thus a routing table is ought to be *updated* and *complete*. The two ways through which a router can receive information are:

- **Static Routing**: In static routing, the routing information is fed into the routing tables manually. It does not only become a time-taking task but gets prone to errors as well. The manual updating is also required in case of statically configured routers when change in the topology of the network or in the layout takes place. Thus static routing is feasible for tinniest environments with minimum of one or two routers.

- **Dynamic Routing**: For larger environment dynamic routing proves to be the practical solution. The process involves use of peculiar routing protocols to hold communication. The purpose of these protocols is to enable the other routers to transfer information about to other routers, so that the other routers can build their own routing tables.

Routing messages between networks

When a node on one network needs to send a message to a node on another network, this packet will be picked up by the router and passed on to the other network. Many nodes are programmed with a so-called 'default gateway', which is the address of the router that is to take care of all packets not for other nodes on the same network.

Routers maintain a so-called **routing table** to keep track of **routes**: which connections (to different networks) are to be used for which faraway networks. Some of these routes are programmed in manually, but many are "learned" automatically by the router. Modern routers inform each other about new routes and no longer working routes to make this as efficient as possible.

Figure 10.5 below illustrates how routers (and behind them, entire networks) may be connected. There are now multiple routes from the node at the left to the node at the right. Since routers transmit IP packets, and IP packets are all independent of one another, each packet can travel along a different route to its destination.

The TCP protocol that runs in the transport layer above does not notice this, although a user may notice if suddently the connection seems faster or slower. That could be caused by packets now following a different route that is faster or slower than the old one.

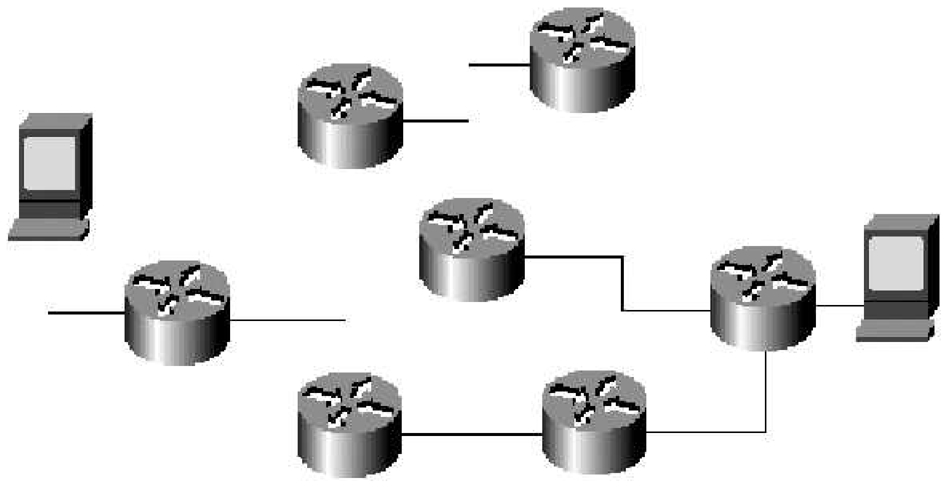


Figure 10.5 : how two nodes on different networks can communicate with each other

Security of routing

Routing data packets in this way is very efficient, but not very secure. Every router in between the source and the destination can examine every packet that comes through.

An additional risk is *traffic analysis*. A router can see where packets come from and where they go to. Even if the content of the packets is encrypted, the source or destination address itself already reveals something about the communication. For example, a corporate IP address that sends data to a newspaper website may indicate leaking of business secrets.

Onion routing with systems like Tor avoid even this risk, although they are much slower than traditional routing systems.

Check your progress

Q1. Explain bridges in detail.

Q2. Why a router needed for a network? Elaborate your answer.

10.3 Gateways

Gateway is a device which is used to connect multiple networks and passes packets from one network to the other network. Acting as the 'gateway' between different networking systems or computer programs, a gateway is a device which forms a link between them. It allows the computer programs, either on the same computer or on different computers to share information across the network through protocols. A router is also a gateway, since it interprets data from one network protocol to another.

The Gateway controls traffic that travels from the inside network to the Internet and provides security from traffic that wants to enter the inside network from the Internet. The Gateway is often the router that home owners or businesses use to connect to the Internet. Because some networks may have a number of connections to several other internal networks the term "Default Gateway" is used to signify the route for traffic which is the first choice.

The default gateway is the machine IP number that you need to access to get to the rest of the network or the Internet. Gateways connect two different kinds of networks, like your work network and the Internet. You will need to know the specific address of your gateway to connect to the Internet.

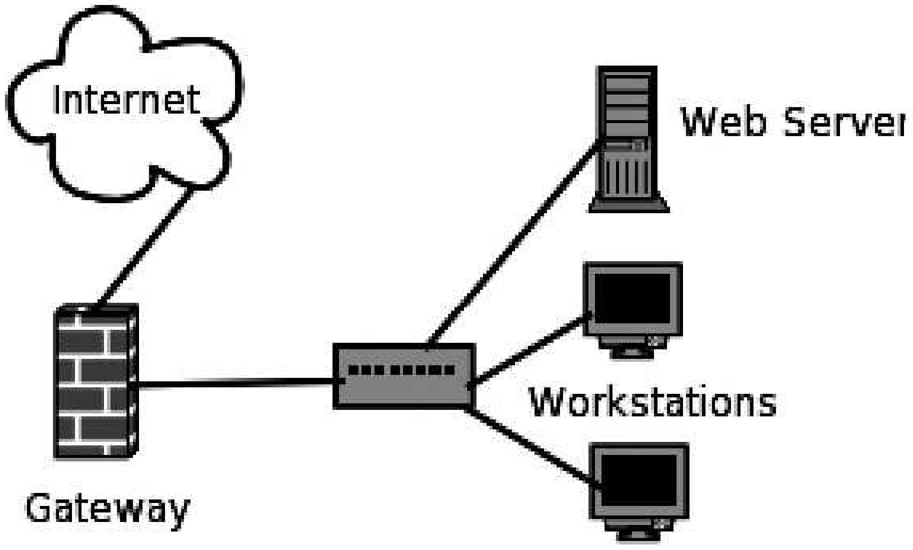


Figure 10.6: Gateway

In the figure 10.6 above the Gateway IP Address would be 192.168.5.222, using the example addresses from above. This gateway could be a router that would have two network connections, one to the internal network and one to the Internet. The IP Address of the workstations and web server would need to be addresses on the 192.168.5.0 network to connect to the gateway. Notice that the workstations and server all connect to a hub/switch before they connect to the gateway since the gateway will probably have only one connection.

When you add a computer to your home network that needs access to the Internet, one of the required settings is the Gateway. Your computer needs to know how to get to the Internet, thus the Gateway. You then need to supply the IP Address of your Gateway so that computer has access.

Examples of Gateway

- E-mail gateways-for example, a gateway that receives Simple Mail Transfer Protocol (SMTP) e-mail, translates it into a standard X.400 format, and forwards it to its destination
- Gateway Service for NetWare (GSNW), which enables a machine running Microsoft Windows NT Server or Windows Server to be a gateway for Windows clients so that they can access file and print resources on a NetWare server
- Gateways between a Systems Network Architecture (SNA) host and computers on a TCP/IP network, such as the one provided by Microsoft SNA Server

- A packet assembler/disassembler (PAD) that provides connectivity between a local area network (LAN) and an X.25 packet-switching network

10.4 Internet Architecture

Internet architecture is defined by a meta-network, a constantly changing collection of thousands of individual networks intercommunicating with a common protocol.

The Internet's architecture is described in its name, a short from of the compound word "inter-networking". This architecture is based in the very specification of the standard *TCP/IP* protocol, designed to connect any two networks which may be very different in internal hardware, software, and technical design. Once two networks are interconnected, communication with TCP/IP is enabled end-to-end, so that any node on the Internet has the near magical ability to communicate with any other no matter where they are. This openness of design has enabled the Internet architecture to grow to a global scale.

In practice, the Internet technical architecture looks a bit like a multi-dimensional river system, with small tributaries feeding medium-sized streams feeding large rivers. For example, an individual's access to the Internet is often from home over a modem to a local Internet service provider who connects to a regional network connected to a national network. At the office, a desktop computer might be connected to a local area network with a company connection to a corporate Intranet connected to several national Internet service providers. In general, small local Internet service providers connect to medium-sized regional networks which connect to large national networks, which then connect to very large bandwidth networks on the Internet *backbone*. Most Internet service providers have several redundant network cross-connections to other providers in order to ensure continuous availability.

The companies running the Internet backbone operate very high bandwidth networks relied on by governments, corporations, large organizations, and other Internet service providers. Their technical infrastructure often includes global connections through underwater cables and satellite links to enable communication between countries and continents. As always, a larger scale introduces new phenomena: the number of packets flowing through the switches on the backbone is so large that it exhibits the kind of complex non-linear patterns usually found in natural, analog systems like the flow of water or development of the rings of Saturn.

Each communication *packet* goes up the hierarchy of Internet networks as far as necessary to get to its destination network where local *routing* takes over to deliver it to the addressee. In the same way, each level in the hierarchy pays the next level for the bandwidth they use, and then the large backbone companies settle up with each other. Bandwidth is priced by large Internet service providers by several methods, such as at a fixed rate for constant availability of a certain number of megabits per second, or by a variety of use methods that amount to a cost per gigabyte. Due to economies of scale and efficiencies in management, bandwidth cost drops dramatically at the higher levels of the architecture.

The internet architecture can be broadly classified into three layers. The very first layer consists of Internet Backbones and very high speed network lines. The National Science Foundation (NSF) created the first high-speed backbone in 1987 called NSFNET, it was a T1 line that connected 170 smaller networks together and operated at 1.544 Mbps (million bits per second). IBM, MCI and Merit worked with NSF to create the backbone and developed a T3 (45Mbps) backbone the following year. Backbones are typically fiber optic trunk lines. The trunk line has multiple fiber optic cables combined together to increase the capacity. Fiber optic cables are designated OC-48 can transmit 2,488 Mbps (2.488 Gbps). The nodes are known as Network Access Point (NAPs). The second layer is usually known as Internet Service Provider (ISP). The ISPs are connected to the Backbones at NAP's with high speed lines.

The end users which are part of third layer are connected to ISPs by dial up or leased lines and modems. The speed of communication is usually 1400 bps to 2048 kbps.

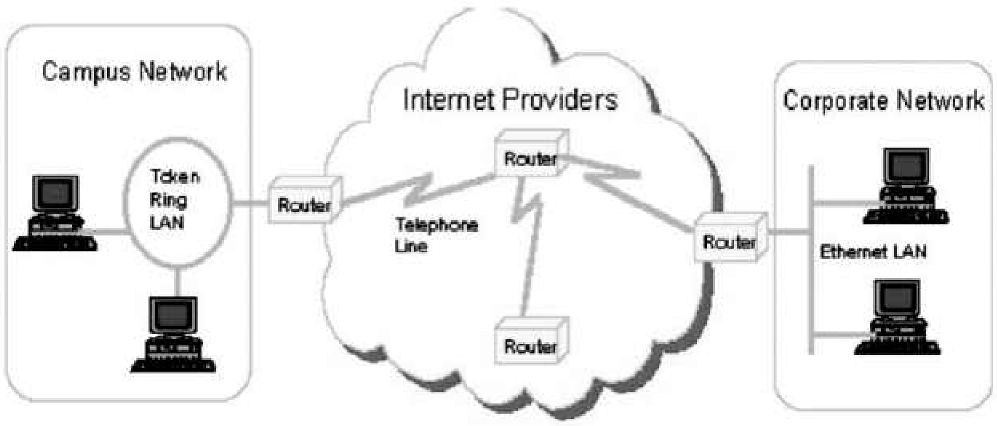


Figure 10.7: ISP

In the real Internet, dozens of large Internet providers interconnect at NAPs (Network Access Point) in various cities, and trillions of bytes of data flow between the individual networks at these points. The Internet is a collection of huge corporate networks interconnected with one other at the NAPs, backbones and routers to talk to each other. A message can leave one computer and travel halfway across the world through several different networks and arrive at another computer in a fraction of a second.

The routers determine where to send information from one computer to another. Routers are specialized computers that send messages to their destinations along thousands of pathways. It joins two networks, passing information from one to the other.

Check your progress

Q1. What is a gateway? Explain default gateway.

Q2. What do you mean by internet architecture?

10.5 Addressing

A **network address** serves as a unique identifier for a computer on a network. When set up correctly, computers can determine the addresses of other computers on the network and use these addresses to send messages to each other.

One of the best known form of network addressing is the Internet Protocol (IP) address. IP addresses consist of four bytes (32bits) that uniquely identify all computers on the public Internet.

Another popular form of address is the Media Access Control (MAC) address. MAC addresses are six bytes (48 bits) that manufacturers of network adapters burn into their products to uniquely identify them.

Internet vs. local area network

When a group of computers are connected together within a relatively small area, it is referred to as a local area network (LAN). If a LAN is available only to certain people (such as employees of a company), it is called a private or internal network. The Internet is a public network because it is accessible to many users and computers from different networks. The network shown in Figure 10.8 is a LAN that can be used to connect to the Internet.

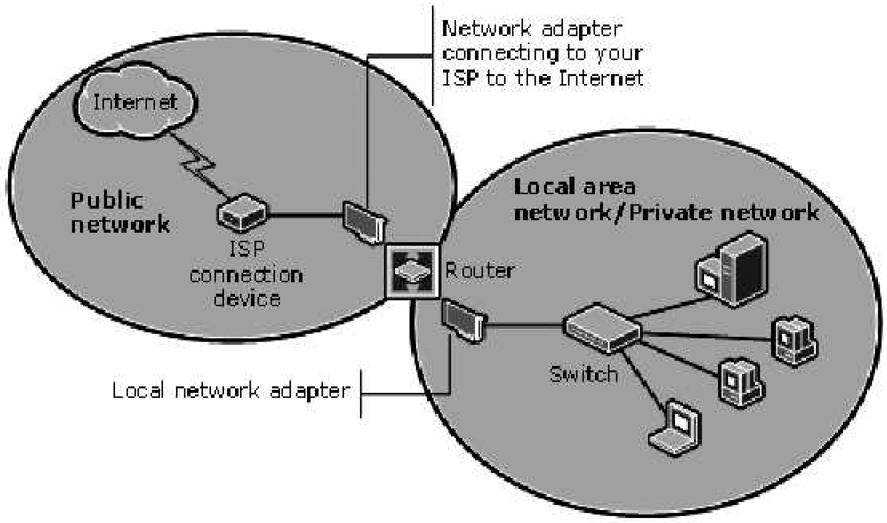

Network adapter connecting to your ISP to the Internet
Internet
Public network
ISP connection device
Router
Local area network/Private network
Switch
Local network adapter

Figure 10.8: LAN that can be used to connect to the Internet.

You can help to physically secure your small business LAN by configuring your hardware or software so that the router acts as a gateway to the Internet. A gateway is a combination of hardware and software that connects two different types of networks, for example a private network and a public network. There must be at least two network adapters installed on a gateway, one to connect to the Internet (ISP network adapter) and the other to connect to the private or local network (local network adapter), as shown in Figure 10.8.

Public vs. private addressing

An IP address is a unique numerical value that is used to identify a computer on a network. There are two kinds of IP addresses, public (also called globally unique IP addresses) and private.

- **Public IP addresses** are assigned by the Internet Assigned Numbers Authority (IANA). The addresses are guaranteed to be globally unique and reachable on the Internet. This assures that multiple computers do not have the same IP address.

  An Internet service provider (ISP) obtains a range of public IP addresses from IANA, and then the ISP assigns the addresses to customers to use when they connect to the Internet through the ISP.

  Public IP addresses are routable on the Internet, which means that a computer with a public IP address is visible to other computers on the Internet.

- **Private IP addresses** cannot be used on the Internet. IANA has set aside three blocks of IP addresses that cannot be used on the global Internet. These three blocks of addresses are private IP addresses, and they are used for networks that do not directly connect to the Internet.

Address Classes

IP (Internet Protocol) address is a unique 32 bit number assigned to each machine connected to the Internet (or an IP network), either permanently or temporarily (by dial up connection or similar). A typical IP address looks like -

207.153.234.217

IP addresses are usually denoted as 4 decimal numbers (also called **octets**), separated by dots. This format of representing an IP address is known as "dotted quad" or "dotted decimal" notation. (There's also base 10 notation, but it is rather uncommon). Each octet is represented in binary format by a single byte or 8 bits, thus 4 octets form a 32 bit IP address as shown above. (Note: This is valid for IP version 4 only). Theoretically, 32 bit IP addresses can accommodate almost 2^{32} = 4,294,967,296 (approximately 4.3 billion) machines on an IP network, such as Internet. In reality however, the number is much less than this number because many IP address ranges are reserved for special purpose and should not be used for another purpose.

An IP address consists of two parts, one identifying the network and one identifying the node or host. Each network can have "X" number of nodes or hosts under it. In the class system, the class of the address determines which part belongs to the network address, and which part belongs to the host address. In newer CIDR (Classless Inter-Domain Routing) system, this information is specified along with the IP address itself, thus allowing more efficient and practical allocation of IP addresses.

IP Classes

The class system has become "old" in the ever demanding Internet paradigm, it has been (partially) replaced by CIDR system, which allows more efficient and practical allocation of IP addresses for the need of smaller networks. Nevertheless, it is important to know and understand what class system is and how it works, which has been the backbone of the IP.

IP addresses have been classified into 5 classes and special purpose addresses, depending upon the value in the first octet. Viz-

- **Class A -** This is a class for very large networks, such as IBM which holds IP addresses in the range - 9.0.0.0 - 9.255.255.255. (Almost 16,777,216 IP addresses).
    - **First Octet** - - The first octet is between 1 to 126 (Starts with binary bit - 0).
    - **Network Address** - The n/w address is denoted by first 8 bits or first octet.
    - **Host/Node Address -** Host address is denoted by last 24 bits or last 3 octets.

This Network-Host IP configuration for class A can be shown as -

network.host.host.host

Each can have 1 to 3 decimals. Thus forming 126 network addresses (2^7 - 1) and each of them capable of having 16,777,214 (2^{24} -2) host addresses.

0	7 bits of the network address	24 bits of host address

- **Class B -** This is a class for medium-sized networks.
  - **First Octet - -** The first octet is between 128 to 191 (Starts with binary bits - 10).
  - **Network Address -** The n/w address is denoted by first 16 bits or first 2 octets.
  - **Host/Node Address -** Host address is denoted by last 16 bits or last 2 octets.

  This Network-Host IP configuration for class B can be shown as -

  network.network.host.host

  Each can have 1 to 3 decimals. Thus forming 16,384 network addresses (2^{14}) and each of them capable of having 65,534 (2^{16} -2) host addresses.

1	0	14 bits of the network address	16 bits of host address

- **Class C -** This is a class for small-sized networks.
  - **First Octet - -** The first octet is between 192 to 223. (Starts with binary bits - 110).
  - **Network Address -** The n/w address is denoted by first 24 bits or first 3 octets.
  - **Host/Node Address -** Host address is denoted by last 8 bits or last octet.

  This Network-Host IP configuration for class C can be shown as -

  network.network.network.host

Each can have 1 to 3 decimals. Thus forming 2,097,152 network addresses (2^{21}) and each of them capable of having 254 (2^8 -2) host addresses.

1	1	0	21 bits of the network address	8 bits of host address

- **Class D** - This is a class meant for multicasting only, for sending multicast messages to other groups of host machines.
  - **First Octet** - - The first octet is between 224 to 239. (Starts with binary bits - 1110).

  The class D is a special purpose reserved class, and addresses in this range are not assigned as IP addresses on an IP network, including Internet.

1	1	1	0	28 bits for some sort of group address

- **Class E** - This is a class meant for experimental purpose only.
  - **First Octet** - - The first octet is between 240 to 255. (Starts with binary bits - 1111).

  The class E is also a special purpose reserved class, and addresses in this range are not assigned as IP addresses on an IP network, including Internet. The IP address 255.255.255.255 in this range is also known as Broadcast.

1	1	1	1	28 bits

Note that all the figures given here (networks, possible hosts etc.) are only indicative, and not accurate because of various reserved IP addresses for numerous reasons. Possibly, for all practical purposes a valid Internet IP address will be from class A, B or C with exceptions of Loopback Addresses and Private Networks.

Network Type	Address Range	Normal Netmask	Comments
Class A	001.x.x.x to 126.x.x.x	255.0.0.0	For very large networks
Class B	128.1.x.x to 191.254.x.x	255.255.0.0	For medium size networks
Class C	192.0.1.x to 223.255.254.x	255.255.255.0	For small networks
Class D	224.x.x.x to 239.255.255.255		Used to support multicasting
Class E	240.x.x.x to 247.255.255.255		

Figure 10.9: Summary Network Type

Loopback

Addresses 127.0.0.0 to 127.255.255.255 are reserved for loopback, for internal testing on a local machine. 127.0.0.1 Typically refers to your own local machine, you can test this - you should always be able toping 127.0.0.1, irrespective of connectivity to the network, as it represents your own machine.
IP addresses in this range are *never* valid Internet addresses.

Private Networks

There are 3 IP network addresses reserved for private networks.

- 10.0.0.0 to 10.255.255.255
- 172.16.0.0 to 172.31.255.255
- 192.168.0.0 to 192.168.255.255

They may be used by anyone setting up internal IP networks, such as organization LAN behind a proxy server or a router. It is recommended to use them because routers on the Internet should (and will) never forward packets coming from these addresses. These addresses are meaningful only for the network to which they belong (Intranet). Due to this, two or more organizations may have same IP address falling in this range assigned to two or more individual machines without any conflict.

Base 10

An IP address can also be represented as a decimal (base 10) number formed by 32 bits. Although IP addresses are often written in the standard "dotted quad" notation, they can also be converted into "base 10" numbers to obfuscate URLs.

A simple URL "http://192.18.97.35/" can also be represented as "http://3222429987/" which will point to the same resource. To make it even more complicated, a garbage or meaningless data can also be added to the left of it which ends with an "@" (at the rate) sign, such as "http://SomeGarbageHere@3222429987/", which is also the same URL. Spammers and clever webmasters often use it to obfuscate the source. Consider following example, all these URLs listed below point to Sun's Java site.

http://java.sun.com/ - Normal DNS URL.
http://192.18.97.35/ - URL using "dotted quad" IP.
http://3222429987/ - Obfuscated base 10 URL, nevertheless takes you to the same site.
http://asgdahdashjk@3222429987/ - More complicated base 10 URL, does it make any sense?

Valid IP addresses converted to base 10 are in the range - 16777216 to 4294967295.

You can use IP Decoder tool given here to convert base 10 to standard IP addresses and vice a versa, however make sure that you delete an "@" sign and everything to the left of it before entering a base 10 URL in the IP decoder.

CIDR

CIDR stands for Classless Inter-Domain Routing, which is a new addressing scheme for the Internet which allows for more efficient allocation of IP addresses than the old classes address scheme. It has been developed primarily to take care of following issues -

- To overcome the shortage of allocable IP addresses.
- To assign IP addresses more efficiently, in order to cater to various sized networks.

A Class C network is capable of having 254 hosts, but not all class C networks needed all these hosts, as a result a very small percentage of the assigned IP addresses were used. This was even more prominent for Class A and Class B networks. CIDR was developed to be a much more efficient method of assigning addresses. CIDR allows networks as small as 32 hosts to networks as big as 524,288 hosts, thus reducing lot of "wasted" IP addresses. A typical CIDR IP address looks like -

207.17.10.36/26

Where "/26" indicates that first 26 bits are used for network address and rest 6 bits are used for hosts (nodes), allowing about 62 hosts, which is approximately one fourth (1/4) of Class C.

The Internet today is a mixture of CIDR and old Classes scheme, while CIDR becoming more and more popular and classes scheme more or less obsolete. Most new routers support CIDR and standards recommend it. It is not feasible to cover it in more details here, I'll probably write a new article about CIDR once I know more about it.

10.6 Summary

In this unit you have learnt about internetworking like Bridges, Routers, and Gateways. You have also learnt about Internet Architecture and Addressing.

- Bridges is a device used to connect two separate Ethernet **networks** into one extended Ethernet. **Bridges** only forward packets between **networks** that are destined for the other **network**. Term used by Novell to denote a **computer** that accepts packets at the **network** layer and forward them to another **network**.
- A **router** is a **networking** device that forwards data packets between **computer networks**. **Routers** perform the "traffic directing" functions on the Internet. A data packet is typically forwarded from one **router** to another through the **networks** that constitute the internetwork until it reaches its destination node.
- A gateway is a node (router) in a computer network, a key *stopping point* for data on its way to or from other networks. Thanks to gateways, we are able to communicate and send data back and forth. The Internet wouldn't be any use to us without gateways (as well as a lot of other hardware and software).
- The Internet is a worldwide, publicly accessible network of interconnected computer networks that transmit data by packet switching using the standard Internet Protocol (IP). It is a "network of networks" that consists of millions of smaller domestic, academic, business, and government networks, which together carry various information and services, such as electronic mail, online chat, file transfer, and the interlinked web pages and other documents of the World Wide Web.
- Computers communicate over the Internet using the IP protocol (*Internet Protocol*), which uses numerical addresses, called **IP addresses**, made up of four whole numbers (4 bytes) between 0 and 255 and written in the format xxx.xxx.xxx.xxx. For example, *194.153.205.26* is an IP address given in technical format. These addresses are used by networked computers to communicate, so each computer on a network has a unique IP address on that network.

10.7 Review Questions

Q1. Explain the working of the bridges. Why do we need bridges in a network?

Q2. What is a router? How router forward a packet in a network?

Q3. Describe gateway in detail.

Q4. Why do we need internet architecture? Explain.

Q5. Define all types of classes used in a network.

RIL-089

UNIT-XI

Transport Layer-I

Structure

11.0 Introduction

This unit focused on transport layer. In this unit there are six main sections. In this unit you will learn about transport layer. The transport layer is the fourth layer of the OSI reference model. It provides transparent transfer of data between end systems using the services of the network layer (e.g. IP) below to move PDUs of data between the two communicating systems. The transport service is said to perform "peer to peer" communication, with the remote (peer) transport entity. The data communicated by the transport layer is encapsulated in a transport layer PDU and sent in a network layer SDU. The network layer nodes (i.e. Intermediate Systems (IS)) transfer the transport PDU intact, without decoding or modifying the content of the PDU. In this way, only the peer transport entities actually communicate using the PDUs of the transport protocol. In section one that is Sec. 11.13. 1 Design issues. Next section i.e. Sec. 11.2 describe the Quality of Services. In Sec 11.3 Primitives Connection Management has been described. In the last section you will learn about Addressing in transport layer. In Sec. 11.5 and 11.6 you will find summary and review questions respectively.

Objectives

After studying this unit you should be able to:

- ❖ Define design issues in transport layer.
- ❖ Describe Quality of Services.
- ❖ Express Primitives Connection Management.

- Define Addressing.

11.1 Design issues

The transport layer delivers the message from one process to another process running on two different hosts. Thus, it has to perform number of functions to ensure the accurate delivery of message. The various functions of transport layer are:

- Establishing, Maintaining & Releasing Connection
- Addressing
- Data Transfer
- Flow Control
- Error Control
- Congestion Control

Establishing, Maintaining & Releasing Connection:

The transport layer establishes, maintains & releases end-to-end transport connection on the request of upper layers. Establishing a connection involves allocation of buffers for storing user data, synchronizing the sequence numbers of packets etc. A connection is released at the request of upper layer.

Addressing:

In order to deliver the message from one process to another, an addressing scheme is required. Several process may be running on a system at a time. In order to identify the correct process out of the various running processes, transport layer uses an addressing scheme called port number. Each process has a specific port number.

Data Transfer:

Transport layer breaks user data into smaller units and attaches a transport layer header to each unit forming a TPDU (Transport Layer Data Unit). The TPDU is handed over to the network layer for its delivery to destination. The TPDU header contains port number, sequence number, acknowledgement number, checksum and other fields.

Flow Control:

Like data link layer, transport layer also performs flow control. However, flow control at transport layer is performed end-to-end rather than node-to-node. Transport Layer uses a sliding window protocol to perform flow control.

Error Control:

Transport layer also provides end-to-end error control facility. Transport layer deals with several different types of errors: Error due to damaged bits. Error due to non-delivery of TPDUs. Error due to duplicate delivery of TPDUs. Error due to delivery of TPDU to a wrong destination.

Congestion Control:

Transport layer also handles congestion in the networks. Several different congestion control algorithms are used to avoid congestion.

11.2 Quality of Services

QoS (Quality of Service) refers to a broad set of networking technologies and techniques designed to guarantee predicable levels of network performance. Elements of network performance within the scope of QoS include availability (uptime), bandwidth (throughput), latency (delay), and error rate (packet loss). Quality of Service (QoS) refers to the capability of a network to provide better service to selected network traffic over various technologies, including Frame Relay, Asynchronous Transfer Mode (ATM), Ethernet and 802.1 networks, SONET, and IP-routed networks that may use any or all of these underlying technologies. The primary goal of QoS is to provide priority including dedicated bandwidth, controlled jitter and latency (required by some real-time and interactive traffic), and improved loss characteristics. Also important is making sure that providing priority for one or more flows does not make other flows fail. QoS technologies provide the elemental building blocks that will be used for future business applications in campus, WAN, and service provider networks.

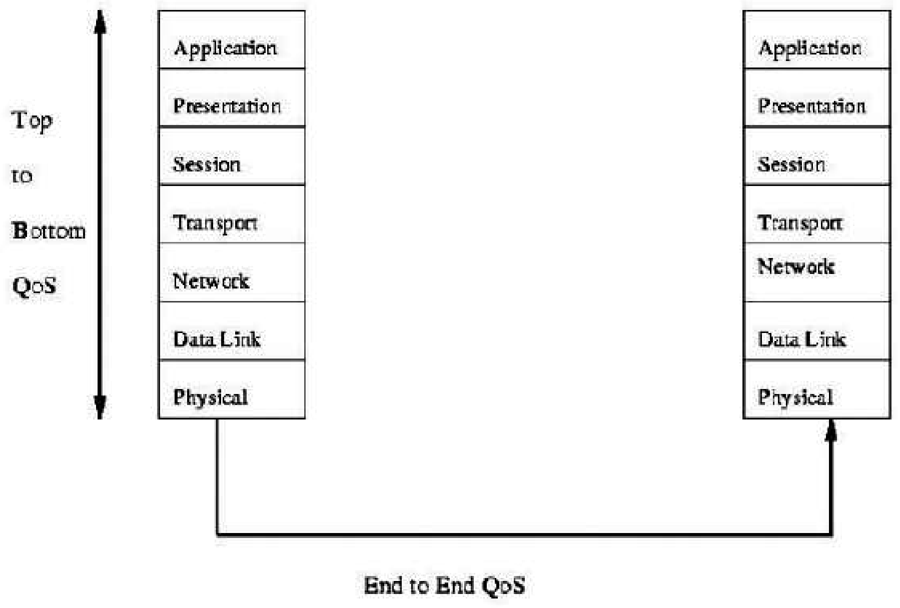


Figure 11.1-Level of QoS

Why do we need QoS?

