SRINIVAS UNIVERSITY

CITY CAMPUS PANDESHWAR
MANGALURU- 575001.

INSTITUTE OF COMPUTER & INFORMATION SCIENCE

BACKGROUND STUDY MATERIAL

PRINCIPLES OF TCP/IP

V SEMESTER B.C.A

Compiled by

Faculty

TEACHING PLAN

SUBJECT: PRINCIPLES OFTCP/IP

SUBJECT CODE: 17BCASD44

FACULTY: PROF. SRIDHARA ACHARYA

Learning Objective of the paper:

The main objective of this paper is to make the student understand the concept of communication in detail. Using this knowledge the students can work in the field of communication standard, protocol design, encryption and decryption standards etc. This knowledge is very essential to work in any web based client/server applications.

Learning Outcome of the paper:

Students will learn the basics of communication, various protocols, datagram, IPV4 and IPV6, classfull and classless addressing scheme, domain name system etc. Overall this paper will make the students to learn the complete knowledge in communication.

Unit 1

8 Hours

Session 1: Evaluation of open network, Layering of communication process
Session 2: Layering and Standardization.
Session 3: Internet concept, Internet architectural model.
Session 4: Internet Addresses, Link layer encapsulation
Session 5: Physical addresses, IEEE & MACs
Session 6: ARP operations, Cache & timeouts.
Session 7: RARP overview, operations, Primary and Backup RARP Servers.
Session 8: Loopback Interface

Unit 2

8 Hours

Session 9: IP Routing Principles
Session 10: Routing IP Datagrams,
Session 11: Table – Driven IP forwarding
Session 12: Routing information Protocol (RIP)
Session 13: The open SPF Protocol (OSPF)
Session 14: Delay metric (HELLO)
Session 15: Static Vs Dynamic Interior Routers (BGP), Trace route program
Session 16: CIDR- Subnetting, VLSM, Supernetting

Unit 3

8 Hours

Session 17: UDP Header, UDP Checksum
Session 18: UDP Multiplexing, Demultiplexing & Ports
Session 19: Maximum Datagram Size.
Session 20: Sliding Windows.
Session 21: TCP-Passive and Active opens.
Session 22: RTT Estimation
Session 23: TCP connection establishment.
Session 24: TCP Connection Termination.

Unit 4

8 Hours

Session 25: Server side silly window avoidance,
Session 26: Delayed Acknowledgement and Nagles Algorithm,
Session 27: receiver side silly window avoidance,

Session 28: TCP Timers
Session 29: Multicasting – IP Multicast Addresses, IGMP.
Session 30: DNS – Basics, Resolution, Caching, DNS Message Format,
Session 31: TELNET Protocol,
Session 32: Rlogin – protocol.

Unit 5

8 Hours

Session 33: FTP – Protocol, Process Model.,
Session 34 TFTP
Session 35: NFS
Session 36: SMTP
Session 37: POP
Session 38: IMAP, MIME.
Session 39: IPV6 – Features, Datagram format,
Session 40: Use of Multiple Headers, IPV4 Vs IPV6.

6 Hours

Value added Chapter Internet security and firewall
Session 41: Introduction
Session 42: Protecting resources
Session 43: Information Policy
Session 44: Internet Security.
Session 45: Firewalls and internet access
Session 46:Assignment Questions

Text Book:
1. Douglas E Comer, **Internetworking with Principles, Protocols and Architecture** Vol- I, 5th Edition, PHI Learning 2010.

Reference Books

1. Peter Loshin, **Clearly Explained**, Elsevier India 1999
2. Behrouz A. Forouzan, **Protocol Suit,** 2nd edition, Tata Mc-Grow-Hill publications,2003

Scheme of Evaluation:
The paper carries 100 marks out of which 50 marks will be allotted to external examination and 50 marks will be allotted to the internal assessment.
Internal assessment marks will be calculated as follows
1. Performance in 2 IA examinations will be converted out of 30 marks
2. Attendance 10 marks
3. Assignment 10 marks.

Total 50 marks.

External examination marks will be as follows

1.	1 marks questions 10 out of 12	1 X 10 = 10 marks.
2.	One full question out of 2 full questions in each unit carries	8 X 5 = 40 marks
		Total 50 marks.

In order to clear this paper minimum 50% marks must be scored both in internal and well as external examination.

****

UNIT I

CHAPTER 1

EVOLUTION OF OPEN NETWORKS

1.1 Motivation for networking

Internet communication has become a fundamental part of life. The World Wide Web contains information about such diverse subjects as atmospheric conditions, crop production, stock prices, and airline traffic. Groups establish electronic mailing lists so they can share information of common interest. Professional colleagues exchange business correspondence electronically, and relatives exchange personal greetings. Although it appears to operate as a unified network, the Internet is not engineered from a single networking technology because no technology suffices for all uses. Instead, networking hardware is designed for specific situations and budgets. Some groups need high-speed networks to connect computers in a single building. Because low-cost hardware that works well inside a building cannot span large geographic distances, an alternative must be used to connect machines thousands of miles apart.

In the past twenty-five years, a technology has been created that makes it possible to interconnect many disparate physical networks and operate them as a coordinated unit. Known as *internetworking,* the technology forms the basis for the Internet by accommodating multiple, diverse underlying hardware technologies, providing a way to interconnect the networks, and defining a set of communication conventions that the networks use to interoperate. The internet technology hides the details of network hardware, and permits computers to communicate independent of their physical network connections.

The internet technology described in this book is an example of *open system interconnection.* It is called *open* because, unlike proprietary communication systems available from one specific vendor, the specifications are publicly available. Thus, anyone can build the software needed to communicate across an internet. More important, the entire technology has been designed to foster communication among machines with diverse hardware architectures, to use almost any packet switched network hardware, to accommodate a wide variety of applications, and to accommodate arbitrary computer operating systems.

1.2 Internet

U.S. government agencies realized the importance and potential of internet technology many years ago, and funded research that made possible a global Internet. This book discusses principles and ideas that resulted from research funded by the *Defense Advanced Research*

Projects Agency (DARPA). The DARPA technology includes a set of network standards that specify the details of how computers communicate, as well as a set of conventions for interconnecting networks and forwarding traffic. Officially named the *Internet Protocol Suite* and commonly referred to as (after the names of its two main standards), it can be used to communicate across any set of interconnected networks. For example, can be used to interconnect a set of networks within a single building, within a physical campus, or among a set of campuses. Although the technology is noteworthy by itself, it is especially interesting because its viability has been demonstrated on a large scale. It forms the base technology for the global Internet that connects over 650 million individuals in homes, schools, corporations, and government labs in virtually all populated areas. An outstanding success, the Internet demonstrates the viability of the technology and shows how it can accommodate a wide variety of underlying hardware technologies.

1.3 History and Scope of Internet

Part of what makes the technology so exciting is its universal adoption as well as the size and growth rate of the global Internet. DARPA began working toward an internet technology in the mid 1970s, with the architecture and protocols taking their current form around 1977-79. At that time, DARPA was known as the primary funding agency for packet-switched network research, and pioneered many ideas in packet switching with its well-known *ARPANET.* The ARPANET used conventional point-to point leased line interconnections, but DARPA also funded exploration of packet switching over radio networks and satellite communication channels. Indeed, the growing diversity of network hardware technologies helped force DARPA to study network interconnection, and pushed internetworking forward. The availability of research funding from DARPA caught the attention and imagination of several research groups, especially those researchers who had previous experience using packet switching 'on the ARPANET. DARPA scheduled informal meetings of researchers to share ideas and discuss results of experiments. Informally, the group was known as the *Internet Research Group.* By 1979, so many researchers were involved in the effort that DARPA created an informal committee to coordinate and guide the design of the protocols and architecture of the emerging Internet. Called the *Internet Control and Configuration Board (ICCB),* the group met regularly until 1983, when it was reorganized.

The global Internet began around 1980 when DARPA started converting machines attached to its research networks to the new protocols. The ARPANET, already in place, quickly became the backbone of the new Internet and was used for many of the early experiments with . The transition to Internet technology became complete in January 1983 when the Office of the

Secretary of Defense mandated that all computers connected to long-haul networks use . At the same time, the *Defense Communication Agency (DCA)* split the ARPANET into *two* separate networks, one for further research and one for military communication. The research part retained the name ARPANET; the military part, which was somewhat larger, became known as the *military network, (MILNET).*

To encourage university researchers to adopt and use the new protocols, DARPA made an implementation available at low cost. At that time, most university computer science departments were running a version of the UNIX operating system available in the University of California's *Berkeley Software Distribution,* commonly called *Berkeley UNIX* or *BSD UNIX.* By funding Bolt Beranek and Newman, Incorporated *(BBN)* to implement its TCPIIP protocols for use with UNIX and funding Berkeley to integrate the protocols with its software distribution, DARPA was able to reach over 90% of university computer science departments. The new protocol software came at a particularly significant time because many departments were just acquiring second or third computers and connecting them together with local area networks. The departments needed communication protocols that provided application services such as file transfer.

Besides a set of utility programs, Berkeley UNIX provided a new operating system abstraction known as a *socket* that allowed application programs to access communication protocols. A generalization of the UNIX mechanism for I/O, the socket interface has options for several types of network protocols in addition to . The introduction of the socket abstraction was important because it allowed programmers to use TCPIIP protocols with little effort. The socket interface has become a de facto standard, now used in most operating systems.

Realizing that network communication would soon be a crucial part of scientific research, the National Science Foundation *(NSF)* took an active role in expanding the TCPIIP Internet to reach as many scientists as possible. In the late 1970s, NSF funded a project known as the *Computer Science NETwork (CSNET),* which had as its goal connecting all computer scientists. Starting in 1985, NSF began a program to establish access networks centered around its six supercomputer centers, and in 1986 expanded networking efforts by funding a new wide area backbone network, known as the *NSFNET backbone.* NSF also provided seed money for regional networks, each of which connected major scientific research institutions in a given area. Within seven years of its inception, the Internet had grown to span hundreds of individual networks located throughout the United States and Europe. It connected nearly 20,000 computers at universities, government, and corporate research laboratories. Both the size and the use of the Internet continued to grow much faster than anticipated. By late 1987, it was estimated that the growth had reached 15% per month. By 2005, the global Internet reached nearly 300 million computers in 209 countries.

Early adoption of protocols and growth of the Internet was not limited to government-funded projects. Major .computer corporations connected to the Internet as did many other large corporations including: oil companies, the auto industry" electronics firms, pharmaceutical companies, and telecommunications carriers. Medium and small companies began connecting in the 1990s. In addition, many companies used protocols on their internal corporate intranets even if they chose not to be part of the global Internet.

Chapter 2

LAYERING OF COMMUNICATION PROCESS

2.1Introduction

This chapter makes a conceptual leap by describing a scheme that al1ows us to col1ect the diverse network technologies into a coordinated whole. The primary goal is a system that hides the details of underlying network hardware while providing universal communication services. The primary result is a high-level abstraction that provides the framework for al1design decisions.

2.2Application level inter connection

Designers have taken two different approaches to hiding network details, using application programs to handle heterogeneity or hiding details in the operating system. Early heterogeneous network interconnections provided uniformity through application level programs cal1ed *application gateways.* In such systems, an application-level program, executing on each computer in the network, understands the details of the network connection for that computer, and interoperates across those connections with application programs on other computers. For example, some electronic mail systems consist of mail programs that are each configured to forward a memo to a mail program on the next computer. The path from source to destination may involve many different networks, but that does not matter as long as the mail systems on all the machines cooperate by forwarding each message. Using application programs to hide network details may seem natural at' first, but such an approach results in limited, cumbersome communication. Adding new functionality to the system means building a new application program for each computer. Adding new network hardware means modifying existing programs (or creating new programs) for each possible application. On a given computer, each application program must understand the network connections for the computer, resulting in duplication of code. Users who are experienced with networking understand that once the interconnections grow to hundreds or thousands of networks, no one can possibly build all the necessary application programs. Furthermore, success of the step-at-a-time communication scheme requires correctness of all application programs executing along the path. When an intermediate program fails, the source and destination remain unable to detect or control the problem. Thus, systems that use intermediate applications programs cannot guarantee reliable communication.

2.3 Network level inter connection

The alternative to providing interconnection with application-level programs is a system based on network-level interconnection. A network-level interconnection provides a mechanism that delivers small packets of data from their original source to their ultimate destination without using intermediate application programs. Switching small units of data instead of files or large messages has several advantages. First, the scheme maps directly onto the underlying network hardware, making it extremely efficient. Second, network-level interconnection separates data communication activities from application programs, permitting intermediate computers to handle network traffic without understanding the applications that are sending or receiving it. Third, using network connections keeps the entire system flexible, making it possible to build general purpose communication facilities. Fourth, the scheme allows network managers to add new network technologies by modifying or adding a single piece of new network level software, while application programs remain unchanged. The key to designing universal network-level interconnection can be found in an abstract communication system concept known as *internetworking.* The internetwork, or *internet,* concept is an extremely powerful one. It detaches the notions of communication from the details of network technologies and hides low-level details from the user. More important, it drives all software design decisions and explains how to handle physical addresses and routes. After reviewing basic motivations for internetworking, we will consider the properties of an internet in more detail. We begin with two fundamental observations about the design of communication systems:

- No single network hardware technology can satisfy all constraints.
- Users desire universal interconnection.

The first observation is an economic as well as technical one. Inexpensive Local Area Networks that provide high speed communication only cover short distances; wide area networks that span long distances cannot supply local communication cheaply. Because no single network technology satisfies all needs, we are forced to consider multiple underlying hardware technologies. The second observation is self-evident. Ultimately, users would like to be able to communicate between any two points. In particular, we desire a communication system that is not constrained by the boundaries of physical networks. The goal is to build a unified, cooperative interconnection of networks that supports a universal communication service. Within each network, computers will use underlying technology-dependent communication facilities. New software, inserted between the technology-dependent communication mechanisms and application programs, will hide the low-level details and make the collection of networks appear to be a single large network. Such an interconnection scheme is called an *internetwork* or *internet.* The idea of building an internet follows a' standard pattern- of system design: researchers imagine a high-level computing facility and work from available computing technology, adding layers of software until they have a system that efficiently implements the imagined high-level facility. The next section shows the first step of the design process by defining the goal more precisely.

2.4The Conceptual Layers Of Protocol Software

Think of the modules of protocol software on each machine as being stacked vertically into layers, as in Figure 2.1 Each layer takes responsibility for handling onepart of the problem.

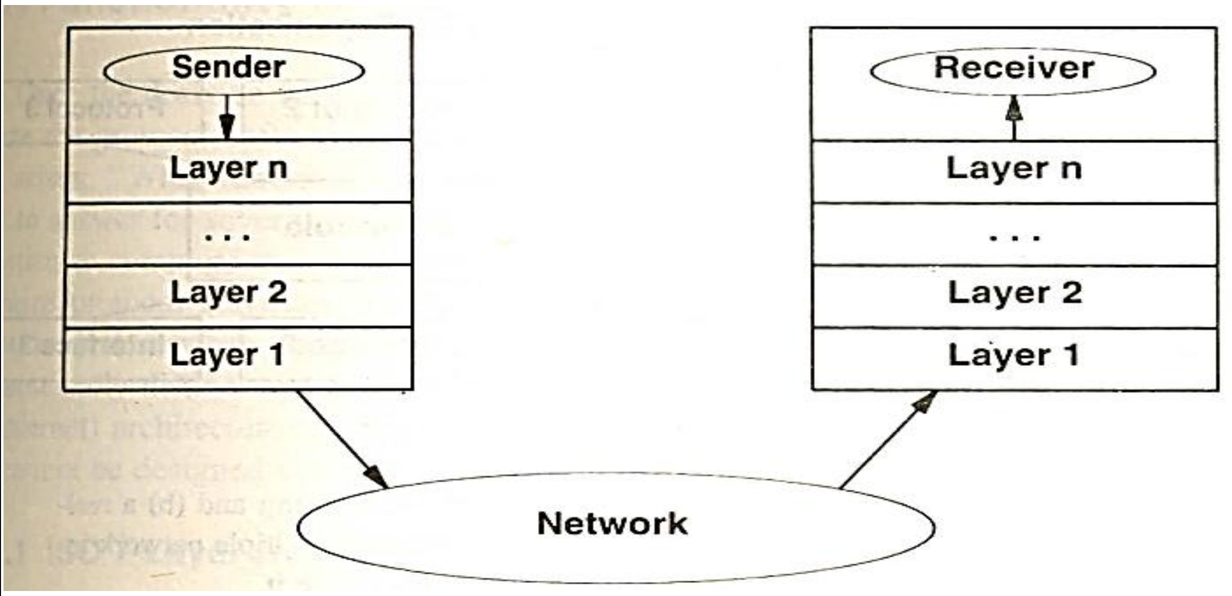


Figure 2.1 Conceptual organization of protocol software in layers

Conceptually, sending a message from an application program on one machine to an application program on another means transforming the message down through successive layers of protocol software on the sender's machine, forwarding the message across the network, and transforming the message up through successive layers of protocol software on the receiver's machine.

In practice, the protocol software is much more complex than the simple model of Figure 2.1 indicates. Each layer makes decisions about the correctness of the message and chooses an appropriate action based on the message type or destination address.

For example, one layer on the receiving machine must decide whether to keep the message or forward it to another machine. Another layer must decide which application program should receive the message. To understand the difference between the conceptual organization of protocol software and the implementation details, consider the comparison shown in Figure 2.2. The conceptual diagram in Figure 2.2a shows an Internet layer between a high level protocol layer and a network interface layer. The realistic diagram in Figure 2.2b shows that the IP software may communicate with multiple high-level protocol modules and with multiple network interfaces.

Although a diagram of conceptual protocol layering does not show all details, it does help explain the general concept. For example, Figure 11.3 shows the layers of protocol software used by a message that traverses three networks. The diagram shows only the network interface and Internet Protocol layers in the routers because only those layers are needed to receive, route, and send datagram. We understand that any machine attached to two networks must have two network interface modules, even though the conceptual layering diagram shows only a single network interface layer in each machine.

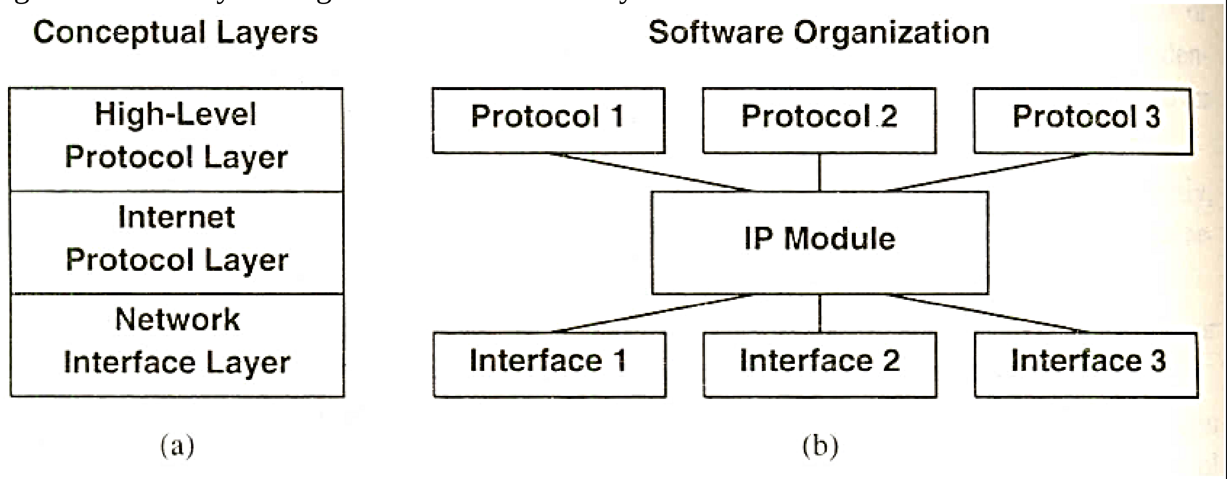


Figure 2.2 A comparison of (a) conceptual protocol layering and (b) a realistic view of software organization showing multiple network interfaces below IP and multiple protocols above it.

As Figure 2.3 shows, a sender on the original machine transmits a message which the IP layer places in a datagram and sends across network 1. On intermediate routers, the datagram passes up to the IP layer which sends it back out again (on a different network). Only when it reaches the final destination machine, does IP extract the message and pass it up to higher layers of protocol software.

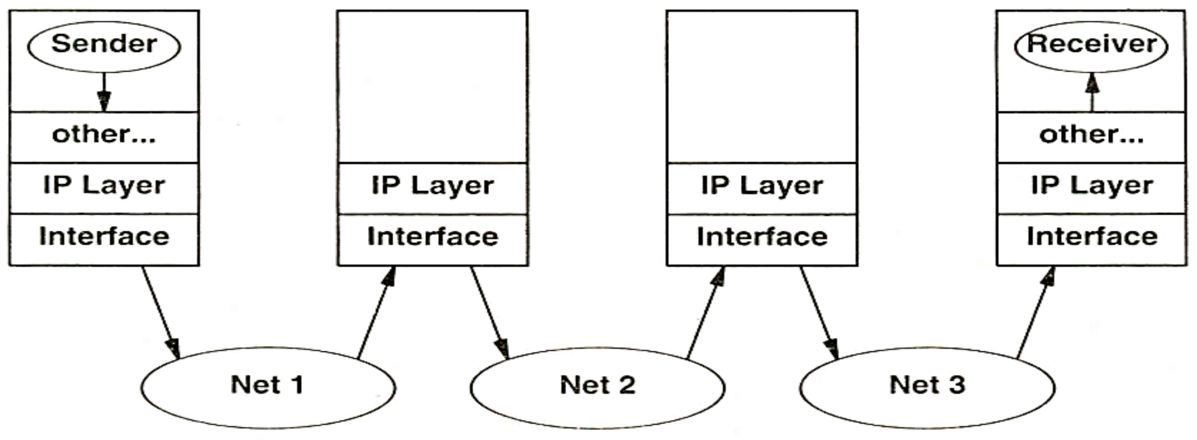


Figure 2.3 The path of a message traversing the Internet from the senderthrough two intermediate routers to the receiver. Intermediaterouters only send the datagram to the 1P software layer.

2.5Functionality Of The Layers

Once the decision has been made to partition the communication problem and organize the protocol software into modules that each handle one sub problem, the question arises: "what functionality should reside in each module?" The question is not easy to answer for several reasons. First, given a set of goals and constraints governing a particular communication problem, it is possible to choose an organization that will optimize protocol software for that problem. Second, even when considering general network-level services such as reliable transport, it is possible to choose from among fundamentally distinct approaches to solving the problem. Third, the design of network (or internet) architecture and the organization of the protocol software are interrelated; one cannot be designed without the other.

2.6 IS0 7-Layer Reference Mode

Two ideas about protocol layering dominate the field. The first, based on early work done by the International Organization for Standardization (ISO), is known as ISO's Reference Model of Open System Interconnection, often referred to as the IS0 model. The IS0 model contains 7 conceptual layers organized as Figure 2.4 shows.

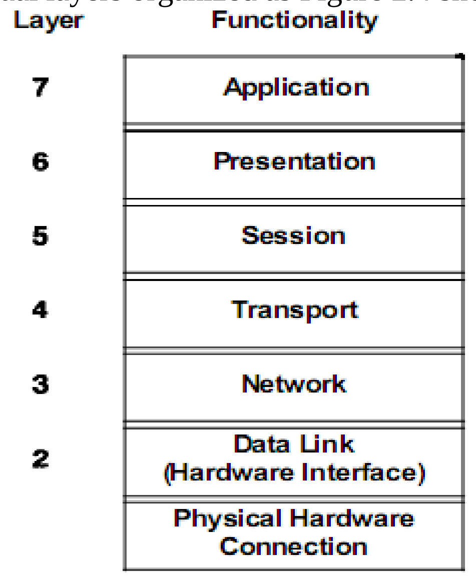


Figure 2.4 ISO 7 layer reference model for protocol software.

The IS0 model, built to describe protocols for a single network, does not contain a specific layer for internetwork routing in the same way protocols do.

2.7 The TCPAP 5-Layer Reference Model

The second major layering model did not arise from a standards committee, butcame instead from research that led to the TCPIIP protocol suite. With a little work, the IS0 model can be stretched to describe the TCPAP layering scheme, but the underlying assumptions are different enough to warrant distinguishing the two.

Broadly speaking, TCPDP software is organized into five conceptual layers - four software layers that build on a fifth layer of hardware. Figure 2.5 shows the conceptual layers as well as the form of data as it passes between them.

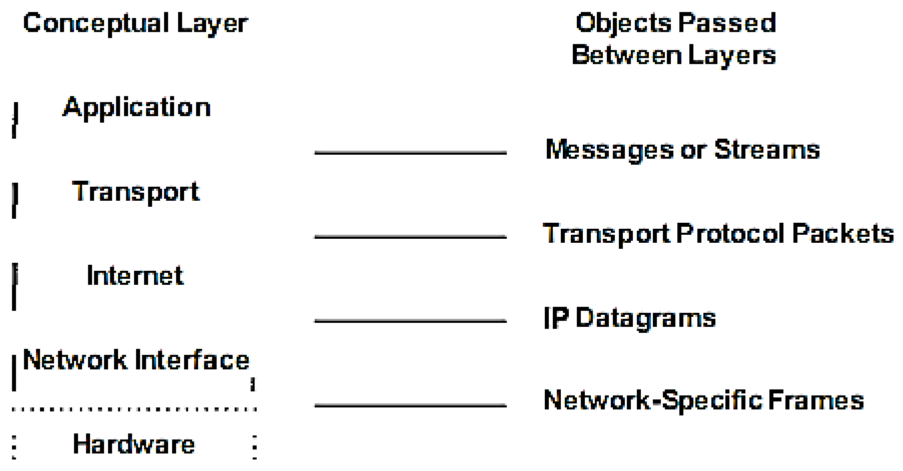


Figure 2.5 The 4 conceptual layers of TCPIIP software above the hardware layer, and the form of objects passed between layers. The layer labeled network interface is sometimes called the data link layer.

Application Layer. At the highest layer, users invoke application programs that access services available across a TCPDP internet. An application interacts with one of the transport layer protocols to send or receive data. Each application program chooses the style of transport needed, which can be either a sequence of individual messages or a continuous stream of bytes. The application program passes data in the required form to the transport layer for delivery.

Transport Layer. The primary duty of the transport layer is to provide communication from one application program to another. Such communication is often called end-to-end. The transport layer may regulate flow of information. It may also provide reliable transport, ensuring that data arrives without error and in sequence. To do so, transport protocol software arranges to have the receiving side send back acknowledgements and the sending side retransmit lost packets. The transport software divides the stream of data being transmitted into small pieces (sometimes called packets) and passes each packet along with a destination address to the next layer for transmission.

Although Figure 2.5 uses a single block to represent the application layer, a general purpose computer can have multiple application programs accessing an internet at one time. The transport layer must accept data from several user programs and send it to the next lower layer. To do so, it adds additional information to each packet, including codes that identify which application program sent it and which application program should receive it, as well as a checksum. The receiving machine uses the checksum to verify that the packet arrived intact, and uses the destination code to identify the application program to which it should be delivered.

Internet Layer. As we have already seen, the Internet layer handles communication from one machine to another. It accepts a request to send a packet from the transport layer along with an identification of the machine to which the packet should be sent. It encapsulates the packet in an IP datagram, fills in the datagram header, uses the routing algorithm to determine whether to deliver the datagram directly or send it to a router, and passes the datagram to the appropriate network interface for transmission.

The Internet layer also handles incoming datagram, checking their validity, and uses the routing algorithm to decide whether the datagram should be processed locally or forwarded. For datagrams addressed to the local machine, software in the internet layer deletes the datagram header, and chooses from among several transport protocols the one that will handle the packet. Finally, the Internet layer sends and receives ICMP error and control messages as needed.

Network Interface Layer. The lowest layer software comprises a network interface layer, responsible for accepting IP datagram and transmitting them overa specific network. A network interface may consist of a device driver (e.g., when the network is a local area network to which the machine attaches directly) or a complex subsystem that uses its own data link protocol (e.g., when the network consists of packet switches that communicate with hosts using HDLC).

2.8 The Protocol Layering Principle

Independent of the particular layering scheme used or the functions of the layers, the operation of layered protocols is based on a fundamental idea. The idea, called the layering principle, can be summarized succinctly:

Layered protocols are designed so that layer n at the destination receives exactly the same object sent by layer n at the source. The layering principle explains why layering is such a powerful idea. It allows the protocol designer to focus attention on one layer at a time, without worrying about how other layers perform. For example, when building a file transfer application, the designer considers only two copies of the application program executing on two computers, and concentrates on the messages they need to exchange for file transfer. The designer assumes that the application on one host receives exactly the data that the application on the other host sends.

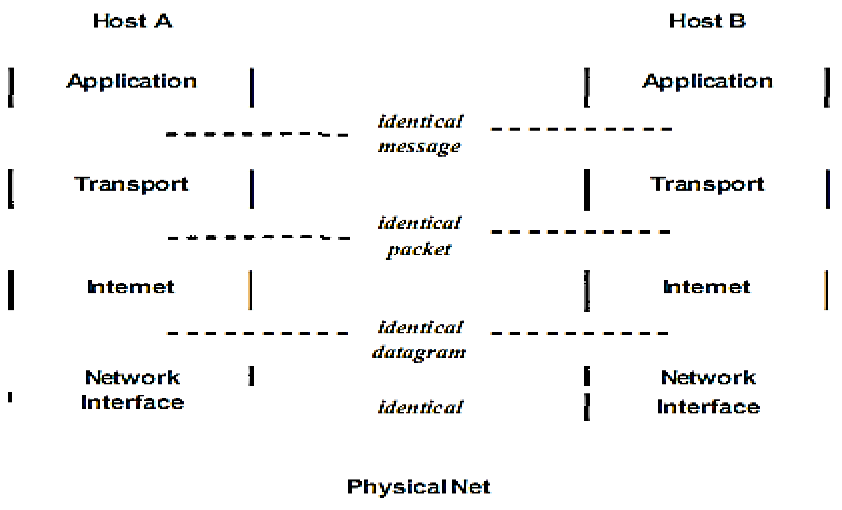


Figure 2.6 illustrates how the layering principle works:

Figure 2.6 The path of a message as it passes from an application on one host to an application on another. Layer n on host B receives exactly the same object that layer n on host A sent.

2.8 Layering in a Internet Environment

Our statement of the layering principle is somewhat vague, and the illustration in Figure 2.6 skims over an important issue because it fails to distinguish between transfers from source to ultimate destination and transfers across multiple networks. Figure 2.7 illustrates the distinction, showing the path of a message sent from an application program on one host to an application on another through a router.

As the figure shows, message delivery uses two separate network frames, one for the transmission from host A to router R, and another from router R to host B. The network layering principle states that the frame delivered to R is identical to the frame sent by host A. By contrast, the application and transport layers deal with end-toend issues and are designed so the software at the source communicates with its peer at the ultimate destination. Thus, the layering principle states that the packet received by the transport layer at the ultimate destination is identical to the packet sent by the transport layer at the original source.

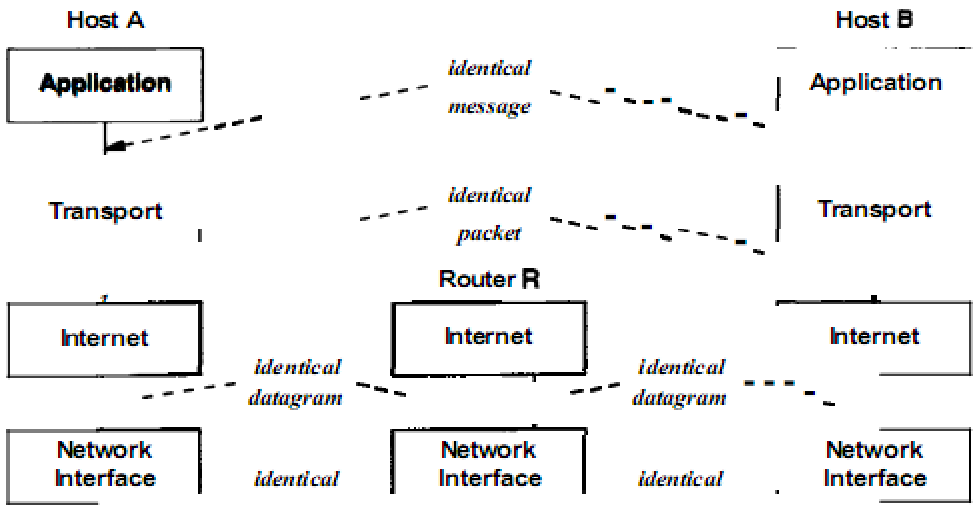


Figure 2.7 The layering principle when a router is used. The frame delivered to router R is exactly the frame sent from host A, but differs from the frame sent between R and B.

It is easy to understand that in higher layers, the layering principle applies across end-to-end transfers, and that at the lowest layer it applies to a single machine transfer. It is not as easy to see how the layering principle applies to the Internet layer. On one hand, we have said that hosts attached to an internet should view it as a large, virtual network, with the IP datagram taking the place of a network frame. In this view, datagrams travel from original source to ultimate destination, and the layering principle guarantees that the ultimate destination receives exactly the datagram that the original source sent. On the other hand, we know that the datagram header contains fields, like a time to live counter, that change each time the datagram passes through a router.

Thus, the ultimate destination will not receive exactly the same datagram as the source sent. We conclude that although most of the datagram stays intact as it passes across an internet, the layering principle only applies to datagrarns across single machine transfers. To be accurate, we should not view the Internet layer as providing end-to-end service.

2.9 Internet architecture

We have seen how computers connect to individual networks. The question arises, "How are networks interconnected to form an internetwork?" The answer has two parts. Physically, two networks can only be connected by a computer that attaches to both of them. A physical attachment does not provide the interconnection we have in mind, however, because such a connection does not guarantee that the computer will cooperate with other machines that wish to communicate. To have a viable internet, we need special computers that are willing to transfer packets from one network to another. Computers that interconnect two networks and pass packets from one to the other are called *internet gateways* or *internet routers.*

Consider an example consisting of two physical networks shown in Figure . In the figure, router *R* connects to both network 1and network 2. For *R* to act as a router, it must capture packets on network 1that are bound for machines on network 2 and transfer them. Similarly, *R* must capture packets on network 2 that are destined for machines on network1 and transfer them.

In the Figure 2.8 clouds are used to denote physical networks because the exact hardware is unimportant. Each network can be a LAN or a WAN, and each may have many computers attached or a few computers attached.

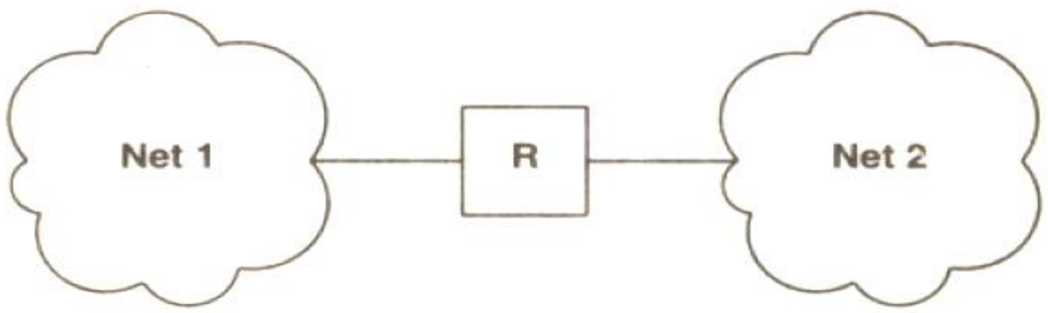


Figure 2.8Two physical networks interconnected by *R*, a router (IP gateway).

2.10 Interconnection through IP routers

In an actual internet that includes many networks and routers, each router needs to know about the topology of the internet beyond the networks to which it connects. For example, Figure 2.9 shows, three network interconnected by two routers.

In this example, router R_1 must transfer from network1 to network 2 all packets destined for computers on either network 2 or network 3. For a large internet composed of many networks, the router's task of making decisions about where to send packets becomes more complex. The idea of a router seems simple, but it is important because it provides a way to interconnect networks, not just computers. In fact, we have already discovered the principle of interconnection used throughout an internet:

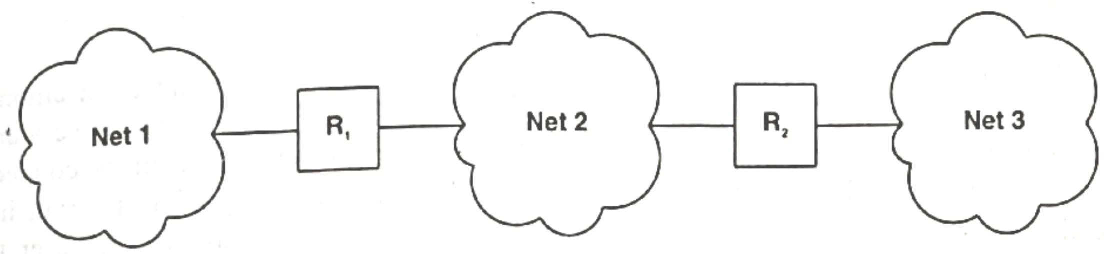


Figure2.9: Three networks interconnected by two routers.

In a TCPIIP internet, special computers called* IP routers *or* IP gateways *provide interconnections among physical networks.

You might suspect that routers, which must each know how to forward packets toward their destination, are large machines with enough primary or secondary memory to hold information about every computer in the internet to which they attach. In fact, routers used with internets are usually small computers. They often have little disk storage and modest main memories. The trick to building a small internet router lies in the following concept:

Routers use the destination network, not the destination computer, when forwarding a packet.

If packet forwarding is based on networks, the amount of information that a router needs to keep is proportional to the number of networks in the internet, not the number of computers. Because routers playa key role in internet communication, we will return to them in later chapters and discuss the details of how they operate and how they learn about routes. For now, we will assume that it is possible and practical to have correct routes for all networks in each router in the internet. We will also assume that only routers provide connections between physical networks in an internet.

2.11 The users view

Remember that is designed to provide a universal interconnection among computers independent of the particular networks to which they attach. Thus, we want a user to view an internet as a single, virtual network to which all machines connect despite their physical connections. In addition to routers that interconnect physical networks, software is needed on each computer to allow application programs to use an internet as if it were a single, physical network.

The advantage of providing interconnection at the network level now becomes clear. Because application programs that communicate over the internet do not know the details of underlying connections, they can be run without change on any computer. Because the details of each machine's physical network connections are hidden in the internet software, only the internet software needs to change when new physical connections are added or existing connections are removed. In fact, it is possible to optimize the internal structure of the internet by altering physical connections while application programs are executing.

A second advantage of having communication at the network level is more subtle: users do not have to understand, remember, or specify how networks connect or what traffic they carry. Application programs can be written that communicate independent of underlying physical connectivity. In fact, network managers are free to change interior parts of the underlying internet architecture without changing application software in most of the computers attached to the internet (of course, network software must be reconfigured when a computer moves to a new network). Routers do not provide direct connections among all pairs of networks. It, may be necessary for traffic traveling from one computer to another to pass through several routers as the traffic crosses intermediate networks. Thus, networks participating in an internet are analogous to-highways in the U.S. interstate system: each net agrees to handle transit traffic in exchange for the right to send traffic throughout the internet. Typical users are unaffected and unaware of extra traffic on their local network.

CHAPTER 3
INTERNET ADDRESSES

3.1 Introduction

Think of an internet as a large network like any other physical network. The difference, of course, is that the internet is a virtual structure, imagined by its designers, and implemented entirely in software. Thus, the designers are free to choose packet formats and sizes, addresses, delivery techniques, and so on; nothing is dictated by hardware. For addresses, the designers of chose a scheme analogous to physical network addressing in which each host on the internet is assigned a 32-bit integer address called its *internet address* or I/P *address.* The clever part of internet addressing is that the integers are carefully chosen to make routing efficient. Specifically, an IP address encodes the identification of the network to which a host attaches as well as the identification of a unique host on that network. We can summarize:
Each host on a TCPI/IP internet is *assigned a unique 32-bit internet address that* is *used in all communication with that host.*

3.2 The original classful addressing scheme

The details of IP addresses help clarify the abstract ideas. For now, we give a simplified view and expand it later. In the simplest case, each host attached to an internet is assigned a 32-bit universal identifier as its internet address. A prefix of an IP address identifies a network. That is, the IP addresses in all hosts on a given network share a common prefix. Conceptually, each address is a pair *(netid, hostid),* where *netid*identifies a network, and *hostid*identifies a host on that network. In practice, however, the partition into prefix and suffix is not uniform throughout the entire internet because the designers did not specify a single boundary. In the original addressing scheme, which is known as *classful,* each IP address had one of the first three forms shown in Figure 3.1.

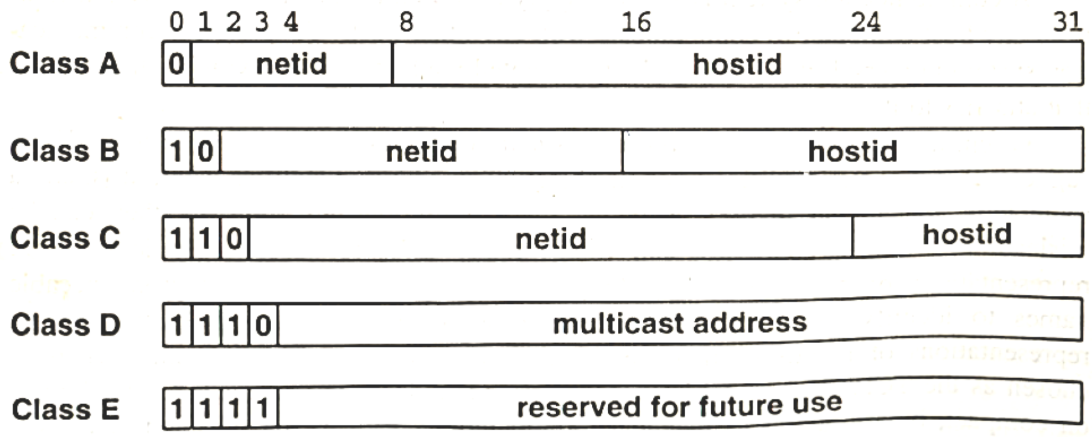


Figure:3.1 The five forms of Internet (lP) addresses used with the original classful addressing scheme.The three primary classes, A. Band C. can be distinguished by the first three bits.

In the classful addressing scheme, each address is said to be *self-identifying* because the boundary between prefix and suffix can be computed from the address alone, without reference to external information. In particular, the class of an address can be determined from the three high-order bits; with two bits being sufficient to distinguish among the three primary classes. Class *A* addresses, used for the handful of networks that have more than 2^{16}(i.e., 65,536) hosts, devote 7 bits to netid and 24 bits to hostid. Class *B* addresses, used for intermediate size networks that have between 2^{8} (i.e., 256) and 216hosts, allocate 14 bits to the netid and 16 bits to the hostid. Finally, class C addresses, used for networks that have less than 28hosts, allocate 21 bits to the netid and only 8 bits to the hostid. Note that the IP address was originally defined in such a way that it was possible to extract the hostid or netid portions quickly. Efficiency was especially important for routers, which use the netid portion of an address when deciding where to send a packet. We will return to the discussion of efficient route lookup after examining recent changes and extensions to the addressing scheme.

3.3 Address specify Network connections

To simplify the discussion, we said that an internet address identifies a host, but that is not strictly accurate. Consider a router that attaches to two physical networks. How can we assign a single IP address if the address encodes a network identifier as well as a host identifier? In fact, we cannot. When conventional computers have two or more physical connections they are called *multi-homed hosts.* Multi-homed hosts and routers require multiple IP addresses. Each address corresponds to one of the machine's network connections. Looking at multi-homed hosts leads to the following important idea: *Because IP addresses encode both a network and a host on that network, they do not specify an individual computer, but a connection to a network.* Thus, a router connecting *n* networks has *n* distinct IP addresses, one for each network connection.

3.4 Network and directed broadcast addresses

We have already cited the major advantage of encoding network information in internet addresses: it makes efficient routing possible. Another advantage is that internet addresses can refer to networks as well as hosts. By convention, hostid*0* is never assigned to an individual host. Instead, an IP address with hostid portion equal to zero is used to refer to the network itself. In summary:

Internet addresses can be used to refer to networks as well as individual hosts. By convention, an address that has all bits of the hosted equal to 0 is *reserved to refer to the network.* .

Another significant advantage of the internet addressing scheme is that it includes a *directed broadcast address* that refers to all hosts on the network. According to the standard, any address with the hostid consisting of all Isis reserved for directed broadcast.

When a packet is sent to such an address, a single copy of the packet is transferred across the internet from the source. Routers along the path use the netid portion of the address when choosing a path; they do not look at the host portion. Once the packet reaches a router attached to the final network, that router examines the host portion of the address to determine how to deliver the packet. If it finds all *1s,* the router broadcasts the packet to all hosts on the network. On many network technologies (e.g., Ethernet), broadcasting is as efficient as unicast transmission; on others,' broadcasting is supported by the network software, but requires substantially more delay than single transmission. Some network hardware does not support broadcast at all. Thus, having an IP directed broadcast address does not guarantee the availability or efficiency of broadcast delivery. In summary,

IP addresses can be used to specify a directed broadcast in which a packet is *sent to all computers on a network; such addresses map to hardware broadcast, (I' available By convention, a directed broadcast address has a valid netid and has a hostid with all bits set to 1.*

3.5 Limited broadcast

The broadcast address we just described is known as *directed* because it contains both a valid network ID and the broadcast hostid. A directed broadcast address can be interpreted unambiguously at any point in an internet because it uniquely identifies the target network in addition to specifying broadcast on that network. Directed broadcast addresses provide a powerful (and somewhat dangerous) mechanism that allows a remote system to send a single packet that will be broadcast on the specified network. From an addressing point of view, the chief disadvantage of directed broadcast is that it requires knowledge of the network address.

Another form of broadcast address, called a *limited broadcast address* or *local network broadcast address*,provides a broadcast address for the local network independent of the assigned IP address. The local broadcast address consists of thirty-two *1s* (hence, it is sometimes called the "all *1s"* broadcast address). A host may use the limited broadcast address as part of a startup procedure before it learns its IP address or the IP address prefix for the local network. Once the host learns the correct IP address for the local network, however, it should use directed broadcast. As a general rule, protocols restrict broadcasting *to* the smallest possible set of machines. We will see how this rule affects multiple networks that share addresses in the chapter on subnet addressing.

3.6 The All -0s address

We have seen that a field consisting of is can be interpreted to mean "all," as in "all hosts" on a network. In general, internet software interprets fields consisting of *Os t*o mean “this”. The interpretation appears throughout the literature. Thus, an IP address with hostid*0* refers *to* "this" host, and an internet address with network ID *0* refers to "this" network. Of course, it is only meaningful to use such an address in a context where it can be interpreted unambiguously. For example, if a machine receives a packet in which the netid portion of the destination address is *0* and the hostid portion *of* the destination address matches its address, the receiver interprets the netid field to mean "this" network (i.e., the network over which the packet arrived). Using netid*0* is especially important in those cases where a host wants to communicate over a network but does not yet know the network IP address. The host uses network ID *0* temporarily, and other hosts on the network interpret the address as meaning "this" network. In most cases, replies will have the network address fully specified, allowing the original sender to record it for future use. Chapters 9 and 23 will discuss in detail mechanisms a host can use to determine the network ID of the local network.

3.7 Subnet and classes Extensions

The addressing scheme described so far requires a unique network prefix for each physical network. Although that was, indeed, the original plan, it did not last long. In the 1980s as Local Area Network technologies became increasingly popular, it became apparent that requiring a unique prefix for each physical network would exhaust the address space quickly. Consequently, an addressing extension was developed to conserve network prefixes. Known as *subnet addressing,* the scheme allows multiple physical networks to share a prefix. In the 1990s, a second extension was devised that ignored the classful hierarchy and allowed the division between prefix and suffix to occur at an arbitrary point. called*classless addressing* or *supernetting,* the scheme allows more complete utilization of the address space. Chapter 10 will consider details of the subnet and supernet addressing extensions. For now, it is only important to know that the addressing scheme has been extended, and that the original classful scheme described in this chapter is no longer the most widely used.

3.8 Dotted decimal Notation

When communicated to humans, either in technical documents or through application programs, IP addresses are written as four decimal integers separated by decimal points, where each integer gives the value of one octet of the IP address. Thus, the 32-bit internet address

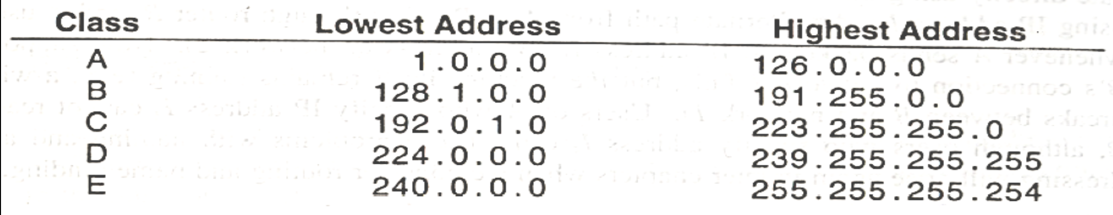
    10000000 00001010 00000010 00011110

is written

    128.10.2.30

We will use dotted decimal notation when expressing IP addresses throughout the remainder of this text. Indeed, most software that displays or requires a human to enter an IP address uses dotted decimal notation. For example, the UNIX' *netstat*command, which displays information about routes and connections, and application programs such as *telnet* and *ftp* all use dotted

decimal notation when accepting or displaying IP addresses. Thus, when classful addressing is used, it is helpful to understand the relationship between IP address classes and dotted decimal numbers. Some values are reserved for special purposes.
The table in Figure 3.2 summarizes the range of values for each class.



Class	Lowest Address	Highest Address
A	1.0.0.0	126.0.0.0
B	128.1.0.0	191.255.0.0
C	192.0.1.0	223.255.255.0
D	224.0.0.0	239.255.255.255
E	240.0.0.0	255.255.255.254

Figure 3.2 The range of dotted decimal values that correspond to each IP address class.

3.9 Loopback address

The table in Figure 3.2 shows that not all possible addresses have been assigned to classes. In particular, the network prefix 127.0.0.0, a value from the class A range, is reserved for *loopback,* and is intended for use in testing and for inter-process communication on the local computer. When any program uses the loopback address as a destination, the protocol software in the computer processes the data without sending traffic across any network. The literature explicitly states that a packet sent to a network 127 address should never appear on any network. Furthermore, a host or router should never propagate routing or reachability information for network number 127; it is not a network address.

CHAPTER 4

ADDRESS RESOLUTION PROBLEM

4.1 Introduction

address is a scheme in which each host is assigned a 32-bit address and an internet behaves like a virtual network, using only the assigned addresses when sending and receiving packets.Two machines on a given physical network can communicate only if they know each other's physical network address. But how a host or a router maps an IP address to the correct physical address when it needs to send a packet across a physical net? This chapter considers that mapping, showing how it is implemented for the two most common physical network address schemes.

4.2 The address Resolution Problem

Consider two machines A and B that connect to the same physical network. Each has an assigned IP address ZA and ZB and a physical address PA and PB. The goal is to devise low-level software that hides physical addresses and allows higher-level programs to work only with internet addresses. Ultimately, however, communication must be carried out by physical networks using whatever physical address scheme the underlying network hardware supplies. Suppose machine A wants to send a packet to machine B across a physical network to which they both attach, but A has only B's internet address IB. The question arises: how does A map that address to B's physical address, PB? Address mapping must be performed at each step along a path from the original source to the ultimate destination. In particular, two cases arise. First, at the last step of delivering a packet, the packet must be sent across one physical network to its final destination. The computer sending the packet must map the final destination's Internet address to the destination's physical address. Second, at any point along the path from the source to the destination other than the final step, the packet must be sent to an intermediate router. Thus, the sender must map the intermediate router's Internet address to a physical address.

The problem of mapping high-level addresses to physical addresses is known as the address resolution problem and has been solved in several ways. Some protocol suites keep tables in each machine that contain pairs of high-level and physical addresses. Others solve the problem by encoding hardware addresses in high-level addresses. Using either approach exclusively makes high-level addressing awkward at best. This chapter discusses two techniques for address resolution used by TCPIIP protocols and shows when each is appropriate.

4.3 Two types of Physical addresses.

There are two basic types of physical addresses, exemplified by the Ethernet, which has large, fixed physical addresses, and proNET, which has small, easily configured physical addresses. Address resolution is difficult for Ethernet-like networks, but easy for networks like proNET.

4.4 Resolution through Direct Mapping

Consider a proNETtoken ring network. proNET uses small integers for physical addresses and allows the user to choose a hardware address when installing an interface board in a computer. The key to making address resolution easy with such network hardware lies in observing that as long as one has the freedom to choose both IP and physical addresses, they can be selected such that parts of them are the same. Typically, one assigns IP addresses with the hostid portion equal to 1, 2, 3, and so on, and then, when installing network interface hardware, selects a physical address that corresponds to the IP address. For example, the system administrator would select physical address 3 for a computer with the IP address 192.5.48.3 because 192.5.48.3 is a class C address with the host portion equal to 3. For networks like proNET, computing a physical address from an IP address is trivial. The computation consists of extracting the host portion of the IP address. Extraction is computationally without reference to external data. Finally, new computers can be added to the network without changing existing assignments or recompiling code. Conceptually, choosing a numbering scheme that makes address resolution efficient means selecting a function f that maps IP addresses to physical addresses. The designer may be able to select a physical address numbering scheme as well, depending on the hardware. Resolving IP address IA means computing

$$PA=f(IA)$$

We want the computation off to be efficient. If the set of physical addresses is constrained, it may be possible to arrange efficient mappings other than the one given in the example above. For instance, when using IP over a connection-oriented network such as ATM, one cannot choose physical addresses. On such networks, one or more computers (servers) store pairs of addresses, where each pair contains an Internet address and the corresponding physical address. Typically, such servers store the pairs in a table in memory to speed searching. To guarantee efficient address resolution in such cases, software can use a conventional hash function to search the table. Exercise 4.1 suggests a related alternative.

4.5 Resolution through Dynamic Binding

To understand why address resolution is difficult for some networks, consider Ethernet technology. In which each Ethernet interface is assigned a 48- bit physical address when the device is manufactured. As a consequence, when hardware fails and requires that an Ethernet interface be replaced, the machine's physical address changes. Furthermore, because the Ethernet address is 48 bits long, there is no hope it can be encoded in a 32-bit IP address. Designers of protocols found a creative solution to the address resolution problem for networks like the Ethernet that have broadcast capability. The solution allows new hosts or routers to be added to the network without recompiling code, and does not require maintenance of a centralized database. To avoid maintaining a table of mappings, the designers chose to use a low-level protocol to bind addresses dynamically. Termed the Address Resolution Protocol (ARP), the protocol provides a mechanism that is both reasonably efficient and easy to maintain.

As Figure 4.1 shows, the idea behind dynamic resolution with ARP is simple: when host A wants to resolve IP address ZB, it broadcasts a special packet that asks the host with IP address le to respond with its physical address, PB. AU hosts, including B, receive the request, but only host B recognizes its IP address and sends a reply that contains its physical address. When A receives the reply, it uses the physical address to send the internet packet directly to B. We can summarize:

The Address Resolution Protocol, ARP, allows a host to find the physical address of a target host on the same physical network, given only the target's IP address.

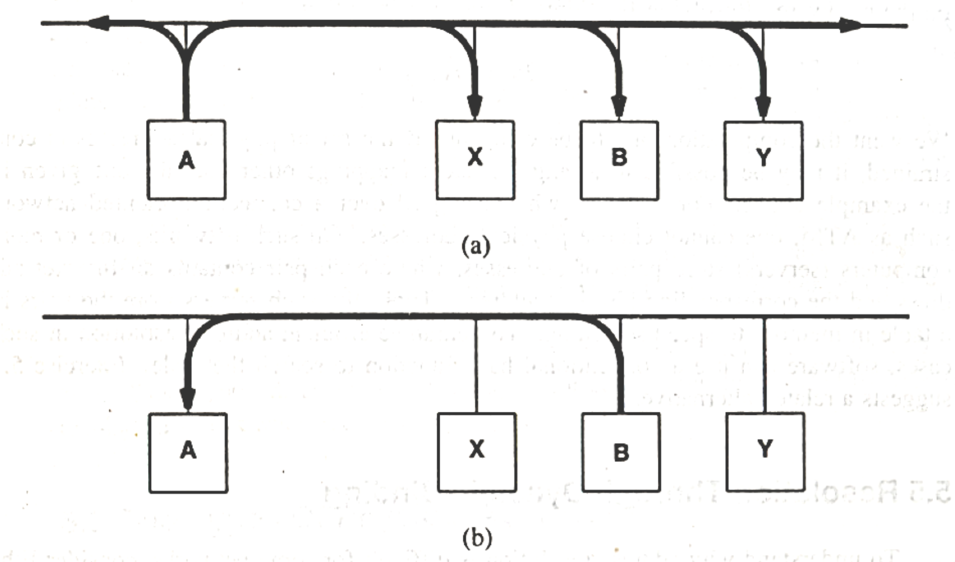


Figure 4.1 The ARP protocol. To determine P_B, 8's physical address, from I_B, its IP address, (a) host *A*broadcasts an ARP request containing $I_B$to all machines on the net, and (b) host 8 responds with an ARP reply that contains the pair *(I_B, P_B)*.ARP using Dynamic Bynding.

4.6 The address Resolution cache

The ARP protocol. To determine P_B, 8's physical address, from I_B, its IP address, (a) host *A*broadcasts an ARP request containing $I_B$to all machines on the net, and (b) host 8 responds with an ARP reply that contains the pair *(I_B, P_B)*.

4.7 ARP cache time outs

To reduce communication costs, computer that use ARP maintain a cache of recently acquired IP-to-physical address bindings. That is, whenever a computer sends an ARP request and receives an ARP reply, It saves the IP address and corresponding hardware address information in its cache for successive lookups. When transmitting a hardware computer always looks in its cache for a binding before sending an ARP packet, If it finds the desired binding in its ARP cache, the computer need not broadcast on the network. Thus when two computers on a network communicate, they begin with ARP request and response and then repeatedly transfer packets without using ARP for each one. Experience shows that because most network communication involves more than one packet transfer, even a small cache is worthwhile.

The ARP cache provides an example of *soft state,* a technique commonly used in network protocols. The name describes a situation in which information call become "stale" without warning. In the case of ARP, consider two computers, *A* and *B,* both connected to an Ethernet. Assume *A* has sent an ARP request, and *B* has replied. Further assume that after the exchange *B* crashes. Computer *A* will not receive any notification of the crash. Moreover, because it already has address binding information for *B* in its ARP cache, computer *A* will continue to send packets to *B.* The Ethernet hardware provides no indication that *B* is not on-line because Ethernet does not have guaranteed delivery. Thus, *A* has no way of knowing when information in its ARP cache has become incorrect. To accommodate soft state, responsibility for correctness lies with the owner of the information. Typically, protocols that implement soft state use timers, with the state information being deleted when the timer expires. For example, whenever address binding information is placed in an ARP cache, the protocol requires a timer to be set, with a typical timeout being 20 minutes. When the timer expires, the information must be removed. After removal there are two possibilities. If no further packets are sent to the destination, nothing occurs. If a packet must be sent to the destination and there is no binding present in the cache, the computer follows the normal procedure of broadcasting ARP request and obtaining the binding. If the destination is still reachable the binding will again be placed in the ARP cache. If not, the sender will discover that the destination is off-line.

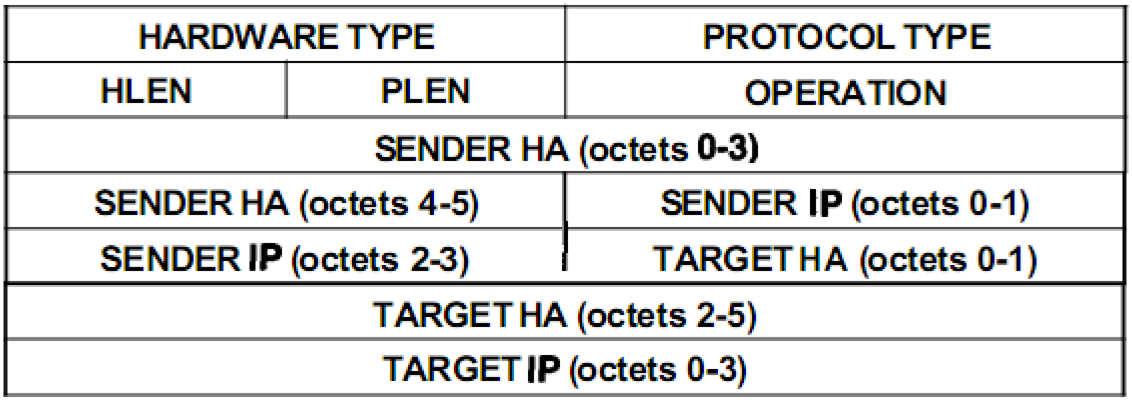
The use of soft state in ARP has advantages and disadvantages. The chief advantage arise
from autonomy. First, a computer can determine when information in its ARP cache should be revalidated independent of other computers. Second, a sender that does not need successful communication with the receiver or a third party to determine sender has become invalid; if a target does not respond to an ARP request, the sender will declare the target to be down. Third, the scheme does not rely on network hardware to provide reliable transfer.

4.8 ARP Protocol Format

Unlike most protocols, the data in ARP packets does not have a fixed-format header. Instead, to make ARP useful for a variety of network technologies, the length of fields that contain addresses depend on the type of network. However, to make it possible to interpret an arbitrary ARP message, the header includes fixed fields near the beginning that specify the lengths of the addresses found in succeeding fields. In fact, the ARP message format is general enough to allow it to be used with arbitrary physical addresses and arbitrary protocol addresses. The example in Figure 5.3 shows the 28- octet ARP message format used on Ethernet hardware (where physical addresses are 48-bits or 6 octets long), when resolving IP protocol addresses (which are 4 octets long).

Figure 4.2 shows an ARP message with 4 octets per line, a format that is standard throughout this text. Unfortunately, unlike most of the remaining protocols, the variable-length fields in ARP packets do not align neatly on 32-bit boundaries, making the diagram difficult to read.

For example, the sender's hardware address, labeled SENDER HA, occupies 6 contiguous octets, so it spans two lines in the diagram.



HARDWARE TYPE		PROTOCOL TYPE	
HLEN	PLEN	OPERATION	
SENDER HA (octets 0-3)			
SENDER HA (octets 4-5)		SENDER IP (octets 0-1)	
SENDER IP (octets 2-3)		TARGET HA (octets 0-1)	
TARGET HA (octets 2-5)			
TARGET IP (octets 0-3)			

Figure 4.2:An example of the ARPW message format when used for IP- to-Ethernet address resolution. The length of fields depends on the hardware and protocol address lengths, which are 6 octets for an Ethernet address and 4 octets for an IP address.

Field HARDWARE TYPE specifies a hardware interface type for which the sender seeks an answer; it contains the value 1 for Ethernet. Similarly, field PROTOCOL TYPE specifies the type of high-level protocol address the sender has supplied; it contains 0800,, for IP addresses. Field OPERATION specifies an ARP request (I), ARP response (2), RARP request (3), or RARP response (4). Fields HLEN and PLEN allow ARP to be used with arbitrary networks because they specify the length of the hardware address and the length of the high-level protocol address. The sender supplies its hardware address and IF' address, if known, in fields SENDER HA and SENDER IP. When making a request, the sender also supplies the target hardware address (RARP) or target IP address (ARP), using fields TARGET HA or TARGET IP. Before the target machine responds, it fills in the missing addresses, swaps the target and sender pairs, and changes the operation to a reply. Thus, a reply carries the IP and hardware addresses of the original requester, as well as the IP and hardware addresses of the machine for which a binding was sought.

4.9 Reverse Address Resolution (RARP)

RARP is no longer important in the Internet, but was once an essential protocol used to bootstrap systems that did not have stable storage. In essence, RARP allows a system to obtain an IP address at startup. The procedure is straightforward: when it boots, the system broadcasts a RARP request and waits for a response. Another computer on the network must be configured to listen for RARP requests and generate a RARP reply that contains the requester's IP address. Once the reply arrives, the system continues to boot, and uses IP for all communication. When ·it makes a RARP request, a system must identify itself so the computer receiving the request can place the correct IP address in the reply. Although any unique hardware identification suffices (e.g., the CPU serial number), RARP uses an obvious ID: the system's MAC address. That is, a booting system places its MAC address in the RARP request, and receives its IP address in the RARP reply.

Interestingly, RARP uses the same packet format as ARP. A RARP request is formed by filling in the target protocol address field, changing the message type from request to reply, and sending the reply back directly to the machine making the request. Like an ARP message, a RARP message is sent from one machine to another encapsulated in the data portion of a network frame. For example, an Ethernet frame carrying a RARP request has the usual preamble, Ethernet source and destination addresses, and packet type fields in front of the frame. The frame type contains the value 803516 to identify the contents of the frame as a RARP message.

Assignment Questions

Unit 1

Answer the following Multiple choice questions. Each carries 1 marks.

1. What are the two levels of interconnections?
   - **a. Application and Network**
   - b. Application and Packet
   - c. Network and Packet
   - d. Physical and Application
2. What are IP routers?
   - a. Devices used for connecting networks
   - **b. Devices used for connecting network and routing**
   - **c.** Devices used for routing
   - **d.** Devices used for handling IP
3. What are application gateways?
   - a. Devices
   - b. Routers
   - **c. Application level interconnecting**
   - **d.** Software
4. What do you mean by internetworking?
   - **a. The connection of different physical networks**
   - b. The connection of a few networks.
   - c. The connection of different computers
   - d. Connection of networks present inside a building.
5. What are the two portions of IP Address?
   - **a. Network and host**
   - b. Network and sub network
   - c. Sub network and host
   - d. Host, network and sub network
6. What is the size of an IP address in IPv4?
   - a. 16 bit
   - **b. 32 bit**
   - c. 64 bit
   - d. 128 bit
7. What are the various classes of IP addressing in classful addressing scheme?
   - a. X, Y, Z, A and B

   b. **A, B, C, D, and E**
   c. M, N, G, O and P
   d. A, B, C, M and N
8. In class A scheme the entire networking can be organized in to how many network?
   a. **128**
   b. 255
   c. 64
   d. 32
9. In class B scheme the entire networking can be organized in to how many?
   a. 128
   b. 256
   c. 640
   d. **16384**
10. In class C scheme the entire networking can be organized in to how many network?
    a. **2097152**
    b. 16384
    c. 640
    d. 128
11. What do you mean by broadcasting?
    a. Sending the packet to every workstation of all the neighboring networks.
    b. **Sending the copy of the message packet to all the workstations of a given network.**
    c. Sending a msg packet to an individual workstation.
    d. Sending the copy to a specific host.
12. What do you mean by Unicasting?
    a. **Reply from the specific workstation to specific destination in the network**.
    b. Reply from all to a specific destination
    c. Reply from a specific to all.
    d. Reply from all to all.
13. What do you mean by multicasting?
    a. Sending the packet to every workstation of all the neighboring networks.
    b. Sending the copy of the message packet to all the workstations of a given network.
    c. Sending a msg packet to an individual workstation.
    d. **Sending the copy to a specific multiple hosts**.