Today's networks need to support multiple kinds of traffic over single network links. Different kinds of traffic demand different treatments from the network. Just like a first class passenger is ready to pay the premium for superior service, customers of today are ready to pay extra for preferential treatment of their traffic. And just as we cannot have a separate airplane for each first class passenger, similarly we cannot have separate network connections for each of our customers. Therefore much of the bulk of network traffic have to flow through lines where first class traffic and other classes of traffic have to share the bandwidth (just like economy class passengers share the airplane with first class passengers). We can only differentiate at places where the traffic flows through active network elements which have the capability to differentiate. Examples of such entities are routers, switches and gateways. Therefore, the need to design networks which

1. Can deliver multiple classes of service - that is they should be QoS conscious.
2. Is Scalable - so that network traffic can increase without affecting network performance
3. Can support emerging network intensive, mission critical applications which will be the key determinant of a companies success in the global world.

The basic architecture introduces the three fundamental pieces for QoS implementation:

- QoS identification and marking techniques for coordinating QoS from end-to-end between network elements
- QoS within a single network element (for example, queuing, scheduling, and traffic-shaping tools)
- QoS policy, management, and accounting functions to control and administer end-to-end traffic across a network

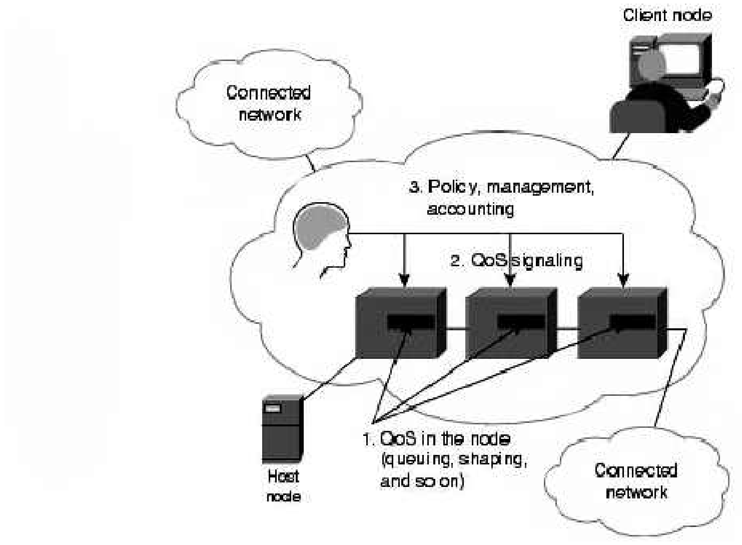

Client node
Connected network
3. Policy, management, accounting
2. QoS signaling
1. QoS in the node (queuing, shaping, and so on)
Host node
Connected network

Figure 11.2- A Basic QoS Implementation Has Three Main Components

Building a Network with QoS

QoS involves prioritization of network traffic. QoS can be targeted at a network interface, toward a given server or router, or at specific applications. A network monitoring system must typically be deployed as part of a QoS solution to ensure that networks are performing at the desired level.

QoS is especially important for Internet applications such as video-on-demand, voice over IP (VoIP) systems and other consumer services where high-performance and high-quality streaming is involved.

Traffic Shaping and Traffic Policing

Some people use the terms **traffic shaping** and QoS interchangeably as shaping is one of the most common techniques used in QoS.

As an alternative to limiting traffic, ISPs could theoretically instead charge premium prices to support certain types of applications on their networks to make money. Some net neutrality observers fear this may tempt providers to engage in anti-competitive practices, such as charging exorbitantly high fees that lock out smaller businesses from Internet ecommerce.

Check your progress

Q1. Explain the design issues of transport layer.

Q2. What do you mean by QoS? Explain in detail.

11.3 Primitives Connection Management

The transport services

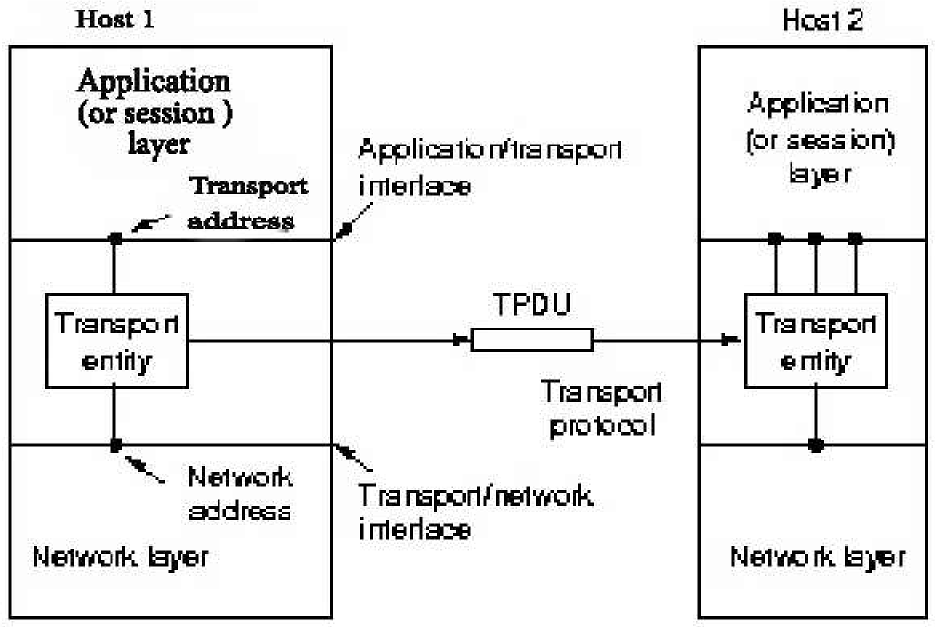


Figure 11.3: Services provided to the upper layers

The **transport entity** can be in the operating system kernel, in a separate user process, in a library package bound into network applications, or on a network interface card. There are 2 types of transport services, connection-oriented and connectionless, just as in the network layer, and they are very similar to them. Why are there then 2 separate layers? The answer is subtle, but crucial. The network layer is part of the communication subnet and is run by the carrier, at least for the WAN. The users have no control over this, what happens if the offered connection-oriented service is unreliable, frequently loses packets or its routers crash from time to time? The user cannot put in better routers or put more error handling in the data link layer.

In essence, the transport layer makes it possible for the transport service to be more reliable than the underlying network service, lost packets and mangled data can be detected and compensated for. Furthermore, the transport service primitives can be designed to be independent of the network service primitives, which may vary considerably from network to network (e.g. a connectionless LAN service may be quite different than a connection-oriented WAN service)

Due to the transport layer, application programs can be written using a standard set of primitives so that they can work on a wide variety of networks without having to worry about dealing with different subnet interfaces and unreliable transmissions. The bottom 4 layers are therefore sometimes seen as the **transport service provider** and the higher layers as the **transport service user**.

Transport service primitives

Primitive	**TPDU sent**	**Meaning**
Listen	none	Block until some process tries to connect
Connect	Connection request	Actively attempt to establish a connection
Send	Data	Send information
Receive	none	Block until a Data TPDU arrives
Disconnect	Disconnection req.	This side wants to release the connection

Table-11.1 Transport Services

The primitives for a simple transport service. TPDU means Transport Protocol Data Unit. To the transport users, a connection is a reliable bit pipe: one user stuffs bits in and they magically appear at the other end.

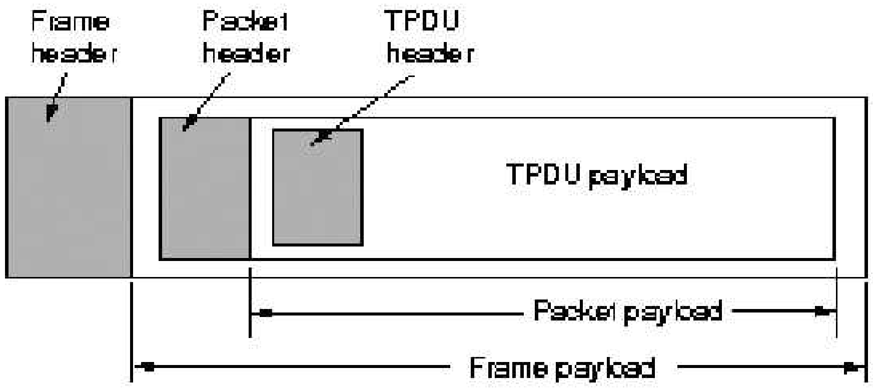


Figure 11-4: Nesting of TPDU, Packets and Frames

Disconnection, meaning no more data needs to be send and the used buffer space can be released, has 2 variants. In the asymmetric variant, either transport user can issue a disconnect primitive, which results in a disconnect TPDU being sent to the remote transport entity. Upon arrival the connection is released. In the symmetric variant, each direction is closed separately, independent of the other one. When one side does a disconnect it means it has no more data to send, but is still willing to accept data from its partner. A connection is released when both sides have done a disconnect.

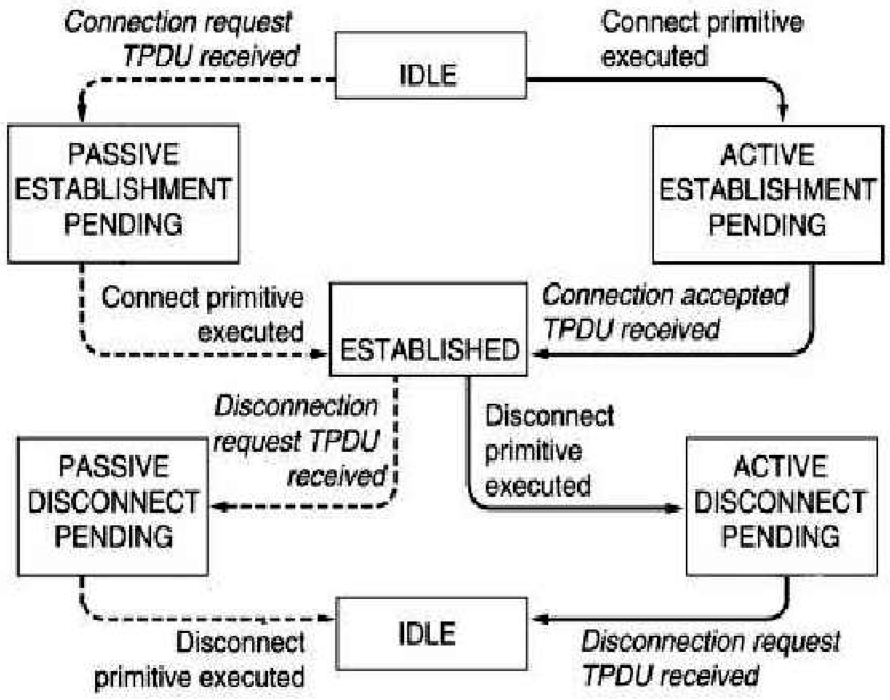

Connection request TPDU received
IDLE
Connect primitive executed
PASSIVE ESTABLISHMENT PENDING
ACTIVE ESTABLISHMENT PENDING
Connect primitive executed
ESTABLISHED
Connection accepted TPDU received
Disconnection request TPDU received
Disconnect primitive executed
PASSIVE DISCONNECT PENDING
ACTIVE DISCONNECT PENDING
Disconnect primitive executed
IDLE
Disconnection request TPDU received

Figure 11.5: Connection establishment and release

A state diagram for connection establishment and release for these simple primitives. Each transition is triggered by some event, either a primitive executed by the local transport user or an incoming packet (labeled in italics).

The solid lines show the client's state sequence, the dashed lines the server ones.

Berkeley sockets

Primitive	**Meaning**
SOCKET	Create a new communication end point. The parameters specify the addressing format, the type of service (e.g. reliable byte stream) and the protocol. It returns an ordinary file descriptor for use in succeeding calls, the same way as an OPEN call does.
BIND	Binds a local address to a socket.
LISTEN	Allocates space to queue incoming calls for the case that several clients try to connect at the same time, it does not block.
ACCEPT	Blocks a server until a connect request TPDU from a client arrives. A new socket with the same properties as the original LISTEN one is then created by the transport entity and a file descriptor for it is returned. The server can then fork off a process or tread to handle the connection on the new socket and go back to waiting for the next connection on the original socket by issuing a new ACCEPT primitive.
CONNECT	Issued by a client after it has created a socket, it blocks the client and actively starts the connection process. When it completes (the appropriate TPDU is received from the server), the client process is unblocked and the connection is established.
SEND	Sends some data over the connection.
RECEIVE	Receives some data from the connection
CLOSE	Symmetric connection release.

Table 11.2- Berkeley sockets

Socket

Normally, a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

On the client-side: The client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client tries to rendezvous with the server on the server's machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. This is usually assigned by the system.

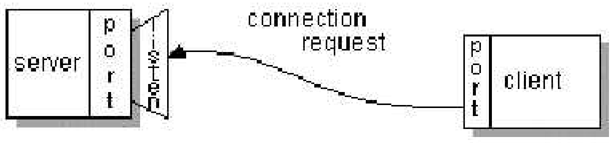


Figure 11.6: On the Client side

If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bound to the same local port and also has its remote endpoint set to the address and port of the client. It needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client.

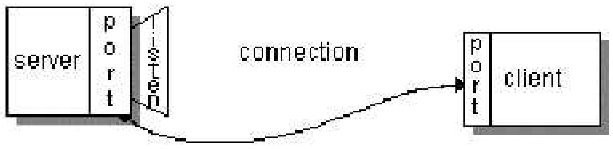


Figure 11.7: On the Server side

On the client side, if the connection is accepted, a socket is successfully created and the client can use the socket to communicate with the server.

The client and server can now communicate by writing to or reading from their sockets.

A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

An endpoint is a combination of an IP address and a port number. Every TCP connection can be uniquely identified by its two endpoints. That way you can have multiple connections between your host and the server.

The java.net package in the Java platform provides a class, Socket, that implements one side of a two-way connection between your Java program and another program on the network. The Socket class sits on top of a platform-dependent implementation, hiding the details of any particular system from your Java program. By using the java.net.Socket class instead of relying on native code, your Java programs can communicate over the network in a platform-independent fashion.

Additionally, java.net includes the ServerSocket class, which implements a socket that servers can use to listen for and accept connections to clients. This lesson shows you how to use the Socket and ServerSocket classes.

If you are trying to connect to the Web, the URL class and related classes (URLConnection, URLEncoder) are probably more appropriate than the socket classes. In fact, URLs are a relatively high-level connection to the Web and use sockets as part of the underlying implementation

Where is Socket Used?

A Unix Socket is used in a client-server application framework. A server is a process that performs some functions on request from a client. Most of the application-level protocols like FTP, SMTP, and POP3 make use of sockets to establish connection between client and server and then for exchanging data.

Socket Types

There are four types of sockets available to the users. The first two are most commonly used and the last two are rarely used.

Processes are presumed to communicate only between sockets of the same type but there is no restriction that prevents communication between sockets of different types.

- **Stream Sockets** - Delivery in a networked environment is guaranteed. If you send through the stream socket three items "A, B, C", they will arrive in the same order – "A, B, C". These sockets use TCP (Transmission Control Protocol) for data transmission. If delivery is impossible, the sender receives an error indicator. Data records do not have any boundaries.

- **Datagram Sockets** - Delivery in a networked environment is not guaranteed. They're connectionless because you don't need to have an open connection as in Stream Sockets – you build a packet with the destination information and send it out. They use UDP (User Datagram Protocol).

- **Raw Sockets** - These provide users access to the underlying communication protocols, which support socket abstractions. These sockets are normally datagram oriented, though their exact characteristics are dependent on the interface provided by the protocol. Raw sockets are not intended for the general user; they have been provided mainly for those interested in developing new communication protocols, or for gaining access to some of the more cryptic facilities of an existing protocol.

- **Sequenced Packet Sockets** - They are similar to a stream socket, with the exception that record boundaries are preserved. This interface is provided only as a part of the Network Systems (NS) socket abstraction, and is very important in most serious NS applications. Sequenced-packet sockets allow the user to manipulate the Sequence Packet Protocol (SPP) or Internet Datagram Protocol (IDP) headers on a packet or a group of packets, either by writing a prototype header along with whatever data is to be sent, or by specifying a default header to be used with all outgoing data, and allows the user to receive the headers on incoming packets.

11.4 Addressing

Transport layer addresses are called **TSAP** (Transport Service Access Point). In Internet, they consists of an (IP address, local port) pair, in ATM of an (AAL, SAP) pair. An IP address is an example of a **NSAP**.

How does the user process on host 1 know that the address of the server, e.g. the time-of-day server, is attached to TSAP 122? One possibility is, that it has always been there and everyone knows it. This might work for a small number of key services that never change, but, in general, user processes often want to talk to other user processes that only exist for a short time and do not have a TSAP that is known in advance. Furthermore, if there are potentially many server processes, most of which are rarely used, it is wasteful to have each of them active and listening to a stable TSAP all day long.

In the **initial connection protocol**, used by UNIX hosts, each machine that wishes to offer services to remote users, has a special **process server** that acts as a proxy for less-heavily used servers. It listens to a set of ports at the same time, waiting for a TCP connection request. If a request comes in, the process server spawns off the requested server, allowing it to inherit the existing connection with the user. The new server does the work and the process server goes back listening for new requests.

Often a **name server** or **directory server** is used. It listens at a well know port for clients which can ask for the TSAP corresponding with a given service name, e.g. time-of-day. After receiving the TSAP, the client disconnects and establishes a connection with the service of which it now knows the TSAP. When a new service is created, it must register itself with the name server, giving it its name (typically an ASCII string) and the address of its TSAP.

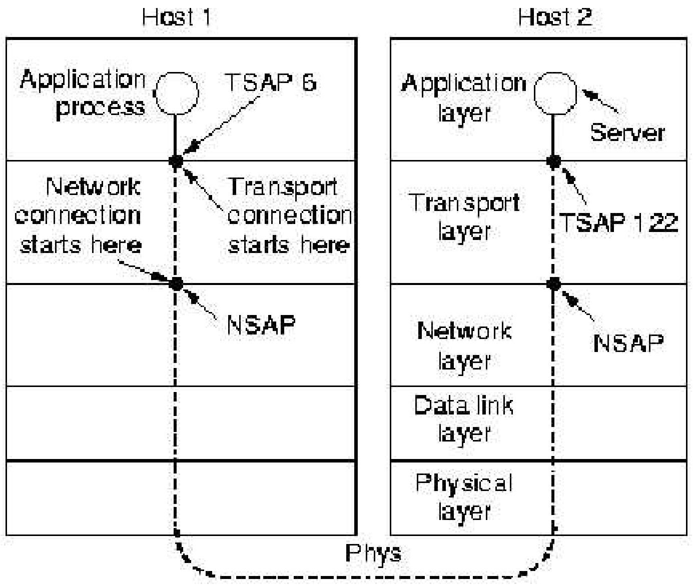


Figure 11.8: Addressing

Check your progress

Q1. Define TPDU.

Q2. What is a socket?

11.5 Summary

In this unit you have learnt about Design issues of transport layer, Quality of Services, Primitives Connection Management and Addressing.

- ❖ In computer networking, the **transport layer** is a conceptual division of methods in the layered architecture of protocols in the network stack in the Internet Protocol Suite and the Open Systems Interconnection (OSI). The protocols of the **layer** provide host-to-host communication services for applications.
- ❖ **Quality of service** (QoS) is the overall performance of a telephony or computer network, particularly the performance seen by the users of the network. To quantitatively measure quality of service, several related aspects of the network service are often considered, such as error rates, bit rate, throughput, transmission delay, availability, jitter, etc. Quality of service is particularly important for the transport of traffic with special requirements. In particular, developers have introduced technology to allow computer networks to become as useful as telephone networks for audio conversations, as well as supporting new applications with even stricter service demands.

- A network **socket** is an endpoint of a connection across a computer network. Today, most communication between computers is based on the Internet Protocol; therefore most network **sockets** are Internet **sockets**.

11.6 Review Questions

Q1. Explain different types of design issue in transport layer.

Q2. What is the use of QoS in computer network?

Q3. Why we need socket? Explain different types of socket.

RIL-089

UNIT-XII
Transport Layer-II

Structure

12.0 Introduction

This block focus on more details in transport layer. There are seven section in this unit. In the Sec 12.1 you will learn about Connection Establishment and Releases. In the next section i.e. Sec. 12.2 Use of Timers has been defined. In Sec. 12.3 you will learn about Flow Control and Buffering. As you have studied flow control earlier at the **transport** level **flow control** will allow the **transport** protocol entity in a host to restrict the **flow** of data over a logical connection from the **transport** protocol entity in another host. However, one of the services of the network level is to prevent congestion. Next section defined Multiplexing. You will find Crash Recovery in Sec. 12.5. In Sec. 12.6 and 12.7 you will find summary and review questions respectively.

Objectives

After studying this unit you should be able to:

- ❖ Define Connection Establishment and Releases
- ❖ Describe Use of Timers
- ❖ Define Flow Control and Buffering
- ❖ Express Multiplexing
- ❖ Describe Crash Recovery.

12.1 Connection Establishment and Releases

To aid in our understanding the connect, accept, and close functions and to help us debug TCP applications using the netstat program, we must understand how TCP connections are established and terminated, and TCP's state transition diagram.

Three-Way Handshake

The following scenario occurs when a TCP connection is established:

1. The server must be prepared to accept an incoming connection. This is normally done by calling socket, bind, and listen and is called a *passive open*.
2. The client issues an *active open* by calling connect. This causes the client TCP to send a "synchronize" (SYN) segment, which tells the server the client's initial sequence number for the data that the client will send on the connection. Normally, there is no data sent with the SYN; it just contains an IP header, a TCP header, and possible TCP options (which we will talk about shortly).
3. The server must acknowledge (ACK) the client's SYN and the server must also send its own SYN containing the initial sequence number for the data that the server will send on the connection. The server sends its SYN and the ACK of the client's SYN in a single segment.
4. The client must acknowledge the server's SYN.

The minimum number of packets required for this exchange is three; hence, this is called TCP's *three-way handshake*. We show the three segments in Figure 12.1

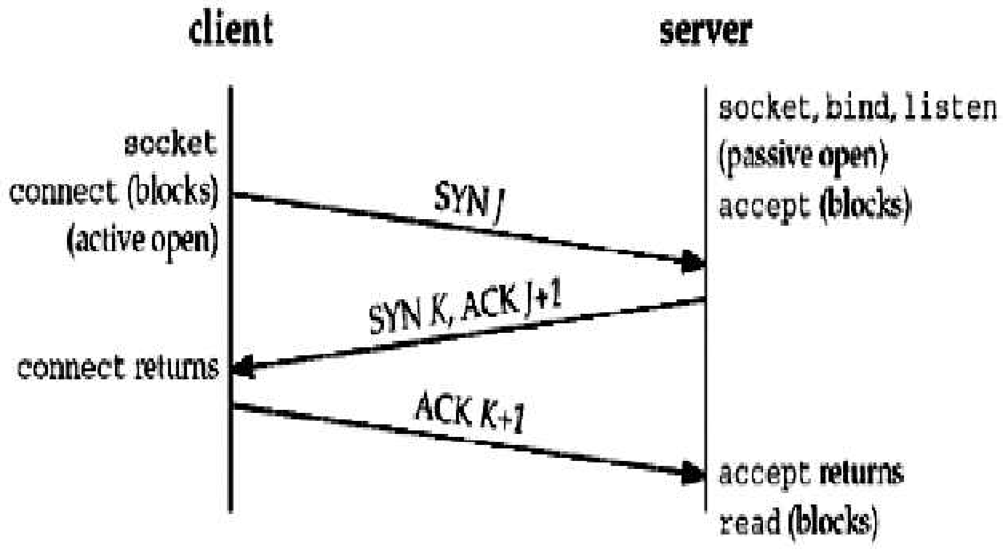


Figure 12.1 TCP three-way handshake.

We show the client's initial sequence number as *J* and the server's initial sequence number as *K*. The acknowledgment number in an ACK is the next expected sequence number for the end sending the ACK. Since a SYN occupies one byte of the sequence number space, the acknowledgment number in the ACK of each SYN is the initial sequence number plus one. Similarly, the ACK of each FIN is the sequence number of the FIN plus one.

An everyday analogy for establishing a TCP connection is the telephone system [Nemeth 1997]. The socket function is the equivalent of having a telephone to use bind is telling other people your telephone number so that they can call you listen is turning on the ringer so that you will hear when an incoming call arrives connect requires that we know the other person's phone number and dial it accept is when the person being called answers the phone. Having the client's identity returned by accept (where they identify is the client's IP address and port number) is similar to having the caller ID feature show the caller's phone number. One difference, however, is that accept returns the client's identity only after the connection has been established, whereas the caller ID feature shows the caller's phone number before we choose whether to answer the phone or not. If the DNS is used it provides a service analogous to a telephone book. getaddrinfo is similar to looking up a person's phone number in the phone book. getnameinfo would be the equivalent of having a phone book sorted by telephone numbers that we could search, instead of a book sorted by name.

TCP Options

Each SYN can contain TCP options. Commonly used options include the following:

- MSS option. With this option, the TCP sending the SYN announces its *maximum segment size*, the maximum amount of data that it is willing to accept in each TCP segment, on this connection. The sending TCP uses the receiver's MSS value as the maximum size of a segment that it sends. We will see how to fetch and set this TCP option with the TCP_MAXSEG socket option.

- Window scale option. The maximum window that either TCP can advertise to the other TCP is 65,535, because the corresponding field in the TCP header occupies 16 bits. But, high-speed connections, common in today's Internet (45 Mbits/sec and faster, as described in RFC 1323 [Jacobson, Braden, and Borman 1992]), or long delay paths (satellite links) require a larger window to obtain the maximum throughput possible. This newer option specifies that the advertised window in the TCP header must be scaled (left-shifted) by 0–14 bits, providing a maximum window of almost one gigabyte (65,535 x 2^{14}). Both end-systems must support this option for the window scale to be used on a connection. We will see how to affect this option with the SO_RCVBUF socket option.

To provide interoperability with older implementations that do not support this option, the following rules apply. TCP can send the option with its SYN as part of an active open. But, it can scale its windows only if the other end also sends the option with its SYN. Similarly, the server's TCP can send this option only if it receives the option with the client's SYN. This logic assumes that implementations ignore options that they do not understand, which is required and common, but unfortunately, not guaranteed with all implementations.

- Timestamp option. This option is needed for high-speed connections to prevent possible data corruption caused by old, delayed, or duplicated segments. Since it is a newer option, it is negotiated similarly to the window scale option. As network programmers there is nothing we need to worry about with this option.

These common options are supported by most implementations. The latter two are sometimes called the "RFC 1323 options," as that RFC [Jacobson, Braden, and Borman 1992] specifies the options. They are also called the "long fat pipe options," since a network with either a high bandwidth or a long delay is called a *long fat pipe*.

TCP Connection Termination

While it takes three segments to establish a connection, it takes four to terminate a connection.

1. **One application calls close first, and we say that this end performs the *active close*. This end's TCP sends a FIN segment, which means it is finished sending data.**

2. The other end that receives the FIN performs the *passive close*. The received FIN is acknowledged by TCP. The receipt of the FIN is also passed to the application as an end-of-file (after any data that may have already been queued for the application to receive), since the receipt of the FIN means the application will not receive any additional data on the connection.

3. Sometime later, the application that received the end-of-file will close its socket. This causes its TCP to send a FIN.

4. The TCP on the system that receives this final FIN (the end that did the active close) acknowledges the FIN.

Since a FIN and an ACK are required in each direction, four segments are normally required. We use the qualifier "normally" because in some scenarios, the FIN in Step 1 is sent with data. Also, the segments in Steps 2 and 3 are both from the end performing the passive close and could be combined into one segment. We show these packets in Figure 12.2.

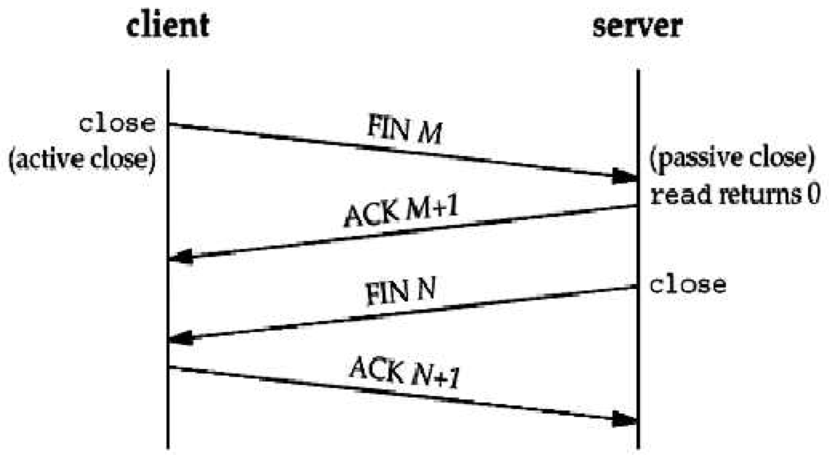


Figure 12.2- Packets exchanged when a TCP connection is closed.

A FIN occupies one byte of sequence number space just like a SYN. Therefore, the ACK of each FIN is the sequence number of the FIN plus one.

Between Steps 2 and 3 it is possible for data to flow from the end doing the passive close to the end doing the active close. This is called a *half-close function*. The sending of each FIN occurs when a socket is closed. We indicated that the application calls close for this to happen, but realize that when a Unix process terminates, either voluntarily (calling exit or having the main function return) or involuntarily (receiving a signal that terminates the process), all open descriptors are closed, which will also cause a FIN to be sent on any TCP connection that is still open.

Although we show the client in Figure 12.2 performing the active close, either end—the client or the server—can perform the active close. Often the client performs the active close, but with some protocols (notably HTTP/1.0), the server performs the active close.

TCP State Transition Diagram

The operation of TCP with regard to connection establishment and connection termination can be specified with a *state transition diagram*. We show this in Figure 12.3.