14. Give the address of the loopback address and mention the need for the same.
    a. 10.1.1.1
    b. 127.0.0.0
    c. 192.168.0.1
    d. 255.255.255.255
15. What is ARP?
    a. Address resolution protocol.
    b. Address resolution problem.
    c. Either a or b
    d. None of the above.
16. What is address resolution cache?
    a. A RAM to store IP address
    b. A small memory to store IP-MAC address pair.
    c. Memory to store ARP
    d. None of the above.
17. What is the purpose of HARDWARE TYPE field in the ARP protocol format?
    a. To explain the computer hardware
    b. To explain the network media
    c. To explain the network device in the computer.
    d. To explain the network protocol.
18. What is the purpose of PROTOCOL TYPE field in the ARP protocol format?
    a. To explain the computer hardware
    b. To explain the network media
    c. To explain the network device in the computer.
    d. To explain the network protocol.
19. What is the purpose of HLEN and PLEN field in the ARP protocol format?
    a. Hardware address length and protocol length
    b. Header length and protocol length
    c. Header and payload length
    d. Hardware and payload length.
20. What is the purpose of OPERATION field in the ARP protocol format?
    a. Field specifies ARP request and response.
    b. Current operation of the ARP packet
    c. A useless field
    d. A useful field.

21. What is RARP?
    a. Repeated Address Resolution Protocol
    b. Responsive Address Resolution Protocol
    c. Reverse Address Resolution Protocol
    d. None of the above
22. What is IAB?
    a. Internet Architecture Board.
    b. Internet Addressing Board.
    c. Intranet Access Board.
    d. Internet Access Board.
23. What is MILNET?
    a. Milman Network
    b. Military Network
    c. Mil Network
    d. None of the above.
24. What is DARPA?
    a. Defense Advanced Research Project Agency
    b. Different Advanced Research Project Agency
    c. Defense Advanced Rework Project Agency.
    d. Different Advanced Rework Project Agency.
25. CSNET stands for
    a. Communication network
    b. Computer Science Network
    c. Common Network
    d. None of the above

Essay questions

1. Explain the application level interconnection.
2. Explain the network level interconnection.
3. Explain the architecture of internet.
4. With the help of a diagram explain the process of interconnection through IP routers.
5. Explain the users’ view of internetworking.
6. With the help of a diagram explain the conceptual layer of a protocol software.

7. With the help of a diagram explain The path of a message traversing the Internet from the sender through two intermediate routers to the receiver. Intermediate routers only send the datagram to the 1P software layer.
8. List out the 7 layers of ISO OSI reference model.
9. List and explain the five layers of the model.
10. Explain the layering principles of within a single network.
11. Explain the layering principles using routers for different network.

12. Explain the classful addressing scheme with the help of a diagram.
13. Explain how to broadcast the packet to the entire network?
14. What do you mean by limited broadcasting?
15. Explain the dotted decimal representation of IP addressing.
16. Explain the loopback addressing. Also explain the use of the same.

17. Write a note on Address resolution problem.
18. Explain how to resolve the physical address using direct mapping.
19. Explain how to resolve the physical address using dynamic binding.
20. Write a note on functioning of Address resolution cache.
21. Write a note on RARP.
22. With the help of a diagram explain the ARP protocol format.

****

UNIT II
CHAPTER 5
IP ROUTING PRINCIPLES AND ROUTING OF IP DATAGRAM

5.1 The IPV4 Datagram

The analogy between a network and a internet is strong. On a physical network, the unit of transfer is a frame that contains a header and data, where header gives information such as the (physical) source and destination addresses. e internet calls its basic transfer unit an *Internet datagram*, sometimes referred to as an *IP datagram* or merely a *datagram.* Like a typical physical network frame, a datagram is divided into header and data areas. Also like a frame, the datagram header contains the source and destination addresses and a type field that identifies the contents of the datagram. The difference, of course, is that the datagram header contains IP addresses whereas the frame header contains physical addresses. Figure 5.1 shows the general form of a datagram:

Datagram Header	Datagram Data area

Figure 5.1: IPv4 Datagram

IP specifies the header format including the source and destination IP addresses. IP does not specify the format of the data area; it can be used to transport arbitrary data.

5.2 Datagram format

The following figure 5.2 Figure shows the arrangement of fields in a datagram:

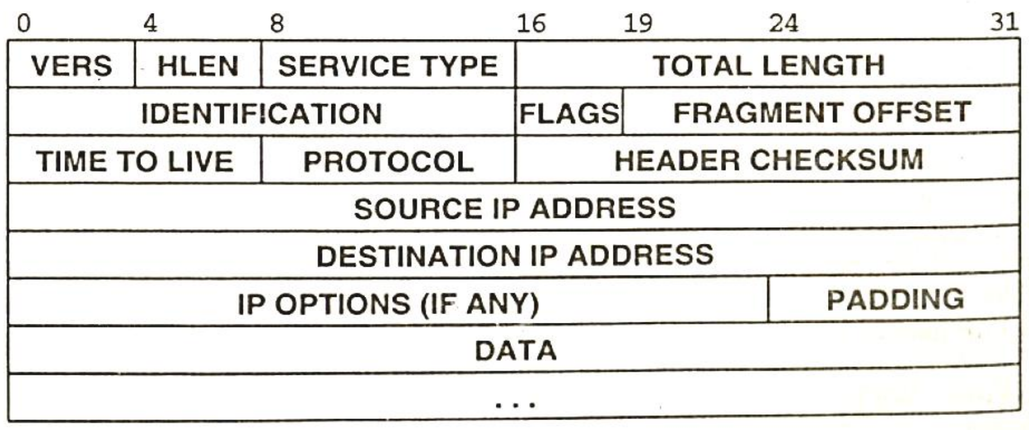


Figure 5.2: Format of an Internet datagram, the basic unit of transfer in a internet.

Because datagram processing occurs in software, the contents and format are not constrained by any hardware. For example, the first 4-bit field in a datagram *(VERS)* contains the version of the IP protocol that was used to create the datagram. It is used to verify that the sender, receiver, and any routers in between them agree on the format of the datagram. All IP software is required to check the version field before processing a datagram to ensure it matches the format the software expects. If standards change, a machines will reject datagram with protocol versions that differ from theirs, preventing them from misinterpreting datagram contents according to an outdated format. The current IP protocol version is 4. Consequently, the term *IPv4* is often used to denote the current protocol.

The header length field *(HLEN)*, also 4 bits, gives the datagram header length measured in 32-bit words. As we will see, all fields in the header have fixed length except for the *IP OPTIONS* and corresponding *PADDING* fields. The most common header, which contains no options and no padding, measures 20 octets and has a header length field equal to 5.

The *TOTAL LENGTH* field gives the length of the IP datagram measured in octets, including octets in the header and data. The size of the data area can be computed by subtracting the length of the header *(HLEN)* from the *TOTAL LENGTH.* Because the *TOTALLENGTH* field is 16 bits long, the maximum possible size of an IP datagram is 216 or 65,535 octets. In most applications this is not a severe limitation. It may become more important in the future if higher speed networks can carry data packets larger than 65,535octets.

5.3 Direct and Indirect delivery

Loosely speaking, we can divide routing into two forms: direct delivery and indirect delivery. Direct delivery, the transmission of a datagram from one machine across a single physical network directly to another, is the basis on which all internet communication rests. Two machines can engage in direct delivery only if they both attach directly to the same underlying physical transmission system (e.g., a single Ethernet). Indirect delivery occurs when the destination is not on a directly attached network, forcing the sender to pass the datagram to a router for delivery.

5.4 Datagram delivery over a single Network

It is very much clear that one machine on a given physical network can send a physical frame directly to another machine on the same network. To transfer an IP datagram, the sender encapsulates the datagram in a physical frame, maps the destination IP address into a physical address, and uses the network hardware to deliver it.

Transmission of an IP datagram between two machines on a single physical network does not involve routers. The sender encapsulates the datagram in a physical frame, binds the destination IP address to a physical hardware address, and sends the resulting frame directly to the destination.

How does the sender know whether the destination lies on a directly connected network? The test is straightforward. We know that IP addresses are divided into a network-specific prefix and a host-specific suffix. To see if a destination lies on one of the directly connected networks, the sender extracts the network portion of the destination IP address and compares it to the network portion of its own IP address(es). A match means the datagram can be sent directly. Here we see one of the advantages of the Internet address scheme, namely:

Because the internet addresses of all machines on a single network include a common network pre& and extracting that pre& requires only a few machine instructions, testing whether a machine can be reached directly is extremely efficient.

From an internet perspective, it is easiest to think of direct delivery as the final step in any datagram transmission, even if the datagram traverses many networks and intermediate routers. The final router along the path between the datagram source and its destination will connect directly to the same physical network as the destination.

Thus, the final router will deliver the datagram using direct delivery. We can think of direct delivery between the source and destination as a special case of general purpose routing - in a direct route the datagram does not happen to pass through any intervening routers.

5.5 Indirect Delivery

Indirect delivery is more difficult than direct delivery because the sender must identify a router to which the datagram can be sent. The router must then forward the datagram on toward its destination network. To visualize how indirect routing works, imagine a large internet with many networks interconnected by routers but with only two hosts at the far ends. When one host wants to send to the other, it encapsulates the datagram and sends it to the nearest outer. We know that the host can reach a router because all physical networks are interconnected, so there must be a router attached to each network. Thus, the originating host can reach a router using a single physical network. Once the frame reaches the router, software extracts the encapsulated datagram, and the IP software selects the next router along the path towards the destination. The datagram is again placed in a frame and sent over the next physical network to a second router, and so on, until it can be delivered directly. These ideas can be summarized:

Routers in a TCP/IP internet form a cooperative, interconnected structure. Datagram pass from router to router until they reach a router that can deliver the datagram directly.

How can a router know where to send each datagram? How can a host know which router to use for a given destination? The two questions are related because they both involve IP routing. We will answer them in two stages, considering the basic table-driven routing algorithm in this chapter and postponing a discussion of how routers learn new routes until later.

5.6 Table – Driven IP forwarding

The usual IP routing algorithm employs an Internet routing table (sometimes called an IP routing table) on each machine that stores information about possible destinations and how to reach them. Because both hosts and routers route datagrams, both have IP routing tables. Whenever the IP routing software in a host or router needs to transmit a datagram, it consults the routing table to decide where to send the datagram.

What information should be kept in routing tables? If every routing table contained information about every possible destination address, it would be impossible to keep the tables current. Furthermore, because the number of possible destinations is large, machines would have insufficient space to store the information.

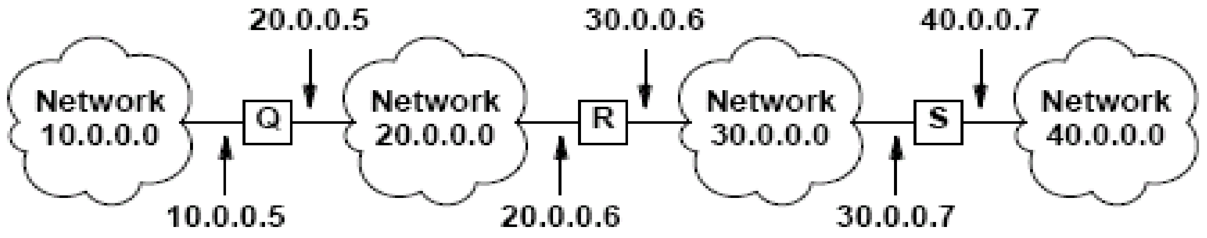
Conceptually, we would like to use the principle of information hiding and allow machines to make routing decisions with minimal information. For example, we would like to isolate information about specific hosts to the local environment in which they exist and arrange for machines that are far away to route packets to them without knowing such details. Fortunately, the IP address scheme helps achieve this goal. Recall that IP addresses are assigned to make all machines connected to a given physical network share a common prefix (the network portion of the address). We have already seen that such an assignment makes the test for direct delivery efficient. It also means that routing tables only need to contain network prefixes and not full IP addresses.

5.7 Next-Hop forwarding

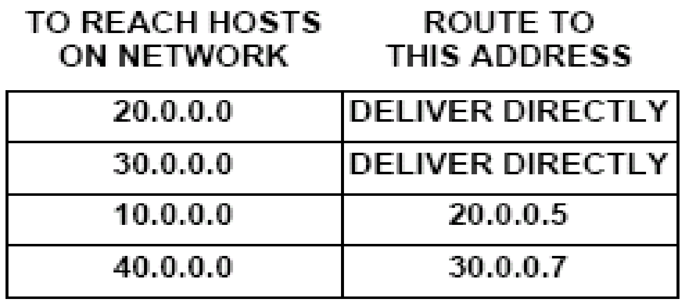
Using the network portion of a destination address instead of the complete host address makes routing efficient and keeps routing tables small. More important, it helps hide information, keeping the details of specific hosts confined to the local environment in which those hosts operate. Typically, a routing table contains pairs (N, R), where N is the IP address of a destination network, and R is the IP address of the "next" router along the path to network N. Router R is called the next hop, and the idea of using a routing table to store a next hop for each destination is called next-hop routing. Thus, the routing table in a router R only specifies one step along the path from R to a destination network - the router does not know the complete path to a destination.

It is important to understand that each entry in a routing table points to a router that can be reached across a single network. That is, all routers listed in machine M's routing table must lie on networks to which M connects directly. When a datagram is ready to leave M, IP software locates the destination IP address and extracts the network portion. M then uses the network portion to make a routing decision, selecting a router that can be reached directly.
In practice, we apply the principle of information hiding to hosts as well. We insist that although hosts have IP routing tables, they must keep minimal information in their tables. The idea is to force hosts to rely on routers for most routing.

Figure 5.3 shows a concrete example that helps explain routing tables. The example internet consists of four networks connected by three routers. In the figure, the routing table gives the routes that router R uses. Because R connects directly to networks 20.0.0.0 and 30.0.0.0, it can use direct delivery to send to a host on either of those networks (possibly using ARP to find physical addresses). Given a datagram destined for a host on network 40.0.0.0, R routes it to the address of router S, 30.0.0.7. S will then deliver the datagram directly. R can reach address 30.0.0.7 because both R and S attach directly to network 30.0.0.0.



(a)



TO REACH HOSTS ON NETWORK	ROUTE TO THIS ADDRESS
20.0.0.0	DELIVER DIRECTLY
30.0.0.0	DELIVER DIRECTLY
10.0.0.0	20.0.0.5
40.0.0.0	30.0.0.7

Figure 5.3 (a) An example intemet with 4 networks and 3 routers, and (b) the routing table in R.

As Figure 5.3 demonstrates, the size of the routing table depends on the number of networks in the intemet; it only grows when new networks are added. However, the table size and contents are independent of the number of individual hosts connected to the networks. We can summarize the underlying principle:

To hide information, keep routing tables small, and make routing decisions efficient, IP routing software only keeps information about destination network addresses, not about individual host addresses.

Choosing routes based on the destination network ID alone has several consequences. First, in most implementations, it means that all traffic destined for a given network takes the same path. As a result, even when multiple paths exist, they may not be used concurrently. Also, all types of traffic follow the same path without regard to the delay or throughput of physical networks. Second, because only the final router along the path attempts to communicate with the destination host, only it can determine if the host exists or is operational. Thus, we need to arrange a way for that router to send reports of delivery problems back to the original source. Third, because each router forwards traffic independently, datagram traveling from host A to host B may follow an entirely different path than datagrams traveling from host B back to host A. We need to ensure that routers cooperate to guarantee that two-way communication is always possible.

5.8 Default Routers and Host specific Rout

Another technique used to hide information and keep routing table sizes small consolidates multiple entries into a default case. The idea is to have the IP routing software first look in the routing table for the destination network. If no route appears in the table, the routing routines send the datagram to a default router.

Default routing is especially useful when a site has a small set of local addresses and only one connection to the rest of the internet. For example, default routes work well in host computers that attach to a single physical network and reach only one router leading to the remainder of the internet. The routing decision consists of two tests: one for the local net and a default that points to the only router. Even if the site contains a few local networks, the routing is simple because it consists of a few tests for the local networks plus a default for all other destinations.

Although we said that all routing is based on networks and not on individual hosts, most IP routing software allows per-host routes to be specified as a special case. Having per-host routes gives the local network administrator more control over network use, permits testing, and can also be used to control access for security purposes. When debugging network connections or routing tables, the ability to specify a special route to one individual machine turns out to be especially useful.

5.9 Routing information Protocol (RIP)

5.9.1 History of RIP

One of the most widely used IGPs is the Routing Information Protocol (RIP), also known by the name of a program that implements it, routed. The routed software was originally designed at the University of California at Berkeley to provide consistent routing and reachability information among machines on their local networks. It relies on physical network broadcast to make routing exchanges quickly. It was not designed to be used on large, wide area networks (although vendors now sell versions of RIP adapted for use on WANs). Based on earlier internetworking research done at Xerox Corporation's Palo Alto Research Center (PARC), routed implements a protocol derived from the Xerox NS Routing Information Protocol (RIP), but generalizes it to cover multiple families of networks.

Despite minor improvements over its predecessors, the popularity of RIP as an IGP does not arise from its technical merits alone. Instead, it is the result of Berkeley distributing routed software along with their popular BSD UNIX systems. Thus, many sites adopted and installed routed, and started using RIP without even considering its technical merits or limitations. Once installed and running, it became the basis for local routing, and research groups adopted it for larger networks.

Perhaps the most startling fact about RTP is that it was built and widely adopted before a formal standard was written. Most implementations were derived from the Berkeley code, with interoperability among them limited by the programmer's understanding of undocumented details and subtleties. As new versions appeared, more problems arose. An RFC standard appeared in June 1988, and made it possible for vendors to ensure interoperability.

5.9.2 RIP operation

The underlying RIP protocol is a straightforward implementation of distance-vector routing for local networks. It partitions participants into active and passive (i.e., silent) machines. Active participants advertise their routes to others; passive participants listen to RIP messages and use them to update their routing table, but do not advertise. Only a router can run RIP in active mode; a host must use passive mode.

A router running RIP in active mode broadcasts a routing update message every 30 seconds. The update contains information taken from the router's current routing database. Each update contains a set of pairs, where each pair contains an IP network address and an integer distance to that network. RIP uses a hop count metric to measure distances. In the RIP metric, a router is defined to be one hop from a directly connected network?, two hops from a network that is reachable through one other router, and so on. Thus, the number of hops or the hop count along a path from a given source to a given destination refers to the number of routers that a datagram encounters along that path. It should be obvious that using hop counts to calculate shortest paths does not always produce optimal results. For example, a path with hop count 3 that crosses three Ethernets may be substantially faster than a path with hop count 2 that crosses two satellite connections. To compensate for differences in technologies, many RIP implementations allow managers to configure artificially high hop counts when advertising connections to slow networks. Both active and passive RIP participants listen to all broadcast messages, and update their tables according to the distance-vector algorithm described earlier. For example, in the internet of Figure 16.2, router R, will broadcast a message on network 2 that contains the pair (1, I), meaning that it can reach network 1 at cost 1. Routers R, and R, will receive the broadcast and install a route to network 1 through R, (at cost 2). Later, routers R, and R, will include the pair (1,2) when they broadcast their RIP messages on network 3. Eventually, all routers and hosts will install a route to network 1.

RIP specifies a few rules to improve performance and reliability. For example, once a router learns a route from another router, it must apply hysteresis, meaning that it does not replace the route with an equal cost route. In our example, if routers R, and R, both advertise network 1 at cost 2, routers R, and R, will install a route through the one that happens to advertise first. We can summarize:

To prevent oscillation among equal cost paths, RIP specifies that existing routes should be retained until a new route has strictly lower cost.

What happens if the first router to advertise a route fails (e.g., if it crashes)? RIP specifies that all listeners must timeout routes they learn via RIP. When a router installs a route in its table, it starts a timer for that route. The timer must be restarted whenever the router receives another RIP message advertising the route. The route becomes invalid if 180 seconds pass without the route being advertised again.

RIP must handle three kinds of errors caused by the underlying algorithm. First, because the algorithm does not explicitly detect routing loops, RIP must either assume participants can be trusted or take precautions to prevent such loops. Second, to prevent instabilities RIP must use a low value for the maximum possible distance (RIP uses 16). Thus, for internets in which legitimate hop counts approach 16, managers must divide the internet into sections or use an alternative protocol. Third, the distance vector algorithm used by RIP can create a slow convergence or count to infinity problem, in which inconsistencies arise because routing update messages propagate slowly across the network. Choosing a small infinity (16) helps limit slow convergence, but does not elirninate it.

5.10 HELLO PROTOCOL

The HELLO protocol provides an example of an IGP that uses a routing metric other than hop count. Although HELLO is now obsolete, it was significant in the history of the Internet because it was the IGP used among the original NSFNET backbone "fuzzball" routers?. HELLO is significant to us because it provides an example of a protocol that uses a metric of delay.

HELLO provides two functions: it synchronizes the clocks among a set of machines, and it allows each machine to compute shortest delay paths to destinations. Thus, HELLO messages carry timestamp information as well as routing information. The basic idea behind HELLO is simple: each machine participating in the HELLO exchange maintains a table of its best estimate of the clocks in neighboring machines. Before transmitting a packet, a machine adds its timestamp by copying the current clock value into the packet. When a packet arrives, the receiver computes an estimate of the current delay on the link by subtracting the timestamp on the incoming packet from the local estimate for the current clock in the neighbor. Periodically, machines poll their neighbors to reestablish estimates for clocks.

HELLO messages also allow participating machines to compute new routes. The protocol uses a modified distance-vector scheme that uses a metric of delay instead of hop count. Thus, each machine periodically sends its neighbors a table of destinations it can reach and an estimated delay for each. When a message arrives from machine X, the receiver examines each entry in the message and changes the next hop to X if the route through X is less expensive than the current route (i.e., any route where the delay to X plus the delay from X to the destination is less than the current delay to the destination).

5.11 The open SPF Protocol (OSPF)

To encourage the adoption of link state technology, a working group of the Internet Engineering Task Force has designed an interior gateway protocol that uses the link state algorithm called Open SPF (OSPF), the new protocol tackles several ambitious goals.

- As the name implies, the specification is available in the published literature. Making it an open standard that anyone can implement without paying license fees has encouraged many vendors to support OSPF. Consequently, it has become a popular replacement for proprietary protocols.

- OSPF includes type of service routing. Managers can install multiple routes to a given destination, one for each priority or type of service. When routing a datagram, a router running OSPF uses both the destination address and type of service field in an IP header to choose a route. OSPF is among the first TCP/IP protocols to offer type of service routing.

- OSPF provides load balancing. If a manager specifies multiple routes to a given destination at the same cost, OSPF distributes traffic over all routes equally. Again, OSPF is among the first open IGPs to offer load balancing; protocols like RIP compute a single route to each destination.

- To permit growth and make the networks at a site easier to manage, OSPF allows a site to partition its networks and routers into subsets called areas. Each area is self contained; knowledge of an area's topology remains hidden from other areas. Thus, multiple groups within a given site can cooperate in the use of OSPF for routing even though each group retains the ability to change its internal network topology independently.

- The OSPF protocol specifies that all exchanges between routers can be authenticated. OSPF allows a variety of authentication schemes, and even allows one area to choose a different scheme than another area. The idea behind authentication is to guarantee that only trusted routers propagate routing information. To understand why this could be a problem, consider what can happen when using RIP1, which has no authentication. If a malicious person uses a personal computer to propagate RIP messages advertising low cost routes, other routers and hosts running RIP will change their routes and start sending datagrams to the personal computer.

- OSPF includes support for host-specific, subnet-specific, and classless routes as well as classful network-specific routes. All types may be needed in a large internet.

- To accommodate multi-access networks like Ethernet, OSPF extends the SPF algorithm We described the algorithm using a point-to-point graph and said that each router running SPF would periodically broadcast link status messages about each reachable neighbor. If K routers attach to an Ethernet, they will broadcast K2 reachability messages. OSPF minimizes broadcasts by allowing a more complex graph topology in which each node represents either a router or a network. Consequently, OSPF allows every multi-access network to have a designated gateway (i.e., a designated router) that sends link status messages on behalf of all routers attached to the network; the messages report the status of all links from the network to routers attached to the network. OSPF also uses hardware broadcast capabilities, where they exist, to deliver link status messages.

- To permit maximum flexibility, OSPF allows managers to describe a virtual network topology that abstracts away from the details of physical connections. For exarnple, a manager can configure a virtual link between two routers in the routing graph even if the physical connection between the two routers requires communication across a transit network.

- OSPF allows routers to exchange routing information learned from other (external) sites. Basically, one or more routers with connections to other sites learn information about those sites and include it when sending update messages. The message format distinguishes between information acquired from external sources and information acquired from routers interior to the site, so there is no ambiguity about the source or reliability of routes.

5.12 Static Vs Dynamic Interior Routers (BGP)

Computer scientists use the term Exterior Gateway Protocol (EGP) to denote any protocol used to pass routing information between two autonomous systems. Currently a single exterior protocol is used in most internets. Known as the Border Gate way Protocol (BGP), it has evolved through four (quite different) versions. Each version is numbered, which gives rise to the formal name of the current version: BGP-4.

When a pair of autonomous systems agree to exchange routing information, each must designate a router that will speak BGP on its behalf; the two routers are said to become BGP peers of one another. Because a router speaking BGP must communic ate with a peer in another autonomous system, it makes sense to select a machine that is near the "edge" of the autonomous system. Hence, BGP terminology calls the machine a border gateway or border router.

BGP is unusual in several ways. Most important, BGP is neither a pure distance vector protocol nor a pure link state protocol. It can be characterized by the following:

- Inter-Autonomous System Communication. Because BGP is designed as an exterior gateway protocol, its primary role is to allow one autonomous system to communicate with another.

- Coordination Among Multiple BGP Speakers. If an autonomous system has multiple routers each communicating with a peer in an outside autonomous system, BGP can be used to coordinate among routers in the set to guarantee that they all propagate consistent information.

- Propagation Of Reachability Information BGP allows an autonomous system to advertise destinations that are reachable either in or through it, and to learn such information from another autonomous system.
- Next-Hop Paradigm. Like distance-vector routing protocols, BGP supplies next hop information for each destination.

- Policy Support. Unlike most distance-vector protocols that advertise exactly the routes in the local routing table, BGP can implement policies that the local administrator chooses. In particular, a router running BGP can be configured to distinguish between the set of destinations reachable by computers inside its autonomous system and the set of destinations advertised to other autonomous systems.
- Reliable Transport. BGP is unusual among protocols that pass routing information because it assumes reliable transport. Thus, BGP uses TCP for all communication.

- Path Information. In addition to specifying destinations that can be reached and a next hop for each, BGP advertisements include path information that allows a receiver to learn a series of autonomous systems along a path to the destination.

- Incremental Updates. To conserve network bandwidth, BGP does not pass full information in each update message. Instead, full information is exchanged once, and then successive messages carry incremental changes called deltas.

- Support For Classless Addressing. BGP supports CIDR addresses. That is, rather than expecting addresses to be self-identifying, the protocol provides a way to send a mask along with each address.

- Route Aggregation. BGP conserves network bandwidth by al lowing a sender to aggregate route information and send a single entry to represent multiple, related destinations.

- Authentication. BGP allows a receiver to authenticate messages (i.e., verify the identity of a sender).

CHAPTER 6
CIDR SUBNETTING

6.1 Introduction

The chief advantage of dividing an IP address into two parts arises from the size of the routing tables required in routers. Instead of keeping one routing entry per destination host, a router can keep one routing entry per network, and examine only the network portion of a destination address when making routing decisions.

Recall that the original IP addressing scheme accommodated diverse network sizes by dividing host addresses into three primary classes. Networks assigned class *A*addresses partition the 32 bits into an 8-bit network portion and a 24-bit host portion. Class *B* addresses partition the 32 bits into 16-bit network and host portions, while class C partitions the address into a 24-bit network portion and an 8-bit host portion.

To understand some of the address extensions in this chapter, it will be important to realize that individual sites have the freedom to modify addresses and routes as long as the modifications remain invisible to other sites. That is; a site can choose to assign and use IP addresses in unusual ways internally as long as:

- All hosts and routers at the site agree to honor the site's addressing scheme.
- Other sites on the Internet can treat addresses as a network prefix and a host suffix.

6.2 Proxy ARP

The terms *proxy ARP*, *promiscuous ARP*, and *the ARP hack* refer to a second technique used to map a single IP network prefix into two physical addresses. The technique, which only applies to networks that use ARP to bind internet addresses to physical addresses, can best be explained with an example. Figure illustrates the situation.

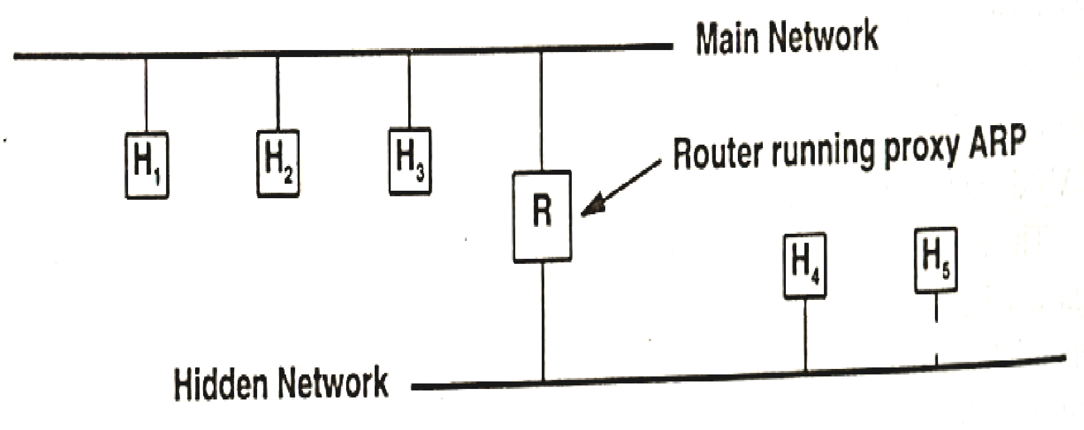


Figure6.1: Proxy ARP technique (the ARP hack)

Proxy ARP technique (the ARP hack) allows one network address to be shared between two physical nets. Router *R* answers ARP requests on each network for hosts on the other network, giving its hardware address and then routing datagrams correctly when they arrive. In essence, *R* lies about IP-lo-physical address bindings.

In the figure two network share a single IP network address. Imagine that labeled *Main Network* was the original network, and that the second, labeled network a hidden network, was added later. The router connecting the two networks, *R, kn*ows which host lie on which physical network and uses ARP to maintain the illusion that only one network exists, To make the illusion work, *R* keeps the location of hosts completely hidden, allowing all other machines on the

network to communicate as if directly connected. In our example when host *H_1*needs to communicate with host $H_{4,}$ it first invokes ARP to map *H_4*'s IP address into a physical address. Once it has a physical address. *H_1*can send the datagram directly to that physical address.
Because *R* runs proxy ARP software, it captures the broadcast ARP request from *H* decides that the machine in question lies on the other physical network, and responds to the ARP request by sending its own physical address. *HI* receives the ARP response, installs the mapping in its ARP table, and then uses the mapping to send datagrams destined for *H_4* to *R.* When *R* receives a datagram, It searches a special, routing table to determine how to route the datagram. *R* must forward datagrams destined for *H* over the hidden network. To allow hosts on the hidden network to reach hosts on 4, the main network, *R* performs the proxy ARP service on that network as well.

Routers using the proxy ARP technique are taking advantage of an important Feature of the ARP protocol, namely, trust. ARP is based on the idea that all machines cooperate and that any response is legitimate. Most hosts install mappings obtained through ARP without checking their validity and without maintaining consistency. Thus, it may happen that the ARP table maps several IP addresses to the same physical address, but that does not violate the protocol specification.

Some implementations of ARP are not as lax as others. In particular, ARP implementations designed to alert managers to possible security violations will inform them whenever two distinct IP addresses map to the same physical hardware address. The purpose of alerting the manager is to warn about *spoofing,* a situation in which one machine claims to be another in order to intercept packets, Host implementations of ARP that warn managers of possible spoofing cannot be used on networks that have Proxy ARP routers because the software will generate messages frequently.

The chief advantage of proxy ARP is that it can be added to a single router on a network without disturbing the routing tables in other hosts or routers on that network. Thus, proxy ARP completely hides the details of physical connections.

The chief disadvantage of proxy ARP is that it does not work for networks unless they use ARP for address resolution, Furthermore, it does not generalize to more complex network topology (e.g.: multiple routers interconnecting two physical networks) nor does it b support a reasonable form of routing, In fact most implementations of proxy ARP in rely on managers to maintain tables of machines and addresses manually, making it both time consuming and prone to errors.

6.3 Subnet Addressing

The second technique used to allow a single network address to span multiple physical networks is called *subnet addressing, subnet routing,* or *subnetting.* Subnetting is the most widely used of the three techniques because it is the most general and because it has been standardized. In fact, subnetting is a required part of IP addressing. The easiest way to understand subnet addressing is to imagine that a site has a single class *B* IP network address assigned to it, but it has two or more physical networks. Only local routers know that there are multiple physical nets and how to route traffic among them; routers in other autonomous systems route all traffic as if there were a single physical network. Figure 6.2 shows an example.

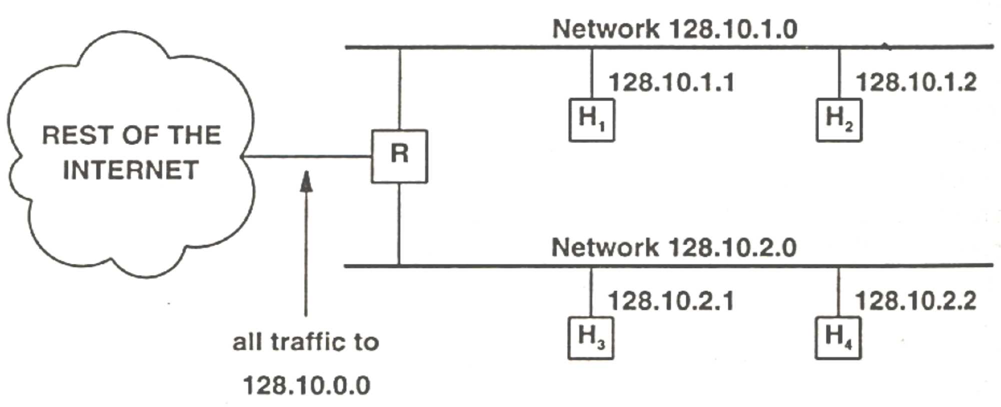


Figure 6.2 A site with two physical networks using subnet addressing to label them with a single class *B* network address.

A site with two physical networks using subnet addressing to label them with a single class *B* network address. Router *R* accepts all traffic for net 128.10.0.0 and chooses a physical network based on the third octet of the address.

In the example, the site is using the single class *B* network address *128.10.0.0* for two networks. Except for router *R*, all routers in the internet route as if there were a single physical net. Once a packet reaches *R*, it must be sent across the correct physical network to its destination. To make the choice of physical network efficient, the local site has chosen to use the third octet of the address to distinguish between the two networks. The manager assigns machines on one physical net addresses of the form *128.10.1.*X, and machines on the other physical net addresses of the form *128.10.2.*X, where X, the final octet of the address, contains a small integer used to identify a specific host. To choose a physical network, *R* examines the third octet of the destination address and routes datagrams with value 1 to the network labeled *128.10.1.0* and those with value 2 to the network labeled *128.10.2.0.*

Conceptually, adding subnets only changes the interpretation of IP addresses slightly. Instead of dividing the 32-bit IP address into a network prefix and a host suffix, subnetting divides the address into a *network portion* and a *local portion.* The interpretation of the network portion remains the same as for networks that do not use subnetting. As before, reach ability to the network must be advertised to outside autonomous systems; all traffic destined for the network will follow the advertised route. The interpretation of the local portion of an address is left up to the site (within the constraints of the formal standard for subnet addressing). To summarize:

We think of a 32-bit IP address as having an internet portion and a local portion, where the internet portion identifies a site, possibly with multiple physical networks, and the local portion identifies a physical network and host at that site.

The example of Figure showed subnet addressing with a class *B* address that had a 2-octet internet portion and a 2-octet local portion. To make routing among the physical networks efficient, the site administrator in our example chose to use one octet of the local portion to identify a physical network, and the other octet of the local portion to identify a host on that network, as Figure6.3 shows.

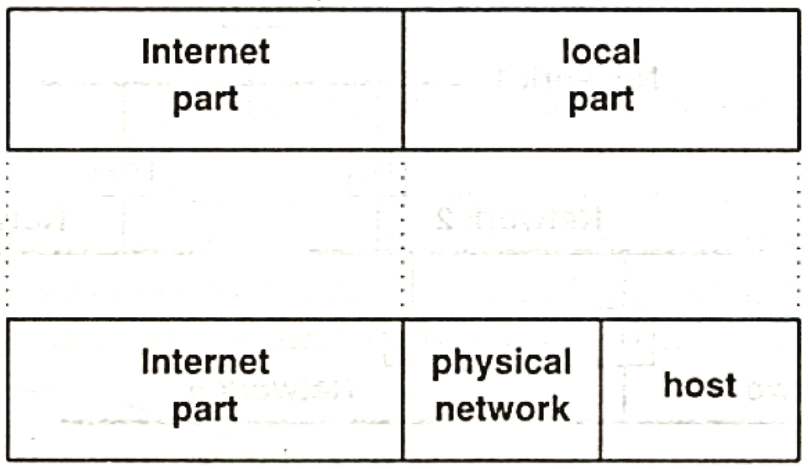


Figure 6.3: Redistribution of Class B address into network ,subnetwork and host portion.

The result is a form of *hierarchical addressing* that leads to corresponding *hierarchical routing.* The top level of the routing hierarchy (i.e., other autonomous systems in the internet) uses the first two octets when routing, and the next level (i.e., the local site) uses an additional octet. Finally, the lowest level (i.e., delivery across one physical network) uses the entire address.
Hierarchical addressing is not new; many systems have used it before. The best example is the U.S. telephone system, where a 10-digit phone number is divided into a 3-digit area code, 3-digit exchange, and 4-digit connection. The advantage of using hierarchical addressing is that it accommodates large growth because it means a given router does not need to know as much detail about distant destinations as it does about local ones. One disadvantage is that choosing a hierarchical structure is difficult, and it often becomes difficult to change a hierarchy once it has been established.

6.4 Variable – Length Subnets
We have implied that choosing a subnet addressing scheme is synonymous with choosing how to partition the local portion of an IP address into physical net and host parts. Indeed, most sites that implement subnetting use a fixed-length assignment. It should be clear that the designers did not choose a specific division for subnetting because no single partition of the local part of the address works for all organizations some need many networks with few hosts per network, while others need a few networks with many hosts attached to each. The designers realized that the same problem can exist within a single organization. To allow maximum autonomy, the TCPAP subnet standard provides even more flexibility than indicated above. An organization may select a subnet partition on a per network basis. Although the technique is known as variable-length subnetting, the name is slightly misleading because the value does not "vary" over time - once a partition has been selected for a particular network, the partition never changes. All hosts and routers attached to that network must follow the decision; if they do not, datagrams can be lost or rnisrouted. We can summarize:

To allow maximum flexibility in choosing how to partition subnet addresses, the subnet standard permits variable-length subnetting in which the partition can be chosen independently for each physical network. Once a subnet partition has been selected, all machines on that network must honour it.

The chief advantage of variable-length subnetting is flexibility: an organization can have a mixture of large and small networks, and can achieve higher utilization of the address space. However, variable-length subnetting has serious disadvantages. Most important, values for subnets must be assigned carefully to avoid address ambiguity, a situation in which an address is interpreted differently depending on the physical network. For example, an address can appear to match two different subnets. As a result, invalid variable-length subnets may make it impossible for all pairs of hosts to communicate. Routers cannot resolve such ambiguity, which means that an invalid assignment can only be repaired by renumbering. Thus, network managers are discouraged from using variable-length subnetting.

6.5 Implementation of subnets with masks

Specifying subnet masks in binary is both awkward and prone to errors. Therefore, most software allows alternative representations. Sometimes, the representation follows whatever conventions the local operating system uses for representation of binary quantities, (e.g., hexadecimal notation).

Most IP software uses dotted decimal representation for subnet masks; it works best when sites choose to align subnetting on octet boundaries. For example, many sites choose to subnet class *B* addresses by using the third octet to identify the physical net and the fourth octet to identify hosts as on the previous page. In such cases, the subnet mask has dotted decimal representation *255.255.255.0,* making it easy to write and understand.

6.6 Subnet mask Representation

The literature also contains examples of subnet addresses and subnet masks represented in braces as a 3-tuple:

{ <network number> , <subnet number> , <host number> }

In this representation, the value -1 means "all ones." For example, if the subnet mask for a class *B* network is *255.255.255.0,* it can be written {-1, -1, *0*}.

The chief disadvantage of the 3-tuple representation is that it does not accurately specify how many bits are used for each part of the address; the advantage is that it abstracts away from the details of bit fields and emphasizes the values of the three parts of the address. To see why address values are sometimes more important than bit fields, consider the 3-tuple:

{ 128.10,-1,0}

which denotes an address with a network number *128.10,* all ones in the subnet field, and all zeroes in the host field. Expressing the same address value using other representations requires a

32-bit subnet mask as well as a 32-bit IP address, and forces readers to decode bit fields before they can deduce the values of individual fields. Further more, the 3-tuple representation is independent of the IP address class or the size of the subnet field. Thus, the 3-tuple can be used to represent sets of addresses or abstract ideas. For example, the 3-tuple:

{ <network number>, -1, -1 }

denotes "addresses with a valid network number, a subnet field containing all ones, and a host field containing all ones". We will see additional examples later in this chapter.

6.7 Classless Addressing and Super netting

The addressing scheme described so far requires a unique network prefix for each physical network. Although that was, indeed, the original plan, it did not last long. In the 1980s as Local Area Network technologies became increasingly popular, it became apparent that requiring a unique prefix for each physical network would exhaust the address space quickly. Consequently, an addressing extension was developed to conserve network prefixes. Known as *subnet addressing,* the scheme allows multiple physical networks to share a prefix. In the 1990s, a second extension was devised that ignored the classful hierarchy and allowed the division between prefix and suffix to occur at an arbitrary point. Called *classless addressing* or *supernetting,* the scheme allows more complete utilization of the address space. Chapter 10 will consider details of the subnet and supernet addressing extensions. For now, it is only important to know that the addressing scheme has been extended, and that the original classful scheme described in this chapter is no longer the most widely used.

6.8 Assignment Questions

Unit II

Multiple Choice Questions

1. The general form of IP datagram contains________
   a. IP header only
   b. IP Data only
   c. IP header and Data
   d. IP header, Data and control signals
2. The VERS field in the IP datagram specifies ______
   a. **IP Version** b. Data Version c. Protocol Version d. All the above
3. The HLEN field in the IP datagram specifies
   a. Length of the total IP Datagram
   b. Length of the IP Header
   c. Length of the Body of the IP datagram
   d. Total length of IP datagram
4. The Time To Live field refers to _____
   a. The time taken by the IP datagram to reach the destination.
   b. The time taken by the IP datagram to start from the source.
   c. The total time that the IP datagram is active.
   d. The time taken by the IP datagram to survive in the network.
5. The direct IP datagram delivery refers to _____
   a. The delivery of the datagram directly to the outside network.
   b. The delivery of the datagram to the same network
   c. The delivery of the datagram to router.
   d. The simple way of delivery of datagram.
6. The indirect delivery refers to______
   a. Delivery of datagram to a single network.
   b. Delivery of datagram to a multiple networks.
   c. Delivery of datagram to a different network.
   d. Delivery of datagram to both single and multiple networks.
7. The IP routing table is needed for
   a. IP datagram delivery to an inside network
   b. IP datagram delivery to outside network
   c. IP datagram delivery to internetwork.
   d. All the above.
8. The router is useful when a site has a small set of local addresses and only one connection to the rest of the internet is _____
   a. Host specific routers
   b. Table driven routers
   c. Default routers
   d. All the above
9. The purpose of the routing information protocol is
   a. To inform the routing to the routers.
   b. To dynamically update the routing table with latest updating.
   c. To route the IP datagram

- **d.** To manage the routers and hosts.

10. The active participants in RIP operation are
- **a.** Advertising themselves.
- **b.** Listening to advertisements.
- **c. Both Advertising and Listening.**
- **d.** None of the Above.

11. The router update routing information once in _______ time.
- **a.** 10 secs. b. 20 secs. **c. 30 secs.** d. 40 secs.

12. Which protocol is used to find out the hop count?
- **a.** RIP b. BGP c. IGP **d. HELLO**

13. The protocol which computes the shortest delay path to the destination is
- **a. HELLO** b. IGP c. BGP d. RIP

14. Any protocol used to pass routing information between two autonomous systems is_____
- **a.** IGP b**. EGP** c. RIP d. ARP

15. The current EGP used for passing routing information between two autonomous system is _________
- **a. BGP** b. IGP c. ARP. d. HELLO

16. The protocol which is used to map a single IP Network prefix into two physical network is called _____
- **a.** ARP **b. Proxy ARP** c. BGP d. HELLO

17. ARP Hack is another name given to _________
- **a.** ARP **b. Proxy ARP** c. BGP d. HELLO

18. Which addressing has three parts in the IP address.
- **a.** ARP **b. subnet addressing** c. Proxy ARP d. Classfull

19. -1 in subnet addressing represents ________
- **a. All ones.** b. ones compliment c. Twos compliment d. None

20. VLSM stands for ______
- **a.** Value added Length Supernet Mask
- **b.** Value Length Subnet Mask
- **c. Variable Length Subnet Mas**
- **d.** None of the above

21. What is OSPF?
- **a. Open Source Shortest Path Finder.**
- **b.** Open Source Smart Path Finder
- **c.** Operation Source Smart Path Finder
- **d.** Optional Source Smart Path Finder.

22. The protocol which uses distance vector algorithm is
- **a.** ARP b. HTTP c. **HELLO** d. FTP

23. The TOTAL LENGTH field in IP datagram is for _____
- **a.** To find the length of the header
- **b.** To find the length of the data
- **c. To find the total length of the IP datagram**
- **d.** None of the above.

24. The maximum size of the IP datagram is
- **a.** 65535 octates b. 12466 octates

c. 565 octates d. 32250 octaes

25. Forwarding the IP datagram is done using

a. RIP b**. Routing Table.** c. switch d. Hub

Answer the following questions. Each carries 2 marks.

1. Draw the general format of IPv4 datagram.
2. What is the maximum size of IP datagram?
3. What are default routers?
4. What do you mean by RIP?
5. Write a note on OSPF.
6. What is BGP?
7. What do you mean by spoofing?
8. How do you redistribute the class B addressing for net id, subnet id and host id?
9. What are subnet masks?
10. Give an example for 3 touple representation of class b addressing with subnetworks.

Essay Questions

1. With the help of a diagram explain the format of IP datagram.
2. Explain how to deliver a datagram over a single network?
3. Explain how the indirect delivery is functioning?
4. Explain the process of Table Driven IP forwarding.
5. Write a note on next hop forwarding.
6. Explain the history of RIP.
7. Write a note on HELLO protocol.
8. Explain The various goals of OSPF protocol.
9. List and explain the characteristics of BGP protocol.
10. With the help of a diagram explain Proxy ARP.
11. Explain the concept of subnet addressing with the help of a diagram.
12. Explain the need and functioning of subnet masks.
13. What are variable length subnets? Explain.
14. Write a note on supernetting.

UNIT III
CHAPTER 7
UDP (USER DATAGRAM PROTOCOL)

7.1 User datagram Protocol

In the protocol suite, the User Datagram Protocol or UDP provides the primary mechanism that application programs use to send datagrams to other application programs. UDP provides protocol ports used to distinguish among multiple programs executing on a single machine. That is, in addition to the data sent, each UDP message contains both a destination port number and a source port number, making it possible for the UDP software at the destination to deliver the message to the correct recipient and for the recipient to send a reply.

UDP uses the underlying internet layer to transport a message from one machine to another, and provides the same unreliable, connectionless datagram delivery semantics as IP. It does not use acknowledgements to make sure messages arrive; it does not order incoming messages, nor does it provide feedback to control the rate at which information flows between the machines. Thus, UDP messages can be lost, duplicated, or arrive out of order. Furthermore, packets can arrive faster than the recipient can processthem. We can summarize:

The User Datagram Protocol (UDP) provides an unreliable connectionless delivery service using IP to transport messages between machines. It uses IP to carry messages, but adds the ability to distinguish among multiple destinations within a given host computer.

An application program that uses UDP accepts full responsibility for handling the problem of reliability, including message loss, duplication, delay, out-of-order delivery, and loss of connectivity. Unfortunately, application programmers often ignore these problems when designing software. Furthermore, because programmers often test network software using highly reliable, low-delay local area networks, testing may not expose potential failures. Thus, many application programs that rely on UDP work well in a local environment, but fail in dramatic ways when used in the global Internet.

7.2 Format of UDP Message

Each UDP message is called a user datagram. Conceptually, a user datagram consists of two parts: a UDP header and a UDP data area. As Figure 7.1 shows, the header is divided into four 16-bit fields that specify the port from which the message was sent, the port to which the message is destined, the message length, and a UDP checksum.

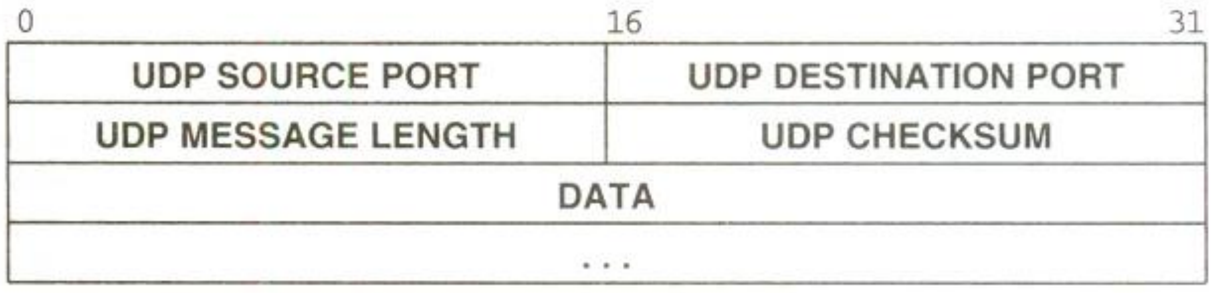


Figure 7.1 The format of fields in a UDP datagram.

The SOURCE PORT and DESTINATION PORT fields contain the 16-bit UDP protocol port numbers used to demultiplex datagrams among the processes waiting to receive them. The SOURCE PORT is optional. When used, it specifies the port to which replies should be sent; if not used, it should be zero.

The LENGTH field contains a count of octets in the UDP datagram, including the UDP header and the user data. Thus, the minimum value for LENGTH is eight, the length of the header alone. The UDP checksum is optional and need not be used at all; a value of zero in the CHECKSUM field means that the checksum has not been computed. The designers chose to make the checksum optional to allow implementations to operate with little computational overhead when using UDP across a highly reliable local area network.

Recall, however, that IP does not compute a checksum on the data portion of an IP datagram. Thus, the UDP checksum provides the only way to guarantee that data has arrived intact and should be used.

Beginners often wonder what happens to UDP messages for which the computed checksum is zero. A computed value of zero is possible because UDP uses the same checksum algorithm as IP: it divides the data into 16-bit quantities and computes the one's complement of their one's complement sum. Surprisingly, zero is not a problem because one's complement arithmetic has two representations for zero: all bits set to zero or all bits set to one. When the computed checksum is zero, UDP uses the representation with all bits set to one.

7.3 Layering and the Checksum computation

The UDP checksum includes a pseudo-header that has fields for the source and destination IP addresses. It can be argued that the destination IP address must be known to the user when sending a UDP datagram, and the user must pass it to the UDP layer. Thus, the UDP layer can obtain the destination IP address without interacting with the IP layer. However, the source IP address depends on the route IP chooses for the datagram, because the IP source address identifies the network interface over which the datagram is transmitted. Thus, unless it interacts with the IP layer, UDP cannot know the IP source address.

We assume that UDP software asks the IP layer to compute the source and (possibly) destination IP addresses, uses them to construct a pseudo-header, computes the checksum, discards the pseudo-header, and then passes the UDP datagram to IP for transmission. An alternative approach that produces greater efficiency arranges to have the UDP layer encapsulate the UDP datagram in an IP datagram, obtain the source address from IP, store the source and destination addresses in the appropriate fields of the datagram header, compute the UDP checksum, and then pass the IP datagram to the IP layer, which only needs to fill in the remaining IP header fields.

Does the strong interaction between UDP and IP violate our basic premise that layering reflects separation of functionality? Yes. UDP has been tightly integrated with the IP protocol. It is clearly a compromise of the pure separation, made for entirely practical reasons. We are willing to overlook the layering violation because it is impossible to fully identify a destination application program without specifying the destination machine, and we want to make the mapping between addresses used by UDP andthose used by IP efficient. One of the exercises examines this issue from a differentpoint of view, asking the reader to consider whether UDP should be separated from IP.

7.4 UDP Multiplexing, Demultiplexing and Ports

UDP software provides another example of multiplexing and demultiplexing. It accepts UDP datagrams from many application programs and passes them to IP for transmission, and it accepts arriving UDP datagrams from IP and passes each to the appropriate application program. Conceptually all multiplexing and demultiplexing between UDP software and application programs occur through the port mechanism. In practice, each application program must negotiate with the operating system to obtain a protocol port and an associated port number before it can send a UDP datagram. Once the port has been assigned, any datagram the application program sends through the port will have that port number in its UDP SOURCE PORT field. While processing input, UDP accepts incoming datagrams from the IP software and demultiplexes based on the UDP destination port, as Figure 7.2 shows.

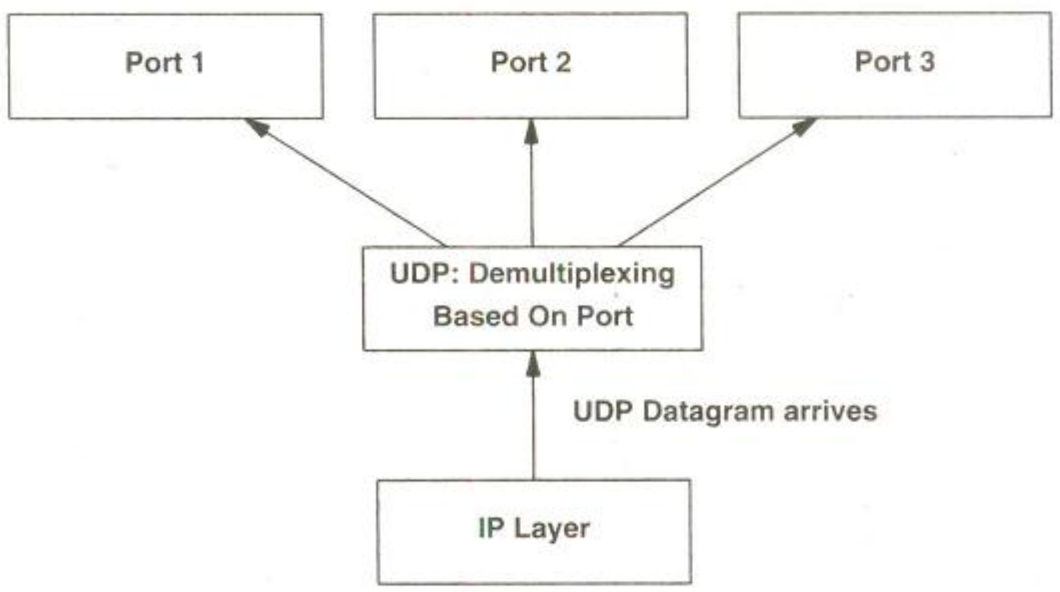


Figure 7.2:Example of demultiplexing one layer above IP. UDP uses the UDP destination port number to select an appropriate destination port for incoming datagrams.

The easiest way to think of a UDP port is as a queue. In most implementations, when an application program negotiates with the operating system to use a given port, the operating system creates an internal queue that can hold arriving messages. Often, the application can specify or change the queue size. When UDP receives a datagram, it checks to see that the destination port number matches one of the ports currently in use.

If not, it sends an ICMP port unreachable error message and discards the datagram. If a match is found, UDP enquires the new datagram at the port where an application program can access it. Of course, an error occurs if the port is full, and UDP discards the incoming datagram.

CHAPTER 8
TCP

8.1 Introduction

The implementation of TCP is also substantially more complex. Although TCP is presented here as part of the Internet protocol suite, it is an independent, general purpose protocol that can be adapted for use with other delivery systems. For example, because TCP makes very few assumptions about the underlying network, it is possible to use it over a single network like an Ethernet, as well as over the global Internet.

8.2 The need for stream Delivery

At the lowest level, computer communication networks provide unreliable packet delivery. Packets can be lost or destroyed when transmission errors interfere with data, when network hardware fails, or when networks become too heavily loaded to accommodate the load presented. Packet switching systems change routes dynamically, deliver packets out of order, deliver them after a substantial delay. or deliver duplicates. Furthermore, underlying network technologies may dictate an optimal packet size or pose other constraints needed to achieve efficient transfer rates.

At the highest level, application programs often need to send large volumes of data from one computer to another. Using an unreliable connectionless delivery system for large volume transfers becomes tedious and annoying, and it requires that programmers build error detection and recovery into each application program. Because it is difficult to design, understand, or modify software that correctly provides reliab ility, few application programmers have the necessary technical background. As a consequence, onegoal of network protocol research has been to find general purpose solutions to the problems of providing reliable stream delivery, making it possible for experts to build a single instance of stream protocol software that all application programs use. Having a single general purpose protocol helps isolate application programs from the details of networking, and makes it possible to define a uniform interface for the stream transfer service.

8.3 Properties of the reliable Delivery Service

The interface between application programs and the reliable delivery service can be characterized by five features:

- Stream Orientation. When two application programs transfer large volumes of data, the data is viewed as a stream of bits, divided into 8-bit octets or bytes. The stream delivery service on the destination machine passes to the receiver exactly the same sequence of octets that the sender passes to it on the source machine.
- Virtual Circuit Connection. Making a stream transfer is analogous to placing a telephone call. Before transfer can start, both the sending and receiving application programs interact with their respective operating systems, informing them of the desire for a stream transfer. Conceptually, one application places a "call" which must be accepted by the other. Protocol software modules in the two operating systems communicate by sending messages across the underlying internet, verifying that the transfer is authorized, and that both sides are ready. Once all details have been settled, the protocol modules inform the application programs that a connection has been established and that transfer can begin. During transfer, protocol software on the two machines continue to communicate to verify that data is received correctly. If the communication fails for any reason (e.g., because network hardware along

the path between the machines fails), both machines detect the failure and report it to the appropriate application programs. We use the term virtual circuit to describe such connections because although application programs view the connection as a dedicated hardware circuit, the reliability is an illusion provided by the stream delivery service.

- Buffered Transfer. Application programs send a data stream across the virtual circuit by repeatedly passing data octets to the protocol software. When transferring data, an application uses whatever size pieces it finds convenient, which can be as small as a single octet. At the receiving end, the protocol software delivers octets from the data stream in exactly the same order they were sent, making them available to the receiving application program as soon as they have been received and verified. The protocol software is free to divide the stream into packets independent of the pieces the application program transfers. To make transfer more efficient and to minimize network traffic, implementations usually collect enough data from a stream to fill a reasonably large datagram before transmitting it across an internet. Thus, even if the application program generates the stream one octet at a time, transfer across an internet may be quite efficient. Similarly, if the application program chooses to generate extremely large blocks of data, the protocol software can choose to divide each block into smaller pieces for transmission. For those applications where data should be delivered without waiting to fill a buffer, the stream service provides a push mechanism that applications use to force immediate transfer. At the sending side, a push forces protocol software to transfer all data that has been generated without waiting to fill a buffer. When it reaches the receiving side, the push causes TCP to make the data available to the application without delay. The reader should note, however, that the push function only guarantees that all data will be transferred; it does not provide record boundaries. Thus, even when delivery is forced, the protocol software may choose to divide the stream in unexpected ways.

- Unstructured Stream. It is important to understand that the stream service does not honor structured data streams. For example, there is no way for a payroll application to have the stream service mark boundaries between employee records, or to identify the contents of the stream as being payroll data. Application programs using the stream service must understand stream content and agree on stream format before they initiate a connection.

- Full Duplex Connection. Connections provided by the stream service allow concurrent transfer in both directions. Such connections are called full duplex. From the point of view of an application process, a full duplex connection consists of two independent streams flowing in opposite directions, with no apparent interaction. The stream service allows an application process to terminate flow in one direction while data continues to flow in the other direction, making the connection half duplex. The advantage of a full duplex connection is that the underlying protocol software can send control information for one stream back to the source in datagrams carrying data in the opposite direction. Such piggybacking reduces network traffic.

8.4 The idea behind sliding windows

Before examining the TCP stream service, we need to explore an additional concept that underlies stream transmission. The concept, known as a sliding window, makes stream transmission efficient. To understand the motivation for sliding windows, To achieve reliability, the sender transmits a packet and then waits for-an acknowledgement before transmitting another. The data only flows between the machines in one direction at anytime, even if the network is capable of simultaneous communication in both directions.

The network will remain completely idle during times that machines delay responses(e.g., while machines compute routes or checksums). If we imagine a network with high transmission delays, the problem becomes clear:

A simple positive acknowledgement protocol wastes a substantial amount of network bandwidth because it must delay sending a new packet until it receives an acknowledgement for the previous packet.

The sliding window technique is a more complex form of positive acknowledgement and retransmission than the simple method discussed above. Sliding window protocols use network bandwidth better because they allow the sender to transmit multiple packets before waiting for an acknowledgement. The easiest way to envision sliding window operation is to think of a sequence of packets to be transmitted as Figure 8.1 shows. The protocol places a small, fixed-size window on the sequence and transmits all packets that lie inside the window.

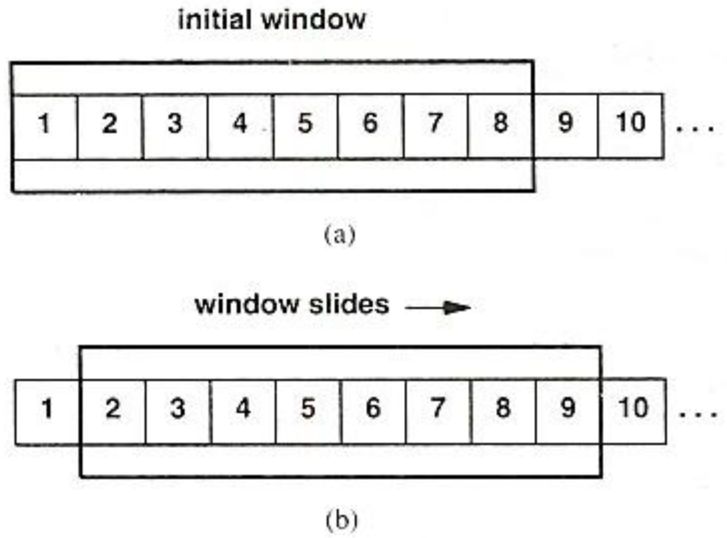


Figure 8.1: (a) A sliding window protocol with eight packets in the window, and (b) The window sliding so that packet 9 can be sent when an acknowledgement has been received for packet 1. Only un acknowledged packets are retransmitted.

We say that a packet is unacknowledged if it has been transmitted but no acknowledgement has been received. Technically, the number of packets that can be unacknowledged at any given time is constrained by the window size, which is limited to a small, fixed number. For example, in a sliding window protocol with window size 8, the sender is permitted to transmit 8 packets before it receives an acknowledgement.

As Figure 8.1 shows, once the sender receives an acknowledgement for the first packet inside the window, it "slides" the window along and sends the next packet. The window continues to slide as long as acknowledgements are received.

The performance of sliding window protocols depends on the window size and the speed at which the network accepts packets. Figure 8.2 shows an example of the operation of a sliding window protocol when sending three packets. Note that the sender transmits all three packets before receiving any 'acknowledgements.

With a window size of 1, a sliding window protocol is exactly the same as our simple positive acknowledgement protocol. By increasing the window size, it is possible to eliminate network idle time completely. That is, in the steady state, the sender can transmit packets as fast as the network can transfer them. The main point is:

Because a well-tuned sliding window protocol keeps the network completely saturated with packets, it obtains substantially higher throughput than a simple positive acknowledgement protocol.

Conceptually, a sliding window protocol always remembers which packets have been acknowledged and keeps a separate timer for each unacknowledged packet. If a packet is lost, the timer expires 'and the sender retransmits that packet. When the sender slides its window, it moves past all acknowledged packets. At the receiving end, the protocol software keeps an analogous window, accepting and acknowledging packets as they arrive. Thus, the window partitions the sequence of packets into three sets: those packets to the left of the window have been successfully transmitted, received, and acknowledged; those packets to the right have not yet been transmitted; and those packets that lie in the window are being transmitted. The lowest numbered packet in the window is the first packet in the sequence that has not been acknowledged.