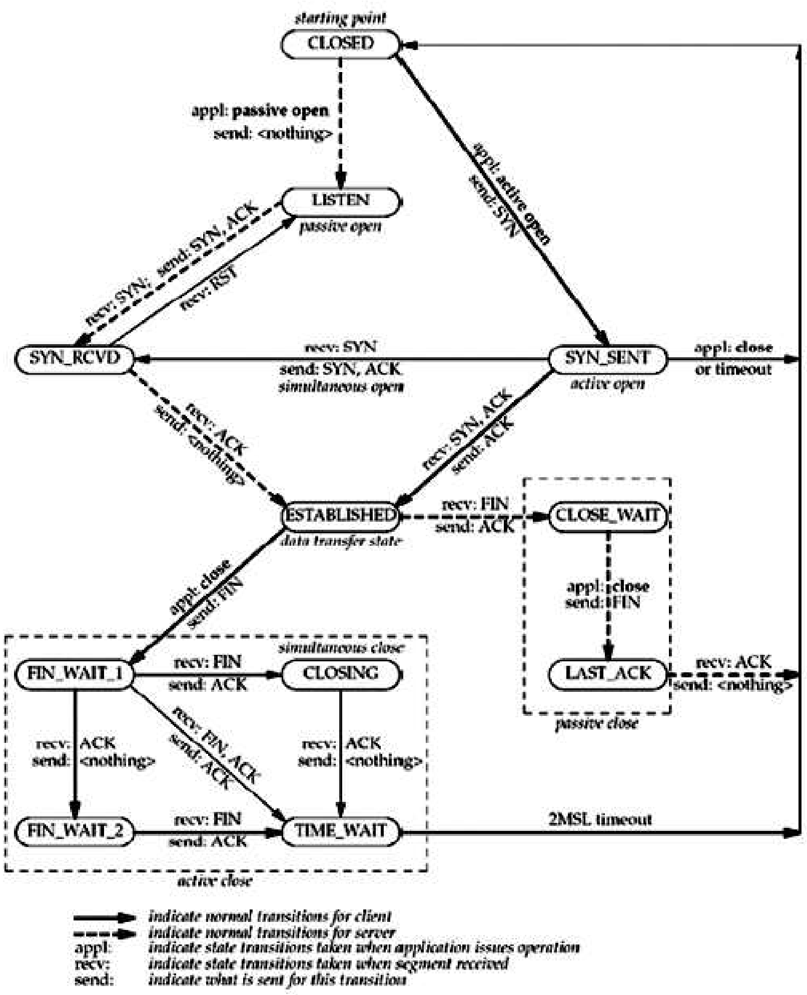


Figure 12.3 TCP state transition diagram.

There are 11 different states defined for a connection and the rules of TCP dictate the transitions from one state to another, based on the current state and the segment received in that state. For example, if an application performs an active open in the CLOSED state, TCP sends a SYN and the new state is SYN_SENT. If TCP next receives a SYN with an ACK, it sends an ACK and the new state is ESTABLISHED. This final state is where most data transfer occurs.

The two arrows leading from the ESTABLISHED state deal with the termination of a connection. If an application calls close before receiving a FIN (an active close), the transition is to the FIN_WAIT_1 state. But if an application receives a FIN while in the ESTABLISHED state (a passive close), the transition is to the CLOSE_WAIT state.

We denote the normal client transitions with a darker solid line and the normal server transitions with a darker dashed line. We also note that there are two transitions that we have not talked about: a simultaneous open (when both ends send SYNs at about the same time and the SYNs cross in the network) and a simultaneous close (when both ends send FINs at the same time). Chapter 18 of TCPv1 contains examples and a discussion of both scenarios, which are possible but rare.

One reason for showing the state transition diagram is to show the 11 TCP states with their names. These states are displayed by netstat, which is a useful tool when debugging client/server applications.

Watching the Packets

Figure 4.4 shows the actual packet exchange that takes place for a complete TCP connection: the connection establishment, data transfer, and connection termination. We also show the TCP states through which each endpoint passes.

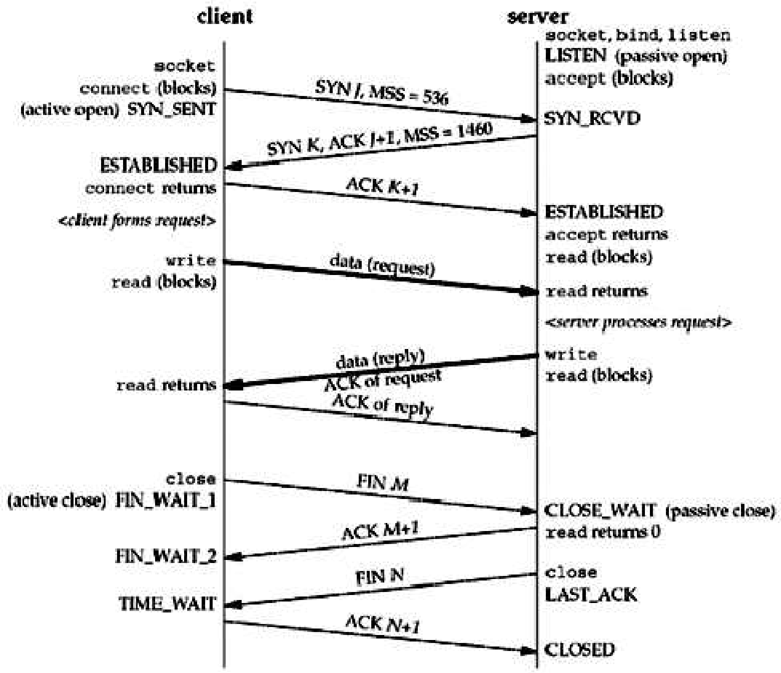


Figure 12.4- Packet exchange for TCP connection.

The client in this example announces an MSS of 536 (indicating that it implements only the minimum reassembly buffer size) and the server announces an MSS of 1,460 (typical for IPv4 on an Ethernet). It is okay for the MSS to be different in each direction.

Once a connection is established, the client forms a request and sends it to the server. We assume this request fits into a single TCP segment (i.e., less than 1,460 bytes given the server's announced MSS). The server processes the request and sends a reply, and we assume that the reply fits in a single segment (less than 536 in this example). We show both data segments as bolder arrows. Notice that the acknowledgment of the client's request is sent with the server's reply. This is called *piggybacking* and will normally happen when the time it takes the server to process the request and generate the reply is less than around 200 ms. If the server takes longer, say one second, we would see the acknowledgment followed later by the reply.

12.2 Use of Timers

Since TCP does not know about the congestion or performance of the underlying network and the network does not provide any guarantees for packet delivery, TCP relies on a time limit to determine if any transmitted segments were lost. If an acknowledgement is not received within a specific time limit then the sender may assume that the segment was lost. Since IP provides no guarantees on the timeliness of delivery, this retransmission timeout (RTO) can only be a best guess. We need a value that is long enough to avoid excessive timeouts and retransmissions but is short enough to avoid excessive waits.

TCP keeps track of the average round-trip time of segments. Since these may fluctuate over time, TCP uses an exponentially weighted moving average (EWMA), which places approximately 10–20% of the weight on the most recent RTT measurement. This average of the RTT is called the smoothed round trip time, or SRTT.

The second thing that TCP keeps track of is how much the recently measured round-trip time deviated from the SRTT. This too is an EWMA function, placing approximately 25% of the weight on the most recent deviation. The average delay is called the round-trip time variation, or RTTVAR. It is an estimate of how much the RTT typically deviates from the average RTT and is an approximation to the standard deviation, which would be slower to compute.

The retransmission timeout (RTO) value is set to a function of the SRTT and RTTVAR:

$$\text{timeout interval} = \text{SRTT} + 4 * \text{RTTVAR}$$

However, whenever a timeout occurs, this value is doubled. The timeout value is doubled for each timed-out retransmission up to a value of 64 seconds. This doubling of the timeout is called an exponential backoff. Whenever an acknowledgement is received without a timeout, the timeout interval is reset to its base value.

TCP uses different types of timer to control and management various tasks:

Keep-alive timer:

- This timer is used to check the integrity and validity of a connection.
- When keep-alive time expires, the host sends a probe to check if the connection still exists.

Retransmission timer:

- This timer maintains stateful session of data sent.
- If the acknowledgement of sent data does not receive within the Retransmission time, the data segment is sent again.

Persist timer:

- TCP session can be paused by either host by sending Window Size 0.
- To resume the session a host needs to send Window Size with some larger value.
- If this segment never reaches the other end, both ends may wait for each other for infinite time.
- When the Persist timer expires, the host re-sends its window size to let the other end know.
- Persist Timer helps avoid deadlocks in communication.

Timed-Wait:

- After releasing a connection, either of the hosts waits for a Timed-Wait time to terminate the connection completely.
- This is in order to make sure that the other end has received the acknowledgement of its connection termination request.
- Timed-out can be a maximum of 240 seconds (4 minutes).

Check your progress

Q1. Define three way hand shaking in transport layer.

Q2. Explain use of timers in transport layer.

12.3 Flow Control and Buffering

In some ways the flow control problem in the transport layer is the same as in the data link layer, but in other ways it is different. The basic similarity is that in both layers a sliding window or other scheme is needed on each connection to keep a fast transmitter from overrunning a slow receiver. The main difference is that a router usually has relatively few lines, whereas a host may have numerous connections. In the data link layer, the sending side must buffer outgoing frames because they might have to be retransmitted. If the subnet provides datagram service, the sending transport entity must also buffer, and for the same reason.

If the receiver knows that the sender buffers all TPDUs until they are acknowledged, the receiver may or may not dedicate specific buffers to specific connections, as it sees fit. The receiver may, for example, maintain a single buffer pool shared by all connections. When a TPDU comes in, an attempt is made to dynamically acquire a new buffer. If one is available, the TPDU is accepted; otherwise, it is discarded. Since the sender is prepared to retransmit TPDUs lost by the subnet, no harm is done by having the receiver drop TPDUs, although some resources are wasted. The sender just keeps trying until it gets an acknowledgement. If the network service is unreliable, the sender must buffer all TPDUs sent, just as in the data link layer.

However, with reliable network service, other trade-offs become possible. In particular, if the sender knows that the receiver always has buffer space, it need not retain copies of the TPDUs it sends. However, if the receiver cannot guarantee that every incoming TPDU will be accepted, the sender will have to buffer anyway. In the latter case, the sender cannot trust the network layer's acknowledgement, because the acknowledgement means only that the TPDU arrived, not that it was accepted. We will come back to this important point later. Even if the receiver has agreed to do the buffering, there still remains the question of the buffer size. If most TPDUs are nearly the same size, it is natural to organize the buffers as a pool of identically-sized buffers, with one TPDU per buffer, as in Figure 4.5 (a).

However, if there is wide variation in TPDU size, from a few characters typed at a terminal to thousands of characters from file transfers, a pool of fixed-sized buffers presents problems. If the buffer size is chosen equal to the largest possible TPDU, space will be wasted whenever a short

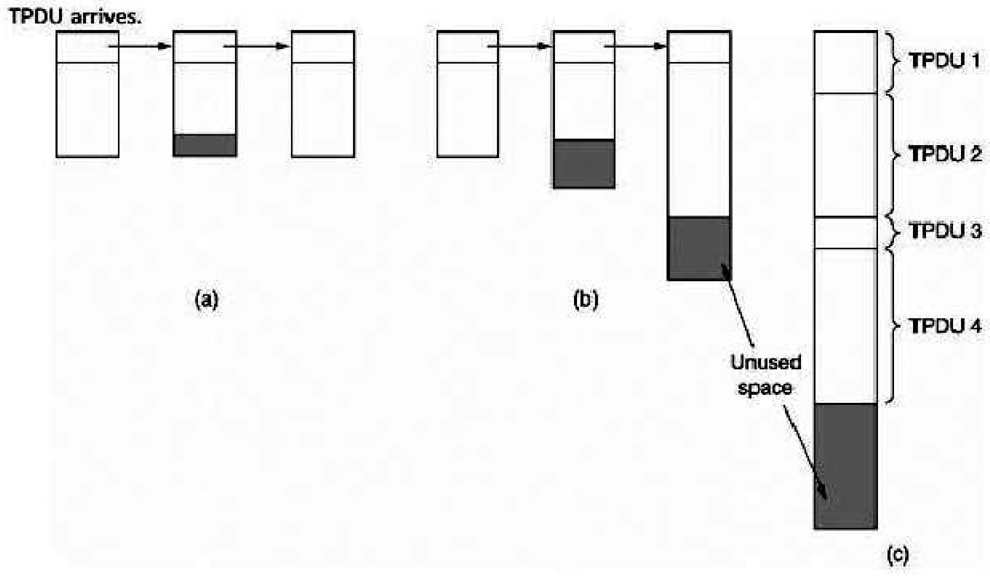


Figure 12.5- (a) Chained fixed-size buffer. (b) Chained Variable-Sized buffer. (c) One large circular buffer per connection.

If the buffer size is chosen less than the maximum TPDU size, multiple buffers will be needed for long TPDUs, with the attendant complexity. Another approach to the buffer size problem is to use variable-sized buffers, as in Figure 12.6 (b).

The advantage here is better memory utilization, at the price of more complicated buffer management. A third possibility is to dedicate a single large circular buffer per connection, as in Figure 12.5 (c).

This system also makes good use of memory, provided that all connections are heavily loaded, but is poor if some connections are lightly loaded. The optimum trade-off between source buffering and destination buffering depends on the type of traffic carried by the connection.

12.4 Multiplexing

Most communication in TCP/IP takes the form of exchanges of information between a program running on one device, and a matching program on another device. Each instance of an application represents a copy of that application software that needs to send and receive information. These application instances are commonly called *processes*. A TCP/IP application process is any piece of networking software that sends and receives information using the TCP/IP protocol suite. This includes both "classic" end-user applications such as the ones described above, as well as support protocols that behave as applications when they send messages. Examples of the latter would include a network management protocol like SNMP, or even the routing protocol BGP (which sends messages using TCP like an application does).

So, a typical TCP/IP host has multiple processes each needing to send and receive datagrams. All of them, however, must be sent using the same interface to the internetwork, using the IP layer. This means that the data from all applications (with some possible exceptions) is "funneled down", initially to the transport layer, where it is handled by either TCP or UDP. From there, messages pass to the device's IP layer, where they are packaged in IP datagrams and sent out over the internetwork to different destinations. The technical term for this is *multiplexing*. This term simply means combining, and its use here is a software analog to the way it is done with signals.

A complementary mechanism is responsible for receipt of datagrams. At the same time that the IP layer multiplexes datagrams from many application processes to be sent out, it receives many datagrams that are intended for different processes. The IP layer must take this stream of unrelated datagrams, and eventually pass them to the correct process (through the transport layer protocol above it). This is the reverse of multiplexing: *demultiplexing*. You can see an illustration of the basic concept behind TCP/IP process multiplexing and demultiplexing in Figure 12.6

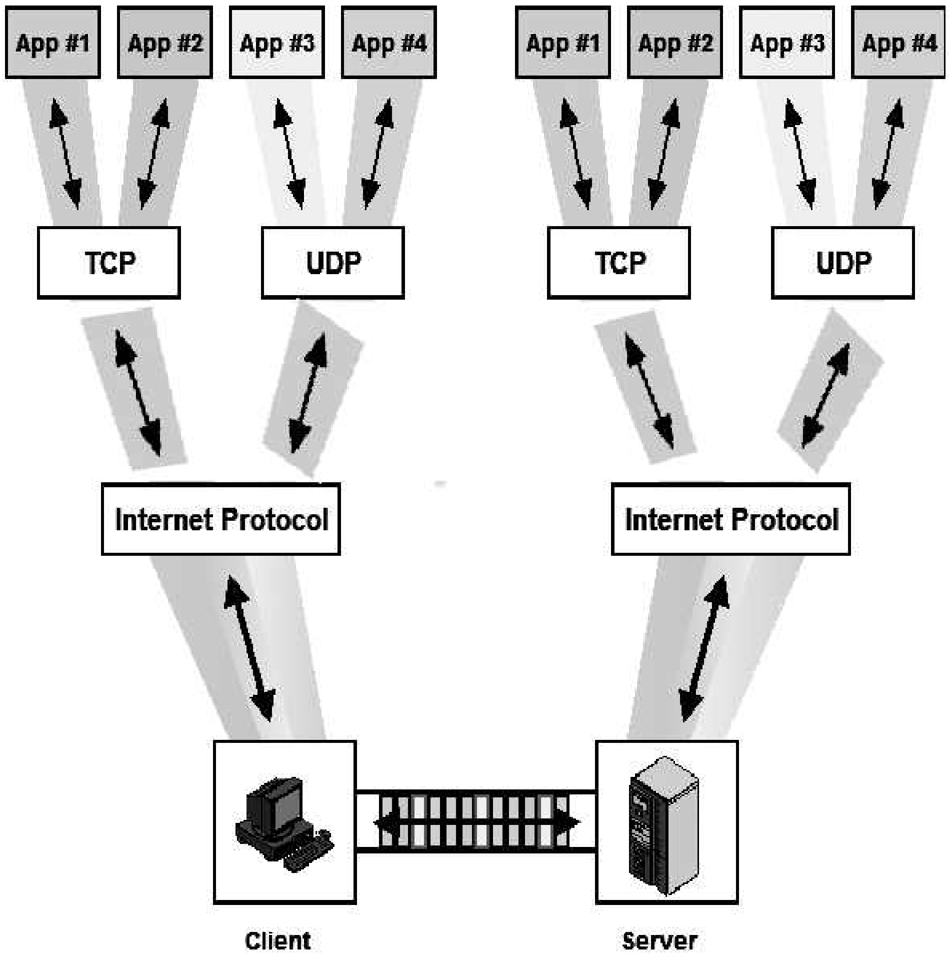


Figure 12.6- Process Multiplexing and Demultiplexing In TCP/IP

In a typical machine running TCP/IP there are many different protocols and applications running simultaneously. This example shows four different applications communicating between a client and server machine. All four are multiplexed for transmission using the same IP software and physical connection; received data is demultiplexed and passed to the appropriate application. IP, TCP and UDP provide the means of keeping distinct the data from each application.

***Key Concept:** TCP/IP is designed to allow many different applications to send and receive data simultaneously using the same Internet Protocol software on a given device. To accomplish this it is necessary to multiplex transmitted data from many sources as it is passed down to the IP layer. As a stream of IP datagrams is received, it is demultiplexed and the appropriate data passed to each application software instance on the receiving host.*

UDP multiplexing and demultiplexing

UDP is an extremely lightweight transport layer protocol on top of IP. Unlike TCP, it does not offer reliable message delivery and it does not guarantee that messages will be received in the order that they were sent.

An incoming frame (e.g., ethernet packet) contains a protocol identifier that identifies the payload (the data part of the frame) as IP data. When that payload is passed to the IP layer, a field in the header of the IP datagram identifies the higher layer protocol as UDP. The UDP layer reads the destination port field in the UDP header and delivers the segment to the socket that is associated with that port number. The kernel maintains a hash table of socket structures that is indexed by a key that is created from the UDP destination port. With UDP, any segments addressed to a specific port number will be delivered to the socket that is identified with that port. We will see that this is different from TCP, which takes performs full demultiplexing based on the source *and* destination.

While UDP does not have the reliability and in-order delivery advantages of TCP (or, as we shall see, rate adjustment to deal with congestion), there are several reasons that make it attractive for certain applications:

- Segments are sent immediately. When a user writes data to a UDP socket, it immediately goes down the layers of the network stack and is transmitted onto the network. TCP may wait for an acknowledgement or for sufficient data in it's transmit buffer instead of transmitting immediately.
- Message boundaries are preserved. TCP treats a communication stream as a sequence of bytes. The number of writes to a socket does not necessarily correspond to the number of messages that will be received at the other end.
- No connection setup overhead. With UDP, the first segment that is sent on the network can contain application data. With TCP, we first need to establish a connection with a three-way handshake, which requires an overhead of sending a segment and receiving an acknowledgement before we can send a segment with data.
- UDP is stateless. The kernel has to keep track of sockets, of course, but does there is no need to keep track of sequence numbers, buffer for out-of-order data, acknowledgements, etc. This uses less kernel memory and makes error recovery and load balancing easier: requests can be redirected to other hosts spontaneously.

- Smaller headers. UDP has an eight byte header compared to TCP's 20-byte header. This leads to smaller packets on the network.

UDP header and checksum

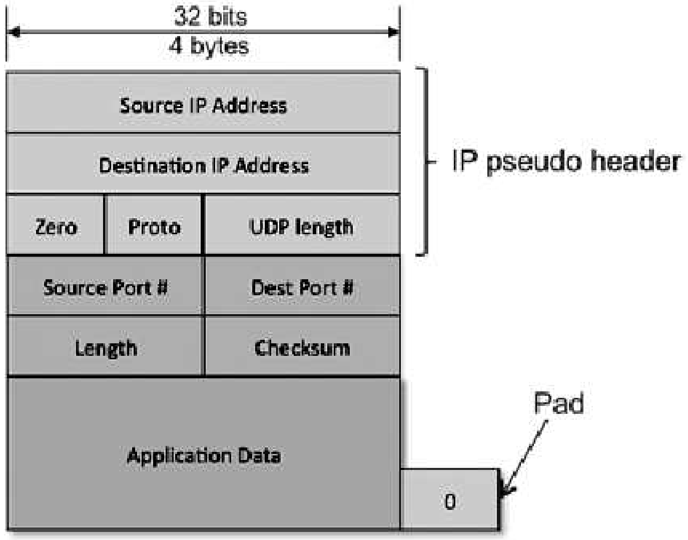


Figure 12.7: UDP header with IP pseudo header

The UDP header (Figure 12.7) is eight bytes long. It contains the source and destination ports, segment length and a checksum. The checksum is a simple error-detecting code that allows the UDP layer to check for segment corruption. If the received segment contains an error, it is dropped and not delivered to the socket. The checksum is computed over the UDP header, application data, and a pseudo IP header. The pseudo IP header contains a subset of fields from the IP header (source address, destination address, transport protocol ID, and UDP segment length). It is included in the checksum computation to ensure that the UDP layer will not get any misrouted segments (this is a safeguard but the IP header has its own checksum, which is computed in the same way).
The checksum is a 16-bit value. If the data being summed does not contain an even number of bytes (i.e., it does not have an integral multiple of 16-bit values), it is padded with a zero byte. The same ones' complement algorithm is used to compute checksums for IP headers, UDP headers, and TCP headers. The value of the checksum field is set to zero during the computation. To compute the checksum, all 16-bit chunks of data are added together. Each time the addition of two numbers results in an overflow, a one is added to the result. Finally, the bits of the final result are inverted.

To validate the checksum, the receiver performs the same arithmetic, generating a checksum for the segment and pseudo IP header. Since the segment checksum is included in the header for this computation, the result for an error-free packet will be all ones (0xffff). This computation reliably detects single bit errors in the segment.

12.5 Crash Recovery

If hosts and routers are subject to crashes, recovery from these crashes becomes an issue. If the transport entity is entirely within the hosts, recovery from network and router crashes is straightforward. If the network layer provides datagram service, the transport entities expect lost TPDUs all the time and know how to cope with them.

If the network layer provides connection-oriented service, then loss of a virtual circuit is handled by establishing a new one and then probing the remote transport entity to ask it which TPDUs it has received and which ones it has not received. In particular, it may be desirable for clients to be able to continue working when servers crash and then quickly reboot. No matter how the client and server are programmed, there are always situations where the protocol fails to recover properly. The server can be programmed in one of two ways: acknowledge first or write first. The client can be programmed in one of four ways: always retransmit the last TPDU, never retransmit the last TPDU, retransmit only in state SO, or retransmit only in state Si. This gives eight combinations, but as we shall see, for each combination there is some set of events that makes the protocol fail. Three events are possible at the server: sending an acknowledgement writing to the output process (W), and crashing (C).

	Strategy used by receiving host					
	First ACK, then write			First write, then ACK		
Strategy used by sending host	AC(W)	AWC	C(AW)	C(WA)	W AC	WC(A)
Always retransmit	OK	DUP	OK	OK	DUP	DUP
Never retransmit	LOST	OK	LOST	LOST	OK	OK
Retransmit in S0	OK	DUP	LOST	LOST	DUP	OK
Retransmit in S1	LOST	OK	OK	OK	OK	DUP

OK = Protocol functions correctly
DUP = Protocol generates a duplicate message
LOST = Protocol loses a message

Figure 12.8- Different combinations of client and server strategy.

The three events can occur in six different orderings: AC(W), AWC, C(AW), C(WA), WAC, and WC(A), where the parentheses are used to indicate that neither A nor W can follow C (i.e., once it has crashed, it has crashed).

Figure 4.8 shows all eight combinations of client and server strategy and the valid event sequences for each one.

Notice that for each strategy there is some sequence of events that causes the protocol to fail. For example, if the client always retransmits, the AWC event will generate an undetected duplicate, even though the other two events work properly.

Check your progress

Q1. Define flow control in transport layer.

Q2. Explain multiplexing and demultiplexing.

12.6 Summary

In this unit you have learnt about Connection Establishment and Releases, Use of Timers, Flow Control and Buffering, Multiplexing and Crash Recovery.

- ❖ TCP communication works in Server/Client model. The client initiates the connection and the server either accepts or rejects it. Three-way handshaking is used for connection management.
- ❖ Client initiates the connection and sends the segment with a Sequence number. Server acknowledges it back with its own Sequence number and ACK of client's segment which is one more than client's Sequence number. Client after receiving ACK of its segment sends an acknowledgement of Server's response.
- ❖ Either of server and client can send TCP segment with FIN flag set to 1. When the receiving end responds it back by Acknowledging FIN, that direction of TCP communication is closed and connection is released.
- ❖ Flow control is a function for the control of the data flow within an OSI layer or between adjacent layers. In other words it limits the amount of data transmitted by the sending transport entity to a level, or rate, that the receiver can manage.
- ❖ Flow control is a good example of a protocol function that must be implemented in *several layers* of the OSI architecture model. At the transport level flow control will allow the transport protocol entity in a host to restrict the flow of data over a logical connection from the transport protocol entity in another host. However, one of the services of the network level is to prevent congestion. Thus the network level also uses flow control to restrict the flow of network protocol data units (NPDUs).
- ❖ The technique to combine two or more data streams in one session is called Multiplexing. When a TCP client initializes a connection with Server, it always refers to a well-defined port number which indicates the application process. The client itself uses a randomly generated port number from private port number pools.
- ❖ TCP is very reliable protocol. It provides sequence number to each of byte sent in segment. It provides the feedback mechanism i.e. when a host receives a packet, it is bound to ACK that packet

having the next sequence number expected (if it is not the last segment).

- When a TCP Server crashes mid-way communication and re-starts its process it sends TPDU broadcast to all its hosts. The hosts can then send the last data segment which was never unacknowledged and carry onwards.

12.7 Review Questions

Q1. Explain connection establish and connection release in transport layer.

Q2. Define types of time in transport layer.

Q3. Write about flow control and buffering mechanism in transport layer.

Q4. Define multiplexing with suitable diagram.

Q5. Describe crash recovery in transport layer.

BIBLIOGRAPHY

Behrouz A. Forouzan. *Data Communications and Networking, Fourth Edition.* McGraw-Hill, 2007.

Larry L. Peterson and Bruce S. Davie. *Computer Networks: A Systems Approach, 4th Edition.* Morgan Kaufmann, 2007.

William Stallings. *Data and Computer Communications, Eighth Edition.* Prentice Hall, 2006.

Andrew S. Tanenbaum. *Computer Networks, Fourth Edition.* Prentice Hall PTR, 2003.

Alan Dennis. *Networking in the Internet Age.* John Wiley & Sons, 2002.

Charles E. Perkins, editor. *Ad Hoc Networking.* Addison-Wesley, 2001.

Radia Perlman. *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols, Second Edition.* Addison-Wesley, 2000.

Dimitri Bertsekas and Robert Gallager. *Data Networks, Second Edition.* Prentice Hall, 1992.

Douglas Comer and David Stevens. *Internetworking with TCP/IP, Volume 3: Client-Server Programming and Applications, Windows Sockets Version.* Prentice-Hall, 1997.

Douglas Comer and David Stevens. *Internetworking with TCP/IP, Volume 3: Client-Server Programming and Applications, BSD Sockets Version, Second Edition.* Prentice-Hall, 1996.

Sidnie Feit. *TCP/IP: Architecture, Protocols, and Implementation.* McGraw-Hill, 1993.

Kevin D. Mitnick and William L. Simon. *The Art of Deception.* Wiley Publishing, Inc., 2002.

Ted G. Lewis. *Network Science: Theory and Applications.* John Wiley & Sons, 2011.

Peter M. Higgins. *Nets, Puzzles, and Postmen: An Exploration of Mathematical Connections.* Oxford University Press, 2007.

[Schwartz 1980] M. Schwartz, Information, Transmission, Modulation, and Noise, McGraw Hill, NY, NY 1980.

[Shacham 1990] N. Shacham, P. McKenney, "Packet Recovery in High-Speed Networks Using Coding and Buffer Management", Proc. IEEE Infocom Conference, (San Francisco, 1990), pp. 124-131.

[Spragins 1991] J. D. Spragins, Telecommunications protocols and design, Addison-Wesley, Reading MA, 1991.

[Strayer 1992] W.T. Strayer, B. Dempsey, A.Weaver, XTP: The Xpress Transfer Protocol, Addison Wesley, Reading MA, 1992.

Uttar Pradesh Rajarshi Tandon
Open University

Master in Computer Science

MCS-108

Data Communication and Computer Networks

Block

4

UPPER LAYER PROTOCOLS

RIL-089

© UPRTOU, Prayagraj. 2019
ISBN : 978-93-83328-25-3

RIL-089

Block-IV Upper Layers Protocols

This is the fourth and last block on Data communication & Networks and having detail description of upper layer protocols of OSI model i.e. session layer, presentation layer and application layer. In the Open Systems Interconnection (OSI) communications model, the **Session layer** (sometimes called the "port layer") manages the setting up and taking down of the association between two communicating end points that is called a *connection*. A connection is maintained while the two end points are communicating back and forth in a conversation or *session* of some duration. Some connections and sessions last only long enough to send a message in one direction. However, other sessions may last longer, usually with one or both of the communicating parties able to terminate it. If we talk about **presentation layer,** it says in the Open Systems Interconnection (OSI) communications model, the presentation layer ensures that the communications passing through are in the appropriate form for the recipient. For example, a presentation layer program may format a file transfer request in binary code to ensure a successful file transfer. Finally the **application layer** is the seventh layer of the OSI model and the only one that directly interacts with the end user. The application layer provides many services, including Simple Mail Transfer Protocol, File transfer, Web surfing, Web chat, Email clients, Network data sharing, Virtual terminals, various file and data operations. The application layer provides full end-user access to a variety of shared network services for efficient OSI model data flow. This layer has many responsibilities, including error handling and recovery, data flow over a network and full network flow. It is also used to develop network-based applications. More than 15 protocols are used in the application layer, including File Transfer Protocol, Telnet, Trivial File Transfer Protocol and Simple Network Management Protocol.