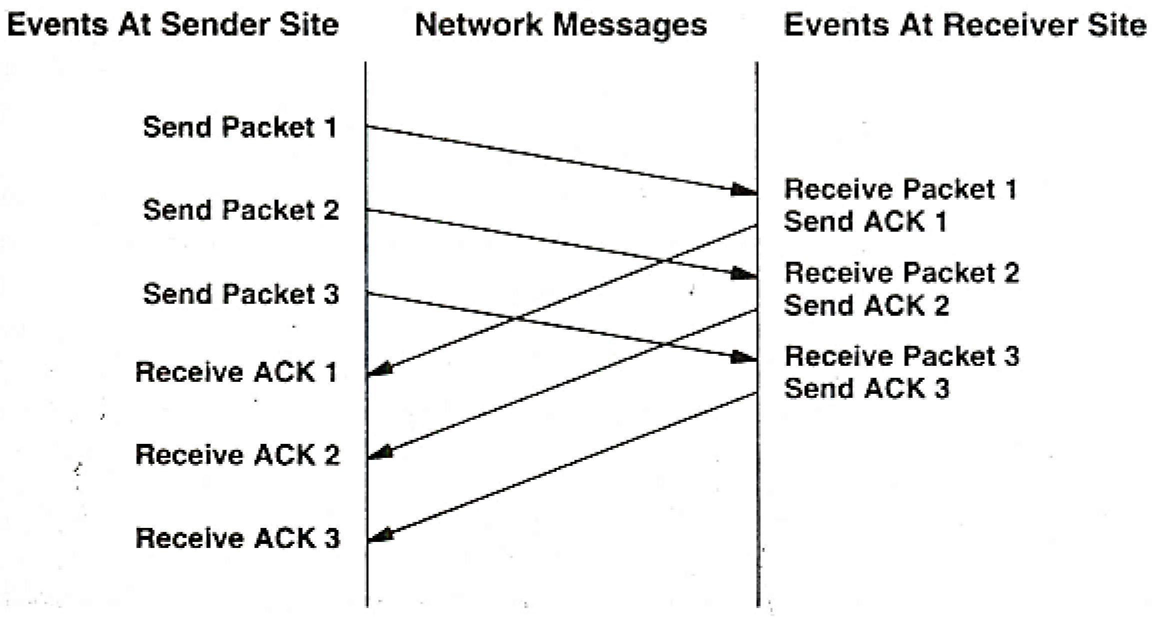


Figure 8.2 An example of three packets transmitted using a sliding window protocol. The key concept is that the sender can transmit all packets in the window without waiting for an acknowledgement.

8.5 The transmission control Protocol

Now that we understand the principle of sliding windows, we can examine the reliable stream service provided by the Internet protocol suite. The service is defined by the Transmission Control Protocol, or TCP. The reliable stream service is so significant that the entire protocol suite is referred to as . It is important to understand that

TCP is a communication protocol but not a piece of software.

The difference between a protocol and the software that implements it is analogous to the difference between the definition of a programming language and a compiler. As in the programming language world, the distinction between definition and implementation sometimes becomes blurred. People encounter TCP software much more frequently than they encounter the protocol specification, so it is natural to think of a particular implementation as the standard. Nevertheless, the reader should try to distinguish between the two.

Exactly what does TCP provide? TCP is complex, so there is no simple answer. The protocol specifies the format of the data and acknowledgements that two computers exchange to achieve a reliable transfer, as well as the procedures the computers use to ensure that the data arrives correctly. It specifies how TCP software distinguishes among multiple destinations on a given machine, and how communicating machines recover from errors like lost or duplicated packets. The protocol also specifies how two computers initiate a TCP stream transfer and how they agree when it is complete.

It is also important to understand what the protocol does not include. Although the TCP specification describes how application programs use TCP in general terms, it does not dictate the details of the interface between an application program and TCP. That is, the protocol documentation only discusses the operations TCP supplies; it does not specify the exact procedures application programs invoke to access these operations.

The reason for leaving the application program interface unspecified is flexibility. In particular, because programmers usually implement TCP in the computer's operating system, they need to employ whatever interface the operating system supplies. Allowing the implement or flexibility makes it possible to have a single specification for TCP that can be used to build software for a variety of machines.

Because TCP assumes little about the underlying communication system, TCP can be used with a variety of packet delivery systems. In particular, because it does not require the underlying system to be reliable or fast, TCP can run over a variety of hardware mechanisms such as a dialup telephone line, a local area network, a high-speed fiber optic network, a satellite connection, or a noisy wireless connection in which many packets are lost. The large variety of delivery systems TCP can use is one of its strengths.

8.6 Passive and active opens

Unlike UDP, TCP is a connection-oriented protocol that requires both endpoints to agree to participate. That is, before TCP traffic can pass across an internet, application programs at both ends of the connection must agree that the connection is desired. To do so, the application program on one end performs a passive open function by contacting its operating system and indicating that it will accept an incoming connection. At that time the operating system assigns a TCP port number for its end of the connection. The application program at the end must then contact its operating system using an active open request to establish a connection. The two TCP software modules communicate to establish and verify a connection. Once a connection has been created, application programs can begin to pass data; the TCP software modules at each end exchange messages that guaranty the reliable delivery.

8.7 Establishing a TCP connection

To establish a connection, TCP uses a three-way handshake. In the simplest case, the handshake proceeds as Figure 8.3 shows.

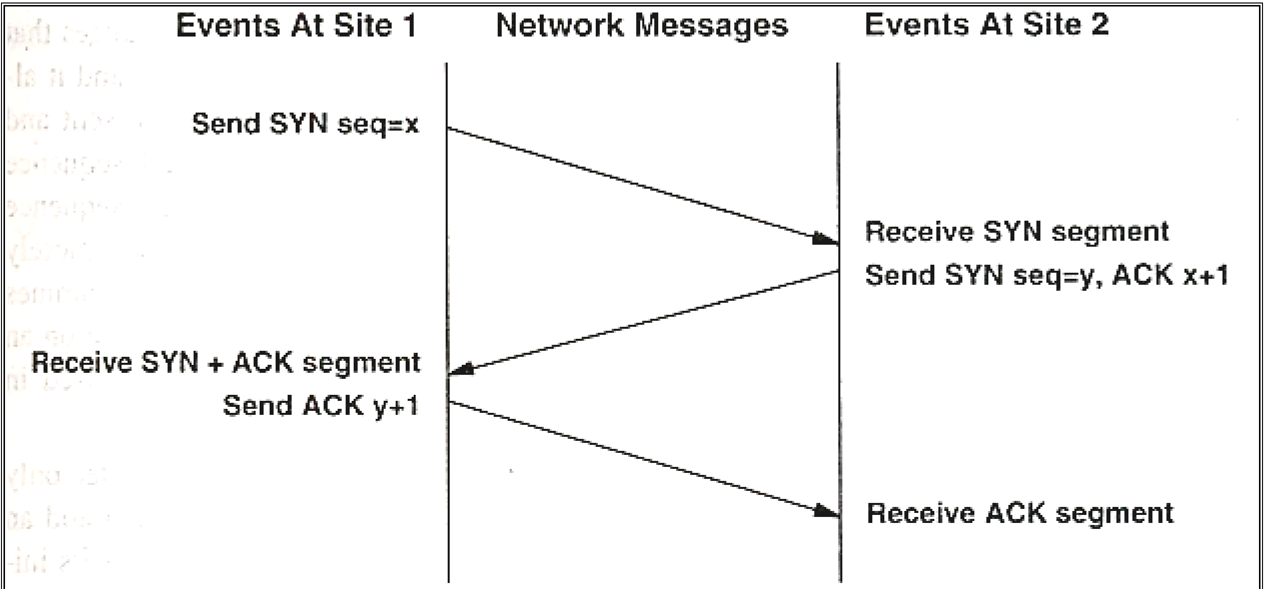


Figure 8.3 The sequence of messages in a three-way handshake. Time proceeds down the page; diagonal lines represent segments sent between sites. SYN segments carry initial sequence number information.

The first segment of a handshake can be identified because it has the SYN bit set in the code field. The second message has both the SYN and ACK bits set, indicating that it acknowledges the first SYN segment as well as continuing the handshake. The final handshake message is only an acknowledgement and is merely used to inform the destination that both sides agree that a connection has been established. Usually, the TCP software on one machine waits passively for the handshake, and the TCP software on another machine initiates it. However, the handshake is carefully designed to work even if both machines attempt to initiate a connection simultaneously. Thus, a connection can be established from either end or from both ends simultaneously. Once the connection has been established, data can flow in both directions equally well. There is no master or slave. The three-way handshake is both necessary and sufficient for correct synchronization between the two ends of the connection. To understand why, remember that TCP builds on an unreliable packet delivery service, so messages can be lost, delayed, duplicated, or delivered out of order. Thus, the protocol must use a timeout mechanism and retransmit lost requests. Trouble arises if retransmitted and original requests arrive while the connection is being established, or if retransmitted requests are delayed until after a connection has been established, used, and terminated. A three-way handshake (plus the rule that TCP ignores additional requests for connection after a connection has been established) solves these problems.

8.8 Closing a TCP connection

Two programs that use TCP to communicate can terminate the conversation gracefully using the close operation. Internally, TCP uses a modified three-way handshake to close connections. Recall that TCP connections are full duplex and that we view them as containing two independent stream transfers, one going in each direction. When an application program tells TCP that it has no more data to send, TCP will close the connection in one direction. To close its half of a connection, the sending TCP finishes transmitting the remaining data, waits for the receiver to acknowledge it, and then sends a segment with the FIN bit set. The receiving TCP acknowledges the FIN segment and informs the application program on its end that no more data is available (e.g., using the operating system's end-of-tile mechanism).

Once a connection has been closed in a given direction, TCP refuses to accept more data for that direction. Meanwhile, data can continue to flow in the opposite direction until the sender closes it. Of course, acknowledgements continue to flow back to the sender even after a connection has been closed. When both directions have been closed, the TCP software at each endpoint deletes its record of the connection.

The details of closing a connection are even more subtle than suggested above because TCP uses a modified three-way handshake to close a connection. Figure 8.4 illustrates the procedure.

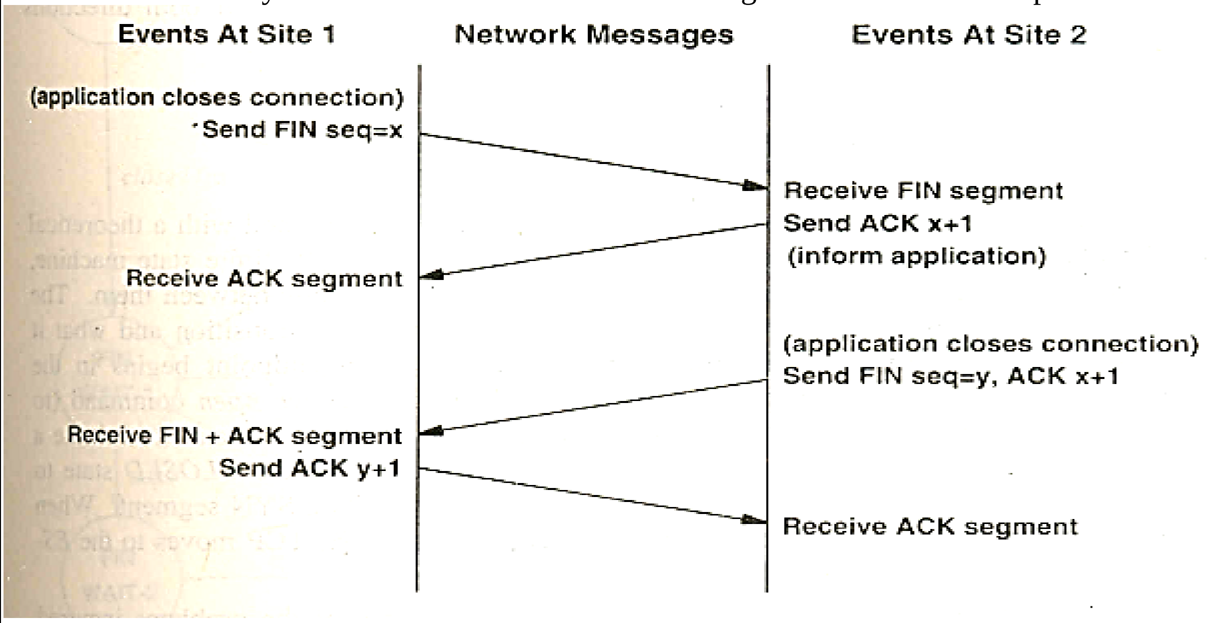


Figure8.4 The modified three-way handshake used to close connections. The site that receives the first FIN segment acknowledges it immediately, and then delays before sending the second FIN segment.

The difference between three-way handshakes used to establish and break connections occurs after a machine receives the initial FIN segment. Instead of generating a second FIN· segment immediately, TCP sends an acknowledgement and then informs the application of the request to shut down. Informing the application program of the request and obtaining a response may take considerable time (e.g., it may involve human interaction). The acknowledgement prevents retransmission of the initial FIN segment during the wait. Finally, when the application program instructs TCP to shut down the connection completely, TCP sends the second FIN segment and the original site replies with the third message, an ACK.

Assignment Questions.

Unit III

MCQ Questions carrying 1 mark each

1. The full form of UDP is
   a. Unified Datagram Protocol
   b. Unnamed Datagram Protocol
   c. User Datagram Protocol
   d. Universal Datagram Protocol
2. The protocol which is used to send the datagram from one application to another application is ____
   a. FTP b. HTTP c. BGP d. **UDP**
3. Multiple programs executing in a single machine is distinguished by ____
   a. Gateways b. **Ports** c. connectors d. software
4. ___________ that uses UDP accepts full responsibility for handling the problem of reliability, including message loss, duplication, delay, out-of-order delivery, and loss of connectivity
   a. An Application Program
   b. A connecting port
   c. A Protocol
   d. Network device
5. The concept of UDP datagram accepting datagrams from multiple application program is called ________
   a. Mixing b. Shuffling. **c. Multiplexing** d. Demultiplexing.
6. From the arrived datagram separating the UDP messages to appropriate software is called.
   a. Mixing b. Shuffling. c. Multiplexing **d. Demultiplexing.**
7. The properties of the reliable delivery services are
   a. Stream orientation b. virtual circuit connection

   c. buffered transfer d. All the above.
8. When two application programs transfer large volumes of data, the data is viewed as __
   a. Bits b. **a stream of bits** c. Individual Bits. d. Data
9. The concept where before transfer of data can start, both sending and receiving application programs interact with each other is called _______
   a. Telephonic connection
   b. Network connection
   c. Virtual connection
   d. Data connection
10. Connections provided by the stream service allow concurrent transfer in both directions is called _______
    a. Simplex b. Half Duplex **c. Full Duplex** d. All the above.
11. The stream service does not honor structured data streams is called _____
    a. **Unstructured stream**b. structured steam c. data stream d. data bytes.
12. The reply from the receiver for all the packets received from source is called ______
    a. Simple reply **b. Acknowledgement** c. Message d. Packet
13. The sliding window protocol is used to ___________

- **a. Send Multiple Packets before receiving the acknowledgement.**
- b. Wait for the acknowledgement before sending the next packet.
- c. Keep on sending the packets to the destination.
- d. Keep on receiving the acknowledgement.

14. TCP is __________ type of protocol
    a. **Connection oriented** b. Connectionless c. Both d. None of the above
15. The protocol that requires both endpoints to agree to participate is _______
    a. UDP **b. TCP** c. IP d. HTTP
16. The application program on one end performs a _________open function by contacting its operating system and indicating that it will accept an incoming connection. At that time the operating system assigns a TCP port number for its end of the connection.
    a. **Passive** b. Active c. Both d. None of the above.
17. To establish a connection, TCP uses ______
    - a. Single hand shake
    - b. Two way hand shake
    - **c. Three way hand shake**
    - d. No hand shake
18. To close a connection, TCP uses ______
    - a. Single hand shake
    - b. Two way hand shake
    - **c. Modified Three way hand shake**
    - d. No hand shake
19. The application programs in TCP can begin to pass the data only _______
    - a. After communicating
    - **b. Establishing a connection.**
    - c. Opening the Ports
    - d. All the Above
20. The application programs start exchanging data only after ___________
    - a. Active Port is open
    - b. Passive Port is open
    - c. Connection is established
    - **d. All the above**

Essay Questions

1. Write a note on UDP.
2. With the help of a diagram explain the UDP message format.
3. Explain the process of multiplexing and Demultiplexing the UDP datagrams.
4. Expand TCP.
5. What are passive opens?
6. What are active opens?
7. List and explain the properties of reliable delivery service.
8. Explain with the help of a diagram three packet transmission using sliding window.
9. What do you mean by sliding window? Explain.
10. How do you establish a TCP connection? Explain with the diagram.
11. How do you close a TCP connection? Explain with the diagram.

UNIT-IV

CHAPTER 9
SILLY WINDOW SYNDROME

9.1 Introduction

Silly Window Syndrome (SWS) is a problem that can arise in poor implementations of the transmission control protocol (TCP) when the receiver is only able to accept a few bytes at a time or when the sender transmits data in small segments repeatedly. The resulting number of small packets, or tinygrams, on the network can lead to a significant reduction in network performance and can indicate an overloaded server or a sending application that is limiting throughput.

In TCP, as data is transmitted the receiver replies with acknowledgements that, among other values, specify a window size - the number of bytes it is currently able to receive. The sender uses this to compute a "usable window" by subtracting the amount of unacknowledged data from the window size provided by the receiver. This process is known as the sliding window algorithm that TCP uses as its flow control protocol.

9.2 Cause for the Silly Window Syndrome.

On the sender's side, silly window syndrome can be caused by an application that only generates very small amounts of data to send at a time. Even if the receiver advertises a large window, the default behavior for TCP would be to send each individual small segment instead of buffering the data as it comes in and sending it in one larger segment.

9.3 Sender side Silly window Avoidance.

A heuristic method where the send TCP must allow the sending application to make "write" calls, and collect the data transferred in each call before transmitting it into a large segment. The sending TCP delays sending segments until it can accumulate reasonable amounts of data, which is known as *clumping.*

9.4 Receive-side silly window avoidance.

A heuristic method that a receiver uses to maintain an internal record of the available window, and delay advertising an increase in window size to the sender until it can advance a significant amount. This amount depends on the receiver's buffer size and maximum segment size. By using this method, it prevents small window advertisements where received applications extract data octets slowly

9.5 Delayed Acknowledgement.

There is a 32 bits (4 bytes) field in the *TCP header* that is known as the **sequence number**. There is also another 32 bits field for the **ACK number** and 9 bits for flags. One of the flags is **a bit for ACK that must be set to 1 for indicating that the segment transmitted contains an acknowledgment of one or more segments already received**. How many segments are acknowledged will depend on the *ACK* number value. If the *ACK* number is the same that the sequence number of the last segment +1, then is acknowledging only one segment. If not, then it is acknowledging several.

<u>How data is transmitted and acknowledged in TCP</u>:
Lets imagine that a *host A* is sending a segment to a *host B* which contains 1000 bytes of data.
Suppose that the initial sequence number of the segment is 10.
Once the host in destination receives the segment, it will take the sequence number (10) and will add 1000 (the number of bytes received) to that sequence number.

10+1000=1010
Then when *host B* wants to acknowledge the reception of that segment to A, it will send a segment back containing an *ACK* number of 1011 and the *ACK* flag bit turned on.

Please note that the ACK number is 1011 instead of 1010.
This is basically saying:

Hey! I have received until byte 1010, please next time you communicate with me start with a segment whose sequence number is 1011. Apart from that, an *ACK* can be *pure* if contains no data and is just for acknowledging one or more segments, or it can be *not pure* when apart from the *ACK* it contains data that is sent back from destination to source (this technique is know as piggybacking.

Sending a pure *ACK* is an opportunity lost; is the lost opportunity of sending an *ACK* with data, instead of just a simple bit of information. In that sense it has an opportunity cost. **Why?** Because of the protocol overhead.

Delayed ACK was invented to reduce the number of *ACKs* required to acknowledge the segments, so protocol overhead is reduced.

Basically *Delayed ACK* is the destination retaining the *ACK* segment a period of time (200 ms in Microsoft Windows by default), expecting one of two things to happen:

1) That a new segment will arrive that will also require acknowledge, so we can ACK two segments in one pure ACK instead of in two separate ACKs; avoiding that way to transmit over the network 41 extra bytes.

2) That, perhaps, before the ACK timer expires, the destination will need to send some data back to the source in which we can include the ACK bit and the right ACK sequence number to acknowledge the segment (piggybacking).

So delayed ACK is basically a bet.

9.6 Nagel's Algorithm.

Named for its creator, John Nagle, the Nagle algorithm is used to automatically concatenate a number of small buffer messages; this process (called *nagling*) increases the efficiency of a network application system by decreasing the number of packets that must be sent. Nagle's algorithm, defined in 1984 as *Ford Aerospace and Communications Corporation Congestion Control in IP/TCP Internetworks* (IETF RFC 896) was originally designed to relieve congestion for a private TCP/IP network operated by Ford, but has since been broadly deployed.

Nagle's document specified a means of dealing with what he called *the small packet problem*, created when an application generates data one byte at a time, causing the network to be overloaded with packets (a situation often referred to as *send-side silly window syndrome*). A single character - one byte of data - originating from a keyboard could result in the transmission of a 41 byte packet consisting of one byte of useful information and 40 bytes of header data. This situation translates into 4000% overhead, which was considered to be acceptable for a lightly loaded network such as the Advanced Research Projects Agency Network (ARPANet - the precursor of the Internet operating at that time), but not so for a heavily loaded network such as Ford's, where it could necessitate retransmissions, cause lost packets, and hamper propagation speed through excessive congestion in switching nodes and gateways. Throughput could be hampered to the extent that connections were aborted. Nagle's algorithm - usually implemented through the insertion of two lines of code into a TCP program - instructs the sender to buffer (store) data if any unacknowledged data is outstanding. Any data sent subsequently is held until the outstanding data is acknowledged (ACKed) or until there is a full packet's worth of data to send.

Although Nagle's algorithm addressed problems that were being experienced within Ford's network, the same problems were beginning to be experienced by ARPANet. Nagling has been broadly implemented across networks, including the Internet, and is generally performed by default - although it is sometimes considered to be undesirable in highly interactive environments, such as some client/server situations. In such cases, nagling may be turned off through use of the TCP_NODELAY sockets option.

9.7 TCP timers

TCP uses several timers to ensure that excessive delays are not encountered during communications. Several of these timers are elegant, handling problems that are not immediately obvious at first analysis. Each of the timers used by TCP is examined in the following sections, which reveal its role in ensuring data is properly sent from one connection to another.

TCP implementation uses four timers –

1. **Retransmission Timer –** To retransmit lost segments, TCP uses retransmission timeout (RTO). When TCP sends a segment the timer starts and stops when the acknowledgment is received. If the timer expires timeout occurs and the segment is retransmitted. RTO (retransmission timeout is for 1 RTT) to calculate retransmission timeout we first need to calculate the RTT(round trip time).

   RTT three types –

   - **Measured RTT(RTTm) –** The measured round-trip time for a segment is the time required for the segment to reach the destination and be acknowledged, although the acknowledgment may include other segments.

   - **Smoothed RTT(RTTs) –** It is the weighted average of RTTm. RTTm is likely to change and its fluctuation is so high that a single measurement cannot be used to calculate RTO.
   - Initially -> No value
   - After the first measurement -> RTTs=RTTm
   - After each measurement -> RTTs= (1-t)*RTTs + t*RTTm
   - Note: t=1/8 (default if not given)
   - **Deviated RTT(RTTd) –** Most implementation do not use RTTs alone so RTT deviated is also calculated to find out RTO.
   - Initially -> No value
   - After the first measurement -> RTTd=RTTm/2
   - After each measurement -> RTTd= (1-k)*RTTd + k*(RTTm-RTTs)
   - Note: k=1/4 (default if not given)

   Retransmission Timeout : RTO calculation – The value of RTO is based on the smoothed round-trip time and its deviation. Most implementations use the following formula to calculate the RTO:

   Initial value -> Original (given in question)

   After any measurement -> RTO=RTTs + 4*RTTd

   #NOTE: At every retransmission the value of RTO doubles. (RTO(new) = RTO(before retransmission) *2) this is explained in Karn’s Algorithm

2. **Persistent Timer –** To deal with a zero-window-size deadlock situation, TCP uses a persistence timer. When the sending TCP receives an acknowledgment with a window size

of zero, it starts a persistence timer. When the persistence timer goes off, the sending TCP sends a special segment called a probe. This segment contains only 1 byte of new data. It has a sequence number, but its sequence number is never acknowledged; it is even ignored in calculating the sequence number for the rest of the data. The probe causes the receiving TCP to resend the acknowledgment which was lost.

3. **Keep Alive Timer** – A keep alive timer is used to prevent a long idle connection between two TCPs. If a client opens a TCP connection to a server transfers some data and becomes silent the client will crash. In this case, the connection remains open forever. So a keep alive timer is used. Each time the server hears from a client, it resets this timer. The time-out is usually 2 hours. If the server does not hear from the client after 2 hours, it sends a probe segment. If there is no response after 10 probes, each of which is 75 s apart, it assumes that the client is down and terminates the connection.

4. **Time Wait Timer –** This timer is used during tcp connection termination. The timer starts after sending the last Ack for 2nd FIN and closing the connection.
   After a TCP connection is closed, it is possible for datagrams that are still making their way through the network to attempt to access the closed port. The quiet timer is intended to prevent the just-closed port from reopening again quickly and receiving these last datagrams.
   The **quiet timer** is usually set to twice the maximum segment lifetime (the same value as the Time-To-Live field in an IP header), ensuring that all segments still heading for the port have been discarded.

Chapter 10
IP MULTICASTING

10.1 Introduction

This chapter explores another feature of IP: multipoint delivery of datagrams. We begin with a brief review of the underlying hardware support. Later sections describe IP addressing for multipoint delivery and protocols that routers use to propagate the necessary routing information.

10.2 Hardware Broadcast

Many hardware technologies contain mechanisms to send packets to multiple destinations simultaneously (or nearly simultaneously). Broadcast delivery means that the network delivers one copy of a packet to each destination. On bus technologies like Ethernet, broadcast delivery can be accomplished with a single packet transmission. On networks composed of switches with point-to-point connections, software must implement broadcasting by forwarding copies of the packet across individual connections until all switches have received a copy.

With most hardware technologies, a computer specifies broadcast delivery by sending a packet to a special, reserved destination address called the broadcast address. For example, Ethernet hardware addresses consist of 48-bit identifiers, with the all Is address used to denote broadcast. Hardware on each machine recognizes the machine's hardware address as well as the broadcast address, and accepts incoming packets that have either address as their destination. The chief disadvantage of broadcasting arises from its demand on resources in addition to using network bandwidth, each broadcast consumes computational resources on all machines. For example.it would be possible to design an alternative internet protocol suite that used broadcast to deliver datagrams on a local network and relied on IP software to discard datagrams not intended for the local machine. However, such a scheme would be extremely inefficient because all computers on the network would receive and process every datagram, even though a machine would discard most of the datagrams that arrived. Thus, the designers of used unicast addressing and address binding mechanisms like ARP to eliminate broadcast delivery.

10.3 Hardware origins of Multicast

Some hardware technologies support a second, less common form of multi-point delivery called multicasting. Unlike broadcasting, multicasting allows each system to choose whether it wants to participate in a given multicast. Typically, a hardware technology reserves a large set of addresses for use with multicast. When a group of machines want to communicate, they choose one particular multicast address to use for communication. After configuring their network interface hardware to recognize the selected multicast address, all machines in the group will receive a copy of any packet sent to that multicast address.

At a conceptual level, multicast addressing can be viewed as a generalization of all other address forms. For example, we can think of a conventional unicast address as a form of multicast addressing in which there is exactly one computer in the multicast group. Similarly, we can think of directed broadcast addressing as a form of multicasting in which all computers on a particular network are members of the multicast group. Other multicast addresses can correspond to arbitrary sets of machines. Despite Its apparent generality, multicasting cannot replace conventional forms of addressing because there is a fundamental difference in the underlying mechanisms that implement forwarding and delivery. Unicast and broadcast addresses identify a computer or a set of computers attached to one physical segment, so forwarding depends on the network topology. A multicast address identifies an arbitrary set of listeners, so the forwarding mechanism must propagate the packet to all segments. For example, considertwo LAN segments connected by an adaptive bridge that has learned host addresses. If a host on segment 1 sends a unicast frame to another host on segment 1, the bridge will not forward the frame to segment 2. If a host uses a multicast address, however, the bridge will forward the frame. Thus, we can conclude:

Although it may help us to think of multicast addressing as a generalization that subsumes unicast and broadcast addresses, the underlying forwarding and delivery mechanisms can make multicast less efficient.

10.4 IP- Multicast

IP multicasting is the internet abstraction of hardware multicasting. It follows the paradigm of allowing transmission to a subset of host computers, but generalizes the concept to allow the subset to spread across arbitrary physical networks throughout an internet. In IP terminology, a given subset is known as a multicast group. IP multicasting has the following general characteristics:

- Group Address. Each multicast group is a unique class D address. A few IP multicast addresses are permanently assigned by the Internet authority, and correspond to groups that always exist even if they have no current members. Other addresses are temporary, and are available for private use.

- Number Of Groups. IP provides addresses for up to 228 simultaneous multicast groups. Thus, the number of groups is limited by practical constraints on routing table size rather than addressing.

- Dynamic Group Membership. A host can join or leave an IP multicast group at any time. Furthermore, a host may be a member of an arbitrary number of multicast groups.

- Use Of Hardware. If the underlying network hardware supports multicast, IP uses hardware multicast to send IP multicast. If the hardware does not support multicast, IP uses broadcast or unicast to deliver IP multicast.

- Inter-network Forwarding. Because members of an IP multicast group can attach to multiple physical networks, special multicast routers are required to forward IP multicast; the capability is usually added to conventional routers.

- Delivery Semantics. IP multicast uses the same best-effort delivery semantics as other IP datagram delivery, meaning that multicast datagrams can be lost, delayed, duplicated, or delivered out of order.

- Membership And Transmission. An arbitrary host may send datagrams to any multicast group; group membership is only used to determine whether the host receives datagrams sent to the group.

10.5 IP- Multicast Addresses

Two types of multicast addresses are permanently assigned and for temporary use. Permanent addresses
are called well-known; they are used for major services on the global Internet as well as for infrastructure maintenance (e.g., multicast routing protocols). Other multicast addresses

correspond to transient multicast groups that are created when needed and discarded when the count of group members reaches zero. Like hardware multicasting, IP multicasting uses the datagram's destination address to specify that a particular datagram must be delivered via multicast. IP reserves class D addresses for multicast; they have the form shown in Figure 9.1.

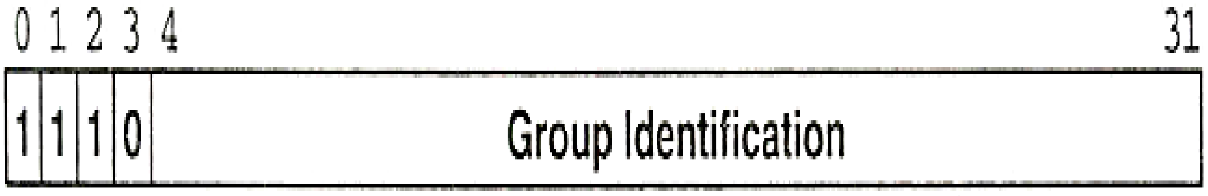


Figure 9.1 The format of class D IP addresses used for multicasting. Bits 4 through 31 identify a particular multicast group.

The first 4 bits contain 1110 and identify the address as a multicast. The remaining 28 bits specify a particular multicast group. There is no further structure in the group bits. In particular, the group field is not partitioned into bits that identify the origin or owner of the group, nor does it contain administrative information such as whether all members of the group are on one physical network.

When expressed in dotted decimal notation, multicast addresses range from

224.0.0.0 through 239.255.255.255

Many parts of the address space have been assigned special meaning. For example, the lowest address, 224.0.0.0, is reserved; it cannot be assigned to any group. Address 224.0.0.1 is permanently assigned to the all systems group, and address 224.0.0.2 is permanently assigned to the all routers group. The all systems group includes all hosts and routers on a network that are participating in IP multicast, whereas the all routers group includes only the routers that are participating. Both of these groups are used for control protocols. and must be on the same local network as the sender; there are no IP multicast addresses that refer to all systems in the Internet or all routers in the Internet. Furthermore, the remaining addresses up through 224.0.0.255 are restricted to a single network (i.e., a router is prohibited from forwarding a datagram sent to any address in the range, and a sender is supposed to set the TIL to 1). Typically, multicast routing protocols use such addresses. Figure 16.2 shows a few examples of permanently assigned addresses.

10.6 Internet Group Management Protocol

To participate in IP multicast on a local network, a host must have software that allows it to send and receive multicast datagrams. To participate in a multicast that spans multiple networks, the host must inform local multicast routers. The local routers contact other multicast routers, passing on the membership information and establishing routes. We will see later that the concept is similar to conventional route propagation among internet routers. Before a multicast router can propagate multicast membership information, it must determine that one or more hosts on the local network have decided to join a multicast group. To do so, multicast routers and hosts that implement multicast must use the Internet Group Management Protocol (IGMP) to communicate group membership information. Because the current version is 3, the protocol described here is officially known as IGMPv3.

IGMP is analogous to ICMP. Like ICMP, it uses IP datagrams to carry messages. Also like ICMP, it provides a service used by IP. Therefore,

Although IGMP uses IP datagrams to carry messages, we think of it as an integral part of IP, not a separate protocol.

Furthermore, IGMP is a standard for ; it is required on all machines that receive IP multicast (i.e., all hosts and routers that participate at level 2). Conceptually, IGMP has two phases. Phase 1: When a host joins a new multicast group, it sends an IGMP message to the group's multicast address declaring itsmembership. Local multicast routers receive the message, and establish necessary routing by propagating the group membership information to other multicast routers throughout the internet. Phase 2: Because membership is dynamic, local multicast routers periodically poll hosts on the local network to determine whether any hosts still remain members of each group. If any host responds for a given group, the routerkeeps the group active. If no host reports membership in a group after several polls, the multicast router assumes that none of the hosts on the network remain in the group, and stops advertising group membership to other multicast routers. To further complicate group membership, IGMP permits an application on a host to install a source address filter that specifies whether the host should include or exclude multicast traffic from a given source address. Thus, it is possible to join a multicast group, but to exclude datagrams sent to the group by a given source. The presence of filters is important because IGMP allows a host to pass the set of filter specifications to the local router along with group membership information. In the case where two applications disagree (i .e., one application excludes a given source and another includes the source), software on the host must rationalize the two specifications and then handle the decision about which applications receive a given datagram locally.