So we will begin the first unit on routing algorithm. In this unit we focused on Optimality Principle of routing algorithm, Shortest Path Routing- Dijkstra, Bellman-Ford and Floyd-Warshall Algorithm.

Second unit begins with Elements of TCP/IP Protocol. In this unit we described User Datagram Protocol, Connection Management, and Finite State Machine.

Third unit begins with session layer protocols. In this unit you will learn about Dialog Management, Synchronization, and OSI Session Primitives Connection Establishment.

In the last, the fourth unit introduced Presentation and Application Layer Protocols. In this unit we have defined Presentation Concepts NMP-Abstract Syntax Notation-I (ANSI-1), Structure of Management, and Management Information Base.

As you study the material, you will find that figures, tables are properly used and these will help to understand the concept. There are many sections in the units to easily understand the topic. Every unit has summary and review questions in the end of the unit which will help you to review yourself.

In your study, you will find that every unit has different length and your study time will vary for each unit.

We hope you enjoy studying the material and once again wish you all the best for your success.

UNIT-XIII
Routing Algorithm

Structure

13.0 Introduction

This is the first unit of this block. This unit focus on routing algorithm. As you have studied earlier that Routing is the process of selecting best paths in a network. In the past, the term routing also meant forwarding network traffic among networks. However, that latter function is better described as forwarding. If we talk about routing algorithm, it is a formula that is stored in the router's memory. The routing algorithm your protocol uses is a major factor in the performance of your routing environment. The purpose of the routing algorithm is to make decisions for the router concerning the best paths for data. In this unit there are four sections. In Sec. 13.1 you will learn about Optimality Principle. Next section described Shortest Path Routing like Dijkstra algorithm, Bellman-Ford algorithm, and Floyd-Warshall Algorithm. In Sec. 13.3 and 13.4 you will find summary and review questions respectively.

Objectives

After studying this unit you should be able to:

- Define routing and it's Optimality Principle.
- Describe Shortest Path Routing.
- Express Dijkstra algorithm, Bellman-Ford algorithm and Floyd-Warshall Algorithm

13.1 Optimality Principle

Routing

In internetworking, the process of moving a packet of data from source to destination is called routing. Routing is usually performed by a dedicated device called a router. Routing is a key feature of the Internet because it enables messages to pass from one computer to another and eventually reach the target machine. Each intermediary computer performs routing by passing along the message to the next computer. Part of this process involves analyzing a *routing table* to determine the best path. Routing is often confused with *bridging*, which performs a similar function. The principal difference between the two is that bridging occurs at a lower level and is therefore more of a hardware function whereas routing occurs at a higher level where the software component is more important. And because routing occurs at a higher level, it can perform more complex analysis to determine the optimal path for the packet.

Routing Algorithm

Routing algorithm is stored in the router's memory. The routing algorithm is a major factor in the performance of your routing environment. The purpose of the routing algorithm is to make decisions for the router concerning the best paths for data. The router uses the routing algorithm to compute the path that would best serve to transport the data from the source to the destination. Note that you do not directly choose the algorithm that your router uses. Rather, the routing protocol you choose for your network determines which algorithm you will use. For example, whereas the routing protocol Routing Information Protocol (RIP) may use one type of routing algorithm to help the router move data, the routing protocol Open Shortest Path First (OSPF) uses another. The routing algorithm cannot be changed. The only way to change it is to change routing protocols. The overall performance of your network depends mainly on the routing algorithm, so you should research the algorithms each protocol uses before deciding which to implement on your network.

Optimality Principle

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

- ❖ It states that if router J is on the optimal path from router I to router K, then the optimal path from J to K also falls along the same route. To see this, call the part of the route from I to J, r1 and the rest of the route r2.

- ❖ If a route better than r2 existed from J to K, it could be concatenated with r1 to improve the route from I to K, contradicting our statement that r1r2 is optimal. As a direct consequence of the optimality principle, we can see that the set of optimal routes from all sources to a given destination form a tree rooted at the destination.

- ❖ Such a tree is called a sink tree and is illustrated in Fig. 1.1, where the distance metric is the number of hops. Note that a sink tree is not necessarily unique; other trees with the same path lengths may exist. The goal of all routing algorithms is to discover and use the sink trees for all routers.

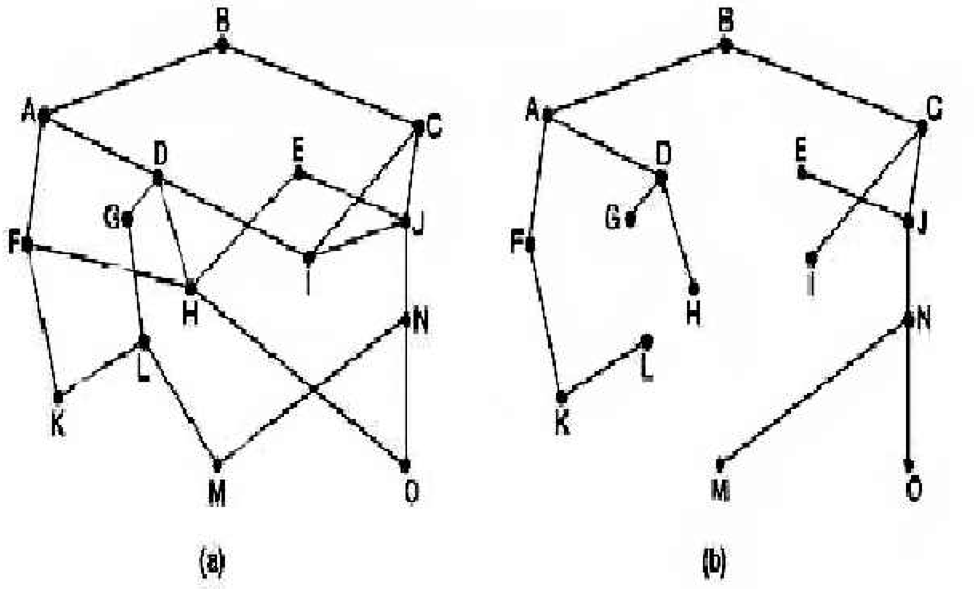


Figure 13.1- ***(a) A subnet. (b) A sink tree for router*** **B.**

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

13.2 Shortest Path Routing

Let us begin our study of feasible routing algorithms with a technique that is widely used in many forms because it is simple and easy to understand. The idea is to build a graph of the subnet, with each node of the graph representing a router and each arc of the graph representing a communication line (often called a link).

13.2.1 Dijkstra algorithm

Algorithm

Let the node at which we are starting be called the initial node. Let the distance of node Y be the distance from the initial node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.

1. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes.
2. Mark all nodes unvisited. Set the initial node as current. Create a set of the unvisited nodes called the unvisited set consisting of all the nodes.
3. For the current node, consider all of its unvisited neighbors and calculate their tentative distances. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B (through A) will be 6 + 2 = 8. If this distance is less than the previously recorded tentative distance of B, then overwrite that distance. Even though a neighbor has been examined, it is not marked as "visited" at this time, and it remains in the unvisited set.
4. When we are done considering all of the neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.
5. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.

6. Select the unvisited node that is marked with the smallest tentative distance, and set it as the new "current node" then go back to step 3.

Description

Note: For ease of understanding, this discussion uses the terms intersection, road and map — however, formally these terms are vertex, edge and graph, respectively.

Suppose you would like to find the shortest path between two intersections on a city map, a starting point and a destination. The order is conceptually simple: to start, mark the distance to every intersection on the map with infinity. This is done not to imply there is an infinite distance, but to note that that intersection has not yet been visited; some variants of this method simply leave the intersection unlabeled. Now, at each iteration, select a current intersection. For the first iteration the current intersection will be the starting point and the distance to it (the intersection's label) will be zero. For subsequent iterations (after the first) the current intersection will be the closest unvisited intersection to the starting point—this will be easy to find. From the current intersection, update the distance to every unvisited intersection that is directly connected to it. This is done by determining the sum of the distance between an unvisited intersection and the value of the current intersection, and relabeling the unvisited intersection with this value if it is less than its current value. In effect, the intersection is relabeled if the path to it through the current intersection is shorter than the previously known paths. To facilitate shortest path identification, in pencil, mark the road with an arrow pointing to the relabeled intersection if you label/re-label it, and erase all others pointing to it. After you have updated the distances to each neighboring intersection, mark the current intersection as visited and select the unvisited intersection with lowest distance (from the starting point) – or lowest label—as the current intersection. Nodes marked as visited are labeled with the shortest path from the starting point to it and will not be revisited or returned to.

Continue this process of updating the neighboring intersections with the shortest distances, then marking the current intersection as visited and moving onto the closest unvisited intersection until you have marked the destination as visited. Once you have marked the destination as visited (as is the case with any visited intersection) you have determined the shortest path to it, from the starting point, and can trace your way back, following the arrows in reverse.

Of note is the fact that this algorithm makes no attempt to direct "exploration" towards the destination as one might expect. Rather, the sole consideration in determining the next "current" intersection is its distance from the starting point. This algorithm therefore "expands outward" from the starting point, iteratively considering every node that is closer in terms of shortest path distance until it reaches the destination. When understood in this way, it is clear how the algorithm necessarily finds the shortest path, however it may also reveal one of the algorithm's weaknesses: its relative slowness in some topologies.

Example

To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph. The concept of a **shortest path** deserves some explanation. One way of measuring path length is the number of hops.

Using this metric, the paths *ABC* and *ABE* in Fig. 13.2 are equally long. Another metric is the geographic distance in kilometers, in which case *ABC* is clearly much longer than *ABE*.

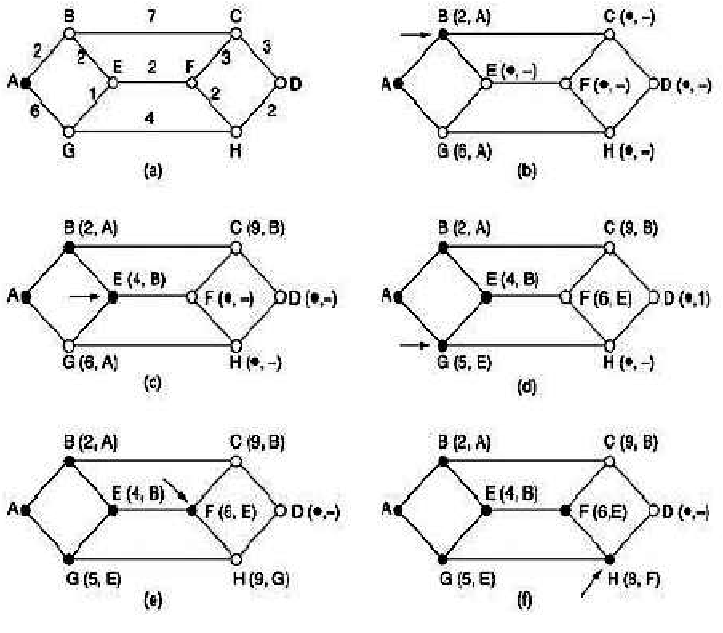


Figure 13.2: ***The first five steps used in computing the shortest path from A to D. The arrows indicate the working node***

However, many other metrics besides hops and physical distance are also possible. For example, each arc could be labeled with the mean queuing and transmission delay for some standard test packet as determined by hourly test runs. With this graph labeling, the shortest path is the fastest path rather than the path with the fewest arcs or kilometers. In the general case, the labels on the arcs could be computed as a function of the distance, bandwidth, average traffic, communication cost, mean queue length, measured delay, and other factors. By changing the weighting function, the algorithm would then compute the "shortest" path measured according to any one of a number of criteria or to a combination of criteria. Several algorithms for computing the shortest path between two nodes of a graph are known.

This one is due to Dijkstra (1959). Each node is labeled (in parentheses) with its distance from the source node along the best known path. Initially, no paths are known, so all nodes are labeled with infinity.

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

- We now start at *B* and examine all nodes adjacent to it. If the sum of the label on *B* and the distance from *B* to the node being considered is less than the label on that node, we have a shorter path, so the node is relabeled.
- After all the nodes adjacent to the working node have been inspected and the tentative labels changed if possible, the entire graph is searched for the tentatively-labeled node with the smallest value.
- This node is made permanent and becomes the working node for the next round. Figure 13.2 shows the first five steps of the algorithm. This algorithm is given in Fig. 13.3.
- The global variables *n* and *dist* describe the graph and are initialized before *shortest_path* is called.
- The only difference between the program and the algorithm described above is that in Fig. 13.3, we compute the shortest path starting at the terminal node, *t*, rather than at the source node, *s*.
- Since the shortest path from *t* to *s* in an undirected graph is the same as the shortest path from *s* to *t*, it does not matter at which end we begin. The reason for searching backward is that each node is labeled with its predecessor rather than its successor.
- When the final path is copied into the output variable, *path*, the path is thus reversed. By reversing the search, the two effects cancel, and the answer is produced in the correct order.

```
#define MAX_NODES 1024                /* maximum number of nodes */
#define INFINITY 1000000000           /* a number larger than every maximum path */
int n, dist[MAX_NODES][MAX_NODES];/* dist[i][j] is the distance from i to j */

void shortest_path(int s, int t, int path[])
{ struct state {                      /* the path being worked on */
     int predecessor;                 /* previous node */
     int length;                      /* length from source to this node */
     enum {permanent, tentative} label; /* label state */
 } state[MAX_NODES];

 int i, k, min;
 struct state *p;

 for (p = &state[0]; p < &state[n]; p++) { /* initialize state */
     p->predecessor = -1;
     p->length = INFINITY;
     p->label = tentative;
 }
 state[t].length = 0; state[t].label = permanent;
 k = t;                               /* k is the initial working node */
 do {                                 /* Is there a better path from k? */
     for (i = 0; i < n; i++)          /* this graph has n nodes */
          if (dist[k][i] != 0 && state[i].label == tentative) {
               if (state[k].length + dist[k][i] < state[i].length) {
                    state[i].predecessor = k;
                    state[i].length = state[k].length + dist[k][i];
               }
          }

     /* Find the tentatively labeled node with the smallest label. */
     k = 0; min = INFINITY;
     for (i = 0; i < n; i++)
          if (state[i].label == tentative && state[i].length < min) {
               min = state[i].length;
               k = i;
          }
     state[k].label = permanent;
 } while (k != s);

 /* Copy the path into the output array. */
 i = 0; k = s;
 do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

Figure 13.3: Dijkstra's algorithm to compute the shortest path through a graph.

How to implement the above algorithm?

We use a boolean array sptSet[] to represent the set of vertices included in SPT. If a value sptSet[v] is true, then vertex v is included in SPT, otherwise not. Array dist[] is used to store shortest distance values of all vertices.

```
// A C / C++ program for Dijkstra's single source shortest path algorithm.

// The program is for adjacency matrix representation of the graph

 #include <stdio.h>

#include <limits.h>

 // Number of vertices in the graph

#define V 9

 // A utility function to find the vertex with minimum distance value, from

// the set of vertices not yet included in shortest path tree
```

```
int minDistance(int dist[], bool sptSet[])
{
  // Initialize min value
  int min = INT_MAX, min_index;

  for (int v = 0; v < V; v++)
   if (sptSet[v] == false && dist[v] <= min)
       min = dist[v], min_index = v;
   return min_index;
}
// A utility function to print the constructed distance array
int printSolution(int dist[], int n)
{
  printf("Vertex   Distance from Source\n");
  for (int i = 0; i < V; i++)
     printf("%d \t\t %d\n", i, dist[i]);
}
// Funtion that implements Dijkstra's single source shortest path algorithm
// for a graph represented using adjacency matrix representation
void dijkstra(int graph[V][V], int src)
{
    int dist[V];     // The output array.  dist[i] will hold the shortest
                  // distance from src to i
     bool sptSet[V]; // sptSet[i] will true if vertex i is included in shortest
                 // path tree or shortest distance from src to i is finalized
    // Initialize all distances as INFINITE and stpSet[] as false
    for (int i = 0; i < V; i++)
       dist[i] = INT_MAX, sptSet[i] = false;
     // Distance of source vertex from itself is always 0
    dist[src] = 0;
     // Find shortest path for all vertices
   for (int count = 0; count < V-1; count++)
   {
    // Pick the minimum distance vertex from the set of vertices not
    // yet processed. u is always equal to src in first iteration.
    int u = minDistance(dist, sptSet);
     // Mark the picked vertex as processed
    sptSet[u] = true;
     // Update dist value of the adjacent vertices of the picked vertex.
    for (int v = 0; v < V; v++)
       // Update dist[v] only if is not in sptSet, there is an edge from
      // u to v, and total weight of path from src to  v through u is
      // smaller than current value of dist[v]
      if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX
                            && dist[u]+graph[u][v] < dist[v])
         dist[v] = dist[u] + graph[u][v];
   }
    // print the constructed distance array
   printSolution(dist, V);
}
// driver program to test above function
int main()
{
  /* Let us create the example graph discussed above */
  int graph[V][V] = {{0, 4, 0, 0, 0, 0, 0, 8, 0},
                {4, 0, 8, 0, 0, 0, 0, 11, 0},
                {0, 8, 0, 7, 0, 4, 0, 0, 2},
                {0, 0, 7, 0, 9, 14, 0, 0, 0},
                {0, 0, 0, 9, 0, 10, 0, 0, 0},
                {0, 0, 4, 0, 10, 0, 2, 0, 0},
                {0, 0, 0, 14, 0, 2, 0, 1, 6},
                {8, 11, 0, 0, 0, 0, 1, 0, 7},
                {0, 0, 2, 0, 0, 0, 6, 7, 0}
               };
   dijkstra(graph, 0);
```

```
    return 0;
}
```

Output:

```
Vertex  Distance from Source
0        0
1        4
2        12
3        19
4        21
5        11
6        9
7        8
8        14
```

13.2.2 Bellman-Ford algorithm

The Bellman–Ford algorithm is an algorithm that computes shortest paths from a single source vertex to all of the other vertices in a weighted digraph. It is slower than Dijkstra's algorithm for the same problem, but more versatile, as it is capable of handling graphs in which some of the edge weights are negative numbers. The algorithm is usually named after two of its developers, Richard Bellman and Lester Ford, Jr., who published it in 1958 and 1956, respectively; however, Edward F. Moore also published the same algorithm in 1957, and for this reason it is also sometimes called the Bellman–Ford–Moore algorithm. Negative edge weights are found in various applications of graphs, hence the usefulness of this algorithm. If a graph contains a "negative cycle", i.e., a cycle whose edges sum to a negative value, then there is no cheapest path, because any path can be made cheaper by one more walk through the negative cycle. In such a case, the Bellman–Ford algorithm can detect negative cycles and report their existence, but it cannot produce a correct "shortest path" answer if a negative cycle is reachable from the source.

Algorithm

Like Dijkstra's Algorithm, Bellman–Ford is based on the principle of relaxation, in which an approximation to the correct distance is gradually replaced by more accurate values until eventually reaching the optimum solution. In both algorithms, the approximate distance to each vertex is always an overestimate of the true distance, and is replaced by the minimum of its old value with the length of a newly found path. However, Dijkstra's algorithm greedily selects the minimum-weight node that has not yet been processed, and performs this relaxation process on all of its outgoing edges; in contrast, the Bellman–Ford algorithm simply relaxes all the edges, and does this |V | − 1 times, where |V | is the number of vertices in the graph. In each of these repetitions, the number of vertices with correctly calculated distances grows, from which it follows that eventually all vertices will have their correct distances. This method allows the Bellman–Ford algorithm to be applied to a wider class of inputs than Dijkstra. Bellman–Ford runs in O (|V|·|E|) time, where |V| and |E| are the number of vertices and edges respectively.

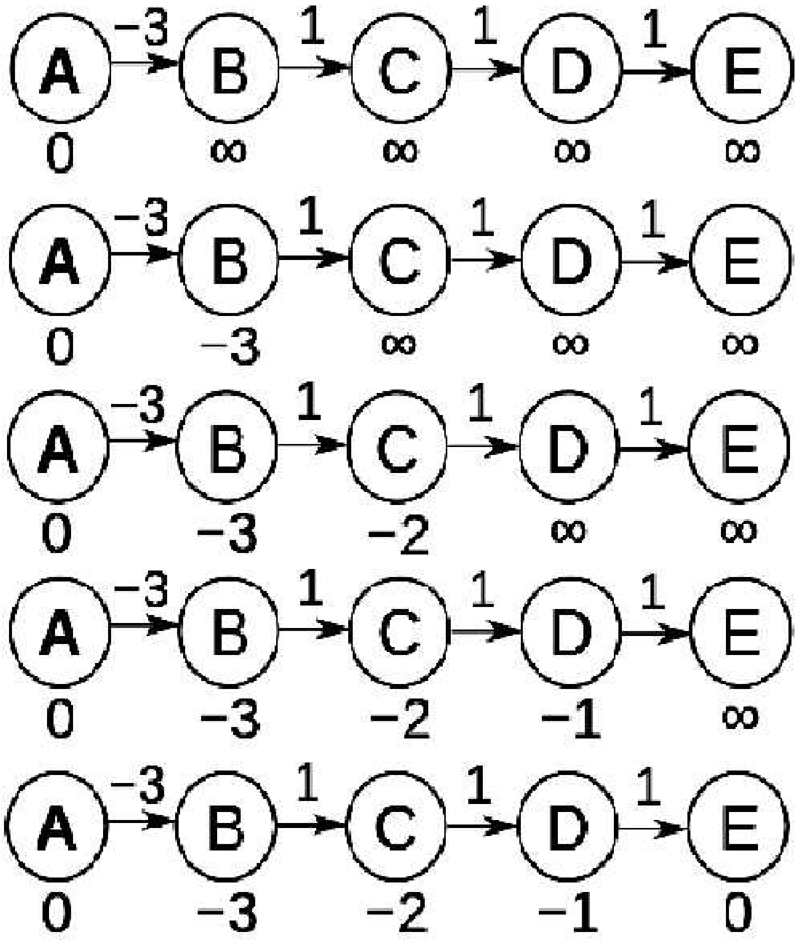


Figure 13.4: Bellman-Ford Algorithm

Note: In this example graph, assuming that A is the source and edges are processed in the worst order, from right to left, it requires the full |V|−1 or 4 iterations for the distance estimates to converge. Conversely, if the

edges are processed in the best order, from left to right, the algorithm converges in a single iteration

Both, the Bellman-Ford algorithm and Dijkstra's algorithm are used to calculate 'metrics' (distance/cost of traversing a link) in routing protocols. Both of them consider only hop count (the number of machines between the source of the message and the destination) as the metric between two nodes. Other factors such as bandwidth and delay can also be used to calculate the metric, but they are used by other complex protocols. The difference between these two algorithms lies in the **type of protocols which use the respective algorithms.**

There are two main types of routing protocols: **distance vector routing (DVR)** protocols and **link state routing (LSR)** protocols. It is assumed that each node knows the cost to reach each of its neighbors.

We have given a detailed explanation here. If you understand the basics, then there is a summarized answer at the end.

The Bellman-Ford algorithm is used by DVR protocols like RIP and RIPv2. In a distance vector routing protocol each router on the network, on which the protocol is running, prepares routing update packets. The information in each routing update packet includes the **list of all the nodes in the network and the corresponding metric costs. This packet is forwarded to all the neighbors of the router.** Similarly the router receives an update from each neighbor and performs the required updates in its routing table. The distances are calculated using the Bellman-Ford algorithm

On the other hand, Dijkstra's algorithm is used by LSR protocols like OSPF. LSR protocols are different from the DVR protocols as **routers implementing these protocols store the entire topology of the network** in their memory. There are 2 stages in building a routing table in LSR protocols. First, a map of the entire network should be stored in every router, and then the shortest distance to each node must be calculated by each router. Even here, the routing updates are prepared periodically and when the topology changes. But here, the update, called a **link state packet, is flooded (LSP) throughout the network,** unlike in the above case, where it is sent only to the neighbors. Each LSP contains an ID for the source, a Sequence Number (to distinguish newer packets from older packets) and the distances from the sender to each of its neighbors. When each router has collected LSPs from each router, it starts creating the routing table including the shortest paths to each node, using Dijkstra's algorithm.

The Bellman-Ford algorithm also has some additional disadvantages. It is not recommended for use in large networks. Also it suffers from something known as the count to infinity problem which we have explained below.

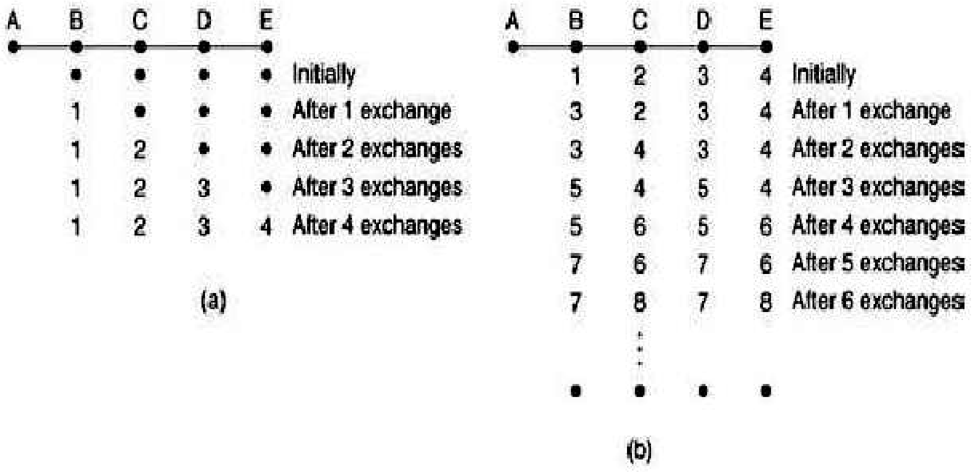


Figure 13.5: Bellman-Ford algorithm

Suppose all nodes have just started up as shown in figure 13.5. Initially no routes exist as shown by the dots. The figure only explains the routes from A. A receives a routing update from B and A learns that it is just 1 hop away so an entry is made in A's routing table. B learns a route to C in this same turn from C's update. It includes this in its next update to A, and hence A learns that C is 1 hop away from B. So the update is made in A that C is 2 hops away (A to B + B to C). Similarly, A learns that D is 2 hops from B and E is 3 hops, from subsequent updates. Thus A learns the routes to each node.

Now assume that A shuts down. The A-B link goes dead. The metric has become infinity. In this situation when B gets an update from C, it notices that there is a path to A via C and hence updates its routing table (B to C + C to A). This change is reflected in B's next update. C updates its link to A (A to B + B to C) and so on. As can be seen, the metric will keep on increasing and go on till infinity, and hence the name 'Count to infinity'. The problem arises because B doesn't realize that the path to A which C advertised passes through B itself!

To resolve this, the maximum hop count is defined which introduces its first disadvantage (large networks). Techniques like split horizon and split horizon with poison reverse are also used to remove this problem

Implementation:

```
// A C / C++ program for Bellman-Ford's single source

// shortest path algorithm.