CHAPTER 11
DOMAIN NAME SYSTEM

11.1 Introduction to Domain Name System (DNS)

The protocols described earlier use 32-bit integers called Internet Protocol addresses (IP addresses) to identify machines. Although such addresses provide a convenient, compact representation for specifying the source and destination in datagrams sent across an internet, users prefer to assign machines pronounceable, easily remembered names.

This chapter considers a scheme for assigning meaningful high-level names to a large set of machines, and discusses a mechanism that maps between high-level machine names and IP addresses. It considers both the translation from high-level names to IP addresses and the translation from IP addresses to high level machine names. The naming scheme is interesting for two reasons. First, it has been used to assign machine names throughout the Internet. Second, because it uses a geographically distributed set of servers to map names to addresses, the implementation of the name mapping mechanism provides a large scale example of the client-server paradigm

11.2 Names for machines

The earliest computer systems forced users to understand numeric addresses for objects like system tables and peripheral devices. Timesharing systems advanced computing by allowing users to invent meaningful symbolic names for both physical objects (e.g., peripheral devices) and abstract objects (e.g., files). A similar pattern has emerged in computer networking. Early systems supported point-to-point connections between computers and used low-level hardware addresses to specify machines. Internetworking introduced universal addressing as well as protocol software to map universal addresses into low-level hardware addresses. Because most computing environments contain multiple machines, users need meaningful, symbolic names to identify them.

Early machine names reflected the small environment in which they were chosen. It was quite common for a site with a handful of machines to choose names based on the machines' purposes. For example, machines often had names like accounting, development, and production. Users find such names preferable to cumbersome hardware addresses.

Although the distinction between address and name is intuitively appealing, it is artificial. Any name is merely an identifier that consists of a sequence of characters chosen from a finite alphabet. Names are only useful if the system can efficiently map them to the object they denote. Thus, we think of an IP address as a low-level name, and we say that users prefer high-level names for machines. The form of high-level names is important because it determines how names are translated to low-level names or bound to objects, as well as how name assignments are authorized. With only a few machines, choosing names is easy - each administrator can choose an arbitrary name and verify that the name is not in use. For example, when its main departmental computer was connected to the Internet in 1980, the Computer Science Department at Purdue University chose the name purdue to identify the connected machine. The list of potential conflicts contained only a few dozen names. By mid 1986, the official list of hosts on the Internet contained 3100 officially registered names and 6500 official aliases. Although the list was growing rapidly in the 1980s, most sites had additional machines (e.g., personal computers) that were not registered. In the current Internet, with hundreds of millions of machines, choosing symbolic names is much more difficult.

11.3 Hierarchical names

How can a naming system accommodate a large, rapidly expanding set of names without requiring a central site to administer it? The answer lies in decentralizing the naming mechanism by delegating authority for parts of the names pace and distributing responsibility for the mapping between names and addresses. The Internet uses such a scheme. Before examining the details, we will consider the motivation and intuition behind it.

The partitioning of a namespace must be defined in a way that supports efficient name mapping and guarantees autonomous control of name assignment. Optimizing only for efficient mapping can lead to solutions that retain a flat names pace and reduce traffic by dividing the names among multiple mapping machines. Optimizing only for administrative ease can lead to solutions that make delegation of authority easy but name mapping expensive or complex.

To understand how the namespace should be divided, consider the internal structure of large organizations. At the top, a chief executive has overall responsibility. Because the chief executive cannot oversee everything, the organization may be partitioned into divisions, with an executive in charge of each division. The chief executive grants each division autonomy within specified limits. More to the point, the executive in charge of a particular division can hire or fire employees, assign offices, and delegate authority, without obtaining direct permission from the chief executive.

Besides making it easy to delegate authority, the hierarchy of a large organization introduces autonomous operation. For example, when an office worker needs information like the telephone number of a new employee, he or she begins by asking local clerical workers (who may contact clerical workers in other divisions). The point is that although authority always passes down the corporate hierarchy, information can flow across the hierarchy from one office to another.

11.4 Delegation of Authority for names

A hierarchical naming scheme works like the management of a large organization. The namespace is partitioned at the top level, and authority for names in subdivisions is passed to designated agents. For example, one might choose to partition the namespace based on site name and to delegate to each site responsibility for maintaining names within its partition. The topmost level of the hierarchy divides the namespace and delegates authority for each division; it need not be bothered by changes within a division.

The syntax of hierarchically assigned names often reflects the hierarchical delegation of authority used to assign them. As an example, consider a namespace with names of the form:

l***ocal.site***

where site is the site name authorized by the central authority, local is the part of name controlled by the site, and the period (".") is a delimiter used to separate them. When the topmost authority approves adding a new site, X, it adds X to the list of valid sites and delegates to site X authority for all names that end in ".X".

11.5 Internet Domain names

The Domain Name System (DNS) is the system that provides name to address mapping for the Internet. DNS has two, conceptually independent aspects. The first is abstract: it specifies the name syntax and rules for delegating authority over names. The second is concrete: it specifies the implementation of a distributed computing system that efficiently maps names to addresses. This section considers the name syntax, and later sections examine the implementation.

The domain name system uses a hierarchical naming scheme known as domain names. As 'in our earlier examples, a domain name consists of a sequence of sub names separated by a delimiter character, the period. In our examples we said that individual sections of the name might represent sites or groups, but the domain name system simply calls each section a label. Thus, the domain name

cs.purdue.edu

contains three labels: cs, purdue, and edu. Any suffix of a label in a domain name is also called a domain. In the above example, the lowest-level domain is cs. purdue. edu, (the domain name for the Computer Science Department at Purdue University), these cond level domain is purdue. edu (the domain name for Purdue University), and the top-level domain is edu (the domain name for educational institutions). As the example shows, domain names are written with the local label first and the top domain last. As we will see; writing them in this order makes it possible to compress messages that contain multiple domain names.

11.6 Domain name Resolution

Although the conceptual tree makes understanding the relationship between servers easy, it hides several subtle details. Looking at the name resolution algorithm will help explain them. Conceptually, domain name resolution proceeds top-down, starting with the root name server and proceeding to servers located at the leaves of the tree. There are two ways to use the domain name system: by contacting name servers one at a time or asking the name server system to perform the complete translation. In either case, the client software forms a domain name query that contains the name to be resolved, a declaration of the class of the name, the type of answer desired, and a code that specifies whether the name server should translate the name completely. It sends the query to a name server for resolution.

When a domain name server receives a query, it checks to see if the name lies in the sub domain for which it is an authority. If so, it translates the name to an address according to its database, and appends an answer to the query before sending it back to the client. If the name server cannot resolve the name completely, it checks to see what type of interaction the client specified. If the client requested complete translation (recursive resolution, in domain name terminology), the server contacts a domain name server that can resolve the name and returns the answer to the client. If the client requested non-recursive resolution (iterative resolution), the name server cannot supply an answer. It generates a reply that specifies the name server the client should contact next to resolve the name. How does a client find a name server at which to begin the search? How does a name server find other name servers that can answer questions when it cannot? The answers are simple. A client must know how to contact at least one name server. To ensure that a domain name server can reach others, the domain system requires thateach server know the address of at least one root server. In addition, a server may know the address of a server for the domain immediately above it (called the parent).

Domain name servers use a well-known protocol port for all communication, so clients know how to communicate with a server once they know the IP address of the machine in which the server executes. There is no standard way for hosts to locate a machine in the local environment on which a name server runs; that is left to whoever designs the client software. In some systems, the address of the machine that supplies domain name service is bound into application programs at compile time, while in others, the address is stored in a file on secondary storage. Many systems obtain the address of a domain server automatically as part of the bootstrap process.

11.7 Caching : The key to efficiency

The cost of lookup for non local names can be extremely high if resolvers send each query to the root server. Even if queries could go directly to the server that has authority for the name, name lookup can present a heavy load to an internet. Thus, to improve the overall performance of a name server system, it is necessary to lower the cost of lookup for nonlocal names.

Internet name servers use caching to optimize search costs. Each server maintains a cache of recently used names as well as a record of where the mapping information for that name was obtained. When a client asks the server to resolve a name, the server first checks to see if it has authority for the name according to the standard procedure. If not, the server checks its cache to see if the name has been resolved recently. Servers report cached information to clients, but mark it as a non authoritative binding, and give the domain name of the server, S, from which they obtained the binding. The local server also sends along additional information that tells the client the binding between S and an IP address. Therefore, clients receive answers quickly, but the information may be out-of-date. If efficiency is important, the client will choose to accept the non authoritative answer and proceed. If accuracy is important, the client will choose to contact the authority and verify that the binding between name and address is still valid.

Caching works well in the domain name system because name to address bindings change infrequently. However, they do change. If servers cached information the first time it was requested and never updated it, entries in the cache could become stale (i.e., incorrect). To keep the cache correct, servers time each entry and dispose of entries that exceed a reasonable time. When a server is asked for the information after it has removed the entry from its cache, the server must go back to the authoritative source and obtain the binding again. More important, servers do not apply a single fixed timeout to all entries, but allow the authority for an entry to configure its timeout. Whenever an authority responds to a request, it includes a Time To Live (TTL) value in the response that specifies how long it guarantees the binding to remain valid. Thus, authorities can reduce network overhead by specifying long timeouts for entries that they expect to remain unchanged, while improving correctness by specifying short timeouts for entries that they expect to change frequently.

Caching is important in hosts as well as in local domain name servers. Most resolver software caches DNS entries in the host. Thus, if a user looks up the same name repeatedly, subsequent lookups can be resolved from the local cache without using the network.

11.8 DNS message format

Looking at the details of messages exchanged between clients and domain name servers will help clarify how the system operates from the view of a typical application program. We assume that a user invokes an application program and supplies the name of a machine with which the application must communicate. Before it can use protocols like TCP or UDP to communicate with the specified machine, the application program must find the machine's IP address. It passes the domain name to a local resolver and requests an IP address. The local resolver checks its cache and returns the answer if one is present. If the local resolver does not have an answer, it formats a message and sends it to the server (i.e., it becomes a client). Although our example only involves one name, the message format allows a client to ask multiple questions in a single message. Each question consists of a domain name for which the client seeks an IP address, a specification of the query class (i.e., internet), and the type of object desired (e.g., address). The server responds by returning a similar message that contains answers to the questions for which the server has bindings. If the server cannot answer all questions, the response will contain information about other name servers that the client can contact to obtain the answers.Responses also contain information about the servers that are authorities for the replies and the IP addresses of those servers. Figure 10.1 shows the message format.

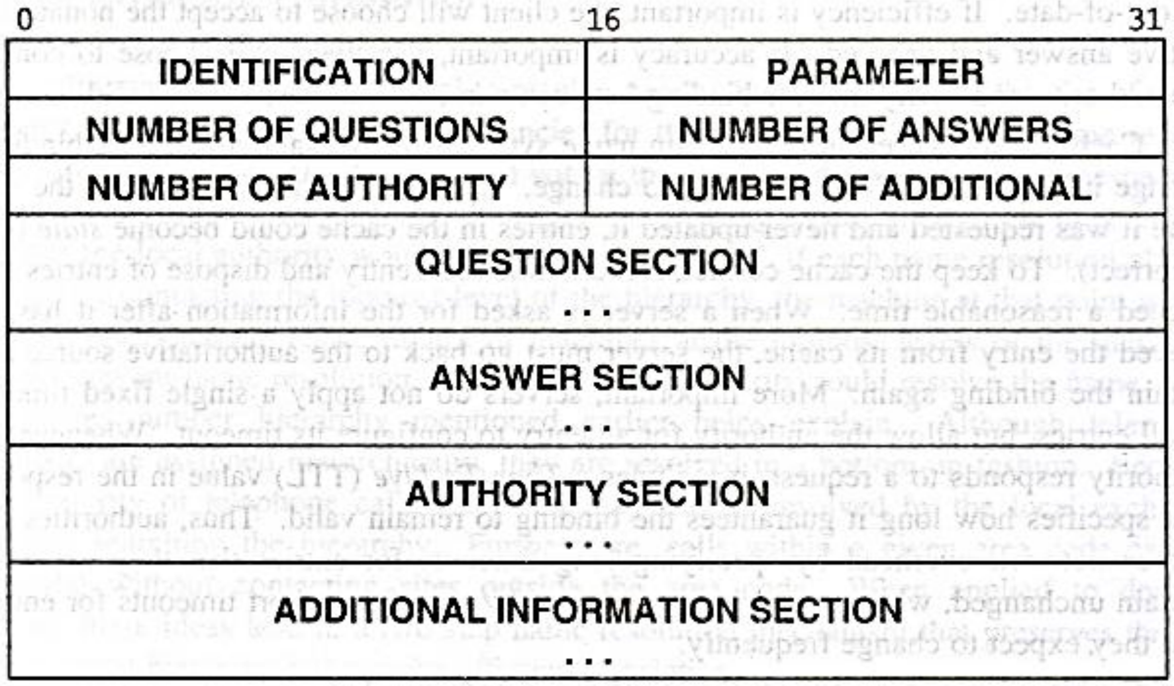


Figure 11.1 Domain name server message format. The QUESTION, ANSWER, AUTHORITY, and ADDITIONAL INFORMATION sections are variable length.

As the figure shows, each message begins with a fixed header. The header contains a unique IDENTIFICATION field that the client uses to match responses to queries, and a PARAMETER field that specifies the operation requested and a response code.

11.9 Telnet Protocol

The protocol suite includes a simple textual remote terminal protocol called TELNET that allows a user to log into a computer across an internet. TELNET establishes a TCP connection, and then passes keystrokes from the user's keyboard directly to the remote computer as if they had been typed on a keyboard attached to the remote machine. TELNET also carries textual output from the remote machine back to the user's screen. The service is called transparent because it gives the appearance that the user's keyboard and display attach directly to the remote machine.

TELNET offers three basic services. First, it defines a network virtual terminal that provides a standard interface to remote systems. Client programs do not have to understand the details of all possible remote systems; they are built to use the standard interface. Second, TELNET includes a mechanism that allows the client and server to negotiate options, and it provides a set of standard options (e.g., one of the options controls whether data passed across the connection uses the standard 7-bit ASCII character set or an 8-bit character set). Finally, TELNET treats both ends of the connection symmetrically. In particular, TELNET does not force client input to come from a keyboard, nor does it force the client to display output on a screen. Thus, TELNET allows an arbitrary program to become a client. Furthermore, either end can negotiate options.

Figure 11.2 illustrates how application programs are used to implement a TELNET client and a TELNET server.

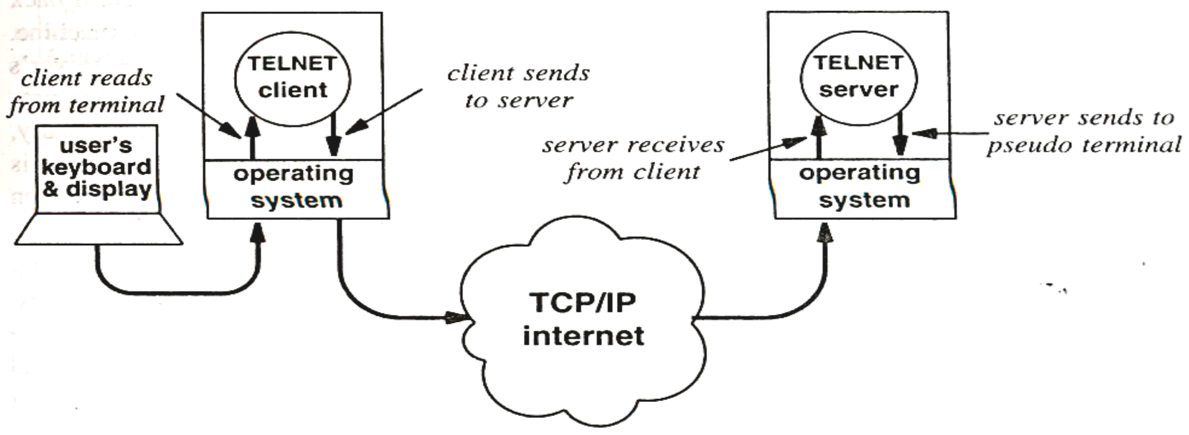

client reads from terminal
TELNET client
client sends to server
user's keyboard & display
operating system
TELNET server
server sends to pseudo terminal
server receives from client
operating system
TCP/IP internet

Figure 11.2:The path of data in a TELNET remote terminal session as it travels from the user's keyboard to the remote operating system.

As the figure shows, when a user invokes TELNET, an application program on the user's machine becomes the client. The client establishes a TCP connection to the server over which they will communicate. Once the connection has been established, the client accepts keystrokes from the user's keyboard and sends them to the server, while it concurrently accepts characters that the server sends back and displays them on the user's screen. The client also checks for an escape character that a user can type to control the client. The server accepts a TCP connection from the client, and then relays data between the TCP connection and the local operating system. In practice, the server is more complex than the figure shows because it must handle multiple, concurrent connections. Usually, a master server process waits for new connections and creates a new slave copy to handle each connection. Thus, the 'TELNET server', shown in Figure 10.2, represents the slave that handles one particular connection. The figure does not show the master server that listens for new requests, nor does it show the slaves handling other connections.

We use the term pseudo terminal to describe the operating system entry point that allows a running program like the TELNET server to transfer characters to the operating system as if they came from a keyboard. It is impossible to build a TELNET server unless the operating system supplies such a facility. If the system supports a pseudo terminal abstraction, the TELNET server can be implemented with application programs.

Each slave server connects a TCP stream from one client to a particular pseudo terminal. Arranging for the TELNET server to be an application-level program has advantages and disadvantages. The most obvious advantage is that it makes modification and control of the server easier than if the code were embedded in the operating system. The obvious disadvantage is inefficiency. Each keystroke travels from the user's key-board through the operating system to the client program, from the client program back through the operating system and across the underlying internet to the server machine. After reaching the destination machine, the data must travel up through the server's operating system to the server application program, and from the server application program back into the server's operating system at a pseudo terminal entry point. Finally, the remote operating system delivers the character to the application program the user is running. Meanwhile, output (including remote character echo if that option has been.

selected) travels back from the server to the client over the same path. Readers who understand operating systems will appreciate that for the implementation shown in Figure 10.2, every keystroke requires computers to switch process context several times. In most systems, an additional context switch is required because the operating system on the server's machine must pass characters from the pseudo terminal back to another application program (e.g., a command interpreter). Although context switching is expensive, the scheme is practical because users do not type at high speed.

11.10 Other Remote Access Technologies

Although TELNET and SSH are part of the protocols, many other remote access protocols have been devised. A few examples illustrate some of the diversity. Rlogin. One of the earliest alternatives to TELNET, devised as part of the BSD Unix operating system, was known as rlogin. Rlogin pioneered an idea that has been used with subsequent services, including SSH: trusted hosts. In essence, the mechanism allows system administrators to choose a set of machines over which login names and file access protections are shared and to establish equivalences among user logins. Users can control access to their personal accounts by authorizing remote login based on remote host and remote user names. Thus, it is possible for a user to have login name X on one machine and Yon another, and still be able to remotely login from one of the machines to the other without typing a password each time. Virtual Network Computing (VNC). Unlike TELNET, rlogin, or SSH, VNC provides a remote desktop capability instead of a textual interface. That is, VNC allows a user on one computer to see an exact copy of the desktop on another computer and to use the keyboard and mouse to interact with the remote computer - the user can see the windows, icons, and other graphics. More important, VNC software runs across multiple platforms. As a result, the client can run on a computer that uses Linux, even if the server runs on a computer that uses Windows.

Remote Desktop Protocol (RDP). Microsoft Corporation has defined a remote desktop protocol for use with their operating system. Like other remote desktop systems, RDP allows a remote computer to see an exact copy of the desktop and to run applications. Also like other remote desktop technologies, RDP can be used across platforms (e.g., a Linux client exists).

11.11 Assignment Questions

Unit IV

Multiple Choice Questions.

1. ____ is a problem that can arise in poor implementations of the transmission control protocol (TCP) when the receiver is only able to accept a few bytes at a time or when the sender transmits data in small segments repeatedly.
   a. Delayed Transmission b. **Silly Window Syndrome**
   c. Silly Window Avoidance d. Fast Transmission
2. The tinygrams are ______
   a. **Small Packets** b. IP Packets c. UDP Packets d. Control Packets
3. The concept of multipoint delivery of datagram is called ______
   a. Multiple Delivery b. All Casting c. **Multicasting** d. Unicasting
4. ________ delivery means that the network delivers one copy of a packet to each destination.
   a. **Broadcast** b. Unicast c. Multicast d. Nocast.
5. The concept of systems choosing many systems of different networks to participate in a network is called _________
   a. Unicast b. **Multicast** c. Broadcast d. Nocast
6. The address reserved for Multicasting is ______
   a. Class A b. Class B c. Class C d**. Class D**
7. The technique of having an acknowledgement that can be pure if contains no data and is just for acknowledging one or more segments, or it can be *not pure* when apart from the *ACK* it contains data that is sent back from destination to source is _____
   a. Tallback b. Small Back c. Tinyback **d. piggyback**
8. The purpose to reduce the number of *ACKs* required to acknowledge the segments is _____
   a. Positive Acknowledgement b. Negative Acknowledgement
   c. Delayed Acknowledgement **d.** Simple Acknowledgement
9. _________ is the destination retaining the *ACK* segment a period of time 200 ms in Microsoft Windows by default
   a. Positive Acknowledgement b. Negative Acknowledgement
   c. Delayed Acknowledgement **d.** Simple Acknowledgement
10. The algorithm that concatenates the small packets is called _________
   a. Routing Algorithm b. ARP algorithm
   c. Hash Algorithm **d. Nagel's Algorithm**
11. ______are used to ensure that excessive delays are not encountered during communications
   a. Acknowledgements b. TCP Connectors
   c. TCP Timers d. IP Timers
12. TCP implementation uses __________ timers
   a. 1 b. 2 c. 3 **d. 4**
13. ________ timer is used during TCP connection termination.
   a. **Time Wait** b. Keep Alive c. Persistant d. Time out
14. Many hardware technologies contain mechanisms to send packets to multiple destinations Simultaneously is called
   a. Software IP Multicast b. Hardware Broadcast
   c**. Hardware Multicast** d. All off the above.
15. __________ is the internet abstraction of hardware multicasting.
   a. Hardware Multicasting b. **IP- Multicasting**
   c. Hybrid Multicasting d. Anycasting
16. To communicate with the group members the IP multicast uses _______ protocol.

a. HTTP b. **IGMP** c. TFTP d. Class D

17. The domain name system uses __________ naming scheme.
    a. normal b. sequential c. **Hierarchical** d. None of the above
18. The domain name to IP address after the domain resolution will be stored in ___
    a. **cache** b. RAM c. Secondary memory d. None
19. The simple textual remote terminal protocol is called__________
    a. R-Login b. **TelNet** c. FTP d. SNMP
20. The BSD Unix version of TelNet protocol is ______
    a. TelNet b. BSD c**. Rlogin** d. IMAP
21. RDP stands for
    a. Remote Desktop Protocol b. Remote Distance Protocol
    c. Remote Destination Protocol d. None of the above

Answer the questions Each carries 2 marks.

1. Write a note on DNS.
2. What are the benefits of DNS over IP address?
3. What do you mean by mapping?
4. What is the use of Telnet protocol?
5. What do you mean by SSH?
6. What is the use of Rlogin?
7. Mention the use of VNC.
8. Expand RDP and mention which company has defined the same?
9. Write a note on multicasting (Define multicasting.)
10. Mention the address range of IP Multicast address.

Essay Questions

1. How to accommodate the large network by assessing the names?
2. Explain DNS with the help of an example.
3. Explain how the resolution of Domain name takes place?
4. With the help of a diagram explain the DNS message format.
5. Write a note on Telnet protocol.
6. With the help of a diagram explain how application programs implement a Telnet client to a Telnet server.
7. Write a note on IP Multicasting.
8. List out the various characteristics of IP Multicast.
9. Explain the Class D addressing scheme for IP Multicasting.
10. What do you mean by IGMP? Explain.

Unit 5
CHAPTER 12
VARIOUS INTERNET PROTOCOLS

12.1 Sharing by File Transfer

The alternative to integrated, transparent on-line access is a file transfer mechanism. Accessing remote data with a transfer mechanism is a two-step process: the user first obtains a local copy of a file and then operates on the copy. Most transfer mechanisms operate outside the local file system (i.e., they are not integrated). A user must invoke a special-purpose client program to transfer files. The client contacts a server on the remote machine and requests a copy of the file. Once the transfer is complete, the user terminates the client and uses an application program on the local system to read or modify the local copy. One advantage of whole-file copying lies in the efficiency of operations - once a local copy of a remote file has been made, application programs can operate on the local copy at high speed.

As with on-line sharing, whole-file transfer between heterogeneous machines can be difficult. The client and server must agree on authorization, notions of file owner ship and access protections, and data formats. Data formats are especially important because translation may change the file slightly (e.g., may modify line termination). As a consequence, if a file is transferred from machine A to B and then back to A, the result can differ from the original.

12.2 The major File Transfer Protocol

File transfer is among the most frequently used applications, and still accounts for a nontrivial amount of Internet traffic. Standard file transfer protocols existed for the ARPANET before became operational. These early versions of file transfer software evolved into a current standard known as the File Transfer Protocol (FTP).

12.3 FTP Features

Given a reliable end-to-end transport protocol like TCP, file transfer might seem trivial. However, as the previous sections pointed out, the details of authorization, naming, and representation among heterogeneous machines make the protocol complex. In addition, FIP offers many facilities beyond the transfer function itself.

- Interactive Access. Although FTP is designed to be used by programs, most implementations also provide an interactive interface that allows humans to interact with remote servers.

- Format Specification. FTP allows the client to specify the type and representation of stored data. For example, the user can specify whether a file contains text or binary data and whether text files use the ASCII or EBCDIC character sets.

- Authentication Control. FTP requires clients to authorize themselves by sending a login name and password to the server before requesting file transfers. The server refuses access to clients that cannot supply a valid login and password.

12.4 FTP Process Model

Like other servers, most FTP server implementations allow concurrent access by multiple clients. Clients use TCP to connect to a server. A single master server process awaits connections and creates a slave process to handle each connection. Unlike most servers, however, the FTP slave process does not perform all necessary computation. Instead, the slave accepts and handles a control connection from the client, but uses an additior.al process and an additional TCP connection to handle each data transfer operation. The control connection carries commands that tell the server which file to transfer. A new TCP connection and a new process on both the client and server side is created for each data transfer operation (i.e., each file transfer). While the exact details of the process architecture depend on the operating systems used, Figure 12.1 illustrates the concept:

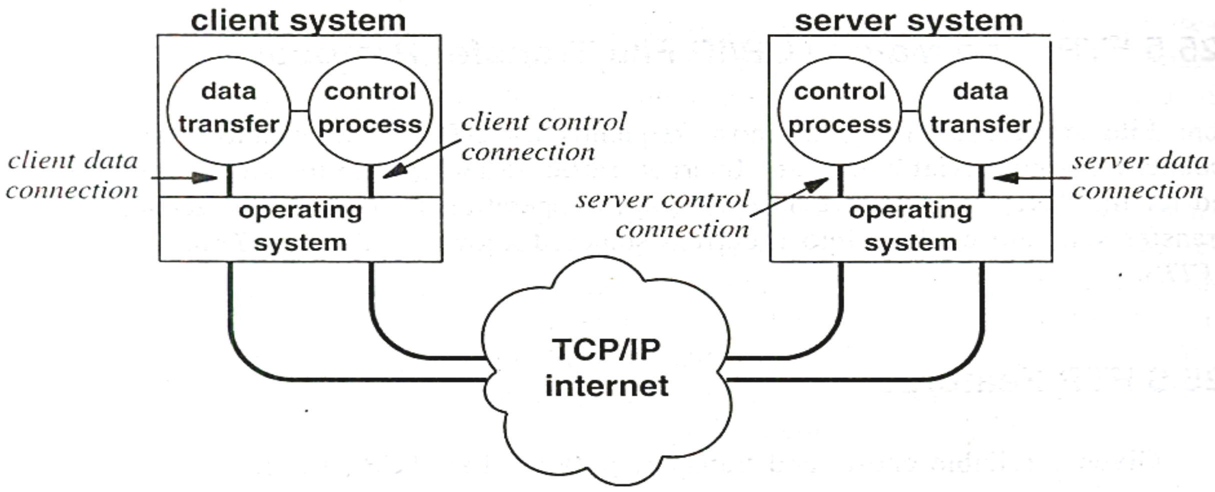


Figure 12.1: An FTP client and server with a TCP control connection between them and a separate TCP connection between their associated data transfer processes.

As the figure shows, the client control process connects to the server control process using one TCP connection, while the associated data transfer processes use their own TCP connection. In general, the control processes and the control connection remain alive as long as the user keeps the FTP session active, and a data transfer connection persists for one file transfer. The idea can be summarized

Data transfer connections and the data transfer processes that use them are created dynamically when needed, but the control connection persists throughout a session. Once the control connection disappears, the session is terminated and the software at both ends terminates all data transfer processes

12.5 TFTP

Although FTP is the most general file transfer protocol in the suite, it is also the most complex and difficult to program. Many applications do not need the full functionality FTP offers, nor can they afford the complexity. For example, FTP requires clients and servers to manage multiple concurrent TCP connections, something that may be difficult or impossible on embedded computers that do not have sophisticated operating systems.

The suite contains a second file transfer protocol that provides inexpensive, unsophisticated service. Known as the Trivial File Transfer Protocol, or (TFTP), it is intended for applications that do not need complex interactions between the client and server. TFTP restricts operations to simple file transfers and does not provide authentication. Because it is more restrictive, TFTP software is much smaller than FTP.

Small size is important in many applications. For example, manufacturers of embedded systems can encode TFTP in read-only memory (ROM) and use it to obtain an initial memory image when the machine is powered on. The program in ROM is called the system bootstrap. The advantage of using TFTP is that it allows bootstrapping code to use the same underlying protocols that the operating system uses once it begins execution. Thus, it is possible for a computer to bootstrap from a server on another physical network.

Unlike FTP, TFTP does not need a reliable stream transport service. It runs on top of UDP or any other unreliable packet delivery system, using timeout and retransmission to ensure that data arrives. The sending side transmits a file in fixed size (512 byte) blocks and awaits an acknowledgement for each block before sending the next.

The receiver acknowledges each block upon receipt. The rules for TFTP are simple. The first packet sent requests a file transfer and establishes the interaction between client and server - the packet specifies a file name and whether the file will be read (transferred to the client) or written (transferred to the server). Blocks of the file are numbered consecutively starting at 1. Each data packet contains a header that specifies the number of the block it carries, and each acknowledgement contains the number of the block being acknowledged. A block of less than 512 bytes signals the end of file. It is possible to send an error message either in the place of data or an acknowledgement; errors terminate the transfer.

Figure 12.2 shows the format of the five TFTP packet types. The initial packet must use operation codes 1 or 2 to specify either a read request or a write request, the FILENAME field to specify the name of a file, and the MODE field to specify whether the client will read the file, write the file, or both.

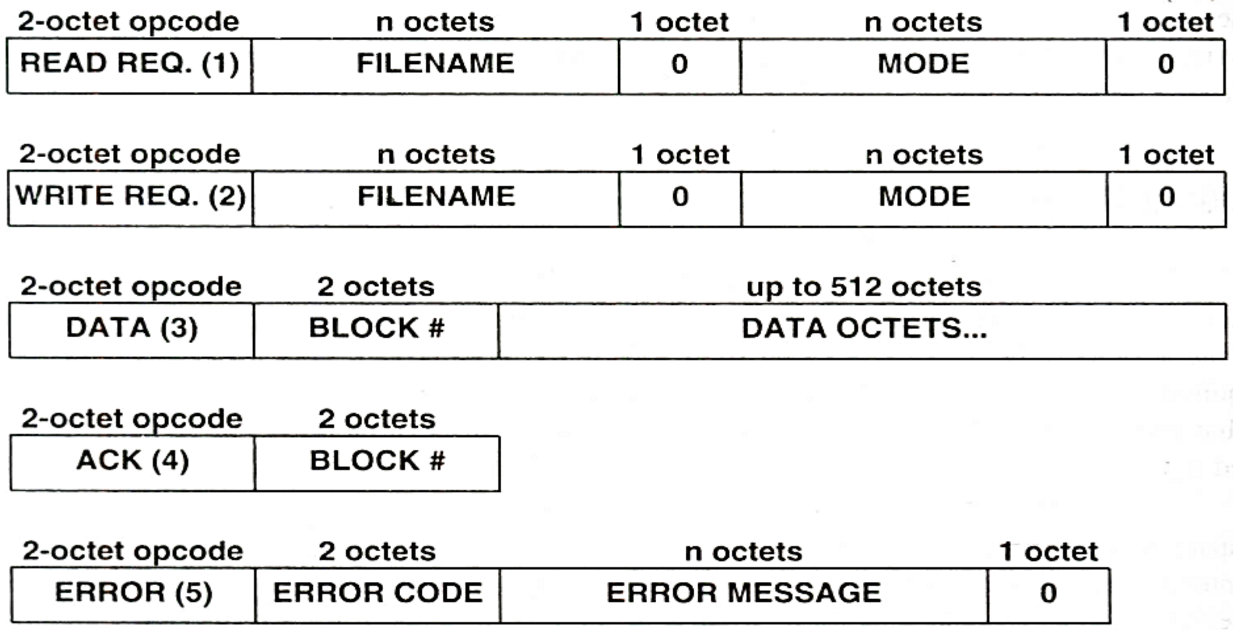


Figure 12.2: The five TFTP message types. Fields are not shown to scale because some are variable length; an initial 2-octet operation code identifies the message format.