 #include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <limits.h>
```

```
 // a structure to represent a weighted edge in graph
struct Edge
{
   int src, dest, weight;
};
 // a structure to represent a connected, directed and
// weighted graph
struct Graph
{
   // V-> Number of vertices, E-> Number of edges
   int V, E;
    // graph is represented as an array of edges.
   struct Edge* edge;
};

// Creates a graph with V vertices and E edges
struct Graph* createGraph(int V, int E)
{
   struct Graph* graph =
        (struct Graph*) malloc( sizeof(struct Graph) );
   graph->V = V;
   graph->E = E;

   graph->edge =
      (struct Edge*) malloc( graph->E * sizeof( struct Edge ) );
    return graph;
}
 // A utility function used to print the solution
void printArr(int dist[], int n)
{
   printf("Vertex   Distance from Source\n");
```

```
    for (int i = 0; i < n; ++i)
        printf("%d \t\t %d\n", i, dist[i]);
}
 // The main function that finds shortest distances from src to
// all other vertices using Bellman-Ford algorithm.  The function
// also detects negative weight cycle
void BellmanFord(struct Graph* graph, int src)
{
    int V = graph->V;
    int E = graph->E;
    int dist[V];
     // Step 1: Initialize distances from src to all other vertices
    // as INFINITE
    for (int i = 0; i < V; i++)
        dist[i]   = INT_MAX;
    dist[src] = 0;
     // Step 2: Relax all edges |V| - 1 times. A simple shortest
    // path from src to any other vertex can have at-most |V| - 1
    // edges
    for (int i = 1; i <= V-1; i++)
    {
        for (int j = 0; j < E; j++)
        {
            int u = graph->edge[j].src;
            int v = graph->edge[j].dest;
            int weight = graph->edge[j].weight;
            if (dist[u] != INT_MAX && dist[u] + weight < dist[v])
                dist[v] = dist[u] + weight;
        }
    }
    // Step 3: check for negative-weight cycles.  The above step
   // guarantees shortest distances if graph doesn't contain
   // negative weight cycle.  If we get a shorter path, then there
   // is a cycle.
   for (int i = 0; i < E; i++)
   {
      int u = graph->edge[i].src;
      int v = graph->edge[i].dest;
      int weight = graph->edge[i].weight;
      if (dist[u] != INT_MAX && dist[u] + weight < dist[v])
         printf("Graph contains negative weight cycle");
   }
    printArr(dist, V);
    return;
}
 // Driver program to test above functions
int main()
{
   /* Let us create the graph given in above example */
   int V = 5;  // Number of vertices in graph
   int E = 8;  // Number of edges in graph
   struct Graph* graph = createGraph(V, E);
    // add edge 0-1 (or A-B in above figure)
   graph->edge[0].src = 0;
   graph->edge[0].dest = 1;
   graph->edge[0].weight = -1;
    // add edge 0-2 (or A-C in above figure)
   graph->edge[1].src = 0;
   graph->edge[1].dest = 2;
   graph->edge[1].weight = 4;
    // add edge 1-2 (or B-C in above figure)
   graph->edge[2].src = 1;
```

```
    graph->edge[2].dest = 2;
    graph->edge[2].weight = 3;
     // add edge 1-3 (or B-D in above figure)
    graph->edge[3].src = 1;
    graph->edge[3].dest = 3;
    graph->edge[3].weight = 2;
     // add edge 1-4 (or A-E in above figure)
    graph->edge[4].src = 1;
    graph->edge[4].dest = 4;
    graph->edge[4].weight = 2;
     // add edge 3-2 (or D-C in above figure)
    graph->edge[5].src = 3;
    graph->edge[5].dest = 2;
    graph->edge[5].weight = 5;
    // add edge 3-1 (or D-B in above figure)
    graph->edge[6].src = 3;
    graph->edge[6].dest = 1;
    graph->edge[6].weight = 1;
     // add edge 4-3 (or E-D in above figure)
    graph->edge[7].src = 4;
    graph->edge[7].dest = 3;
    graph->edge[7].weight = -3;
    BellmanFord(graph, 0);
     return 0;
}
```

Output:

```
Vertex   Distance from Source
0            0
1            -1
2            2
3            -2
4            1
```

13.2.3 Floyd-Warshall Algorithm

In computer science, the Floyd–Warshall algorithm (also known as Floyd's algorithm, Roy–Warshall algorithm, Roy–Floyd algorithm, or the WFI algorithm) is a graph analysis algorithm for finding shortest paths in a weighted graph with positive or negative edge weights (but with no negative cycles, see below) and also for finding transitive closure of a relation R. A single execution of the algorithm will find the lengths (summed weights) of the shortest paths between all pairs of vertices, though it does not return details of the paths themselves. The algorithm is an example of dynamic programming. It was published in its currently recognized form by Robert Floyd in 1962. However, it is essentially the same as algorithms previously published by Bernard Roy in 1959 and also by Stephen Warshall in 1962 for finding the transitive closure of a graph. The modern formulation of Warshall's algorithm as three nested for-loops was first described by Peter Ingerman, also in 1962.

Algorithm

The Floyd–Warshall algorithm compares all possible paths through the graph between each pair of vertices. It is able to do this with only Θ(|V| 3) comparisons in a graph. This is remarkable considering that there may be up to Ω (|V| 2) edges in the graph, and every combination of edges is tested. It does so by incrementally improving an estimate on the shortest path between two vertices, until the estimate is optimal.

Consider a graph G with vertices V numbered 1 through N. Further consider a function shortestPath(i, j, k) that returns the shortest possible path from i to j using vertices only from the set {1,2,...,k} as intermediate points along the way. Now, given this function, our goal is to find the shortest path from each i to each j using only vertices 1 to k + 1.

For each of these pairs of vertices, the true shortest path could be either (1) a path that only uses vertices in the set {1, ..., k} or (2) a path that goes from i to k + 1 and then from k + 1 to j. We know that the best path from i to j that only uses vertices 1 through k is defined by shortest Path (i, j, k), and it is clear that if there were a better path from i to k + 1 to j, then the length of this path would be the concatenation of the shortest path from i to k + 1 (using vertices in {1, ..., k}) and the shortest path from k + 1 to j (also using vertices in {1, ..., k}).

If w(i,j) is the weight of the edge between vertices i and j, we can define shortest Path(i, j, k + 1) in terms of the following recursive formula: the base case is

$$\text{Shortest Path } (i,j,0) = w(i,j)$$

And the recursive case is

$$\text{shortestPath}(i,j,k+1) = \min(\text{shortestPath}(i,j,k),\ \text{shortestPath}(i,k+1,k) + \text{shortestPath}(k+1,j,k))$$

This formula is the heart of the Floyd–Warshall algorithm. The algorithm works by first computing shortestPath(i, j, k) for all (i, j) pairs for k = 1, then k = 2, etc. This process continues until k = n, and we have found the shortest path for all (i, j) pairs using any intermediate vertices.

Example

The algorithm above is executed on the graph on the left below:

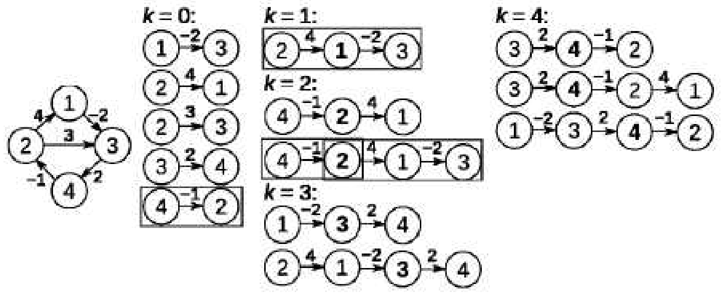


Figure 13.6: The graph

Prior to the first iteration of the outer loop, labelled k=0 above, the only known paths correspond to the single edges in the graph. At k=1, paths that go through the vertex 1 are found: in particular, the path 2→1→3 is found, replacing the path 2→3 which has fewer edges but is longer. At k=2, paths going through the vertices {1,2} are found. The red and blue boxes show how the path 4→2→1→3 is assembled from the two known paths 4→2 and 2→1→3 encountered in previous iterations, with 2 in the intersection. The path 4→2→3 is not considered, because 2→1→3 is the shortest path encountered so far from 2 to 3. At k=3, paths going through the vertices {1,2,3} are found. Finally, at k=4, all shortest paths are found.

Implementation

```
// C Program for Floyd Warshall Algorithm

#include<stdio.h>


// Number of vertices in the graph

#define V 4


/* Define Infinite as a large enough value. This value will be used

  for vertices not connected to each other */

#define INF 99999


// A function to print the solution matrix
```

```
void printSolution(int dist[][V]);

// Solves the all-pairs shortest path problem using Floyd Warshall
algorithm

void floydWarshall (int graph[][V])

{

  /* dist[][] will be the output matrix that will finally have the shortest

    distances between every pair of vertices */

  int dist[V][V], i, j, k;

   /* Initialize the solution matrix same as input graph matrix. Or

     we can say the initial values of shortest distances are based

     on shortest paths considering no intermediate vertex. */

  for (i = 0; i < V; i++)

     for (j = 0; j < V; j++)

        dist[i][j] = graph[i][j];

   /* Add all vertices one by one to the set of intermediate vertices.

    ---> Before start of a iteration, we have shortest distances between all

    pairs of vertices such that the shortest distances consider only the

    vertices in set {0, 1, 2, .. k-1} as intermediate vertices.

    ----> After the end of a iteration, vertex no. k is added to the set of

    intermediate vertices and the set becomes {0, 1, 2, .. k} */

  for (k = 0; k < V; k++)

  {

     // Pick all vertices as source one by one

     for (i = 0; i < V; i++)

     {

        // Pick all vertices as destination for the

        // above picked source

        for (j = 0; j < V; j++)

        {

           // If vertex k is on the shortest path from

           // i to j, then update the value of dist[i][j]
```

```
            if (dist[i][k] + dist[k][j] < dist[i][j])
                dist[i][j] = dist[i][k] + dist[k][j];
        }
    }
  }

  // Print the shortest distance matrix
  printSolution(dist);
}

/* A utility function to print solution */
void printSolution(int dist[][V])
{
  printf ("Following matrix shows the shortest distances"
          " between every pair of vertices \n");
  for (int i = 0; i < V; i++)
  {
     for (int j = 0; j < V; j++)
     {
        if (dist[i][j] == INF)
           printf("%7s", "INF");
        else
           printf ("%7d", dist[i][j]);
     }
     printf("\n");
  }
}

// driver program to test above function
int main()
```

```
{
  /* Let us create the following weighted graph
        10
   (0)------>(3)
    |        /|\
  5 |         |
    |         | 1
   \|/        |
   (1)------>(2)
        3        */
  int graph[V][V] = { {0,   5,  INF, 10},
                      {INF, 0,   3, INF},
                      {INF, INF, 0,   1},
                      {INF, INF, INF, 0}
                    };
   // Print the solution
  floydWarshall(graph);
  return 0;
}
```

Output:

Following matrix shows the shortest distances between every pair of vertices

```
   0     5     8    9
 INF     0     3    4
 INF   INF     0    1
 INF   INF   INF    0
```

Check your progress

Q1. What is a routing? Define.

Q2. How shortest path calculated via any shortest path algorithm?

Q3. What is the count to infinity problem?

13.3 Summary

In this unit you have learnt Optimality Principle, Shortest Path Routing, Dijkstra algorithm, Bellman-Ford algorithm, and Floyd-Warshall Algorithm.

- ❖ Optimality principle based sink tree methodology for adaptive static routing using multiple route configuration scheme is an innovative strategy for integrating dynamic routing protocol attributes seamlessly into static routing.
- ❖ **Shortest path routing** refers to the process of finding paths through a network that have a minimum of distance or other cost metric. Routing of data packets on the Internet is an example involving millions of routers in a complex, worldwide, multilevel network. Optimum routing on the Internet has a major impact on performance and cost.
- ❖ **Dijkstra's algorithm** is an **algorithm** for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. **Dijkstra** in 1956 and published three years later.
- ❖ The **Bellman–Ford algorithm** is an **algorithm** that computes shortest paths from a single source vertex to all of the other vertices in a weighted digraph.
- ❖ **Floyd-Warshall algorithm** is a procedure, which is used to find the shorthest (longest) paths among all pairs of nodes in a graph, which does not contain any cycles of negative lenght.

13.4 Review Questions

Q1. Define Optimality Principle.

Q2. How to calculate Shortest Path Routing in a network?

Q3. What do you understand by Dijkstra algorithm?

Q4. Explain Bellman-Ford algorithm.

Q5. Define Floyd-Warshall Algorithm.

UNIT-XIV

Elements of TCP/IP Protocol

Structure

14.0 Introduction

In this unit we focused on elements of TCP/IP protocols. There are six sections in this unit. In Sec. 14.1 you will learn about TCP/IP protocols. The TCP/IP suite of protocols is the set of protocols used to communicate across the internet. It is also widely used on many organizational networks due to its flexibility and wide array of functionality provided. Microsoft who had originally developed their own set of protocols now is more widely using TCP/IP, at first for transport and now to support other services. In the next section i.e. Sec. 14.2, you will know about user datagram protocol. User Datagram Protocol (UDP) is part of the Internet Protocol suite used by programs running on different computers on a network. UDP is used to send short messages called datagrams but overall, it is an unreliable, connectionless protocol. UDP is officially defined in RFC 768 and was formulated by David P. Reed. In Sec. 14.3 you will know about connection management. Sec. 14.4 describe finite state machine. In Sec. 14.5 and 14.6 you will find summary and review questions respectively.

Objectives

After studying this unit you should be able to:

- ❖ Define Elements of TCP/IP Protocol.
- ❖ Describe User Datagram Protocol.
- ❖ Express Connection Management
- ❖ Define Finite State Machine

14.1 Elements of TCP/IP Protocol

Transmission Control Protocol/Internet Protocol (TCP/IP) is an industry-standard suite of protocols that provide communications in a heterogeneous (made up of dissimilar elements) environment. In addition, TCP/IP provides a routable, enterprise networking protocol and access to the Internet and its resources. Because of its popularity, TCP/IP has become the de facto standard for what's known as *internetworking*, the intercommunication in a network that's composed of smaller networks.

TCP/IP has become the standard protocol used for interoperability among many different types of computers. This interoperability is a primary advantage of TCP/IP. Most networks support TCP/IP as a protocol. TCP/IP also supports routing and is commonly used as an internetworking protocol.

The Internet Standards Process

Because TCP/IP is the protocol of the Internet, it has evolved based on fundamental standards that have been created and adopted over more than 30 years. The future of TCP/IP is closely associated with the advances and administration of the Internet as additional standards continue to be developed. Although no one organization owns the Internet or its technologies, several organizations oversee and manage these new standards, such as the Internet Society and the Internet Architecture Board.

The Internet Society (ISOC) was created in 1992 and is a global organization responsible for the internetworking technologies and applications of the Internet. Although the society's principal purpose is to encourage the development and availability of the Internet, it is also responsible for the further development of the standards and protocols that allow the Internet to function.

The ISOC sponsors the Internet Architecture Board (IAB), a technical advisory group that sets Internet standards, publishes RFCs, and oversees the Internet standards process. The IAB governs the following bodies:

- The Internet Assigned Number Authority (IANA) oversees and coordinates the assignment of protocol identifiers used on the Internet.
- The Internet Research Task Force (IRTF) coordinates all TCP/IP-related research projects.
- The Internet Engineering Task Force (IETF) solves technical problems and needs as they arise on the Internet and develops Internet standards and protocols. IETF working groups define standards known as RFCs.

TCP/IP Terminology

The Internet standards use a specific set of terms when referring to network elements and concepts related to TCP/IP networking. These terms provide a foundation for subsequent chapters. Figure 14.1 illustrates the components of an IP network.

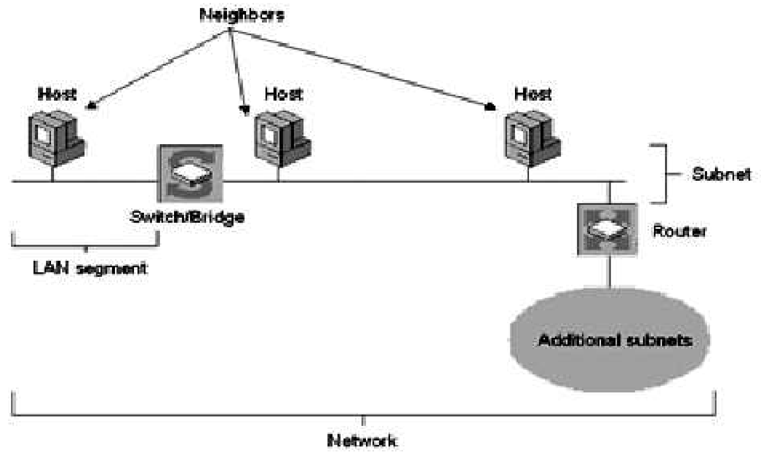


Figure 14.1 Elements of an IP network

Common terms and concepts in TCP/IP are defined as follows:

- **Node:** Any device, including routers and hosts, which runs an implementation of IP.
- **Router:** A node that can forward IP packets not explicitly addressed to itself. On an IPv6 network, a router also typically advertises its presence and host configuration information.
- **Host:** A node that cannot forward IP packets not explicitly addressed to itself (a non-router). A host is typically the source and the destination of IP traffic. A host silently discards traffic that it receives but that is not explicitly addressed to itself.
- **Upper-layer protocol:** A protocol above IP that uses IP as its transport. Examples include Internet layer protocols such as the Internet Control Message Protocol (ICMP) and Transport layer protocols such as the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). (However, Application layer protocols that use TCP and UDP as their transports are not considered upper-layer protocols. File Transfer Protocol [FTP] and Domain Name System [DNS] fall into this category).
- **LAN segment:** A portion of a subnet consisting of a single medium that is bounded by bridges or Layer 2 switches.
- **Subnet:** One or more LAN segments that are bounded by routers and use the same IP address prefix. Other terms for subnet are network segment and link.
- **Network:** Two or more subnets connected by routers. Another term for network is internetwork.

- **Neighbor:** A node connected to the same subnet as another node.
- **Interface:** The representation of a physical or logical attachment of a node to a subnet. An example of a physical interface is a network adapter. An example of a logical interface is a tunnel interface that is used to send IPv6 packets across an IPv4 network.
- **Address:** An identifier that can be used as the source or destination of IP packets and that is assigned at the Internet layer to an interface or set of interfaces.
- **Packet:** The protocol data unit (PDU) that exists at the Internet layer and comprises an IP header and payload.

TCP/IP Protocols

Since TCP/IP is a protocol suite, it is most often discussed in terms of the protocols that comprise it. Each protocol "resides" in a particular layer of the TCP/IP architectural model we saw earlier in this section. Every TCP/IP protocol is charged with performing a certain subset of the total functionality required to implement a TCP/IP network or application. They work together to allow TCP/IP as a whole to operate.

As we mentioned earlier, there are a few TCP/IP protocols that are usually called the "core" of the suite, because they are responsible for its basic operation. Which protocols to include in this category is a matter of some conjecture, but most people would definitely include here the main protocols at the internet and transport layers: the Internet Protocol (IP), Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). These core protocols support many other protocols, to perform a variety of functions at each of the TCP/IP model layers. Still others enable user applications to function.

The organization of protocols in the TCP/IP suite can also be seen at a glance in Figure 14.2.

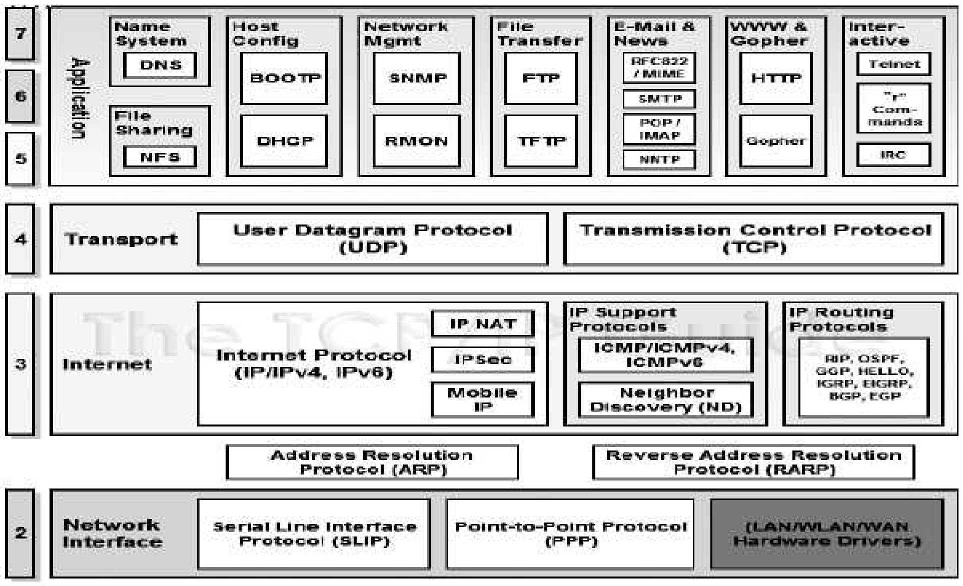


Figure 14.2: TCP/IP Protocols

Layer 1: Host-to-network Layer

1. Lowest layer of the all.
2. Protocol is used to connect to the host, so that the packets can be sent over it.
3. Varies from host to host and network to network.

Layer 2: Internet layer

1. Selection of a packet switching network which is based on a connectionless internetwork layer is called a internet layer.
2. It is the layer which holds the whole architecture together.
3. It helps the packet to travel independently to the destination.
4. Order in which packets are received is different from the way they are sent.
5. IP (Internet Protocol) is used in this layer.

Layer 3: Transport Layer

1. It decides if data transmission should be on parallel path or single path.
2. Functions such as multiplexing, segmenting or splitting on the data is done by transport layer.
3. The applications can read and write to the transport layer.
4. Transport layer adds header information to the data.
5. Transport layer breaks the message (data) into small units so that they are handled more efficiently by the network layer.
6. Transport layer also arrange the packets to be sent, in sequence.

Layer 4: Application Layer

The TCP/IP specifications described a lot of applications that were at the top of the protocol stack. Some of them were TELNET, FTP, SMTP, DNS etc.

1. TELNET is a two-way communication protocol which allows connecting to a remote machine and run applications on it.
2. FTP (File Transfer Protocol) is a protocol that allows File transfer amongst computer users connected over a network. It is reliable, simple and efficient.
3. SMTP (Simple Mail Transport Protocol) is a protocol, which is used to transport electronic mail between a source and destination, directed via a route.
4. DNS (Domain Name Server) resolves an IP address into a textual address for Hosts connected over a network.

TCP/IP Tools in Windows

Table 14.5 lists the TCP/IP diagnostic tools that are included with Windows Server 2003 and Windows XP. You can use these tools to help identify or resolve TCP/IP networking problems.

Tool	Description
ARP	Allows you to view and edit the Address Resolution Protocol (ARP) cache. The ARP cache maps IPv4 addresses to media access control (MAC) addresses. Windows uses these mappings to send data on the local network.
Hostname	Displays the host name of the computer.
IPconfig	Displays current TCP/IP configuration values for both IPv4 and IPv6. Also used to manage DHCP configuration and the DNS client resolver cache.
Lpq	Displays the status of print queues on print servers running Line Printer Daemon (LPD) software.
Nbtstat	Checks the state of current NetBIOS over TCP/IP connections, updates the Lmhosts cache, and determines the registered names and scope ID.
Netsh	Displays and allows you to administer settings for IPv4 or IPv6 on either the local computer or a remote computer.
Netstat	Displays statistics and other information about current IPv4 and IPv6 connections.
NSlookup	Queries a DNS server.
Ping	Tests IPv4 or IPv6 connectivity to other IP nodes.
Route	Allows you to view the local IPv4 and IPv6 routing tables and to modify the local IPv4 routing table.
Tracert	Traces the route that an IPv4 or IPv6 packet takes to a destination.
Pathping	Traces the route that an IPv4 or IPv6 packet takes to a destination and displays information on packet losses for each router and subnet in the path.

Table 14-5 TCP/IP diagnostic tools in Windows

Windows Server 2003 and Windows XP also include command-line tools for data transfer using FTP, Trivial File Transfer Protocol (TFTP), Telnet, and connectivity to UNIX-based resources.

After you have configured TCP/IP, you can use the Ipconfig and Ping tools to verify and test the configuration and connectivity to other TCP/IP hosts and networks.

The Ipconfig Tool

You can use the Ipconfig tool to verify the TCP/IP configuration parameters on a host, including the following:

- For IPv4, the IPv4 address, subnet mask, and default gateway.
- For IPv6, the IPv6 addresses and the default router.

Ipconfig is useful in determining whether the configuration is initialized and whether a duplicate IP address is configured. To view this information, type **ipconfig** at a command prompt.

Here is an example of the display of the Ipconfig tool for a computer that is using both IPv4 and IPv6:

```
C:\>ipconfig
```

Windows IP Configuration

```
Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . : wcoast.example.com
   IP Address. . . . . . . . . . . . : 157.60.139.77
   Subnet Mask . . . . . . . . . . . : 255.255.252.0
   IP Address. . . . . . . . . . . . : 2001:db8:ffff:f282:204:76ff:fe36:7363
   IP Address. . . . . . . . . . . . : fec0::f282:204:76ff:fe36:7363%2
   IP Address. . . . . . . . . . . . : fe80::204:76ff:fe36:7363
   Default Gateway . . . . . . . . . : 157.60.136.1
                                       2001:db8:1:21ad:210:ffff:fed6:58c0
```

Tunnel adapter Automatic Tunneling Pseudo-Interface:

```
   Connection-specific DNS Suffix  . : wcoast.example.com
   IP Address. . . . . . . . . . . . : 2001:db8:ffff:f70f:0:5efe:157.60.139.77
   IP Address. . . . . . . . . . . . : fe80::5efe:157.60.139.77%2
   Default Gateway . . . . . . . . . : fe80::5efe:157.54.253.9%2
```

Type **ipconfig /all** at a command prompt to view the IPv4 and IPv6 addresses of DNS servers, the IPv4 addresses of Windows Internet Name Service (WINS) servers (which resolve NetBIOS names to IP addresses), the IPv4 address of the DHCP server, and lease information for DHCP-configured IPv4 addresses.

The Ping Tool

After you verify the configuration with the Ipconfig tool, use the Ping tool to test connectivity. The Ping tool is a diagnostic tool that tests TCP/IP configurations and diagnoses connection failures. For IPv4, Ping uses ICMP Echo and Echo Reply messages to determine whether a particular IPv4-based host is available and functional. For IPv6, Ping uses ICMP for IPv6 (ICMPv6) Echo Request and Echo Reply messages. The basic command syntax is **ping** *Destination*, in which *Destination* is either an IPv4 or IPv6 address or a name that can be resolved to an IPv4 or IPv6 address.

Here is an example of the display of the Ping tool for an IPv4 destination:

```
C:\>ping 157.60.136.1

Pinging 157.60.136.1 with 32 bytes of data:

Reply from 157.60.136.1: bytes=32 time<1ms TTL=255

Reply from 157.60.136.1: bytes=32 time<1ms TTL=255

Reply from 157.60.136.1: bytes=32 time<1ms TTL=255

Reply from 157.60.136.1: bytes=32 time<1ms TTL=255

Ping statistics for 157.60.136.1:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

To verify a computer's configuration and to test for router connections, do the following:

1. Type **ipconfig** at a command prompt to verify whether the TCP/IP configuration has initialized.
2. Ping the IPv4 address of the default gateway or the IPv6 address of the default router to verify whether they are functioning and whether you can communicate with a node on the local network.
3. Ping the IPv4 or IPv6 address of a remote node to verify whether you can communicate through a router.

If you start with step 3 and you are successful, then you can assume that you would be successful with steps 1 and 2.

Note You cannot use the Ping tool to troubleshoot connections if packet filtering routers and host-based firewalls are dropping ICMP and ICMPv6 traffic.

14.2 User Datagram Protocol

The very fact that the TCP/IP protocol suite bears the name of the Internet Protocol and the Transmission Control Protocol suggests that these are the two key protocols in the suite: IP at the network layer and TCP at the transport layer. It's no wonder, therefore, that many people don't even realize that there is a second transport layer protocol in TCP/IP. Like a shy younger brother, the *User Datagram Protocol (UDP)* sits in the shadows while TCP gets the glory. Its fancier sibling deserves much of this limelight, since TCP is arguably the more important of the two transport layer protocol. However, UDP itself fills a critical niche in the TCP/IP protocol suite, allowing many applications to work at their best when using TCP would be less than ideal.

Note: There is also a protocol that is part of the NetBIOS/NetBEUI protocol suite called the User Datagram Protocol, also abbreviated UDP. The two are of course not the same.

UDP Operation

UDP's only real task is to take data from higher-layer protocols and place it in UDP messages, which are then passed down to the Internet Protocol for transmission. The basic steps for transmission using UDP are:

1. **Higher-Layer Data Transfer:** An application sends a message to the UDP software.
2. **UDP Message Encapsulation:** The higher-layer message is encapsulated into the *Data* field of a UDP message. The headers of the UDP message are filled in, including the *Source Port* of the application that sent the data to UDP, and the *Destination Port* of the intended recipient. The checksum value may also be calculated.
3. **Transfer Message To IP:** The UDP message is passed to IP for transmission.

And that's about it. Of course, on reception at the destination device this short procedure is reversed.

UDP Message Format

Field Name	**Size (bytes)**	**Description**
Source Port	2	***Source Port:*** The 16-bit port number of the process that originated the UDP message on the source device. This will normally be an ephemeral (client) port number for a request sent by a client to a server, or a well-known/registered (server) port number for a reply sent by a server to a client.
Destination Port	2	***Destination Port:*** The 16-bit port number of the process that is the ultimate intended recipient of the message on the destination device. This will usually be a well-known/registered (server) port number for a client request, or an ephemeral (client) port number for a server reply.
Length	2	***Length:*** The length of the entire UDP datagram, including both header and *Data* fields.
Checksum	2	***Checksum:*** An optional 16-bit checksum computed over the entire UDP datagram plus a special "pseudo header" of fields. See below for more information.
Data	Variable	***Data:*** The encapsulated higher-layer message to be sent.

Table 14.6: UDP Message Format

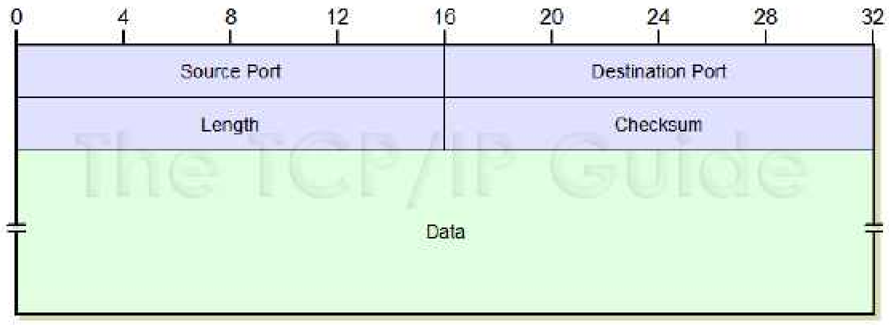


Figure 14.6: UDP Message Format

The Checksum Field and the UDP Pseudo Header

The UDP *Checksum* field is the one area where the protocol actually is a bit confusing. The concept of a checksum itself is nothing new; they are used widely in networking protocols to provide protection against errors. What's a bit odd is this notion of computing the checksum over the regular datagram and also a *pseudo header*. What this means is that instead of calculating the checksum over just the fields in the UDP datagram itself, the UDP software first constructs a "fake" additional header that contains the following fields (Figure 14.7):

- The IP *Source Address* field.
- The IP *Destination Address* field.
- The IP *Protocol* field.
- The UDP *Length* field.

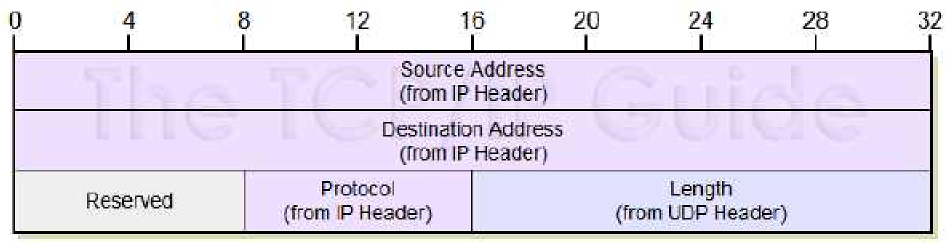

0
4
8
12
16
20
24
28
32
Source Address (from IP Header)
Destination Address (from IP Header)
Reserved
Protocol (from IP Header)
Length (from UDP Header)
The TCP Guide

Figure 14.7: UDP Pseudo Header Format

The total length of this "pseudo header" is 11 bytes. It is padded to 12 bytes with a byte of zeroes and then prepended to the real UDP message. The checksum is then computed over the combination of the pseudo header and the real UDP message, and the value is placed into the *Checksum* field. The pseudo header is used only for this calculation and is then discarded; it is not actually transmitted. The UDP software in the destination device creates the same pseudo header when calculating its checksum to compare to the one transmitted in the UDP header.

Computing the checksum over the regular UDP fields protects against bit errors in the UDP message itself. Adding the pseudo header allows the checksum to also protect against other types of problems as well, most notably the accidental delivery of a message to the wrong destination. The checksum calculation in UDP, including the use of the pseudo header is exactly the same as the method used in TCP (except the *Length* field is different in TCP).

Key Concept: *UDP packages application layer data into a very simple message format that includes only four header fields. One of these is an optional checksum field; when used, the checksum is computed over both*

the real header and a "pseudo header" of fields from the UDP and IP headers, in a manner very similar to how the TCP checksum is calculated.

Note that the use of the *Checksum* field is optional in UDP. If it is not used, it is set to a value of all zeroes. This could potentially create confusion, however, since when the checksum *is* used, the calculation can sometimes result in a value of zero. To avoid having the destination think the checksum was not used in this case, this zero value is instead represented as a value of all ones (65,535 decimal).

Check your progress

Q1. Define TCP/IP protocols.

Q2. Explain UDP protocol.

14.3 Connection Management

The HTTP specifications explain HTTP messages fairly well, but they don't talk much about HTTP connections, the critical plumbing that HTTP messages flow through. If you're a programmer writing HTTP applications, you need to understand the ins and outs of HTTP connections and how to use them.

HTTP connection management has been a bit of a black art, learned as much from experimentation and apprenticeship as from published literature.

TCP Connections

Just about all of the world's HTTP communication is carried over TCP/IP, a popular layered set of packet-switched network protocols spoken by computers and network devices around the globe. A client application can open a TCP/IP connection to a server application, running just about anywhere in the world. Once the connection is established, messages exchanged between the client's and server's computers will never be lost, damaged, or received out of order.

Say you want the latest power tools price list from Joe's Hardware store:

http://www.joes-hardware.com:80/power-tools.html

When given this URL, your browser performs the steps shown in Figure 14.8. In Steps 1-3, the IP address and port number of the server are pulled from the URL. A TCP connection is made to the web server in Step 4, and a request message is sent across the connection in Step 5. The response is read in Step 6, and the connection is closed in Step 7.

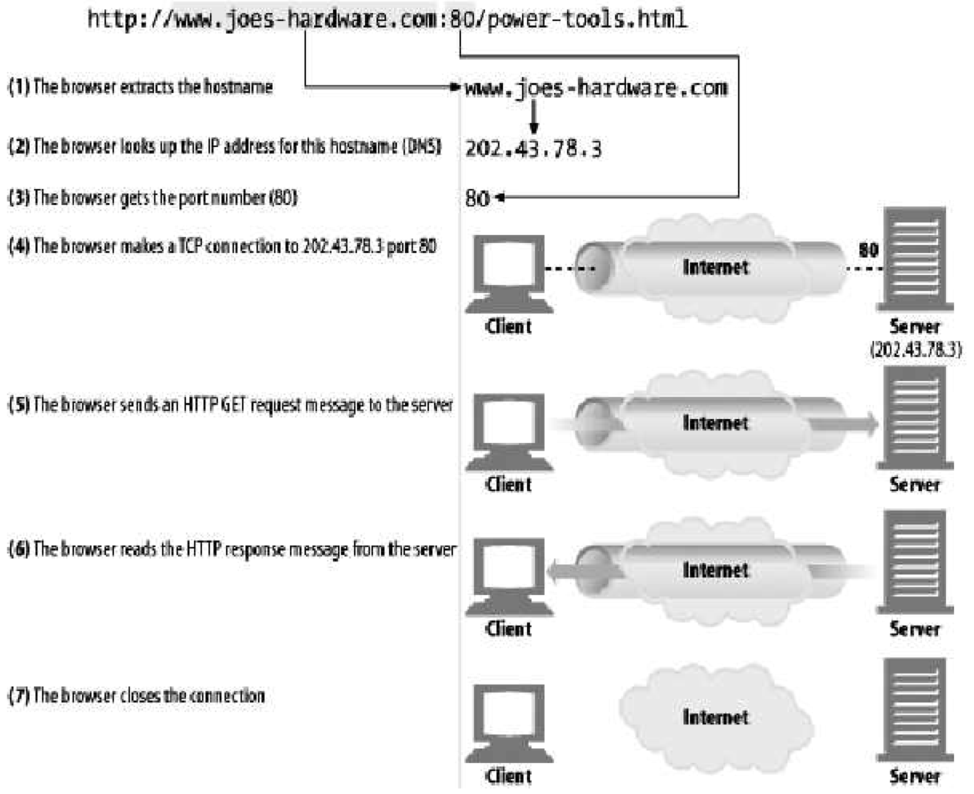

http://www.joes-hardware.com:80/power-tools.html
(1) The browser extracts the hostname
www.joes-hardware.com
(2) The browser looks up the IP address for this hostname (DNS)
202.43.78.3
(3) The browser gets the port number (80)
80
(4) The browser makes a TCP connection to 202.43.78.3 port 80
Internet
80
Client
Server (202.43.78.3)
(5) The browser sends an HTTP GET request message to the server
Internet
Client
Server
(6) The browser reads the HTTP response message from the server
Internet
Client
Server
(7) The browser closes the connection
Internet
Client
Server

Figure 14.8. Web browsers talk to web servers over TCP connections

TCP Reliable Data Pipes

HTTP connections really are nothing more than TCP connections, plus a few rules about how to use them. TCP connections are the reliable connections of the Internet. To send data accurately and quickly, you need to know the basics of TCP.

TCP gives HTTP a *reliable bit pipe* Bytes stuffed in one side of a TCP connection come out the other side correctly, and in the right order (see Figure 14.9).

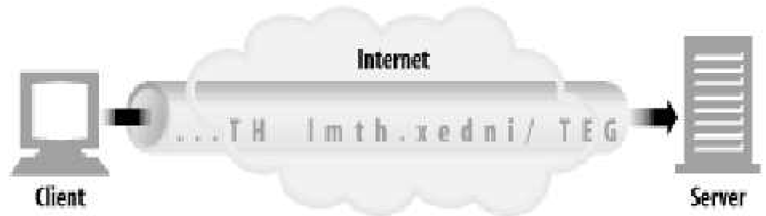


Figure 14.9: TCP carries HTTP data in order, and without corruption

TCP Streams Are Segmented and Shipped by IP Packets

TCP sends its data in little chunks called *IP packets* (or *IP datagrams*). In this way, HTTP is the top layer in a "protocol stack" of "HTTP over TCP over IP," as depicted in Figure 2.10(a). A secure variant, HTTPS, inserts a cryptographic encryption layer (called TLS or SSL) between HTTP and TCP, Figure 14.10 (b).

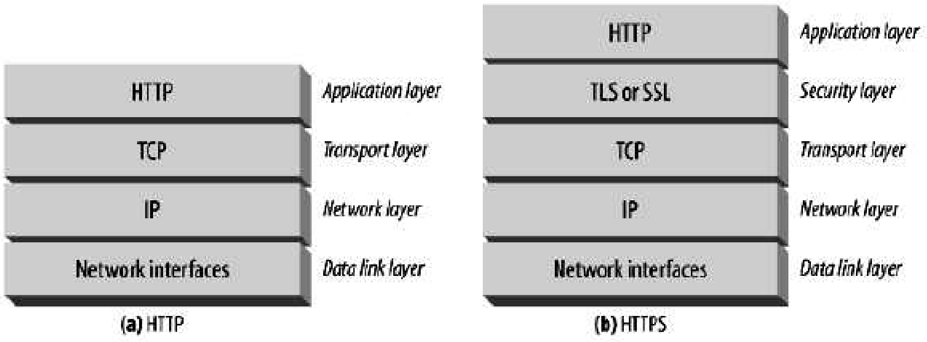


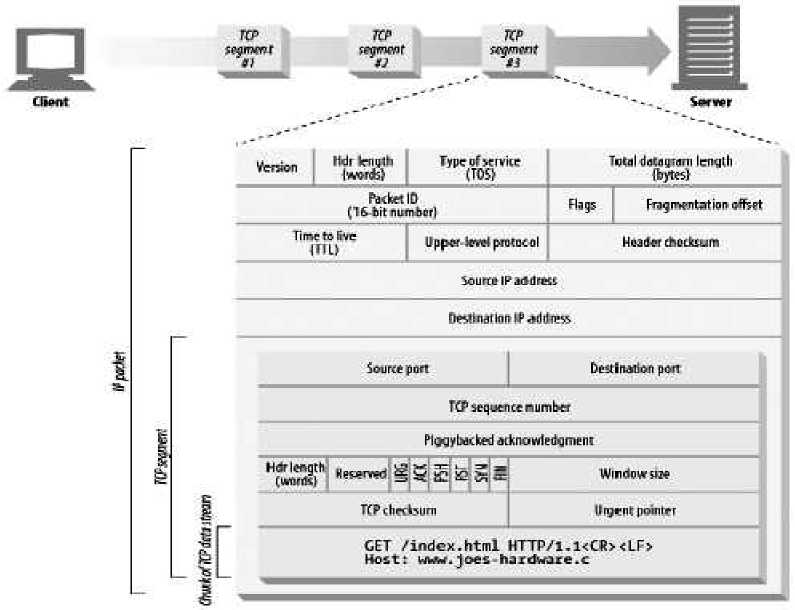
Figure 14.10. HTTP and HTTPS network protocol stacks

When HTTP wants to transmit a message, it streams the contents of the message data, in order, through an open TCP connection. TCP takes the stream of data, chops up the data stream into chunks called segments, and transports the segments across the Internet inside envelopes called IP packets (see Figure 14.11). This is all handled by the TCP/IP software; the HTTP programmer sees none of it.

Each TCP segment is carried by an IP packet from one IP address to another IP address. Each of these IP packets contains:

- An IP packet header (usually 20 bytes)
- A TCP segment header (usually 20 bytes)
- A chunk of TCP data (0 or more bytes)

The IP header contains the source and destination IP addresses, the size, and other flags. The TCP segment header contains TCP port numbers, TCP control flags, and numeric values used for data ordering and integrity checking.



```
GET /index.html HTTP/1.1<CR><LF>
Host: www.joes-hardware.c
```

Figure 14.11: IP packets carry TCP segments, which carry chunks of the TCP data stream

Keeping TCP Connections Straight

A computer might have several TCP connections open at any one time. TCP keeps all these connections straight through *port numbers*.

Port numbers are like employees' phone extensions. Just as a company's main phone number gets you to the front desk and the extension gets you to the right employee, the IP address gets you to the right computer and the port number gets you to the right application. A TCP connection is distinguished by four values:

<source-IP-address, source-port, destination-IP-address, destination-port>

Together, these four values uniquely define a connection. Two different TCP connections are not allowed to have the same values for all four address components (but different connections can have the same values for some of the components).

In Figure 14.12, there are four connections: A, B, C and D. The relevant information for each port is listed in Table 14.6.

Connection	**Source IP address**	**Source port**	**Destination IP address**	**Destination port**
A	209.1.32.34	2034	204.62.128.58	4133
B	209.1.32.35	3227	204.62.128.58	4140
C	209.1.32.35	3105	207.25.71.25	80
D	209.1.33.89	5100	207.25.71.25	80

Table 14.6: TCP connection values

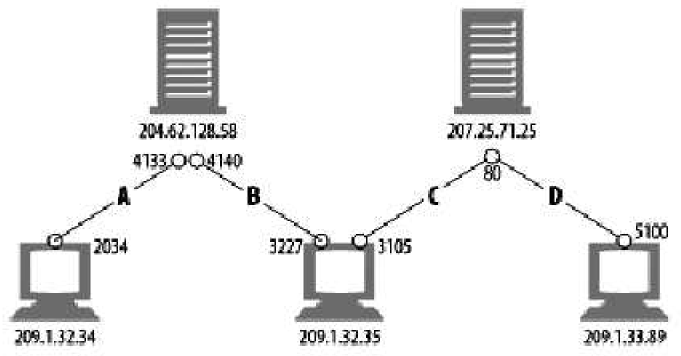


Figure 14.12: Four distinct TCP connections

Note that some of the connections share the same destination port number (C and D both have destination port 80). Some of the connections have the same source IP address (B and C). Some have the same destination IP address (A and B, and C and D). But no two different connections share all four identical values.

14.4 Finite State Machine

In the case of TCP, the finite state machine can be considered to describe the "life stages" of a connection. Each connection between one TCP device and another begins in a null state where there is no connection, and then proceeds through a series of states until a connection is established. It remains in that state until something occurs to cause the connection to be closed again, at which point it proceeds through another sequence of transitional states and returns to the closed state.

The full description of the states, events and transitions in a TCP connection is lengthy and complicated—not surprising, since that would cover much of the entire TCP standard. For our purposes, that level of detail would be a good cure for insomnia but not much else. However, a ***simplified*** look at the TCP FSM will help give us a nice overall feel for how TCP establishes connections and then functions when a connection has been created.

Table 14.7 briefly describes each of the TCP states in a TCP connection, and also describes the main events that occur in each state, and what actions and transitions occur as a result. For brevity, three abbreviations are used for three types of message that control transitions between states, which correspond to the TCP header flags that are set to indicate a message is serving that function. These are:

- ***SYN:*** A *synchronize* message, used to initiate and establish a connection. It is so named since one of its functions is to synchronize sequence numbers between devices.
- ***FIN:*** A ***finish*** message, which is a TCP segment with the *FIN* bit set, indicating that a device wants to terminate the connection.
- ***ACK:*** An *acknowledgment*, indicating receipt of a message such as a *SYN* or a *FIN*.

Again, we have not shown every possible transition, just the ones normally followed in the life of a connection. Error conditions also cause transitions but including these would move us well beyond a "simplified" state machine. The FSM is also illustrated in Figure 14.13, which you may find easier for seeing how state transitions occur.

State	State Description	Event and Transition
CLOSED	This is the default state that each connection starts in before the process of establishing it begins. The state is called "fictional" in the standard. The reason is that this state represents the situation where there is no connection between devices—it either hasn't been created yet, or has just been destroyed. If that makes sense.	**Passive Open:** A server begins the process of connection setup by doing a passive open on a TCP port. At the same time, it sets up the data structure (transmission control block or TCB) needed to manage the connection. It then transitions to the *LISTEN* state.
		Active Open, Send *SYN*: A client begins connection setup by sending a *SYN* message, and also sets up a TCB for this connection. It then transitions to the *SYN-SENT* state.
LISTEN	A device (normally a server) is waiting to receive a *synchronize* (*SYN*) message from a client. It has not yet sent its own *SYN* message.	**Receive Client *SYN*, Send *SYN+ACK*:** The server device receives a *SYN* from a client. It sends back a message that contains its own *SYN* and also acknowledges the one it received. The server moves to the *SYN-RECEIVED* state.
SYN-SENT	The device (normally a client) has sent a *synchronize* (*SYN*) message and is waiting for a matching *SYN* from the other device (usually a server).	**Receive *SYN*, Send *ACK*:** If the device that has sent its *SYN* message receives a *SYN* from the other device but not an *ACK* for its own *SYN*, it acknowledges the *SYN* it receives and then transitions to *SYN-RECEIVED* to wait for the acknowledgment to its *SYN*.
		Receive *SYN+ACK*, Send *ACK*: If the device that sent the *SYN* receives both an acknowledgment to its *SYN* and also a *SYN* from the other device, it acknowledges the *SYN* received and then moves straight to

		the *ESTABLISHED* state.
SYN-RECEIVED	The device has both received a *SYN* (connection request) from its partner and sent its own *SYN*. It is now waiting for an *ACK* to its *SYN* to finish connection setup.	**Receive *ACK*:** When the device receives the *ACK* to the *SYN* it sent, it transitions to the *ESTABLISHED* state.
ESTABLISHED	The "steady state" of an open TCP connection. Data can be exchanged freely once both devices in the connection enter this state. This will continue until the connection is closed for one reason or another.	**Close, Send *FIN*:** A device can close the connection by sending a message with the *FIN* (*finish*) bit sent and transition to the *FIN-WAIT-1* state.
		Receive *FIN*: A device may receive a *FIN* message from its connection partner asking that the connection be closed. It will acknowledge this message and transition to the *CLOSE-WAIT* state.
CLOSE-WAIT	The device has received a close request (*FIN*) from the other device. It must now wait for the application on the local device to acknowledge this request and generate a matching request.	**Close, Send *FIN*:** The application using TCP, having been informed the other process wants to shut down, sends a close request to the TCP layer on the machine upon which it is running. TCP then sends a *FIN* to the remote device that already asked to terminate the connection. This device now transitions to *LAST-ACK*.
LAST-ACK	A device that has already received a close request and acknowledged it, has sent its own *FIN* and is waiting for an *ACK* to this request.	**Receive *ACK* for *FIN*:** The device receives an acknowledgment for its close request. We have now sent our *FIN* and had it acknowledged, and received the other device's *FIN* and acknowledged it, so we go straight to the *CLOSED* state.

FIN-WAIT-1	A device in this state is waiting for an *ACK* for a *FIN* it has sent, or is waiting for a connection termination request from the other device.	**Receive *ACK* for *FIN*:** The device receives an acknowledgment for its close request. It transitions to the *FIN-WAIT-2* state.
		Receive *FIN*, Send *ACK*: The device does not receive an *ACK* for its own *FIN*, but receives a *FIN* from the other device. It acknowledges it, and moves to the *CLOSING* state.
FIN-WAIT-2	A device in this state has received an ACK for its request to terminate the connection and is now waiting for a matching *FIN* from the other device.	**Receive *FIN*, Send *ACK*:** The device receives a *FIN* from the other device. It acknowledges it and moves to the *TIME-WAIT* state.
CLOSING	The device has received a *FIN* from the other device and sent an *ACK* for it, but not yet received an *ACK* for its own *FIN* message.	**Receive *ACK* for *FIN*:** The device receives an acknowledgment for its close request. It transitions to the *TIME-WAIT* state.
TIME-WAIT	The device has now received a *FIN* from the other device and acknowledged it, and sent its own *FIN* and received an *ACK* for it. We are done, except for waiting to ensure the ACK is received and prevent potential overlap with new connections.	**Timer Expiration:** After a designated wait period, device transitions to the *CLOSED* state.

Table 14.7: TCP Finite State Machine (FSM) States, Events and Transitions

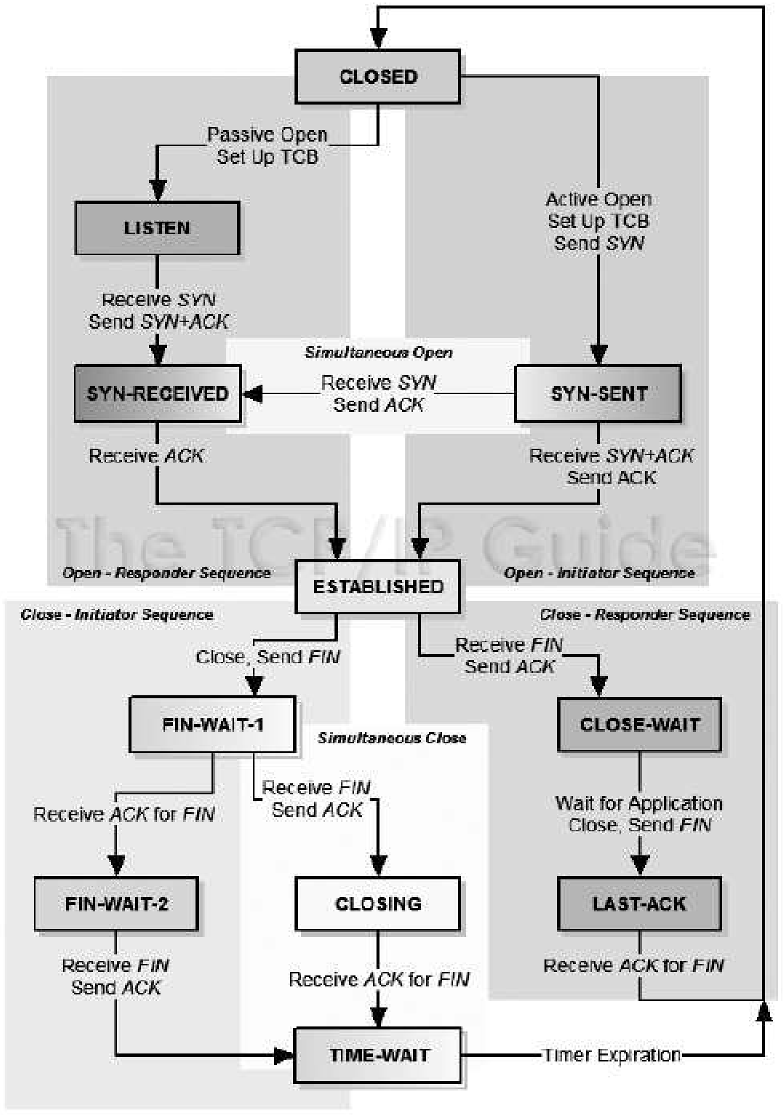


Figure 14.13: The TCP Finite State Machine (FSM)

Finite State Machine Steps Represent the Stages of a Connection

It's important to remember that this state machine is followed for ***each connection***. This means at any given time TCP may be in one state for one connection to socket X, while in another for its connection to socket Y. Also, the typical movement between states for the two processes in a particular connection is not symmetric, because the roles of the devices are not symmetric: one device initiates a connection, the other responds; one device starts termination, the other replies. There is also an alternate path taken for connection establishment and termination if both devices initiate simultaneously (which is unusual, but can happen). This is shown by the color coding in Figure 14.13.

Thus, for example, at the start of connection establishment, the two devices will take different routes to get to *ESTABLISHED*: one device (the server usually) will pass through the *LISTEN* state while the other (the client) will go through *SYN-SENT*. Similarly, one device will initiate connection termination and take the path through *FIN-WAIT-1* to get back to *CLOSED*; the other will go through *CLOSE-WAIT* and *LAST-ACK*. However, if both try to open at once, they each proceed through *SYN-SENT* and *SYN-RECEIVED*, and if both try to close at once, they go through *FIN-WAIT-1*, *CLOSING* and *TIME-WAIT* roughly simultaneously.

***Key Concept:** The TCP finite state machine describes the sequence of steps taken by both devices in a TCP session as they establish, manage and close the connection. Each device may take a different path through the states since under normal circumstances the operation of the protocol is not symmetric—one device initiates either connection establishment or termination, and the other responds.*

Check your progress

Q1. Explain connection management in TCP.

Q2. Define finite state machine in TCP.

14.5 Summary

In this unit you have learnt about Elements of TCP/IP Protocol, User Datagram Protocol, Connection Management and Finite State Machine.

- TCP/IP provides end-to-end data communication specifying how data should be packetized, addressed, transmitted, routed and received. This functionality is organized into four abstraction layers which are used to sort all related protocols according to the scope of networking involved. From lowest to highest, the layers are the link layer, containing communication methods for data that remains within a single network segment (link); the internet layer, connecting independent networks, thus providing internetworking; the transport layer handling host-to-host communication; and the application layer, which provides process-to-process data exchange for applications.
- UDP uses a simple connectionless transmission model with a minimum of protocol mechanism. It has no handshaking dialogues, and thus exposes the user's program to any unreliability of the underlying network protocol. There is no guarantee of delivery, ordering, or duplicate protection. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram.

- TCP communication works in Server/Client model. The client initiates the connection and the server either accepts or rejects it. Three-way handshaking is used for connection management.
- Finite state machines can be used to detect anomalous behavior in TCP traffic by describing the progression of a connection through states as a result of events based on header flags. The method was applied to real traffic to understand its realistic use and it was found that for the time period analyzed here, on the order of 37% of TCP connections do not follow the TCP protocol specifications. The majority of these are a result of malicious activity, and approximately 4% are due to benign anomalies such as unresponsive hosts and misconfigurations. The method may be applied as a network security measure, as a network management tool or as a research tool for the study of TCP behavior on the Internet.

14.6 Review Questions

Q1. Define TCP/IP Protocol with all its layers.

Q2. Why UDP a connection-less protocol? Elaborate your answer.

Q3. What is the difference between TCP and UDP protocols?

Q4. How a TCP establish and release a connection? Explain your answer.

Q5. Why do we need finite state machine in TCP?

UNIT-XV
Session Layer Protocols

Structure

15.0 Introduction

In this unit we focused on session layer protocols. There are five sections in this unit. As we have studied earlier In the Open Systems Interconnection (OSI) communications model, the **Session layer** (sometimes called the "port layer") manages the setting up and taking down of the association between two communicating end points that is called a connection. A connection is maintained while the two end points are communicating back and forth in a conversation or *session* of some duration. Some connections and sessions last only long enough to send a message in one direction. However, other sessions may last longer, usually with one or both of the communicating parties able to terminate it. In Sec. 15.1 you will learn about dialog management. In the next section i.e. Sec. 15.2 you will know synchronization. In section 15.4 we focused on OSI session primitives i.e. connection establishment. In Sec. 15.4 and 15.5 you will find summary and review questions respectively.

Objectives

After studying this unit you should be able to:

- ❖ Define Dialog Management.
- ❖ Describe Synchronization.
- ❖ Express OSI Session Primitives Connection Establishment.

15.1 Session Layer

The session layer of the Open System Interconnection (OSI) model defines how the data is formatted between the devices on either side of the link. This is effectively the manner in which they maintain an open channel between the two devices. However, at lower levels of the OSI model, there is no permanent connection but rather a series of short bursts of data being sent back and forth.

The session layer maintains a conversation over many of these bursts of data; in fact, it can take several bursts of data going back and forth just to establish the structure that will be followed for that session.

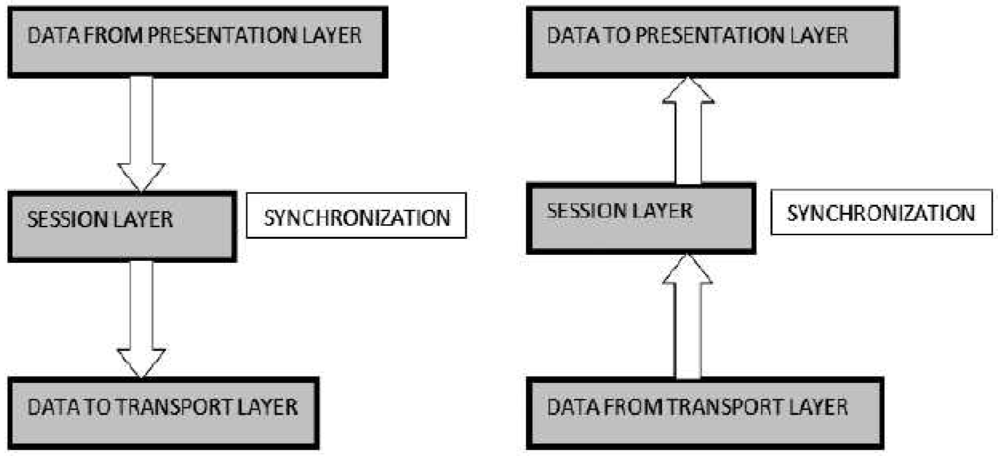

DATA FROM PRESENTATION LAYER
SESSION LAYER
SYNCHRONIZATION
DATA TO TRANSPORT LAYER
DATA TO PRESENTATION LAYER
SESSION LAYER
SYNCHRONIZATION
DATA FROM TRANSPORT LAYER

Figure 15.1: Session Layer

A real-world example of the session layer might be a pair of spies exchanging messages. They would have to establish an order of operations that would be used to pass encoded messages back and forth. This process for passing the messages could be considered a session layer operation and may include steps such as using an agreed upon cipher to encode the message.

Windows file sharing has a session layer component when establishing sessions, as shown in the following figure.

The goal of the client computer in the figure below is to get a list of shares on the server, but it must follow a session setup process in order to get the desired data. The server in this process is in a constant state of listening for connection requests; as the client starts the process off, the session setup process runs like this:

1. The client sends a session request to the server.
2. The server acknowledges the request and includes in the acknowledgement a list of all supported session protocol.

In the case of the Windows server, the list includes older, less secure options such as LANMAN, as well as the newer and more secure NT LANMAN version 2 (NT LM 2).

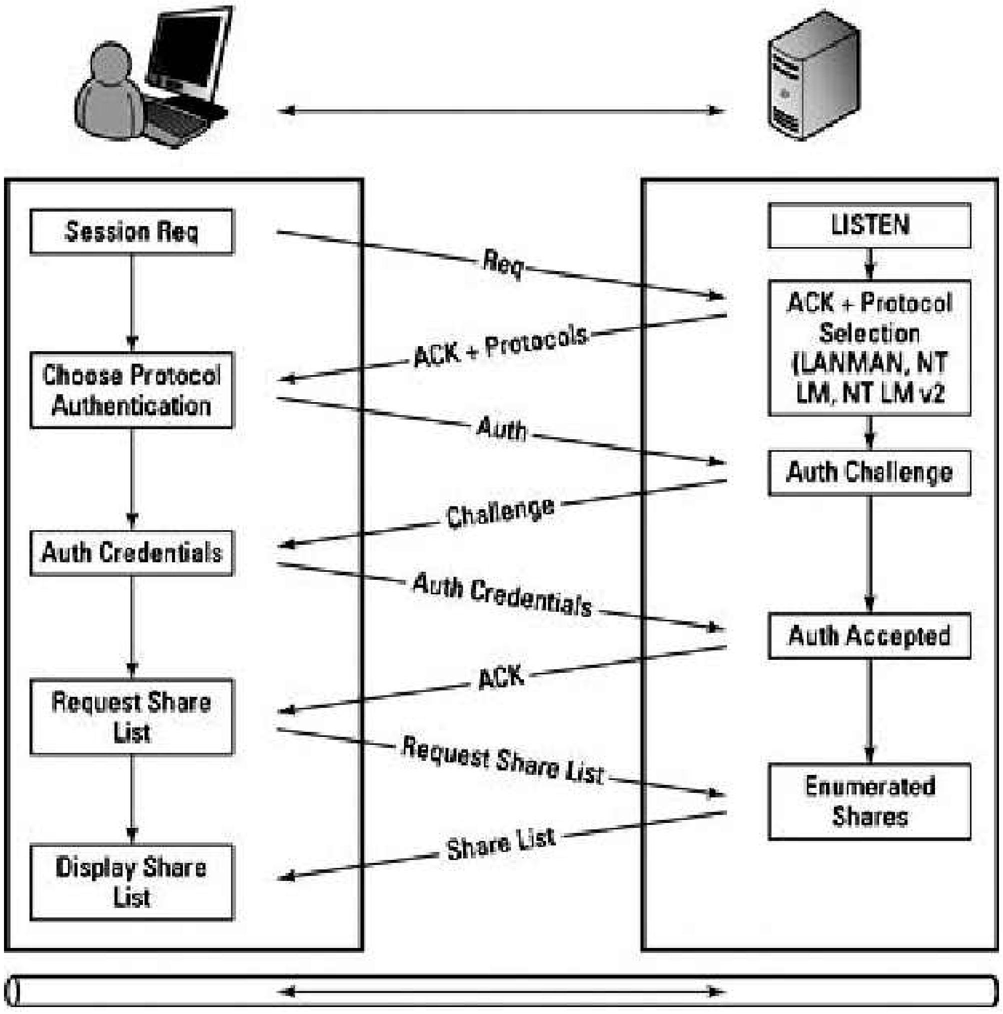


Figure 15.2: Working session layer

3. The client reviews the list of supported protocols and chooses the most secure session protocol that it also supports. At this point, it sends the server the chosen session protocol that they will be using and requests to conduct an authentication. In this case, the authentication will verify a username and password from the server's user account database.

4. The server creates a random string of characters that is included in the password challenge and sends it to the client.

5. The client takes the password challenge string and uses its own password as an encryption key to encrypt the random string.

6. The now encrypted string is then sent back to the server in an authentication credentials package that also includes the user's username.

7. The server retrieves the user's password from its user account database, and uses the password as the encryption key to encrypt the random character string that it sent to the client in the challenge step (Step 4).

8. The server compares the result it calculated to the result listed in the authentication credential package.

9. If the results match, as they do in the illustration, then an acknowledgement (ACK) is sent to the client and the session is now active; but if they do not match, then the server sends a negative acknowledgement (NACK).

   If the client receives a NACK, then it would go back to Step 3 and issue a new authentication request.