Once a read or write request has been made, the server uses the IP address and UDP protocol port number of the client to identify subsequent operations. Thus, neither data messages (the messages that carry blocks from the file) nor ack messages (the messages that acknowledge data blocks) need to specify the file name. The final message type illustrated in Figure 12.2 is used to report errors. Lost messages can be re transmitted after a timeout, but most other errors simply cause termination of the interaction.

TFTP retransmission is unusual because it is symmetric - each side implements a timeout and retransmission. If the side sending data times out, it retransmits the last data block. If the side responsible for acknowledgements times out, it retransmits the last acknowledgement. Having both sides participate in retransmission helps ensure that transfer will not fail after a single packet loss.

While symmetric retransmission guarantees robustness, it can lead to excessive retransmissions. The problem, known as the Sorcerer's Apprentice Bug, arises when an acknowledgement for data packet k is delayed, but not lost. The sender retransmits the data packet, which the receiver acknowledges. Both acknowledgements eventually arrive, and each triggers a transmission of data packet k+ 1. The receiver will acknowledge both copies of data packet k+l, and the two acknowledgements will each cause the sender to transmit data packet k+2. The Sorcerer's Apprentice Bug can also start if the underlying internet duplicates packets. Once started, the cycle continues indefinitely with each data packet being transmitted exactly twice. A fix for the Sorcerer's Apprentice Bug appears in the latest version of TFTP.

12.6 NFS

Initially developed by Sun Microsystems Incorporated, the Network File System (NFS) has been published as an IETF standard for transparent and integrated shared file access. Figure 12.3 illustrates how NFS is embedded in an operating system.

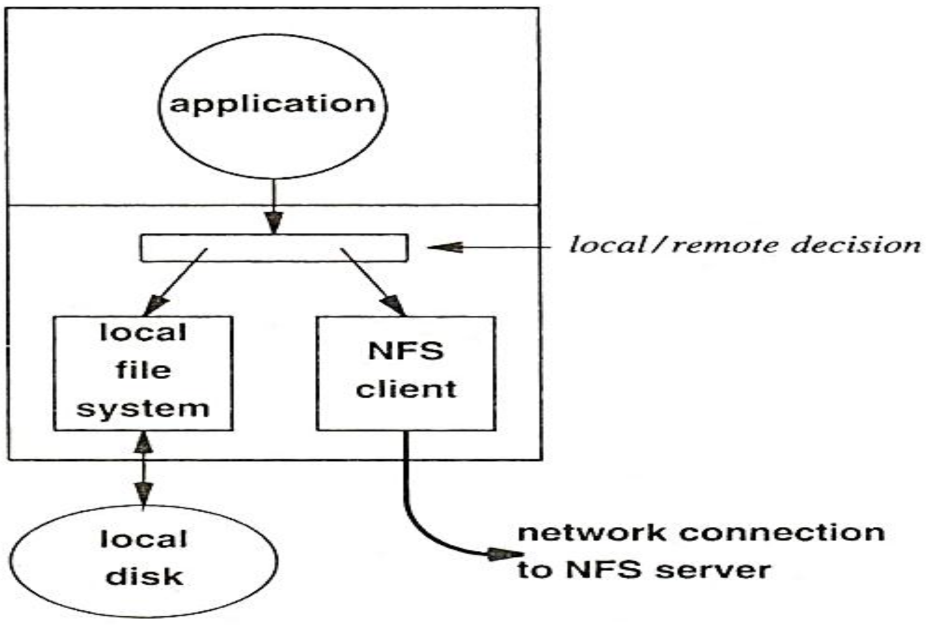


Figure:12.3 Implementation of NFS in an operating system. When an application requests a file operation, the operating system must pass the request to the local file system or to the NFS client software.

When an application program executes, it calls the operating system to open a file, to read data from a file, or to write data into a file. The file access mechanism accepts the request, and automatically passes it to either the local file system software or to the NFS client, depending on whether the file is on the local disk or on a remote machine. When it receives a request, the client software uses the NFS protocol to contact the appropriate server on a remote machine and perform the requested operation. When the remote server replies, the client software returns the results to the application program.

12.7 Simple mail Transfer Protocol (SMTP)

In addition to message formats, the TCP/lP protocol suite specifies a standard for the exchange of mail between machines. That is, the standard specifies the exact format of messages a client on one machine uses to transfer mail to a server on another. The standard transfer protocol is known as the Simple Mail Transfer Protocol(SMTP). SMTP is simpler than the earlier Mail

Transfer Protocol, (MTP). The SMTP protocol focuses specifically on how the underlying mail delivery system passes messages across an internet from one machine to another. It does not specify how the mail system accepts mail from a user or how the user interface presents the user with incoming mail. Also, SMTP does not specify how mail is stored or how frequently the mail system attempts to send messages.

SMTP is surprisingly straightforward. Communication between a client and server consists of readable ASCII text. As with other' application protocols, programs read the abbreviated commands and 3-digit numbers at the beginning of lines; the remaining text is intended to help humans debug mail software. Although SMTP rigidly defines the command format, humans can easily read a transcript of interactions between a client and server because each command appears on a separate line. Initially, the client establishes a reliable stream connection to the server and waits for the server to send a 220 READY FOR MAIL message. (If the server is overloaded, it may delay sending the 220 message temporarily.) Upon receipt of the 220 message, the client sends a HELO command (if the client supports extensions from RFC 2821, the client sends an EHLO command). The end of a line marks the end of a command. The server responds by identifying itself. Once communication has been established, the sender can transmit one or more mail messages and then terminate the connection. The receiver must acknowledge each command. It can also abort the entire connection or abort the current message transfer.

Mail transactions begin with a MAIL command that gives the sender identification as well as a FROM: field that contains the address to which errors should be reported. A recipient prepares its data structures to receive a new mail message, and replies to a MAIL command by sending the response 250. Response 250 means that all is well.

The full response consists of the text 250 OK. After a successful MAIL command, the sender issues a series of RCPT commands that identify recipients of the mail message. The receiver must acknowledge each RCPT command by sending 250 OK or by sending the error message 550 No such user here.

After all RCPT commands have been acknowledged, the sender issues a DATA command. In essence, a DATA command informs the receiver that the sender is ready to transfer a complete mail message. The receiver responds with message 354 Start mail input, and specifies the sequence of characters used to terminate the mail message. The termination sequence consists of 5 characters: carriage return, line feed, period, carriage return, and line feed.

Once it has finished sending all the mail messages, a client issues a QUIT command. The other side responds with command 221, which means it agrees to terminate both sides then close the TCP connection gracefully. SMTP is much more complex than we have outlined here. For example, if a user has moved, the server may know the user's new mailbox address. SMTP allows the server to inform the client about the new address so the client can use it in the future.
When informing the client about a new address, the server may choose to forward the mail that triggered the message, or it may request that the client take the responsibility for forwarding. In addition, SMTP includes Transport Layer Security (TLS) extensions that allow an SMTP session to be encrypted.

12.8 Mail Retrieval and mail box Manipulation Protocols

The SMTP transfer scheme described above implies that a server must remain ready to accept e-mail at all times. The scenario works well if the server runs on a computer that has a permanent Internet connection, but it does not work well for a computer that has intermittent connectivity (e.g., a laptop computer that is disconnected when being moved). It makes no sense for such a computer to run an e-mail server because the server will only be available while the user's computer is connected – all other attempts to contact the server will fail, and e-mail sent to the user will remain undelivered. The question arises, "How can a user without a permanent connection receive e-mail?" The answer to the question lies in a two-stage delivery process. In the first stage, each user is assigned a mailbox on a computer that is always on and has a permanent Internet connection. The computer runs a conventional SMTP server, which always remains ready to accept e-mail. In the second stage, the user connects to the Internet and then runs a protocol that retrieves messages from the permanent mailbox. The protocol transfers the messages to the user'~ computer where they can be read.

Two protocols exist that allow a remote user to access mail in a permanent mail box. Although they have similar functionality, the protocols take opposite approaches: one allows the user to download a copy of messages, and the other allows a user to view and manipulate messages on the server. The next two sections describe the two protocols.

12.9 Post Office Protocol

The most popular protocol used to transfer e-mail messages from a permanent mailbox to a local computer is known as version 3 of the Post Office Protocol (POP3); a secure version of the protocol is known as POP3S. The user invokes a POP3 client, which creates a TCP connection to a POP3 server on the mailbox computer. The user first sends a login and a password to authenticate the session. Once authentication has been accepted, the client sends commands to retrieve a copy of one or more messages and to delete the message from the permanent mailbox. The messages are stored and transferred as text files in 2822 standard format.

Note that the computer with the permanent mailbox must run two servers – an SMTP server accepts mail sent to a user and adds each incoming message to the user's permanent mailbox, and a POP3 server allows a user to extract messages from the mail box and delete them. To ensure correct operation, the two servers must coordinate use of the mailbox so that if a message arrives via SMTP while a user is extracting messages via POP3, the mailbox is left in a valid state.

12.10 Internet Message Access Protocol

Version 4 of the Internet Message Access Protocol (lMAP4) is an alternative to POP3 that allows users to view and manipulate messages; a secure version of IMAP has been defined, and is known as IMAPS. Like POP3, IMAP4 defines an abstraction known as a mailbox; mailboxes are located on the same computer as a server. Also like POP3, a user runs an IMAP4 client that contacts the server to manipulate messages.

Unlike POP3, however, IMAP4 allows a user to access mail messages from multiple locations (e.g., from work and from home), and ensures that all copies are synchronized and consistent. IMAP4 also provides extended functionality for message retrieval and processing. A user can obtain information about a message or examine header fields without retrieving the entire message. In addition a user can search for a specified string and retrieve portions of a message. Partial retrieval is especially useful for slow-speed dialup connections because it means a user does not need to download useless information.

12.11 The MIME Extensions for Non – ASCII Data

The Multipurpose Internet Mail Extensions (MIME) were defined to allow transmission of non-ASCII data through e-mail. MIME does not change or replace protocols such as SMTP, POP3, and IMAP4. Instead, MIME allows arbitrary data to be encoded in ASCII and then transmitted in a standard e-mail message. To accommodate arbitrary data types and representations, each

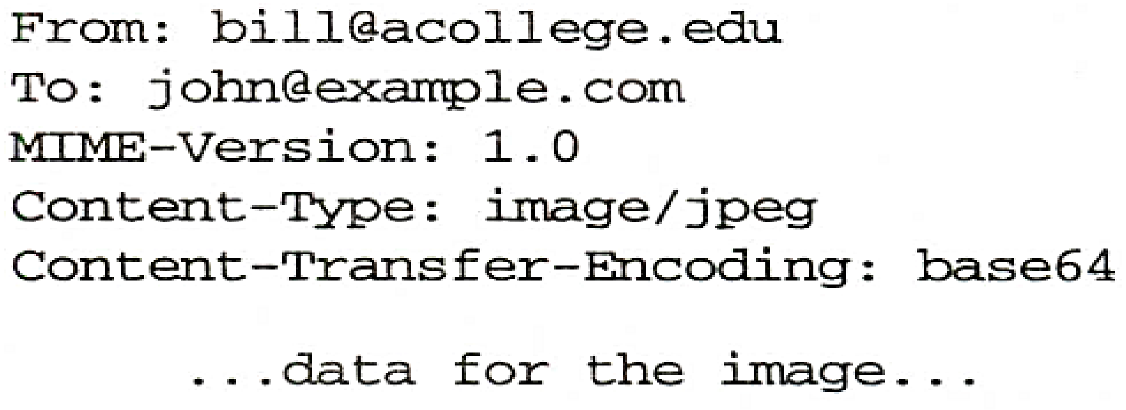
MIME message includes information that tells the recipient the type of the data and the encoding used. MIME information resides in the 2822 mail header - the MIME header lines specify the MIME version, the data type, and the encoding that was used to convert the data to ASCII. For example, Figure 11.4 illustrates a MIME message that contains a photograph in standard JPEG representation. The IPEG image has been converted to a 7-bit ASCII representation using the base64 encoding.

In the figure, the header line MIME- Version: declares that the message was composed using version 1.0 of the MIME protocol. The Content-Type: declaration specifies that the data is a JPEG image, and the Content-Transfer-Encoding: header declares that base64 encoding was used to convert the image to ASCII.

To view the image in a figure, a receiver's mail system must first convert from base64 encoding back to binary, and then run an application that displays a IPEG image on the user's screen. Base64 was chosen to provide sixty-four ASCII characters that have the same representation across various versions of ISO English character sets. Thus, a receiver can be guaranteed that the image extracted from the encoded data is exactly the same as the original image.

The MIME standard specifies that a Content-Type declaration must contain two identifiers, a content type and a subtype, separated by a slash. In the example, image is the content type, and jpeg is the subtype.

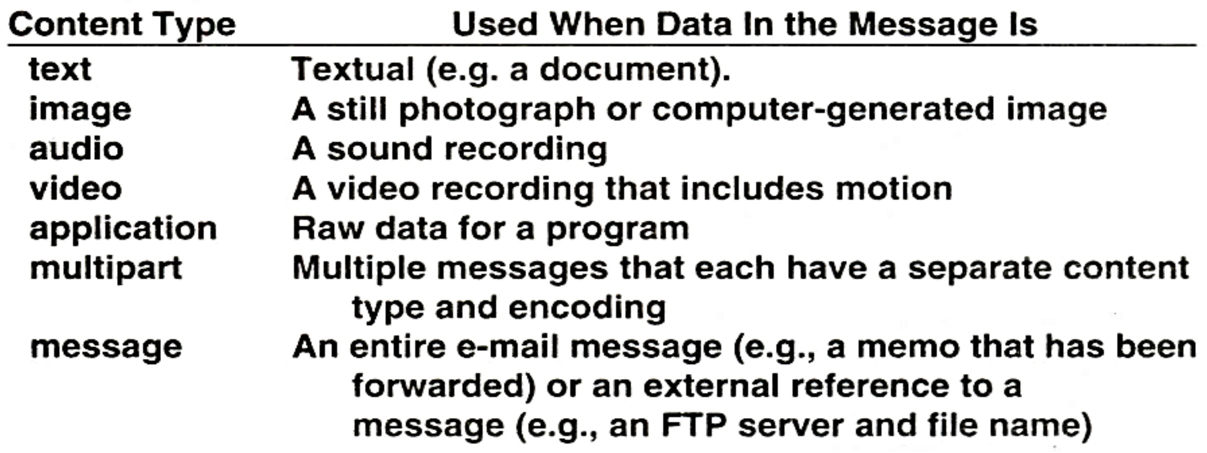
The standard defines seven basic content types, the valid subtypes for each, and transfer encodings. For example, although an image must be of subtype jpeg or gif, text cannot use either subtype. In addition to the standard types and subtypes, MIME permits a sender and receiver to define private content types Figure 12.5 lists the seven basic content types.



```
From: bill@acollege.edu
To: john@example.com
MIME-Version: 1.0
Content-Type: image/jpeg
Content-Transfer-Encoding: base64

        ...data for the image...
```

Figure 12.4:An example MIME message. Lines in the header identify the type of the data as well as the encoding used.



Content Type	Used When Data In the Message Is
text	Textual (e.g. a document).
image	A still photograph or computer-generated image
audio	A sound recording
video	A video recording that includes motion
application	Raw data for a program
multipart	Multiple messages that each have a separate content type and encoding
message	An entire e-mail message (e.g., a memo that has been forwarded) or an external reference to a message (e.g., an FTP server and file name)

Figure 12.5:The seven basic types that can appear in a MIME Content-Type declaration and their meanings.

CHAPTER 13
IPV6 – FEATURES DATAGRAM FORMAT

13.1 Introduction

Evolution of technology is intertwined with evolution of the global Internet for several reasons. First, the Internet is the largest installed internet, so many problems related to scale arise in the Internet before they surface in other internets. Second, funding for research and engineering comes from companies and government agencies that use the operational Internet, so they tend to fund projects that impact the Internet. Third, because most researchers use the global Internet daily, they have immediate motivation to solve problems that will improve service and extend functionality.

With millions of users at tens of thousands of sites around the world depending on the global Internet as part of their daily work environment, it might appear that the Internet is a completely stable production facility. We have passed the early stage of development in which every user was also an expert, and entered a stage in which few users understand the technology. Despite appearances, however, neither the Internet nor the protocol suite is static. Groups discover new ways to use the technology.

Researchers solve new networking problems, and engineers improve the underlying mechanisms. In short, the technology continues to evolve. The purpose of this chapter is to consider the ongoing evolutionary process and examine one of the most significant engineering efforts: a proposed revision of IP. When the proposal is adopted by vendors, it will have a major impact on TCP/TP and the global Internet.

13.2 Why change

The basic technology has worked well for over two decades. Why shouldit change? In a broad sense, the motivation revising the protocols arises from changes in underlying technologies and uses.

- New Computer And Communication Technologies. Computer and network hardware continues to evolve. As new technologies emerge, they are incorporated into the Internet.

- New Applications. As programmers invent new ways to use , additional protocol support is needed. For example, the emphasis on IP telephony has led to investigations of protocols for real-time data delivery.

- Increases In Size And Load. The global Internet has experienced many years of sustained exponential growth, doubling in size every nine months or faster. In 1999, on the average, a new host appeared on the Internet every two seconds.Traffic has also increased rapidly as animated graphics and video proliferate.

13.3Features

The proposed IPv6 protocol retains many of the features that contributed to the success of IPv4. In fact, the designers have characterized IPv6 as being basically the same as IPv4 with a few modifications. For example, IPv6 still supports connectionless delivery (i.e., each datagram is routed independently), allows the sender to choose the size of a datagram, and requires the sender to specify the maximum number of hops a datagram can make before being terminated. As we will see, IPv6 also retains most of the concepts provided by IPv4 options, including facilities for fragmentation and source routing.

Despite many conceptual similarities, IPv6 changes most of the protocol details. For example, IPv6 uses larger addresses, and adds a few new features. More important, IPv6 completely revises the datagram format by replacing IPv4's variable-length options field by a series of fixed-format headers. We will examine details after considering major changes and the underlying motivation for each. The changes introduced by IPv6 can be grouped into seven categories:

- Larger Addresses. The new address size is the most noticeable change. IPv6 quadruples the size of an IPv4 address from 32 bits to 128 bits. The IPv6 address space is so large that it cannot be exhausted in the foreseeable future.

- Extended Address Hierarchy. IPv6 uses the larger address space to create additional levels of addressing hierarchy. In particular, Pv6 can define a hierarchy of ISPs as well as a hierarchical structure within a given site.

- Flexible Header Format. IPv6 uses an entirely new and incompatible datagram format. Unlike the IPv4 fixed-format header, IPv6 defines a set of optional headers.

- Improved Options. Like IPv4, IPv6 allows a datagram to include optional control information. IPv6 includes new options that provide additional facilities not available in IPv4.

- Provision For Protocol Extension. Perhaps the most significant change in IPv6 is a move away from a protocol that fully specifies all details to a protocol that can permit additional features. The extension capability has the potential to allow the IETF to adapt the protocol to changes in underlying network hardware or to new applications.

- Support For Auto configuration And Renumbering. IPv6 provides facilities that allow computers on an isolated network to assign themselves addresses and begin communicating without depending on a router or manual configuration. The protocol also includes a facility that permits a manager to renumber networks dynamically.

- Support For Resource Allocation. IPv6 has two facilities that permit reallocation of network resources: a flow abstraction and a differentiated service specification. The latter will use the same approach as IPv4's differentiated services.

13.4 General Form of an IPv6 Datagram

IPv6 completely changes the datagram format. As Figure 12.1 shows, an IPv6 datagram has a fixed-size base header followed by zero or more extension headers, followed by data.

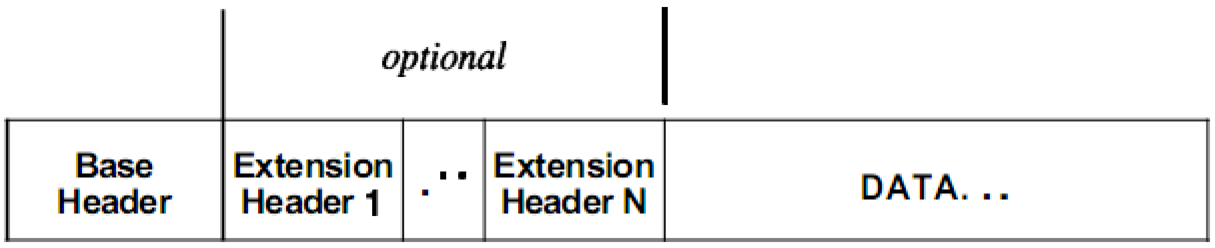


Figure 13.1:The general form of an IPv6 datagram with multiple headers. Only the base header is required; extension headers are optional.

13.5 IPv6 Base Header Format

Interestingly, although it must accommodate larger addresses, an IPv6 base header contains less information than an IPv4 datagram header. Options and some of the fixed fields that appear in an IPV4 datagram header have been moved to extension headers in IPv6. In general, the changes in the datagram header reflect changes in the protocol:

- Alignment has been changed from 32-bit to 64-bit multiples.
- The header length field has been eliminated, and the datagram length field has been replaced by a PAYLOAD LENGTH field.
- The size of source and destination address fields has been increased to 16 octets each.
- Fragmentation information has been moved out of fixed fields in the base header into an extension header.
- The TIME-TO-LIVE field has been replaced by a HOP LIMIT field.
- The SERVICE TYPE is renamed to be a TRAFFIC CLASS field, and extended with a FLOW LABEL field.
- The PROTOCOL field has been replaced by a field that specifies the type of the next header.

Figure 13.2 shows the contents and format of an IPv6 base header. Several fields in an IPv6 base header correspond directly to fields in an IPv4 header. As in IPv4 the initial 4-bit VERS field specifies the version of the protocol; VERS always contains 6 in an IPv6 datagram. As in IPv4, the SOURCE ADDRESS and DESTINATION ADDRESS fields specify the addresses of the sender and intended recipient. In IPv6, however, each address requires 16 octets. The HOP LIMIT field corresponds to the IPv4 TIME-TO-LIVE field. Unlike IPv4, which interprets a time-to-live as a combination of hop count and maximum time, IPV6 interprets the value as giving a strict bound on the maximum number of hops a datagram can make before being discarded.

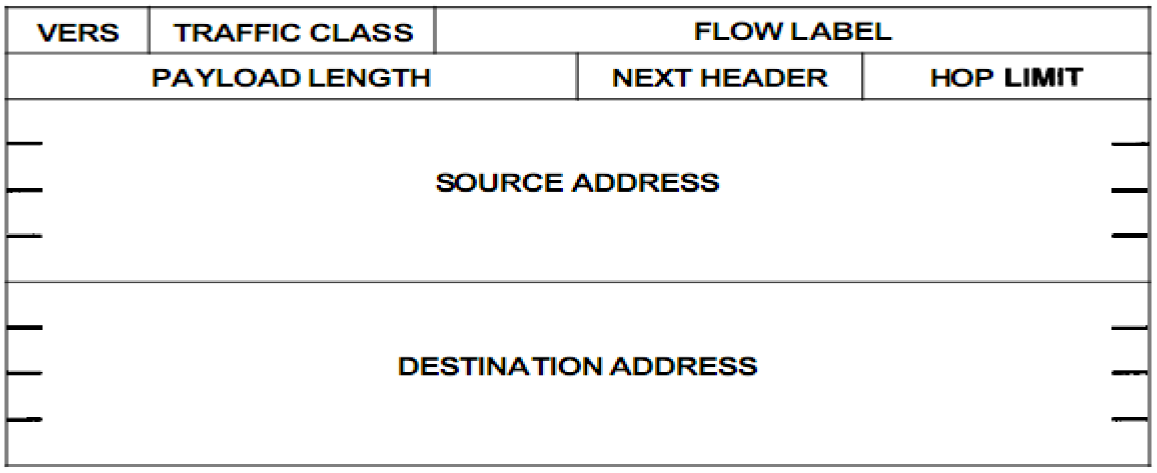


Figure 13.2: The format of the 40-octet IPv6 base header. Each 1-6 datagram begins with a base header.

IPv6 handles datagram length specifications in a new way. First, because the size of the base header is fixed at 40 octets, the base header does not include a field for the header length. Second, IPv6 replaces IPv4's datagram length field by a 16-bit PAY-LOAD LENGTH field that specifies the number of octets carried in the datagram excluding the header itself. Thus, an IPV6 datagram can contain 64K octets of data. Two fields in the base header are used in making forwarding decisions. The IPv4 SERVICE CLASS field has been renamed TRAFFIC CLASS. In addition, a new mechanism in IPv6 supports resource reservation and allows a router to associate each datagram with a given resource allocation. The underlying abstraction, a flow, consists of a path through an internet along which intermediate routers guarantee a specific quality of service. Field FLOW LABEL in the base header contains information that routers use to associate a datagram with a specific flow and priority. For example, two applications that need to send video can establish a flow on which the delay and bandwidth is guaranteed. Alternatively, a network provider may require a subscriber to specify the quality of service desired, and then use a flow to limit the traffic a specific computer or a specific application sends. Note that flows can also be used within a given organization to manage network resources and ensure that all applications receive a fair share. A router uses the combination of datagram source address and flow identifier when associating a datagram with a specific flow. To summarize:

Each IPv6 datagram begins with a 40-octet base header that includes fields for the source and destination addresses, the maximum hop limit, the traffic class, the flow label, and the type of the next header. Thus, an IPv6 datagram must contain at least 40 octets in addition to the data.

13.6 IPv6 Extension Headers

The paradigm of a fixed base header followed by a set of optional extension headers was chosen as a compromise between generality and efficiency. To be totally general, IPv6 needs to include mechanisms to support functions such as fragmentation, source routing, and authentication. However, choosing to allocate fixed fields in the datagram header for all mechanisms is inefficient because most datagrams do not use all mechanisms; the large IPv6 address size exacerbates the inefficiency. For example, when sending a datagram across a single local area network, a header that contains empty address fields can occupy a substantial fraction of each frame. More important, the designers realize that no one can predict which facilities will be needed. The IPV6 extension header paradigm works similar to IPv4 options - a sender can choose which extension headers to include in a given datagram and which to omit. Thus, extension headers provide maximum flexibility. We can summarize:

IPv6 extension headers are similar to IPv4 options. Each datagram includes extension headers for only those facilities that the datagram uses.

Assignment Questions

UNIT V

Multiple choice questions

1. A file with large size can be shared using _______ protocol.
   a. SNMP **b. FTP** c. TFTP d. POP
2. The protocol which is connection oriented for communication is _____
   a. **FTP** b. TFTP c. SNMP d. POP
3. The protocol which uses two separate lines for control and data communication is _________
   a. POP b. SNMP c. TFTP **d. FTP**
4. The protocol which provides an inexpensive and unsophisticated service is ____________
   a. POP b. SNMP **c. TFTP** d. FTP
5. The software which is much smaller and can be encoded within a ROM is _________
   a. POP b. SNMP **c. TFTP** d. FTP
6. The sending side of TFTP transmits the file with a fixed block size of _____ bytes
   a. 512 b. 1024 c. 2048 d. Any random size.
7. Network File System is developed by _______
   a. Microsoft b. Intel **c. Sun Microsystem** d. None

8. The protocol used for simple text mail transmission is
   a. SMTP b. FTP c. TFTP d. MIME
9. The protocol which transfers the mail from a mail box to the local computer is ____________
   a. SMTP b. FTP **c. POP** d. HTTP
10. The protocol which allows the mail messages from multiple locations and ensures that all copies are synchronized and consistent is _______
    a. SMTP b. **IMAP** c. POP d. HTTP
11. The __________ protocol is used to send non ASCII data through e mail
    a. MIME b. POP c. HTTP d. SMTP
12. The variable length option field is the feature in
    a. IPV4 b. IP c.TCP **d. IPV6**
13. The _____ has the flexible header format.
    a. IPV4 b. IP c.TCP **d. IPV6**
14. The _____ has provision for extension.

a. IPV4 b. IP c.TCP **d. IPV6**

15. The ______ has support for autoconfiguration and renumbering.

a. IPV4 b. IP c.TCP **d. IPV6**

16. The ____ has compulsory base header and optional extension header.

a. IPV4 b. IP c.TCP **d. IPV6**

17. The HOP LIMIT field is available in _________

a. IPV4 b. IP c.TCP **d. IPV6**

18. The PAY LOAD LENGTH is available in ________

a. IPV4 b. IP c.TCP **d. IPV6**

19. The TRAFFIC CLASS is available in ___________

a. IPV4 b. IP c.TCP **d. IPV6**

20. The FRAGMENTATION is available in _________ part of IPV6

a. Base Header b. Extension Header c. Any of the above d. Not Available

21. The RCPT command is available in ___________ Protocol.

a. TFTP **b. SMTP** c. FTP d. MIME

22. The protocol which does not require reliable connection for file transmission is

a. FTP **b. TFTP** c. SNMP d. MIME

Answer the following questions Each carries 2 marks.

1. How do you share the file in the internet?
2. What do you mean by TFTP?
3. Who developed NFS.
4. Expand NFS and mention the use of the same.
5. What is SMTP and where is it used?
6. What is the need of IPv6?
7. What is the size of IPv6 datagram?

Essay Questions.

1. Explain the features of FTP.
2. With the help of a diagram explain the process model of FTP.
3. What is the use of TFTP?
4. Differentiate TFTP from FTP.
5. With the aid of a diagram explain the five TFTP packet types.
6. Explain NFS with the help of a diagram.
7. Explain the need and function of SMTP.
8. Write a note on Post Office Protocol.

9. Explain IMAP.
10. With the help of a diagram explain MIME message.
11. Briefly explain the need of IPv6.
12. Explain the features of IPv6.
13. List and explain the seven different categories of changes introduced in IPv6.
14. With the help of a diagram explain the format of IPv6 datagram.

CHAPTER 14
VALUE ADDED CHAPTER INTERNET SECURITY AND FIREWALL

14.1 Introduction

Like the locks used to help keep tangible property secure, computers and data networks need provisions that help keep information secure. Security in an internet environment is both important and difficult. It is important because information has significant value - information can be bought and sold directly or used indirectly to create new products and services that yield high profits. Security in an internet is difficult because security involves understanding when and how participating users, computers, services, and networks can trust one another as well as understanding the technical details of network hardware and protocols. Security is required on every computer and every protocol; a single weakness can compromise the security of an entire network. More important, because supports a wide diversity of users, services, and networks and because an internet can span many political and organizational boundaries, participating individuals and organizations may not agree on a level of trust or policies for handling data.

This chapter considers two fundamental techniques that form the basis for internet security: perimeter security and encryption. Perimeter security allows an organization to determine the services and networks it will make available to outsiders and the extent to which outsiders can use resources. Encryption handles most other aspects of security. We begin by reviewing a few basic concepts and terminology.

14.2 Protecting resources

The terms network security and information security refer in a broad sense to confidence that information and services available on a network cannot be accessed by unauthorized users. Security implies safety, including assurance of data integrity, freedom from unauthorized access of computational resources, freedom from snooping or wiretapping, and freedom from disruption of service. Of course, just as no physical property is absolutely secure against crime, no network is completely secure. Organizations make an effort to secure networks for the same reason they make an effort to secure buildings and offices: basic security measures can discourage crime by making it significantly more difficult.

Providing security for information requires protecting both physical and abstract resources. Physical resources include passive storage devices such as disks and CD-ROMs as well as active devices such as users' computers. In a network environment, physical security extends to the cables, bridges, and routers that comprise the network infrastructure. Indeed, although physical security is seldom mentioned, it often plays an important role in an overall security plan. Obviously, physical security can prevent wiretapping. Good physical security can also eliminate sabotage (e.g., disabling a router to cause packets to be routed through an alternative, less secure path). Protecting an abstract resource such as information is usually more difficult than providing physical security because information is elusive. Information security encompasses many aspects of protection:

- Data integrity. A secure system must protect information from unauthorized change.
- Data availability The system must guarantee that outsiders cannot prevent legitimate access to data (e.g., any outsider should not be able to block customers from accessing a Web site).

- Privacy or confidentiality. The system must prevent outsiders from making copies of data as it passes across a network or understanding the contents if copies are made.

- Authorization. Although physical security often classifies people and resources into broad categories, (e.g., all nonemployees are forbidden from using a particular hallway), security for information usually needs to be more restrictive (e.g., some parts of an employee's record are available only to the personnel office, others are available only to the employee's boss, and others are available to the payroll office).

- Authentication. The system must allow two communicating entities to validate each other's identity.

- Replay avoidance. To prevent outsiders from capturing copies of packets and using them later, the system must prevent a retransmitted copy of a packet from being accepted.

14.3 Information Policy

Before an organization can enforce network security, the organization must assessrisks and develop a clear policy regarding information access and protection. The policy specifies who will be granted access to each piece of information, the rules an individual must follow in disseminating the information to others, and a statement of how the organization will react to violations.An information policy begins with people because:

Humans are usually the most susceptible point in any security scheme. A worker who is malicious, careless, or unaware of an organization's information policy can compromise the best security.