10. At this point the session is set up, and the client can perform the request for the list of shares that are on the server, which would likely lead into a request of a list of files, and then the contents of a specific file.

All of these future operations would be conducted through this single session that has just been created.

Step 10 in the preceding list shows the change from the session layer and the presentation layer. At the session layer, the communication channel to the Windows server components was established, but as the actual request was submitted for the list of shares available on the server, the request used the session layer communication channel, but was actually delivered through to the presentation layer and ultimately the application layer Windows Server service.

Application Program Interfaces (APIs)

The primary job of session layer protocols is to provide the means necessary to set up, manage, and end sessions. In fact, in some ways, session layer software products are more sets of tools than specific protocols. These session-layer tools are normally provided to higher layer protocols through command sets often called *application program interfaces* or *APIs*.

Common APIs include NetBIOS, TCP/IP Sockets and Remote Procedure Calls (RPCs). They allow an application to accomplish certain high-level communications over the network easily, by using a standardized set of services. Most of these session-layer tools are of primary interest to the developers of application software. The programmers use the APIs to write software that is able to communicate using TCP/IP without having to know the implementation details of how TCP/IP works.

For example, the Sockets interface lies conceptually at layer five and is used by TCP/IP application programmers to create sessions between software programs over the Internet on the UNIX operating system.

Windows Sockets similarly lets programmers create Windows software that is Internet-capable and able to interact easily with other software that uses that interface. (Strictly speaking, Sockets is not a protocol, but rather a programming method.)

Session layer functionalities:

- Session management: As its name suggests, the session layer is responsible for managing a session which includes opening, closing and managing a session between end-user application processes. You can think of session layer as the main layer which handles the requests and responses between the two applications. For example, if you are downloading some pictures from Facebook, a network path is defined by the network layer but the requests for the photos and responses by Facebook to you is handled by the session layer. If you have heard of HTTP 1.0 or HTTP 1.1, these application layer protocols somehow follow the session layer orders in delivering the data.
- Authentication: Before establishing a session with some network peer, it is important for one of the computers to know that another peer it is communicating to is a legitimate one. In short terms, you can say that authentication is the process of verifying that "you are who you say you are".
- Authorization: Authorization is more like "Are you authorized to do so?"
- Here is an example:
- If someone knows my email address and password, he can easily authenticate himself as 'XYZ' and he can log in as well. However, since he is not the right person to access my personal email account, so he is not an authorized person to do so.
- So the basic difference between the two is: authentication is the process of verifying that "you are who you say you are", authorization is the process of verifying that "you are permitted to do what you are trying to do". Authorization thus presupposes authentication.
- **Some of the famous session layer's protocols are:**
- Remote procedure call protocol (RPC)
- Point-to-Point Tunneling Protocol (PPTP)
- Session Control Protocol (SCP)
- Session Description Protocol (SDP) etc.

RPC

RPC is a powerful technique for constructing distributed, client-server based applications. It is based on extending the notion of conventional, or local procedure calling, so that the called procedure need not exist in the same address space as the calling procedure. The two processes may be on the same system, or they may be on different systems with a network connecting them. By using RPC, programmers of distributed applications avoid the details of the interface with the network. The transport independence of RPC isolates the application from the physical and logical elements of the data communications mechanism and allows the application to use a variety of transports.

RPC makes the client/server model of computing more powerful and easier to program. When combined with the ONC RPCGEN protocol compiler clients transparently make remote calls through a local procedure interface.

How RPC Works

An RPC is analogous to a function call. Like a function call, when an RPC is made, the calling arguments are passed to the remote procedure and the caller waits for a response to be returned from the remote procedure. Figure 3.3 shows the flow of activity that takes place during an RPC call between two networked systems. The client makes a procedure call that sends a request to the server and waits. The thread is blocked from processing until either a reply is received, or it times out. When the request arrives, the server calls a dispatch routine that performs the requested service, and sends the reply to the client. After the RPC call is completed, the client program continues. RPC specifically supports network applications.

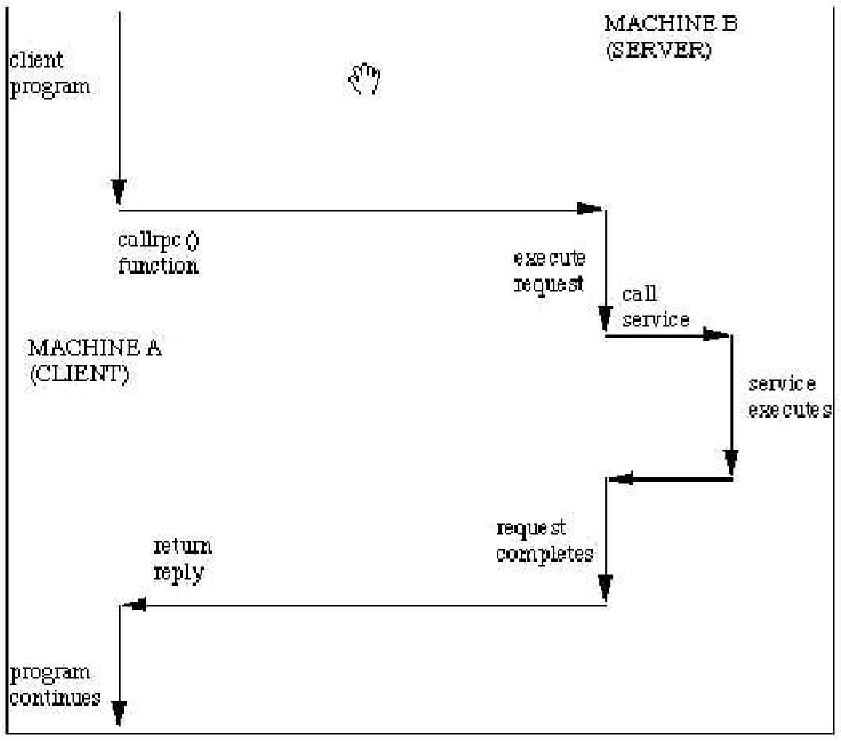


Figure 15.3- RPC working

RPC Application Development

Consider an example:

A client/server lookup in a personal database on a remote machine. Assuming that we cannot access the database from the local machine (via NFS).

We use UNIX to run a remote shell and execute the command this way. There are some problems with this method:

- The command may be slow to execute.
- You require a login account on the remote machine.

The RPC alternative is to

- Establish a server on the remote machine that can respond to queries.
- Retrieve information by calling a query which will be quicker than previous approach.

To develop an RPC application the following steps are needed:

- Specify the protocol for client server communication
- Develop the client program
- Develop the server program

The programs will be compiled separately. The communication protocol is achieved by generated stubs and these stubs and RPC (and other libraries) will need to be linked in.

Defining the Protocol

The easiest way to define and generate the protocol is to use a protocol compiler such as rpcgen.

For the protocol you must identify the name of the service procedures, and data types of parameters and return arguments.

The protocol compiler reads a definition and automatically generates client and server stubs.

rpcgen uses its own language (RPC language or RPCL) which looks very similar to preprocessor directives.

rpcgen exists as a standalone executable compiler that reads special files denoted by a .x prefix.

So to compile a RPCL file you simply do

rpcgen rpcprog.x

This will generate possibly four files:

- rpcprog_clnt.c -- the client stub
- rpcprog_svc.c -- the server stub

- rpcprog_xdr.c -- If necessary XDR (external data representation) filters
- rpcprog.h -- the header file needed for any XDR filters.

The external data representation (XDR) is a data abstraction needed for machine independent communication. The client and server need not be machines of the same type.

Point-to-Point Tunneling Protocol (PPTP)

You can access a private network through the Internet or other public network by using a virtual private network (VPN) connection with the Point-to-Point Tunneling Protocol (PPTP).

PPTP enables the secure transfer of data from a remote computer to a private server by creating a VPN connection across IP-based data networks. PPTP supports on-demand, multiprotocol, virtual private networking over public networks, such as the Internet.

Developed as an extension of the Point-to-Point Protocol (PPP), PPTP adds a new level of enhanced security and multiprotocol communications over the Internet. By using the new Extensible Authentication Protocol (EAP) with strong authentication methods such as certificates, data transfer through a PPTP-enabled VPN connection is as secure as within a single LAN at a corporate site.

PPTP encapsulates IP or IPX protocols inside of PPP datagrams. This means that you can remotely run applications that are dependent upon particular network protocols. The tunnel server performs all security checks and validations, and enables data encryption, which makes it much safer to send information over non-secure networks. You can also use PPTP in private LAN-to-LAN networking.

The IPX/SPX protocol is not available on Windows XP 64-bit Edition (Titanium) and the 64-bit versions of the Windows Server 2003 family.

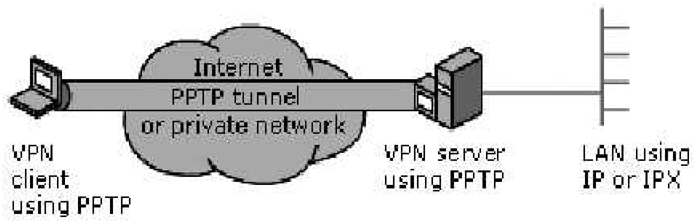


Figure 15.4: Point-to-Point Tunneling Protocol (PPTP)

PPTP requires IP connectivity between your computer and the server. If you are directly attached to an IP LAN and can reach a server, then you can establish a PPTP tunnel across the LAN. If you are creating a tunnel over the Internet, and your normal Internet access is a dial-up connection to an ISP, you must dial up your Internet connection before you can establish the tunnel.

Session Control Protocol (SCP)

Session control protocol (SCP) is a method of creating multiple light-duty connections from a single TCP (Transmission Control Protocol) connection.

Several such lightweight connections can be active simultaneously. SCP is a session layer protocol in the Open Systems Interconnection (OSI) model. In the session layer (layer five in the seven-layer telecommunications model), session layer protocols provide services for coordinating communication between local and remote applications, establishing, managing and terminating connections.

SCP runs on TCP, depending on it to provide connections and reliable service. While SCP is a byte-oriented protocol it also supports message boundary markers. TCP is a set of rules used along with the Internet Protocol (IP) to send data in the form of message units between computers over the Internet.

Other protocols that may be used in this layer include AppleTalk, RPC, ITU x.225 and Zone Information Protocol (ZIP).

Packet Format

SCP headers come in to varieties, compressed and uncompressed. The canonical version of the header is the uncompressed format- the compressed format is conceptually expanded before being processed.

The uncompressed form of an SCP packet consists of a 64 bit header followed by zero or more octets of data. If the COMPRESSION flag in the first byte is set, the header should be interpreted as described in the section on compression.

The header contains three fields; a flag byte, the session identifier, and the packet length.

```
FLAGS
+------+------+------+------+
+SFRP0000| Session ID        |
+------+------+------+------+
|        LENGTH          |
+------+------+------+------+
```

Flags

This field is one octet long. If the least significant bit in the octet is set, then the header should be interpreted as a compressed header. If this bit is not set, then the four high order bits contain flags used to open and close sessions, and to mark application level messages.

Table 15.1

Name	Mask	Description
Compressed Header	XXXX|XXX1	Mark a Compressed Header
SYN	1XXX|0000	Open A New Connection
FIN	X1XX|0000	Close An Existing Connection
PUSH	XX1X|0000	Mark an application level message boundary
RESET	XXX1|0000	Abort the Connection

The RESET flag may not be sent in conjunction with any other flags.

Session Identifiers

Each session is identified by a 24 bit integer.

Sessions created by the client must have even identifiers; those created by the server must have odd identifiers.

Session Identifiers below 1024 not otherwise allocated by this document are reserved to the IANA for allocating to standard sub protocols..

If a party receives a message for a reserved sessions which it does not implement it must return a RESET message on that channel the first time it receives a message on any such channel. At the receiver's option a RESET may be sent for subsequent messages on that channel

Session Identifiers are compared modulo 2^24, using the same algorithm as that for comparing TCP sequence numbers.

Reserved Session Identifiers

Table 15.2

1	Presentation Control Protocol (PCP)
2	Multiplexed Data Algorithm (MDA)

XXXX - Define PCP - select higher-level protocol to talk to.

Define MDA - Wrapped SCP packets with options.

Packet Lengths

The packet length is represented by a 32 bit unsigned integer value. Application level messages may be longer than 2^32-1 octets long; however in this case the messages must be split up into two or more SCP layer packets.

An SCP implementation is free to regenerate SCP packets when so desired; for example, if the underlying transport can make part of an SCP packet available before the entire packet has been received, the implementation is free to deliver the data as and when it becomes available as long as any message boundary marker is only send for the final packet. An implementation may also combine two SCP packets for the same session as long as no packet but the last has a message boundary marker.

Session States

Sessions can exist in several different states; closed, opened for read, opened for write, and opened for read and write. A Session can change its state in response to receiving a packet with the SYN, FIN, or RESET bits set, or in response to an API by the application. The available API calls are open, close, abort.

The following table describes the actions and state transitions that occur in response to a given event. The events are given in order of priority - if multiple flags are present, then the first matched action should be followed, and any remaining flags processed in the resultant successor state.

For example, if a session is in the CLOSED state, and it receives a packet containing a SYN, a FIN, and data, the implementation should check the table for Closed, and look for the first even that matches. In this case, the first match is for SYN; the implementation should accept the Session, remove the SYN from the set of pending events, and move to Open-Receive.

Table 15.3

State	Event	Action	New State
Closed	RESET	Discard	Closed
	SYN (1)	Accept Session	Open-Receive
	FIN	Discard	Closed
	Other	Discard	Closed
Open-Receive	FIN	Close Session, Discard	Closed
	RESET	Abort Session, Discard	Closed
	SYN	Send RESET, Abort Session	Closed
	CLOSE	Send RESET, Close Session	Closed
	ABORT	Send RESET, Close Session	Closed

	DATA-IN	Read Data	Open-Receive
	OPEN	Send SYN, Open for Writing	Open-ReadWrite
Open-ReadWrite	SYN	Send Reset, Close Connection	Closed
	DATA-IN	Read Data	Open-ReadWrite
	DATA-OUT	Write Data	Open-ReadWrite
	Shutdown	Send FIN, Close for Writing	Open-Receive
	FIN	Close for Reading	Open-Write
	RESET	Abort Connection	Closed
	CLOSE	Send FIN + RESET, Close Connection	Closed
Open-Write	RESET	Abort Connection	Closed
	SYN	Open for Reading	Open-ReadWrite
	Data-In	Send RESET, Abort Connection	Closed
	Data-Out	Write Data	Open-Write
	FIN	Send RESET, Abort Connection	Open-Write
	CLOSE	Send FIN, Close Connection	Closed
	Abort	Send Reset, Close Connection	Closed

Session Description Protocol (SDP)

SDP stands for Session Description Protocol. It is used to describe multimedia sessions in a format understood by the participants over a network. Depending on this description, a party decides whether to join a conference or when or how to join a conference.

- The owner of a conference advertises it over the network by sending multicast messages which contain description of the session e.g. the name of the owner, the name of the session, the coding, the timing etc. Depending on these information the recipients of the advertisement take a decision about participation in the session.

- SDP is generally contained in the body part of Session Initiation Protocol popularly called SIP.

- SDP is defined in RFC 2327. An SDP message is composed of a series of lines, called fields, whose names are abbreviated by a single lower-case letter, and are in a required order to simplify parsing

Purpose of SDP

The purpose of SDP is to convey information about media streams in multimedia sessions to help participants join or gather info of a particular session.

- SDP is a short structured textual description.
- It conveys the name and purpose of the session, the media, protocols, codec formats, timing and transport information.
- A tentative participant checks these information and decides whether to join a session and how and when to join a session if it decides to do so.
- The format has entries in the form of <type> = <value>, where the <type> defines a unique session parameter and the <value> provides a specific value for that parameter.
- The general form of a SDP message is: x = parameter1 parameter2 ... parameterN
- The line begins with a single lower-case letter, for example, x. There are never any spaces between the letter and the =, and there is exactly one space between each parameter. Each field has a defined number of parameters.

15.1.1 Dialog Management

There are many different points of view and techniques for achieving application check pointing. Depending on the specific implementation, a tool can be classified as having several properties:

Amount of state saved: This property refers to the abstraction level used by the technique to analyze an application. It can range from seeing each application as a black box, hence storing all application data, to selecting specific relevant cores of data in order to achieve a more efficient and portable operation.

Automatization level: Depending on the effort needed to achieve fault tolerance through the use of a specific check pointing solution.

Portability: Whether or not the saved state can be used on different machines to restart the application.

System architecture: How is the check pointing technique implemented: inside a library, by the compiler or at operating system level.

Each design decision made affects the properties and efficiency of the final product. For instance, deciding to store the entire application state will allow for a more straightforward implementation, since no analysis of the application will be needed, but it will deny the portability of the generated state files, due to a number of non-portable structures (such as application stack or heap) being stored along with application data.

15.1.2 Synchronization

Synchronization refers to one of two distinct but related concepts: synchronization of processes, and synchronization of data. Process synchronization refers to the idea that multiple processes are to join up or handshake at a certain point, so as to reach an agreement or commit to a certain sequence of action. Data synchronization refers to the idea of keeping multiple copies of a dataset in coherence with one another, or to maintain data integrity. Process synchronization primitives are commonly used to implement data synchronization.

Authentication is the act of confirming the truth of an attribute of a datum or entity. This might involve confirming the identity of a person or software program, tracing the origins of an artifact, ensuring that a product is what it's packaging and labeling claims to be.

15.3 OSI Session Primitives: Connection Establishment.

A typical OSI communications scenario can now be sketched. Assume that one end system (local computer) has a client program which requires the services of a remote server program located at a remote computer connected by a communications network. A simplified diagram is shown in the figure below. This shows the peer-to-peer communication.

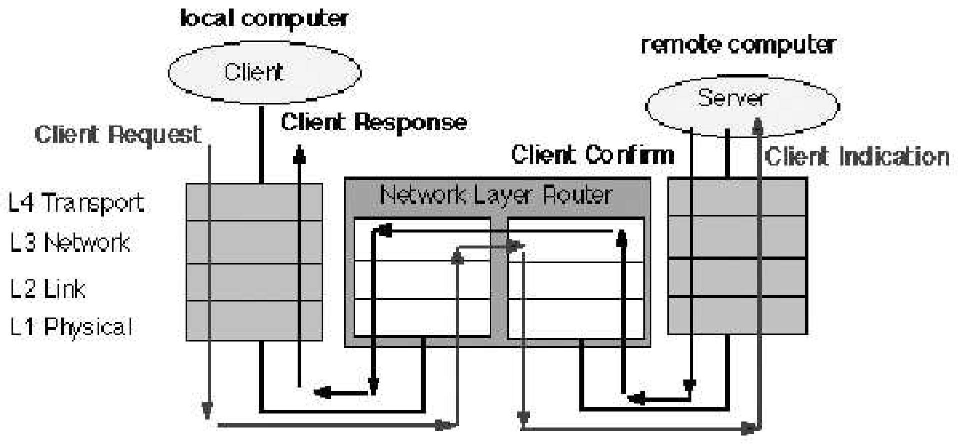


Figure 15.5: Client / Server interaction across a packet data network

Detailed Description

The diagram above provides only a very simplified view of what happens. In fact, a number of primitives are required to perform even such simple communication. The text below performs a detailed analysis of the primitives and PDUs which are exchanged.

The user executes the client program and attempts to send one piece of data to the server (for example,. a request to lookup the address corresponding to a name). The actual data need not be of concern here. Before the client may send the data, it must first establish a connection to the application layer (i.e. communications software library) on the local computer.

The client program uses an A-Associate request function call to start the communication session. Once the application layer has initialized, it contacts the presentation layer, sending a P-Connect request primitive (see below). This establishes the format of the data to be used and the data formats which are to be supported by the system A on the network. This information is sent to the session layer as an SDU with a S-Connect request primitive. The session layer allocates a session identifier, and selects appropriate protocol options to support the services requested by the application layer. The session layer also identifies the intended recipient of the communication (i.e. the remote computer).

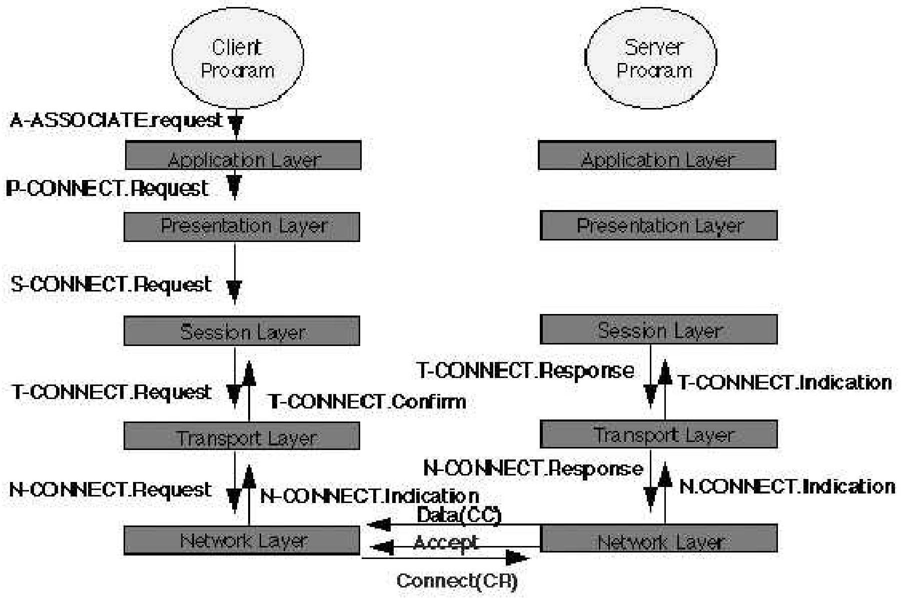


Figure 15.6: Establishment of a connection between client and server (part 1)

The session layer proceeds to request a transport layer connection to the remote system using a T-Connect request. At this stage, the session layer may choose to not send the S-Connect SDU, and instead to wait to see whether the transport connection request succeeds. The transport request identifies the remote service required (i.e. the port identifier for the server) and the type of transport protocol to be used. The basic types of transport service are reliable and best-effort.

At this stage the transport layer requests the network layer to establish a connection to the remote system. The network layer service will normally have established a link layer connection (and a physical layer connection) to the nearest intermediate system. In this case however, we assume that all the layers must be connected.

RIL-089

Finally, the network layer connect packet is sent using the link layer service to the remote system. The system responds by invoking the requested transport layer, and establishing a transport layer connection. If the remote system is able, it will then confirm the establishment of the transport connection, and pass the confirmation back to the local client computer. At this time the two systems have a communications path using the transport layer service.

The transport layer may then use this service to request a connection to the server process on the remote computer. It does this by forwarding the original session layer SDU (which it had previously stored in the session layer of the local computer) as a transport layer packet. This passes across the network layer service, and at the remote system is passed to the session layer protocol.

The received SDU identifies the session ID and the application process with which the client wishes to communicate. The presentation layer is sent a S-Connect indication, containing the presentation options requested by the client and the A-Associate request sent by the client. The application layer now attempts to connect to the server process. (In some cases, a new server program will be activated to handle this new request).

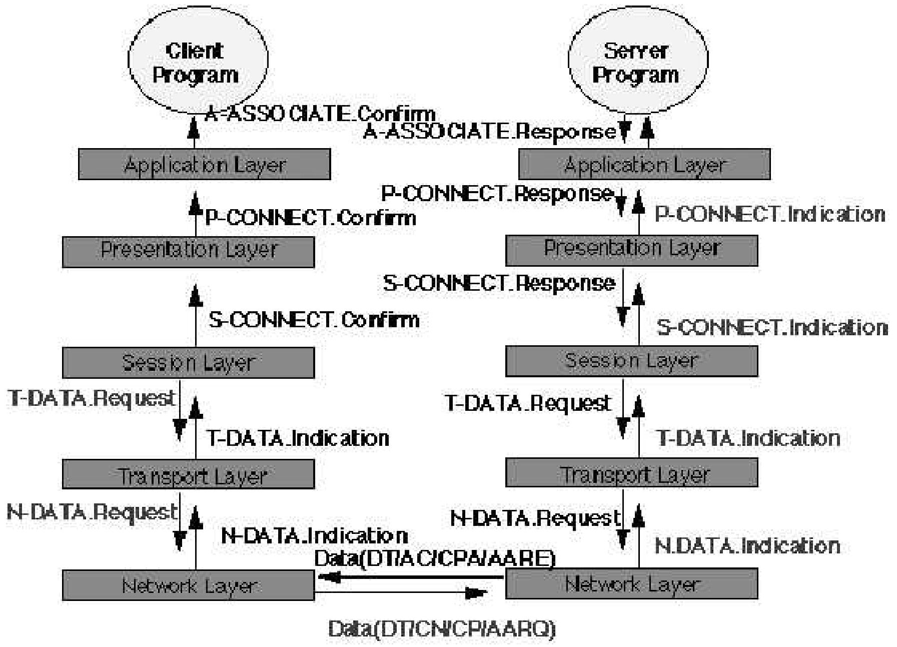


Figure 15.7: Establishment of a connection between client and server (part 2)

Once this has succeeded a response is generated, carrying the final details of the connection. This information is passed as an A-Associate response primitive to the application layer. At each layer additional PCI is added by the remote system, and finally an N-Data primitive is sent across the network layer service. At the receiver each layer verifies the information it receives, and confirms the successful connection. The application layer send an A-Associate confirm message to the client process which is then ready to transmit data.

Transmission of data is accomplished similarly. The client application sends a Data Request primitive, along with the data, to the application layer, which adds its header and passes a Data Request primitive and Data Unit to the presentation layer. This process continues, with addition of appropriate headers, until it reaches the data link layer, which adds its header and passes the individual bits to the physical layer for transmission. The bits are received at the remote physical layer, and passed to the remote system data link layer. When the data link layer recognizes the end of the data unit, it strips off the header and sends the remaining PDU to the network layer (by including it in a Data Indication primitive).

Data Indication primitives, with reduced data units, cascade up the layers until the application data reaches server application. Client application only learns of successful receipt if the server application returns an Acknowledgment (in this case, possibly with the requested data). The Acknowledgment returns across the network. The "association" has now been made and data may be sent across the network between the client and server.

After data transfer is complete, a disconnect phase similar to the connect phase will occur.

Check your progress

Q1. Define session layer.

Q2. What is the use of session layer in OSI-Model?

Q3. What is the purpose of the SCP protocol?

15.4 Summary

In this unit you have learnt about session layer, Dialog Management, Synchronization. OSI Session Primitives Connection Establishment.

- The session layer provides the mechanism for opening, closing and managing a session between end-user application processes, i.e., a semi-permanent dialogue. Communication sessions consist of requests and responses that occur between applications. Session-layer services are commonly used in application environments that make use of remote procedure calls (RPCs).
- An example of a session-layer protocol is the OSI protocol suite session-layer protocol, also known as X.225 or ISO 8327. In case of a connection loss this protocol may try to recover the connection. If a connection is not used for a long period, the session-layer protocol may close it and re-open it. It provides for either full duplex or half-duplex operation and provides synchronization points in the stream of exchanged messages.

- ❖ Other examples of session layer implementations include Zone Information Protocol (ZIP) – the AppleTalk protocol that coordinates the name binding process, and Session Control Protocol (SCP) – the DECnet Phase IV session-layer protocol.
- ❖ Within the service layering semantics of the OSI network architecture, the session layer responds to service requests from the presentation layer and issues service requests to the transport layer.

15.5 Review Questions

Q1. Explain session layer protocols.

Q2. How client server architecture works in session layer?

Q3. Define dialog management in session layer.

Q4. What is the use of synchronization in session layer?

Q5. Explain connection establishment in session layer primitives.

UNIT-XVI

Presentation and Application Layer Protocols

Structure

16.0 Introduction

In this unit we define presentation and application layer protocols. In this unit there are five sections. In Sec.16.1 you will learn about Presentation and Application Layer Protocols. In the presentation layer the functions of encryption and decryption are defined. It converts data formats into a format readable by the application layer. The following are the presentation layer protocols: XDR, TLS, SSL and MIME. If we talk about application layer. The application layer is the seventh layer of the OSI model and the only one that directly interacts with the end user. The application layer provides many services, including Simple Mail Transfer Protocol, File transfer, Web surfing, Web chat, Email clients, Network data sharing, Virtual terminals, Various file and data operations. The application layer provides full end-user access to a variety of shared network services for efficient OSI model data flow. This layer has many responsibilities, including error handling and recovery, data flow over a network and full network flow. It is also used to develop network-based applications. More than 15 protocols are used in the application layer, including File Transfer Protocol, Telnet, Trivial File Transfer Protocol and Simple Network Management Protocol. Its major network device or component is the gateway. In Sec. 16.2, we defined Presentation Concepts NMP- Abstract Syntax Notation-I (ANSI-1). In the next section i.e. Sec. 16.3 you will know about Structure of Management: Management Information Base. In Sec. 16.4 and 16.5 you will find summary and review questions respectively.

Objectives

After studying this unit you should be able to:

- Define Presentation and Application Layer Protocols
- Describe Presentation Concepts NMP- Abstract Syntax Notation-I (ANSI-1)
- Express Structure of Management: Management Information Base.

16.1 Presentation and Application Layer Protocols

Presentation layer

The presentation layer of the Open System Interconnection (OSI) model is responsible for how that data looks or is *formatted*. Consider an example in which spies exchange encoded messages. The manner of passing the messages back and forth is defined by the session layer, but how the messages are encoded (or the cipher the spies used to obscure the message) is the responsibility of the presentation layer.

Naturally, this has to be negotiated between the participants because it would be useless for one spy to encode a message that the other spy did not know how to decode. So with the presentation layer, all participants need to agree with the methods of encoding that are used at this layer.

The same is true in the computer world — all participants, such as the servers and clients, need to agree with how the data will be formatted in order to exchange it. This is why standards for items like the HTML and XML languages allow servers to present data to clients and the clients to display this data to the users.

Tips: Differences among browsers make the actual display of the data slightly different on each browser, partially due to how they honor or interpret the data formatting presented by the web page. This formatting variance is why so many people have multiple web browsers installed on their computers.

Encryption is one of the key translations that takes place at the presentation layer. On outbound traffic from the server, the presentation layer encrypts data that is sent, and on the other end of the connection, it decrypts the data that is sent to the application layer. The following figure 16.1 illustrates the flow of the data between a network client and the server.
If the client computer is running an e-mail program and the server is the user's e-mail server, then on either end of the connection (both the client and the server sides), they are likely using the Simple Mail Transfer Protocol (SMTP) application layer protocol, or rather the encrypted version, SMTPS. The data flow would be as follows:

1. Using SMTPS, the client side of the application passes the text to the presentation layer services and requests encryption.

2. A Transport Layer Security (TLS) component at the presentation layer receives the unencrypted message and proceeds to encrypt the message using standard TLS processes.
3. The encrypted message flows down through the remaining OSI layers, over the physical network to the server.
4. At the server, the message is sent up through all of the layers until it arrives at the presentation layer.

   Now, the servers TLS processes will take over and decrypt the message so that it is clearly readable.
5. The clear text message is then delivered to the SMTP application layer protocol for processing.

In this case, the next step would be to deliver the message to the recipient's mailbox.

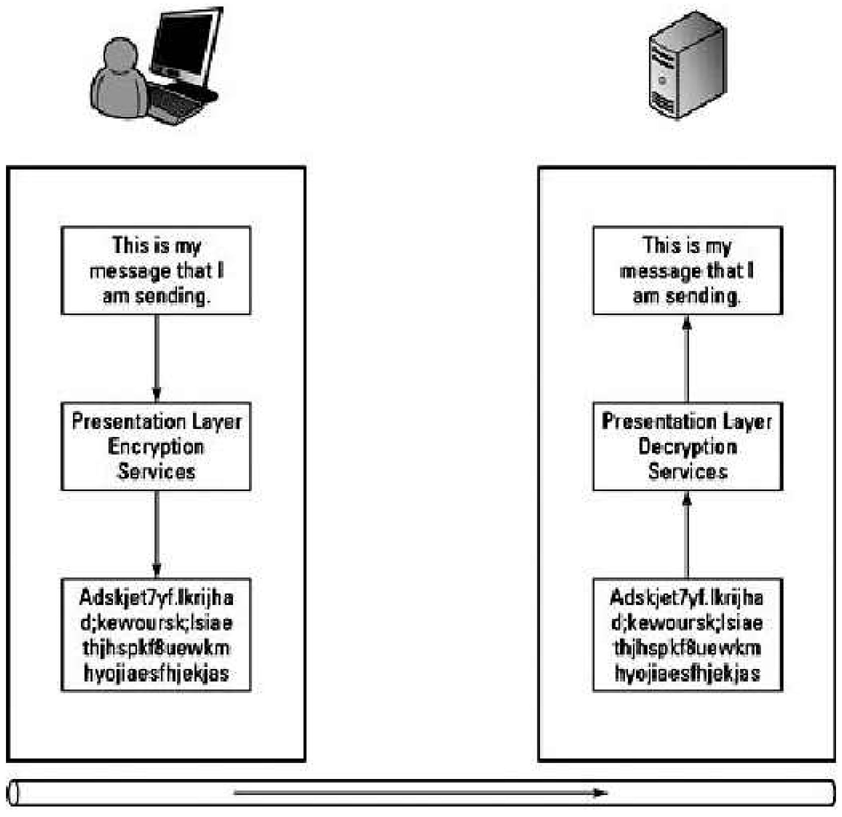

This is my message that I am sending.
Presentation Layer Encryption Services
Adskjet7yf.lkrijha d;kewoursk;lsiae thjhspkf8uewkm hyojiaesfhjekjas
This is my message that I am sending.
Presentation Layer Decryption Services
Adskjet7yf.lkrijha d;kewoursk;lsiae thjhspkf8uewkm hyojiaesfhjekjas

Figure 16.1: Presentation Layer Encryption

Application Layer

Application layer is the top most layer in OSI and TCP/IP layered model. This layer exists in both layered models because of its significance, of interacting with user and user applications. This layer is for applications which are involved in communication system.

A user may or may not directly interacts with the applications. Application layer is where the actual communication is initiated and reflects. Because this layer is on the top of the layer stack, it does not serve any other layers. Application layer takes the help of Transport and all layers below it to communicate or transfer its data to the remote host.

When an application layer protocol wants to communicate with its peer application layer protocol on remote host, it hands over the data or information to the Transport layer. The transport layer does the rest with the help of all the layers below it.

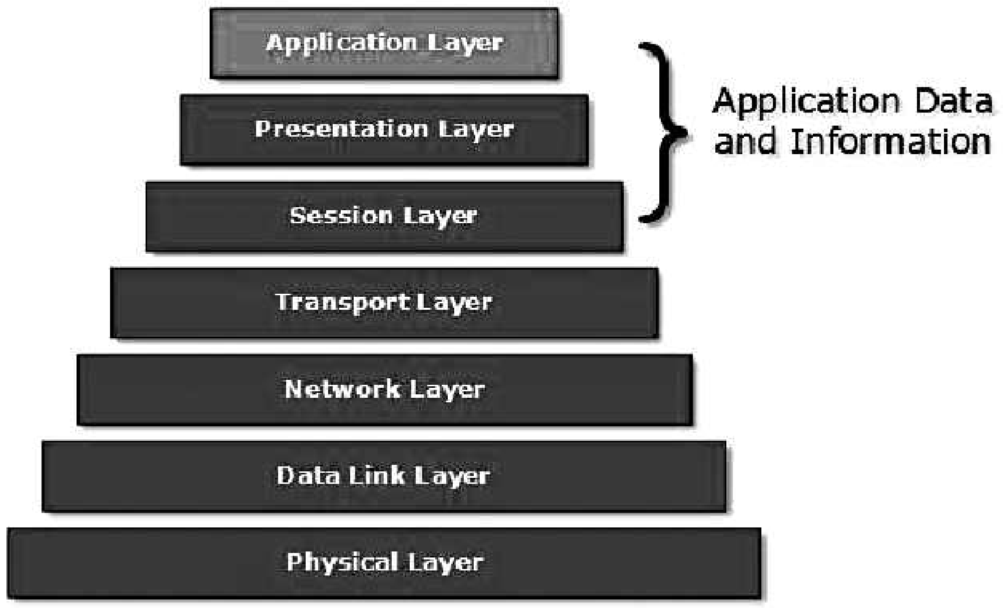


Figure 16.2: Application Layer in OSI

There is an ambiguity in understanding Application Layer and its protocol. Not every user application can be put into Application Layer. Except those applications which interact with the communication system. For example, designing software or text-editor cannot be considered as application layer programs.

On the other hand, when we use a Web Browser, which is actually using Hyper Text Transfer Protocol (HTTP) to interact with the network. HTTP is Application Layer protocol.

Another example is File Transfer Protocol, which helps a user to transfer text based or binary files across the network. A user can use this protocol in either GUI based software like FileZilla or CuteFTP and the same user can use FTP in Command Line mode.

Hence, irrespective of which software you use, it is the protocol which is considered at Application Layer used by that software. DNS is a protocol which helps user application protocols such as HTTP to accomplish its work.

Client Server Model

Two remote application processes can communicate mainly in two different fashions:

- **Peer-to-peer:** Both remote processes are executing at same level and they exchange data using some shared resource.
- **Client-Server:** One remote process acts as a Client and requests some resource from another application process acting as Server.

In client-server model, any process can act as Server or Client. It is not the type of machine, size of the machine, or its computing power which makes it server; it is the ability of serving request that makes a machine a server.

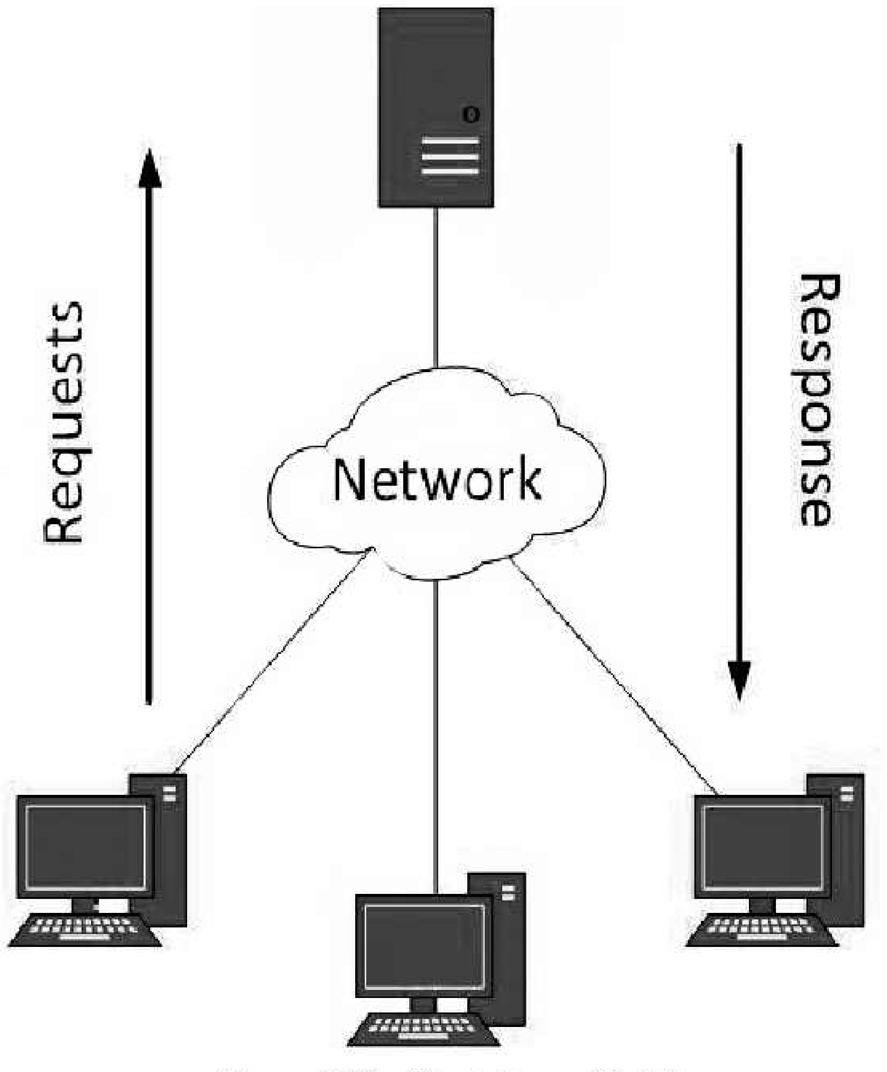


Figure 16.3 : Client-Server Model

A system can act as Server and Client simultaneously. That is, one process is acting as Server and another is acting as a client. This may also happen that both client and server processes reside on the same machine.

Communication

Two processes in client-server model can interact in various ways:

- Sockets
- Remote Procedure Calls (RPC)

Sockets

In this paradigm, the process acting as Server opens a socket using a well-known (or known by client) port and waits until some client request comes. The second process acting as a Client also opens a socket but instead of waiting for an incoming request, the client processes 'requests first'.

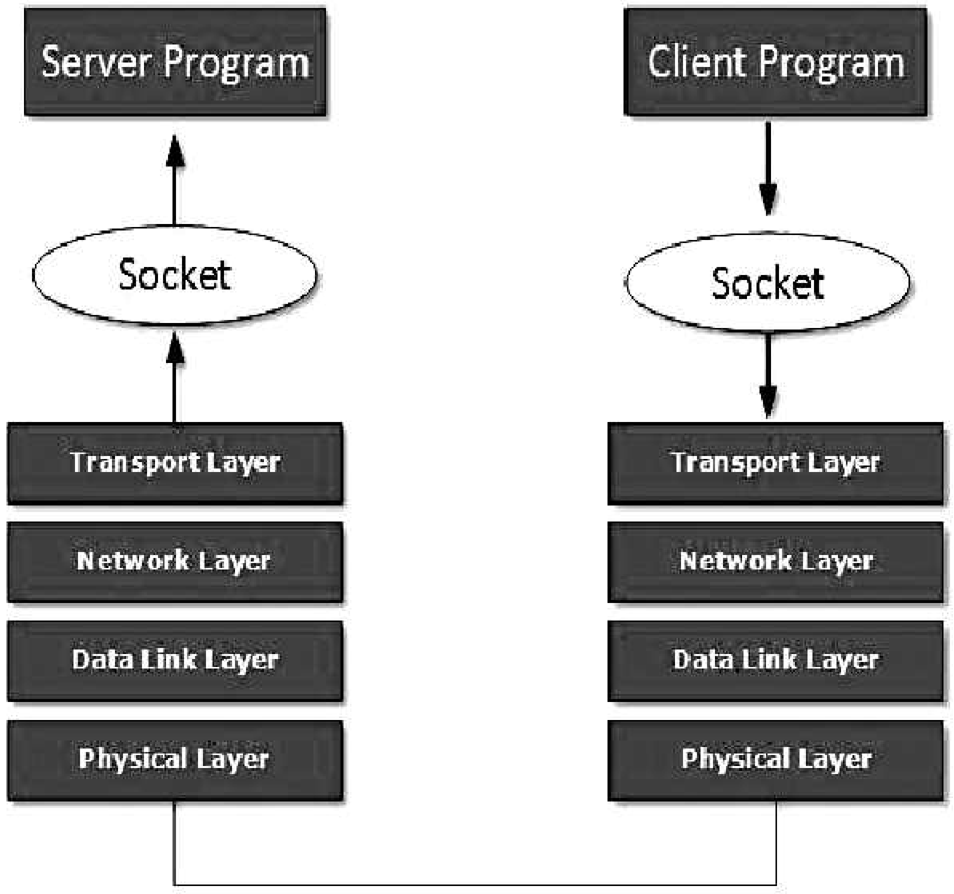


Figure 16.4 : Socket

When the request is reached to server, it is served. It can either be an information sharing or resource request.

Remote Procedure Call

This is a mechanism where one process interacts with another by means of procedure calls. One process (client) calls the procedure lying on remote host. The process on remote host is said to be Server. Both processes are allocated stubs. This communication happens in the following way:

- The client process calls the client stub. It passes all the parameters pertaining to program local to it.
- All parameters are then packed (marshalled) and a system call is made to send them to other side of the network.
- Kernel sends the data over the network and the other end receives it.
- The remote host passes data to the server stub where it is unmarshalled.
- The parameters are passed to the procedure and the procedure is then executed.
- The result is sent back to the client in the same manner.

Application Layer Protocols

There are several protocols which work for users in Application Layer. Application layer protocols can be broadly divided into two categories:

- Protocols which are used by users. For example, E-Mail.
- Protocols which help and support protocols used by users.For example DNS.

Few of Application layer protocols are described below:

Domain Name System

The Domain Name System (DNS) works on Client Server model. It uses UDP protocol for transport layer communication. DNS uses hierarchical domain based naming scheme. The DNS server is configured with Fully Qualified Domain Names (FQDN) and email addresses mapped with their respective Internet Protocol addresses.

A DNS server is requested with FQDN and it responds back with the IP address mapped with it. DNS uses UDP port 53.

Simple Mail Transfer Protocol

The Simple Mail Transfer Protocol (SMTP) is used to transfer electronic mail from one user to another. This task is done by means of email client software (User Agents) the user is using. User Agents help the user to type and format the email and store it until internet is available. When an email is submitted to send, the sending process is handled by Message Transfer Agent which is normally comes inbuilt in email client software.

Message Transfer Agent uses SMTP to forward the email to another Message Transfer Agent (Server side). While SMTP is used by end user to only send the emails, the Servers normally use SMTP to send as well as receive emails. SMTP uses TCP port number 25 and 587.

Client software uses Internet Message Access Protocol (IMAP) or POP protocols to receive emails.

File Transfer Protocol

The File Transfer Protocol (FTP) is the most widely used protocol for file transfer over the network. FTP uses TCP/IP for communication and it works on TCP port 21. FTP works on Client/Server Model where a client requests file from Server and server sends requested resource back to the client.

FTP uses out-of-band controlling i.e. FTP uses TCP port 20 for exchanging controlling information and the actual data is sent over TCP port 21.

The client requests the server for a file. When the server receives a request for a file, it opens a TCP connection for the client and transfers the file. After the transfer is complete, the server closes the connection. For a second file, client requests again and the server reopens a new TCP connection.

Post Office Protocol (POP)

The Post Office Protocol version 3 (POP 3) is a simple mail retrieval protocol used by User Agents (client email software) to retrieve mails from mail server.

When a client needs to retrieve mails from server, it opens a connection with the server on TCP port 110. User can then access his mails and download them to the local computer. POP3 works in two modes. The most common mode the delete mode, is to delete the emails from remote server after they are downloaded to local machines. The second mode, the keep mode, does not delete the email from mail server and gives the user an option to access mails later on mail server.

Hyper Text Transfer Protocol (HTTP)

The Hyper Text Transfer Protocol (HTTP) is the foundation of World Wide Web. Hypertext is well organized documentation system which uses hyperlinks to link the pages in the text documents. HTTP works on client server model. When a user wants to access any HTTP page on the internet, the client machine at user end initiates a TCP connection to server on port 80. When the server accepts the client request, the client is authorized to access web pages.

To access the web pages, a client normally uses web browsers, who are responsible for initiating, maintaining, and closing TCP connections.

HTTP is a stateless protocol, which means the Server maintains no information about earlier requests by clients.

HTTP versions

- HTTP 1.0 uses non persistent HTTP. At most one object can be sent over a single TCP connection.
- HTTP 1.1 uses persistent HTTP. In this version, multiple objects can be sent over a single TCP connection.

Application Layer Network Services

Computer systems and computerized systems help human beings to work efficiently and explore the unthinkable. When these devices are connected together to form a network, the capabilities are enhanced multiple-times. Some basic services computer network can offer are.

Directory Services

These services are mapping between name and its value, which can be variable value or fixed. This software system helps to store the information, organize it, and provides various means of accessing it.

- **Accounting**

  In an organization, a number of users have their user names and passwords mapped to them. Directory Services provide means of storing this information in cryptic form and make available when requested.

- **Authentication & Authorization**

  User credentials are checked to authenticate a user at the time of login and/or periodically. User accounts can be set into hierarchical structure and their access to resources can be controlled using authorization schemes.

- **Domain Name Services**

  DNS is widely used and one of the essential services on which internet works. This system maps IP addresses to domain names, which are easier to remember and recall than IP addresses. Because network operates with the help of IP addresses and humans tend to remember website names, the DNS provides website's IP address which is mapped to its name from the back-end on the request of a website name from the user.

File Services

File services include sharing and transferring files over the network.

RIL-089

- **File Sharing**

One of the reason which gave birth to networking was file sharing. File sharing enables its users to share their data with other users. User can upload the file to a specific server, which is accessible by all intended users. As an alternative, user can make its file shared on its own computer and provides access to intended users.

- **File Transfer**

  This is an activity to copy or move file from one computer to another computer or to multiple computers, with help of underlying network. Network enables its user to locate other users in the network and transfers files.

Communication Services

- **Email**

  Electronic mail is a communication method and something a computer user cannot work without. This is the basis of today's internet features. Email system has one or more email servers. All its users are provided with unique IDs. When a user sends email to other user, it is actually transferred between users with help of email server.

- **Social Networking**

  Recent technologies have made technical life social. The computer savvy peoples, can find other known peoples or friends, can connect with them, and can share thoughts, pictures, and videos.

- **Internet Chat**

  Internet chat provides instant text transfer services between two hosts. Two or more people can communicate with each other using text based Internet Relay Chat services. These days, voice chat and video chat are very common.

- **Discussion Boards**

  Discussion boards provide a mechanism to connect multiple peoples with same interests. It enables the users to put queries, questions, suggestions etc. which can be seen by all other users. Other may respond as well.

- **Remote Access**

  This service enables user to access the data residing on the remote computer. This feature is known as Remote desktop. This can be done via some remote device, e.g. mobile phone or home computer.

Application Services

These are nothing but providing network based services to the users such as web services, database managing, and resource sharing.

- **Resource Sharing**

  To use resources efficiently and economically, network provides a mean to share them. This may include Servers, Printers, and Storage Media etc.

- **Databases**

  This application service is one of the most important services. It stores data and information, processes it, and enables the users to retrieve it efficiently by using queries. Databases help organizations to make decisions based on statistics.

- **Web Services**

  World Wide Web has become the synonym for internet.It is used to connect to the internet, and access files and information services provided by the internet servers.

16.2 Presentation Concepts NMP- Abstract Syntax Notation-I (ANSI-1)

Abstract Syntax Notation (ASN.1) is an ISO standard that addresses the issue of representing, encoding, transmitting, and decoding data structures. It consists of two parts:

1. An abstract syntax that describes data structures in an unambiguous way. The syntax allows programmers to talk about "integers", "character strings", and "structures" rather than bits and bytes.

2. A transfer syntax that describes the bit stream encoding of ASN.1 data objects.

Send: data and additional fields to describe the type of data.

Upon receipt of data, the reverse operation takes place, with the receiver converting from ASN.1 format to the internal representation of the local machine.

Alternative approaches to the data representation problem:

1. The sender have to convert data into the format expected by the receiver, so that the receiver doesn't have to perform any decoding. The disadvantage to this approach is that every sender needs to know how to encode data for every possible target machine.

2. ASN.1 takes the approach of converting everything into a common form, much like the "network standard representation" of TCP/IP. However, this approach has the disadvantage that communication between two identical machines results in (needless) conversions.

ASN.1's abstract syntax is similar in form to that of any high level programming language. For instance, consider the following C structure:

```
struct Student {
  char name[50];    /* ''Foo Bar'' */
  int grad;         /* Grad student? (yes/no) */
  float gpa;        /* 1.1 */
  int id;           /* 1234567890 */
  char bday[8];   /* mm/dd/yy */
}
```

```
Its ASN.1 counterpart is:

  Student ::= SEQUENCE {
    name        OCTET STRING, -- 50 characters
    grad        BOOLEAN,      -- comments preceded
    gpa         REAL,         -- by ''--''
    id          INTEGER,
    bday OCTET STRING   -- birthday
  }
```

Figure 16-5 : C Structure

ASN.1 consists of primitive types, and complex types built from primitive types. The names of primitive types are written (by convention) in upper case, and the following primitive types are defined:

INTEGER:	An integer (of arbitrary length).
BOOLEAN:	TRUE or FALSE.
BIT STRING:	Sequence of zero or more bits. Bit string values are written as '01001101'B (for binary) or '4D'H (in hex).
OCTET STRING:	List of zero or more bytes. Used to represent strings, they have no maximum length.
ANY:	A union of all types (e.g. one of several different possible values).
REAL:	A real number.
NULL:	No type at all, corresponds to "NIL" in C.
OBJECT IDENTIFIER:	The name of an object (e.g. library).When a session is established, both sides negotiate about which ASN.1 objects they will be using.

Primitive types can be combined into more complex types. ASN.1 provides following constructors:

SEQUENCE:	Ordered list of various types (like a C structure). Types can be either primitive or complex types.
SEQUENCE OF:	Ordered list of a single type (e.g., an array).
SET:	Unordered collection of various types.

SET OF:	Unordered collection of a single type. Note: SET and SET OF are similar to SEQUENCE and SEQUENCE OF, except that the order of components is not guaranteed to be preserved at the receiver. Using SET or SET OF may reduce the amount of copying relative to SEQUENCE or SEQUENCE OF.
CHOICE:	Any one type taken from a given list.

16.3 Structure of Management: Management Information Base.

A Management Information Base (MIB) describes a set of managed objects. An SNMP management console application can manipulate the objects on a specific computer if the SNMP service has an extension agent DLL that supports the MIB.

Each managed object in a MIB has a unique identifier. The identifier includes the object's type (such as counter, string, gauge, or address), the object's access level (such as read or read/write), size restrictions, and range information.

The following table contains a partial list of the MIBs that ship with the system. They are installed with the SNMP service in the %systemroot%\system32 directory. For a complete listing of MIBs, refer to the Windows Resource Kit.

MIB	Description
DHCP.MIB	Microsoft-defined MIB that contains object types for monitoring the network traffic between remote hosts and DHCP servers
HOSTMIB.MIB	Contains object types for monitoring and managing host resources
LMMIB2.MIB	Covers workstation and server services
MIB_II.MIB	Contains the Management Information Base (MIB-II), which provides a simple, workable architecture and system for managing TCP/IP-based internets
WINS.MIB	Microsoft-defined MIB for the Windows Internet Name Service (WINS)

RIL-089

Table 16-1 MIB

Windows NT

Typically, you can copy a MIB from the SNMP extension agent that supports the particular MIB. Some additional MIBs are available with the Windows NT 4.0 Resource Kit.

The extension agent DLLs for MIB-II, LAN Manager MIB-II, and the Host Resources MIB are installed with the SNMP service. The extension agent DLLs for the other MIBs are installed when their respective services are installed. At service startup time, the SNMP service loads all of the extension agent DLLs that are listed in the registry.

Users can add other extension agent DLLs that implement additional MIBs. To do this, they must add a registry entry for the new DLL under the SNMP service. They must also register the new MIB with the SNMP management console application. For more information, see the documentation included with your management console application.

MIB Source

MIBs may come from various sources:

- Standard- on the IETF standards track at Proposed, Draft, or full standard. A Proposed Standard can change somewhat due to implementation experience. A Draft Standard changes somewhat less, with more attention to backward compatibility. A full Internet Standard doesn't change much. At all levels these are published as Requests for Comment (RFCs).

- Internet Draft-IETF work in progress. Sometimes the best way to instrument technology is with an Internet Draft MIB, which is typically being worked on by an IETF working group. Such MIBs are somewhat unstable, so it is necessary to capture the specific Internet Draft and to place the MIB within the Cisco Enterprise MIB space (not in the Experimental branch).

- Cisco-Cisco enterprise-specific (also called proprietary or private, even though publicly documented). Such MIBs add instrumentation not covered by standard MIBs. As of IOS Version 10.2, Cisco has old MIBs and new MIBs. The old MIBs are from older software versions and often have somewhat unconventional features.

- Other companies- non-Cisco enterprise-specific. It is occasionally appropriate to implement a MIB defined by some other company, especially when implemented technology they originated and instrumented. This has similar problems to Internet Drafts in that a version of the MIB definition must be captured, but the MIB itself should remain wherever in the MIB space the originating company placed it so as to easily support existing applications.

MIB Objects

A MIB is a tree where the leaves are individual items of data called objects. An object may be, for example, a counter or a protocol status. MIB objects are also sometimes called variables. Note that the SNMP framework uses object in a somewhat different way than does OSI management. An OSI object is a network entity, such as a router or a protocol, which has attributes. These OSI attributes and SNMP objects are essentially the same concept, that is, individual data values. See Appendix A for a detailed description. MIB object consists of the following values:

- Object type-identifies the type of MIB object.
- Syntax-identifies the data type which models the object.
- Access-identifies the maximum level of access and can have one of five values (listed from highest to lowest):
    - Read-create-indicates that instances of the object may be read, written, and created.
    - Read-write-indicates that instance of the object may be read or written, but not created.
    - Read-only-indicates that instances of the object may be read but not written or created.
    - Accessible-for-notify-indicates that instances of the object may only appear in notifications.
    - Not-accessible-indicates that instances of the object may not be directly read, written, or created.
- Status-the status of a managed object can be:
    - Mandatory-indicates that the definition is required and should be implemented.
    - Current-indicates that the definition is current.
    - Deprecated-indicates that the definition will soon be made obsolete and need no longer be implemented.
    - Obsolete-indicates that managed nodes should not implement the object.
- Description-provides a textual description of the managed object

Check your progress

Q1. Define presentation layer.

Q2. Explain application layer.

16.4 Summary

In this unit you have learnt about Presentation and Application Layer Protocols, Presentation Concepts NMP- Abstract Syntax Notation-I (ANSI-1), and Structure of Management: Management Information Base.

- The presentation layer mainly translates data between the application layer and the network format. Data can be communicated in different formats via different sources. Thus, the presentation layer is responsible for integrating all formats into a standard format for efficient and effective communication.
- The presentation layer follows data programming structure schemes developed for different languages and provides the real-time syntax required for communication between two objects such as layers, systems or networks. The data format should be acceptable by the next layers; otherwise, the presentation layer may not perform correctly.
- Network devices or components used by the presentation layer include redirectors and gateways.
- The application layer is a layer in the Open Systems Interconnection (OSI) seven-layer model and in the TCP/IP protocol suite. It consists of protocols that focus on process-to-process communication across an IP network and provides a firm communication interface and end-user services.
- In a managed device, specialized low-impact software modules, called agents, access information about the device and make it available to the network management system (NMS).
- Managed devices maintain values for a number of variables and report those, as required, to the NMS. For example, an agent might report such data as the number of bytes and packets in and out of the device, or the number of broadcast messages sent and received.
- In the Internet network management framework, each variable is referred to as a managed object, which is anything that an agent can access and report back to the NMS.

16.5 Review Questions

Q1. Why presentation layer protocols needed?

Q2. Define all application layer protocols.

Q3. What is ANSI-I? Elaborate your answer.

Q4. Why do we needed MIB (Management Information Base)? Explain.

BIBLIOGRAPHY

Behrouz A. Forouzan. *Data Communications and Networking, Fourth Edition.* McGraw-Hill, 2007.

Larry L. Peterson and Bruce S. Davie. *Computer Networks: A Systems Approach, 4th Edition.* Morgan Kaufmann, 2007.

William Stallings. *Data and Computer Communications, Eighth Edition.* Prentice Hall, 2006.

Andrew S. Tanenbaum. *Computer Networks, Fourth Edition.* Prentice Hall PTR, 2003.

Alan Dennis. *Networking in the Internet Age.* John Wiley & Sons, 2002.

Charles E. Perkins, editor. *Ad Hoc Networking.* Addison-Wesley, 2001.

Radia Perlman. *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols, Second Edition.* Addison-Wesley, 2000.

Dimitri Bertsekas and Robert Gallager. *Data Networks, Second Edition.* Prentice Hall, 1992.

Douglas E. Comer. *Computer Networks and Internets, Sixth Edition.* Prentice-Hall, 2015.

Nader F. Mir. *Computer and Communication Networks, Second Edition.* Prentice-Hall, 2015.

James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach, Sixth Edition.* Addison-Wesley, 2013.

Jeffrey S. Beasley and Piyasat Nilkaew. *Networking Essentials, Third Edition.* Pearson Education, 2012.

Douglas Comer, David Stevens, and Michael Evangelista. *Internetworking with TCP/IP, Volume 2: Design, Implementation, and Internals, Fourth Edition.* Prentice-Hall, 2006.

Douglas Comer and David Stevens. *Internetworking with TCP/IP, Volume 3: Client-Server Programming and Applications, Linux/Posix Sockets Version.* Prentice-Hall, 2001.

Douglas Comer. *Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architecture, Fourth Edition.* Prentice-Hall, 2000.

Warren W. Gay. *Linux Socket Programming By Example.* Que, 2000.

Douglas Comer and David Stevens. *Internetworking with TCP/IP, Volume 3: Client-Server Programming and Applications, Windows Sockets Version.* Prentice-Hall, 1997.

Douglas Comer and David Stevens. *Internetworking with TCP/IP, Volume 3: Client-Server Programming and Applications, BSD Sockets Version, Second Edition.* Prentice-Hall, 1996.

Sidnie Feit. *TCP/IP: Architecture, Protocols, and Implementation.* McGraw-Hill, 1993.