14.4 Internet Security

Internet security is difficult because datagrams traveling from source to destinationoften pass across many intermediate networks and through routers that are not owned or controlled by either the sender or the recipient. Thus, because datagrams can be intercepted or compromised, the contents cannot be trusted. As an example, consider a server that attempts to use source authentication to verify that requests originated from valid customers. Source authentication requires the server to examine the source IP address on each incoming datagram, and only accept requests from computers on an authorized list. Source authentication is weak because it can be broken easily. In particular, an intermediate router can watch traveling to and from the server, and record the IP address of a valid customer. Later the intermediate router can manufacture a request that has the same source address (and intercept the reply). The point is:

An authorization scheme that uses a remote machine's IP address to authenticate its identity does not suffice in an unsecure internet. An imposter who gains control of an intermediate router can obtain access by impersonating an authorized client.

Stronger authentication requires encryption. To encrypt a message, the sender applies a mathematical function that rearranges the bits according to a key which is known only to the sender. The receiver uses another mathematical function to decrypt the message. Careful choices of an encryption algorithm, a key, and the contents of messages can make it virtually impossible for intermediate machines to decode messages or manufacture messages that are valid.

14.5 Firewalls and internet access

Mechanisms that control internet access handle the problem of screening a particular network or an organization from unwanted communication. Such mechanisms can help prevent outsiders from: obtaining information, changing information, or disrupting communication on an organization's intranet. Successful access control requires a careful combination of restrictions on network topology, intermediate information staging, and packet filters.

A single technique known as an internet firewall, has emerged as the basis for internet access control. An organization places a firewall at its connection to external networks (e.g., the global Internet). A firewall partitions an internet into two regions, referred to informally as the inside and outside.

14.6 Encryption and Decryption

Human being from ages had two inherent needs − (a) to communicate and share information and (b) to communicate selectively. These two needs gave rise to the art of coding the messages in such a way that only the intended people could have access to the information. Unauthorized people could not extract any information, even if the scrambled messages fell in their hand.
The art and science of concealing the messages to introduce secrecy in information security is recognized as cryptography.
The word ‘cryptography’ was coined by combining two Greek words, ‘Krypto’ meaning hidden and ‘graphene’ meaning writing.

History of Cryptography

The art of cryptography is considered to be born along with the art of writing. As civilizations evolved, human beings got organized in tribes, groups, and kingdoms. This led to the emergence of ideas such as power, battles, supremacy, and politics. These ideas further fueled the natural need of people to communicate secretly with selective recipient which in turn ensured the continuous evolution of cryptography as well.

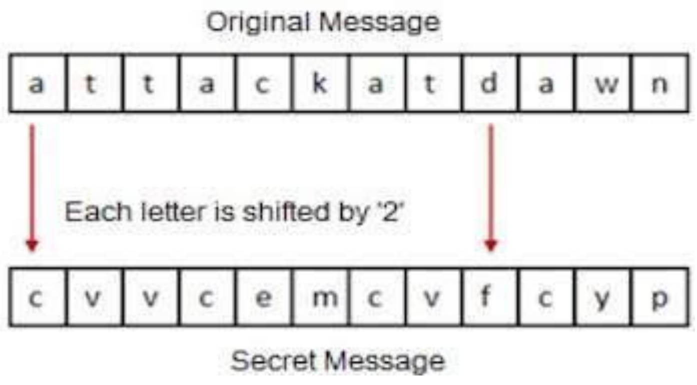
The roots of cryptography are found in Roman and Egyptian civilizations.

Hieroglyph − The Oldest Cryptographic Technique

The first known evidence of cryptography can be traced to the use of ‘hieroglyph’. Some 4000 years ago, the Egyptians used to communicate by messages written in hieroglyph. This code was the secret known only to the scribes who used to transmit messages on behalf of the kings. One such hieroglyph is shown below.

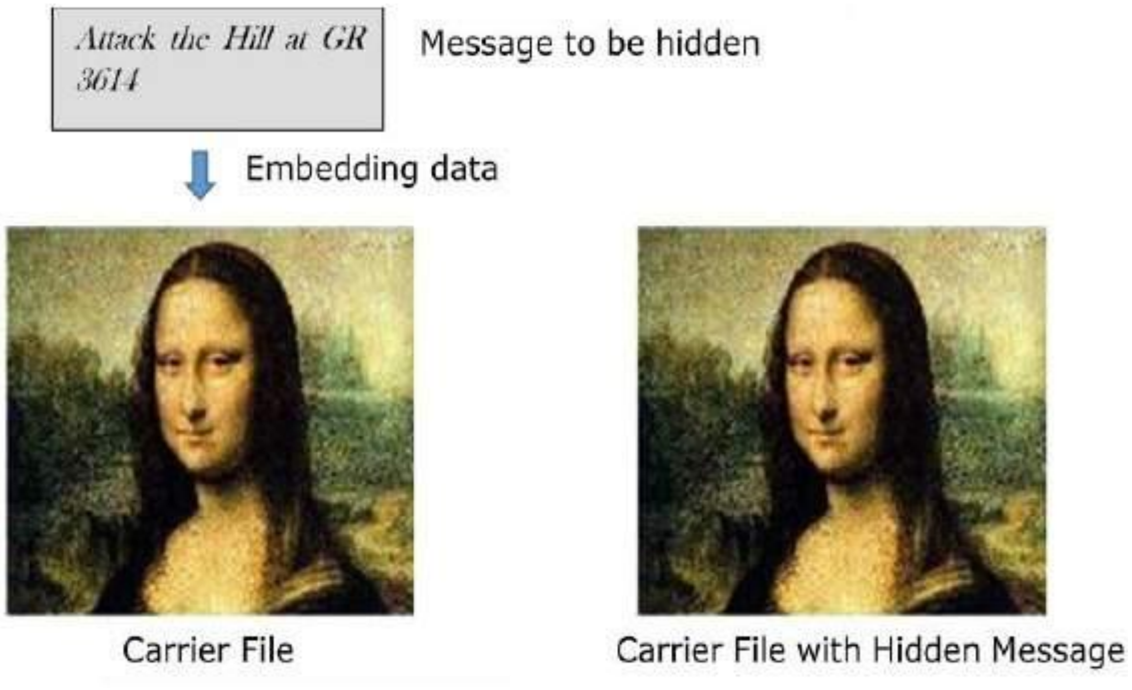
Later, the scholars moved on to using simple mono-alphabetic substitution ciphers during 500 to 600 BC. This involved replacing alphabets of message with other alphabets with some secret rule. This **rule** became a **key** to retrieve the message back from the garbled message.
The earlier Roman method of cryptography, popularly known as the **Caesar Shift Cipher,** relies on shifting the letters of a message by an agreed number (three was a common

choice), the recipient of this message would then shift the letters back by the same number and obtain the original message.



Steganography

Steganography is similar but adds another dimension to Cryptography. In this method, people not only want to protect the secrecy of an information by concealing it, but they also want to make sure any unauthorized person gets no evidence that the information even exists. For example, **invisible watermarking**.
In steganography, an unintended recipient or an intruder is unaware of the fact that observed data contains hidden information. In cryptography, an intruder is normally aware that data is being communicated, because they can see the coded/scrambled message.



Evolution of Cryptography

It is during and after the European Renaissance, various Italian and Papal states led the rapid proliferation of cryptographic techniques. Various analysis and attack techniques were researched in this era to break the secret codes.

- Improved coding techniques such as **Vigenere Coding** came into existence in the 15th century, which offered moving letters in the message with a number of variable places instead of moving them the same number of places.
- Only after the 19th century, cryptography evolved from the ad hoc approaches to encryption to the more sophisticated art and science of information security.

- In the early 20th century, the invention of mechanical and electromechanical machines, such as the **Enigma rotor machine,**provided more advanced and efficient means of coding the information.
- During the period of World War II, both **cryptography** and **cryptanalysis** became excessively mathematical.

With the advances taking place in this field, government organizations, military units, and some corporate houses started adopting the applications of cryptography. They used cryptography to guard their secrets from others. Now, the arrival of computers and the Internet has brought effective cryptography within the reach of common people.

Modern cryptography is the cornerstone of computer and communications security. Its foundation is based on various concepts of mathematics such as number theory, computational-complexity theory, and probability theory.

Characteristics of Modern Cryptography

There are three major characteristics that separate modern cryptography from the classical approach.

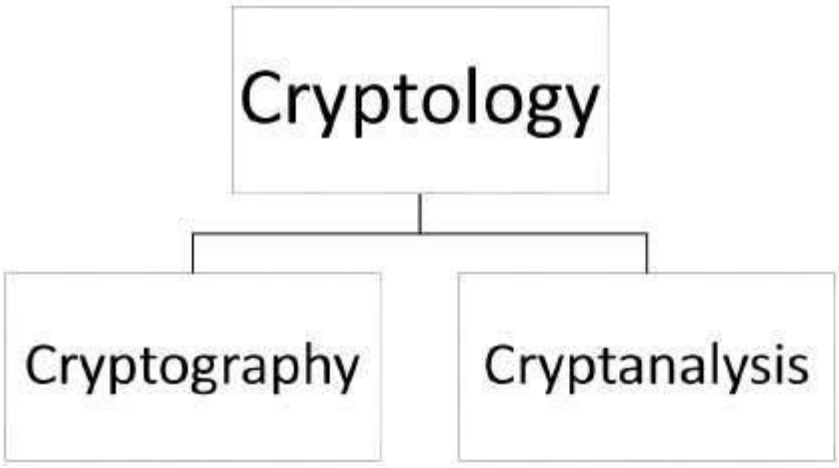
Classic Cryptography	Modern Cryptography
It manipulates traditional characters, i.e., letters and digits directly.	It operates on binary bit sequences.
It is mainly based on 'security through obscurity'. The techniques employed for coding were kept secret and only the parties involved in communication knew about them.	It relies on publicly known mathematical algorithms for coding the information. Secrecy is obtained through a secrete key which is used as the seed for the algorithms. The computational difficulty of algorithms, absence of secret key, etc., make it impossible for an attacker to obtain the original information even if he knows the

	algorithm used for coding.
It requires the entire cryptosystem for communicating confidentially.	Modern cryptography requires parties interested in secure communication to possess the secret key only.

Context of Cryptography

Cryptology, the study of cryptosystems, can be subdivided into two branches −

- Cryptography
- Cryptanalysis



What is Cryptography?

Cryptography is the art and science of making a cryptosystem that is capable of providing information security.

Cryptography deals with the actual securing of digital data. It refers to the design of mechanisms based on mathematical algorithms that provide fundamental information security services. You can think of cryptography as the establishment of a large toolkit containing different techniques in security applications.

What is Cryptanalysis?

The art and science of breaking the cipher text is known as cryptanalysis.

Cryptanalysis is the sister branch of cryptography and they both co-exist. The cryptographic process results in the cipher text for transmission or storage. It involves the study of cryptographic mechanism with the intention to break them. Cryptanalysis is also used during the design of the new cryptographic techniques to test their security strengths.

Note − Cryptography concerns with the design of cryptosystems, while cryptanalysis studies the breaking of cryptosystems.Security Services of Cryptography The primary objective of using cryptography is to provide the following four fundamental information security services. Let us now see the possible goals intended to be fulfilled by cryptography. Confidentiality Confidentiality is the fundamental security service provided by cryptography. It is a security service that keeps the information from an unauthorized person. It is sometimes referred to as **privacy** or **secrecy**. Confidentiality can be achieved through numerous means starting from physical securing to the use of mathematical algorithms for data encryption. Data Integrity

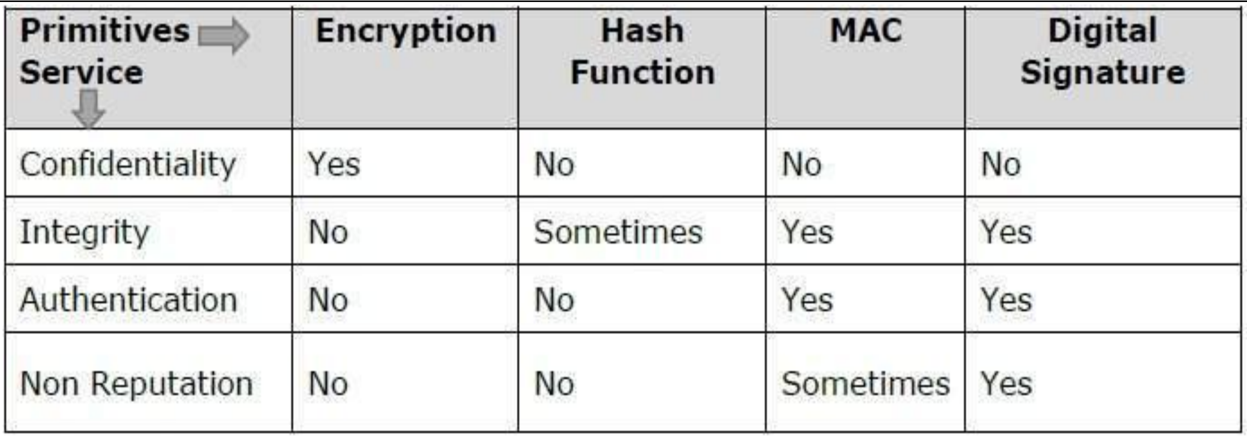
It is security service that deals with identifying any alteration to the data. The data may get modified by an unauthorized entity intentionally or accidently. Integrity service confirms that whether data is intact or not since it was last created, transmitted, or stored by an authorized user. Data integrity cannot prevent the alteration of data, but provides a means for detecting whether data has been manipulated in an unauthorized manner. Authentication Authentication provides the identification of the originator. It confirms to the receiver that the data received has been sent only by an identified and verified sender. Authentication service has two variants −

- **Message authentication** identifies the originator of the message without any regard router or system that has sent the message.
- **Entity authentication** is assurance that data has been received from a specific entity, say a particular website.

Apart from the originator, authentication may also provide assurance about other parameters related to data such as the date and time of creation/transmission. Non-repudiation It is a security service that ensures that an entity cannot refuse the ownership of a previous commitment or an action. It is an assurance that the original creator of the data cannot deny the creation or transmission of the said data to a recipient or third party. Non-repudiation is a property that is most desirable in situations where there are chances of a dispute over the exchange of data. For example, once an order is placed electronically, a purchaser cannot deny the purchase order, if non-repudiation service was enabled in this transaction. Cryptography Primitives Cryptography primitives are nothing but the tools and techniques in Cryptography that can be selectively used to provide a set of desired security services −

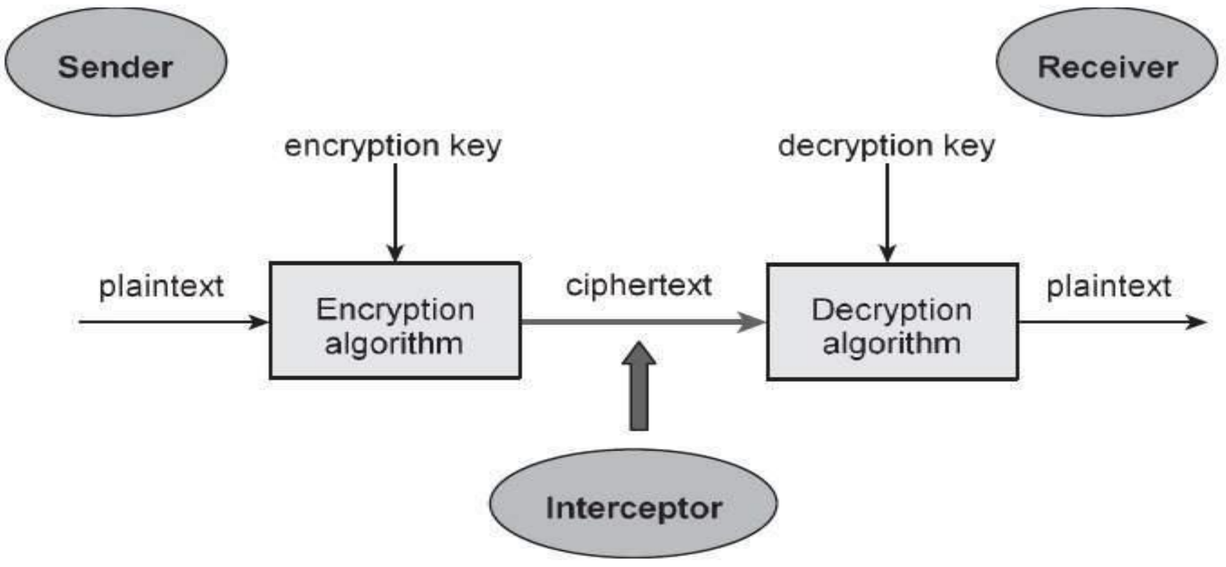
- Encryption
- Hash functions
- Message Authentication codes (MAC)
- Digital Signatures

The following table shows the primitives that can achieve a particular security service on their own.



Primitives ⇨ Service ⇩	Encryption	Hash Function	MAC	Digital Signature
Confidentiality	Yes	No	No	No
Integrity	No	Sometimes	Yes	Yes
Authentication	No	No	Yes	Yes
Non Reputation	No	No	Sometimes	Yes

Note − Cryptographic primitives are intricately related and they are often combined to achieve a set of desired security services from a cryptosystem.
A cryptosystem is an implementation of cryptographic techniques and their accompanying infrastructure to provide information security services. A cryptosystem is also referred to as a **cipher system**.
Let us discuss a simple model of a cryptosystem that provides confidentiality to the information being transmitted. This basic model is depicted in the illustration below −



The illustration shows a sender who wants to transfer some sensitive data to a receiver in such a way that any party intercepting or eavesdropping on the communication channel cannot extract the data.
The objective of this simple cryptosystem is that at the end of the process, only the sender and the receiver will know the plaintext.

Components of a Cryptosystem

The various components of a basic cryptosystem are as follows −

- **Plaintext.** It is the data to be protected during transmission.
- **Encryption Algorithm.** It is a mathematical process that produces a ciphertext for any given plaintext and encryption key. It is a cryptographic algorithm that takes plaintext and an encryption key as input and produces a ciphertext.
- **Ciphertext.** It is the scrambled version of the plaintext produced by the encryption algorithm using a specific the encryption key. The ciphertext is not guarded. It flows on

public channel. It can be intercepted or compromised by anyone who has access to the communication channel.

- **Decryption Algorithm,** It is a mathematical process, that produces a unique plaintext for any given ciphertext and decryption key. It is a cryptographic algorithm that takes a ciphertext and a decryption key as input, and outputs a plaintext. The decryption algorithm essentially reverses the encryption algorithm and is thus closely related to it.
- **Encryption Key.** It is a value that is known to the sender. The sender inputs the encryption key into the encryption algorithm along with the plaintext in order to compute the ciphertext.
- **Decryption Key.** It is a value that is known to the receiver. The decryption key is related to the encryption key, but is not always identical to it. The receiver inputs the decryption key into the decryption algorithm along with the ciphertext in order to compute the plaintext.

For a given cryptosystem, a collection of all possible decryption keys is called a **key space**.

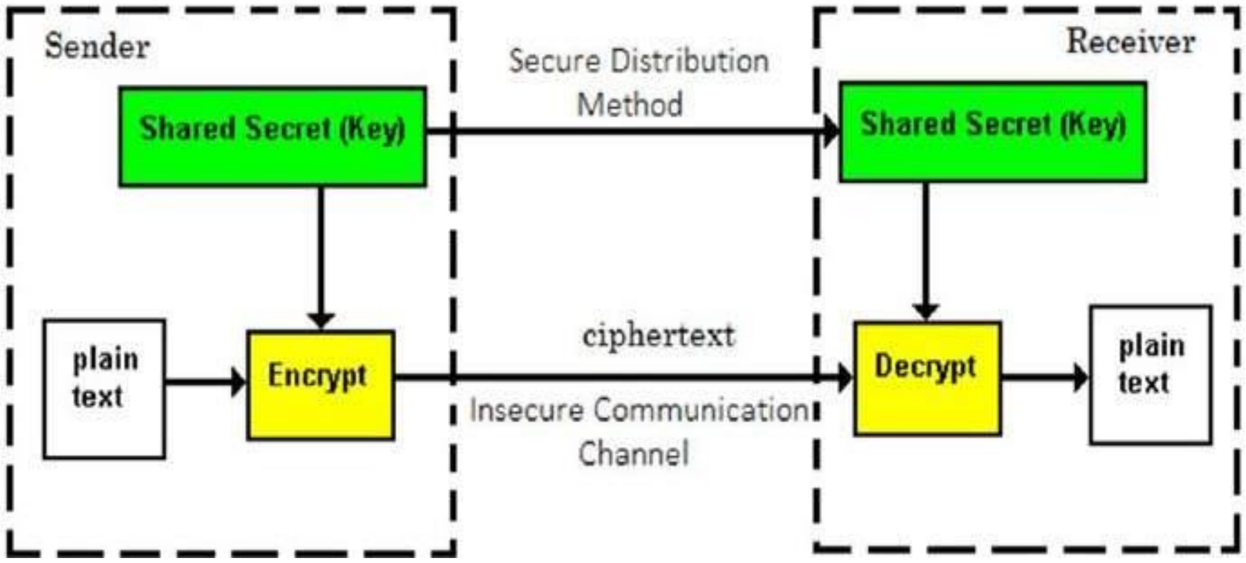
An **interceptor** (an attacker) is an unauthorized entity who attempts to determine the plaintext. He can see the ciphertext and may know the decryption algorithm. He, however, must never know the decryption key. Types of Cryptosystems Fundamentally, there are two types of cryptosystems based on the manner in which encryption-decryption is carried out in the system

- Symmetric Key Encryption
- Asymmetric Key Encryption

The main difference between these cryptosystems is the relationship between the encryption and the decryption key. Logically, in any cryptosystem, both the keys are closely associated. It is practically impossible to decrypt the ciphertext with the key that is unrelated to the encryption key.

Symmetric Key Encryption

The encryption process where **same keys are used for encrypting and decrypting** the information is known as Symmetric Key Encryption. The study of symmetric cryptosystems is referred to as **symmetric cryptography**. Symmetric cryptosystems are also sometimes referred to as **secret key cryptosystems**. A few well-known examples of symmetric key encryption methods are − Digital Encryption Standard (DES), Triple-DES (3DES), IDEA, and BLOWFISH.



Prior to 1970, all cryptosystems employed symmetric key encryption. Even today, its relevance is very high and it is being used extensively in many cryptosystems. It is very unlikely that this encryption will fade away, as it has certain advantages over asymmetric key encryption.

The salient features of cryptosystem based on symmetric key encryption are −

- Persons using symmetric key encryption must share a common key prior to exchange of information.
- Keys are recommended to be changed regularly to prevent any attack on the system.
- A robust mechanism needs to exist to exchange the key between the communicating parties. As keys are required to be changed regularly, this mechanism becomes expensive and cumbersome.
- In a group of **n** people, to enable two-party communication between any two persons, the number of keys required for group is **n × (n – 1)/2**.
- Length of Key (number of bits) in this encryption is smaller and hence, process of encryption-decryption is faster than asymmetric key encryption.
- Processing power of computer system required to run symmetric algorithm is less.

Challenge of Symmetric Key Cryptosystem

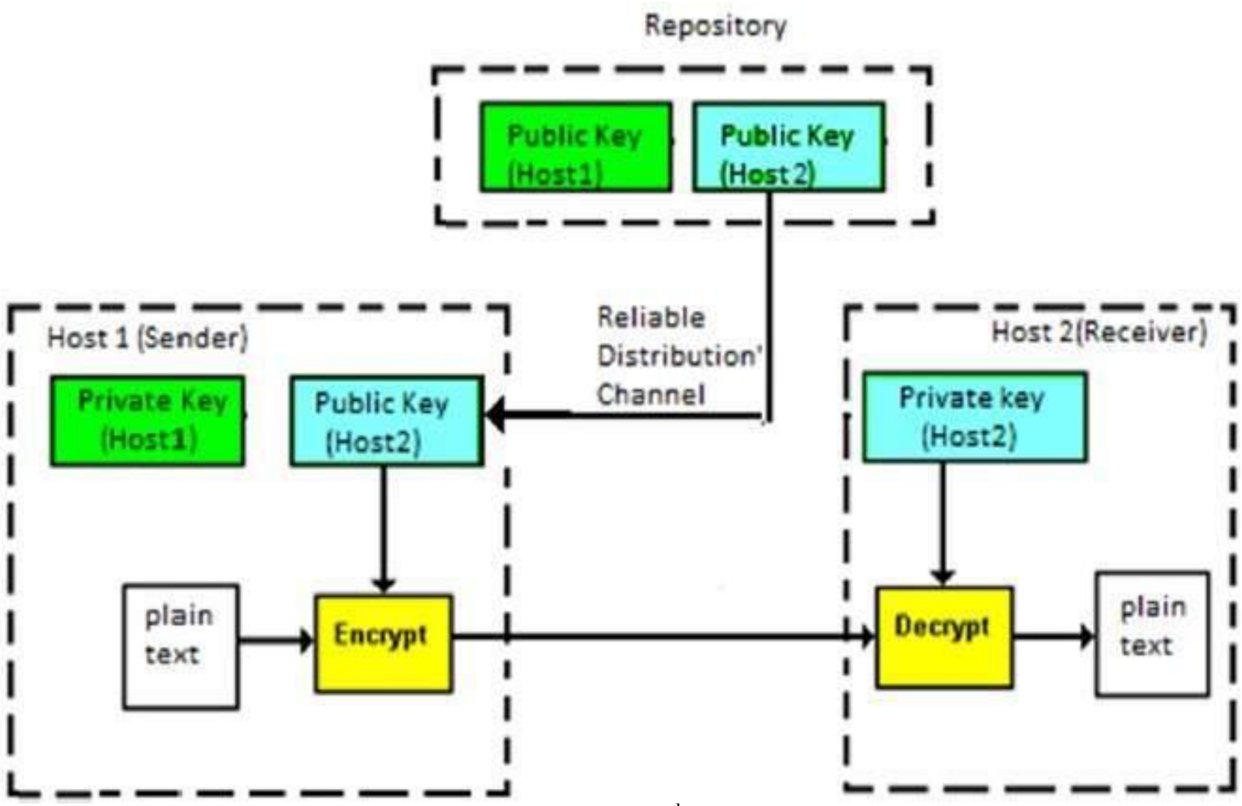
There are two restrictive challenges of employing symmetric key cryptography.

- **Key establishment** − Before any communication, both the sender and the receiver need to agree on a secret symmetric key. It requires a secure key establishment mechanism in place.
- **Trust Issue** − Since the sender and the receiver use the same symmetric key, there is an implicit requirement that the sender and the receiver 'trust' each other. For example, it may happen that the receiver has lost the key to an attacker and the sender is not informed.

These two challenges are highly restraining for modern day communication. Today, people need to exchange information with non-familiar and non-trusted parties. For example, a communication between online seller and customer. These limitations of symmetric key encryption gave rise to asymmetric key encryption schemes.

Asymmetric Key Encryption

The encryption process where **different keys are used for encrypting and decrypting the information** is known as Asymmetric Key Encryption. Though the keys are different, they are mathematically related and hence, retrieving the plaintext by decrypting ciphertext is feasible. The process is depicted in the following illustration −



Asymmetric Key Encryption was invented in the 20th century to come over the necessity of pre-shared secret key between communicating persons. The salient features of this encryption scheme are as follows −

- Every user in this system needs to have a pair of dissimilar keys, **private key** and **public key**. These keys are mathematically related − when one key is used for encryption, the other can decrypt the ciphertext back to the original plaintext.
- It requires to put the public key in public repository and the private key as a well-guarded secret. Hence, this scheme of encryption is also called **Public Key Encryption**.
- Though public and private keys of the user are related, it is computationally not feasible to find one from another. This is a strength of this scheme.
- When *Host1* needs to send data to *Host2,* he obtains the public key of *Host2* from repository, encrypts the data, and transmits.
- *Host2* uses his private key to extract the plaintext.
- Length of Keys (number of bits) in this encryption is large and hence, the process of encryption-decryption is slower than symmetric key encryption.
- Processing power of computer system required to run asymmetric algorithm is higher.

Symmetric cryptosystems are a natural concept. In contrast, public-key cryptosystems are quite difficult to comprehend.

You may think, *how can the encryption key and the decryption key are 'related', and yet it is impossible to determine the decryption key from the encryption key?* The answer lies in the mathematical concepts. It is possible to design a cryptosystem whose keys have this property. The concept of public-key cryptography is relatively new. There are fewer public-key algorithms known than symmetric algorithms.

Challenge of Public Key Cryptosystem

Public-key cryptosystems have one significant challenge − the user needs to trust that the public key that he is using in communications with a person really is the public key of that person and has not been spoofed by a malicious third party. This is usually accomplished through a Public Key Infrastructure (PKI) consisting a trusted third party. The third party securely manages and attests to the authenticity of public keys. When the third party is requested to provide the public key for any communicating person X, they are trusted to provide the correct public key.

The third party satisfies itself about user identity by the process of attestation, notarization, or some other process − that X is the one and only, or globally unique, X. The most common method of making the verified public keys available is to embed them in a certificate which is digitally signed by the trusted third party.

Relation between Encryption Schemes

A summary of basic key properties of two types of cryptosystems is given below −

	Symmetric Cryptosystems	Public Key Cryptosystems
Relation between Keys	Same	Different, but mathematically related
Encryption Key	Symmetric	Public
Decryption Key	Symmetric	Private

Due to the advantages and disadvantage of both the systems, symmetric key and public-key cryptosystems are often used together in the practical information security systems.

Kerckhoff's Principle for Cryptosystem

In the 19th century, a Dutch cryptographer A. Kerckhoff furnished the requirements of a good cryptosystem. Kerckhoff stated that a cryptographic system should be secure even if everything about the system, except the key, is public knowledge. The six design principles defined by Kerckhoff for cryptosystem are −

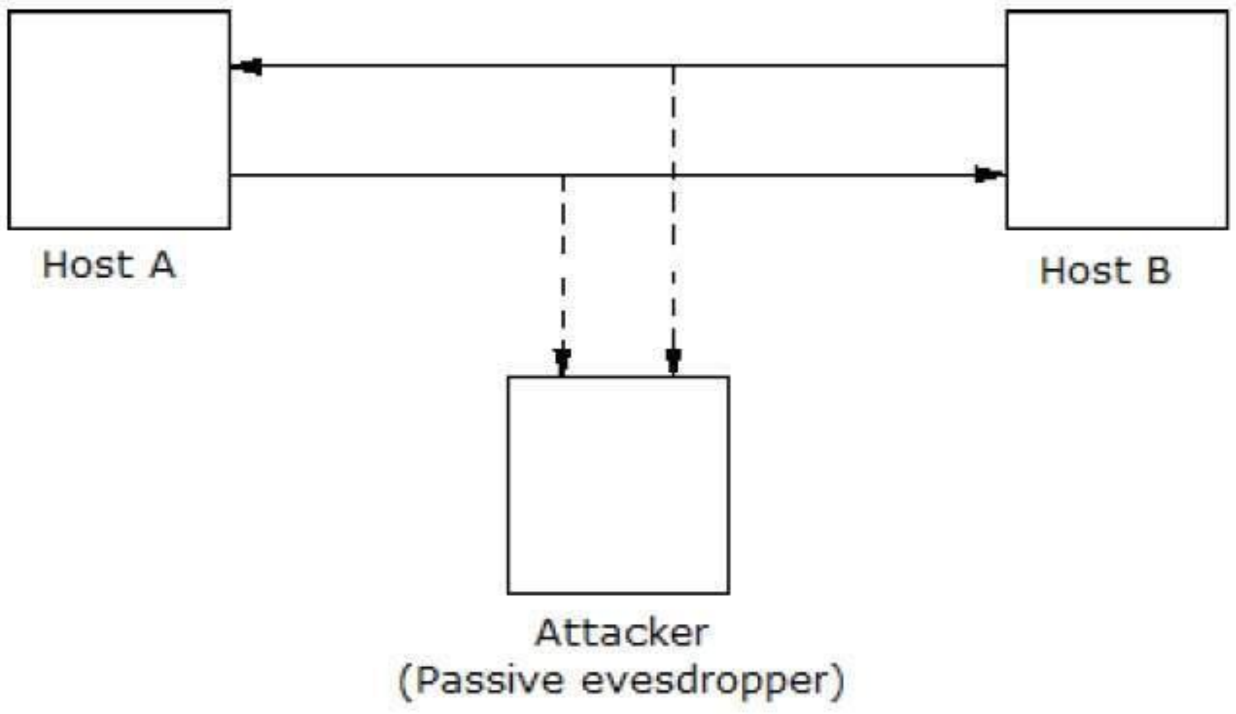
- The cryptosystem should be unbreakable practically, if not mathematically.
- Falling of the cryptosystem in the hands of an intruder should not lead to any compromise of the system, preventing any inconvenience to the user.
- The key should be easily communicable, memorable, and changeable.
- The ciphertext should be transmissible by telegraph, an unsecure channel.
- The encryption apparatus and documents should be portable and operable by a single person.
- Finally, it is necessary that the system be easy to use, requiring neither mental strain nor the knowledge of a long series of rules to observe.

The second rule is currently known as **Kerckhoff principle**. It is applied in virtually all the contemporary encryption algorithms such as DES, AES, etc. These public algorithms are considered to be thoroughly secure. The security of the encrypted message depends solely on the security of the secret encryption key. Keeping the algorithms secret may act as a significant barrier to cryptanalysis. However, keeping the algorithms secret is possible only when they are used in a strictly limited circle. In modern era, cryptography needs to cater to users who are connected to the Internet. In such cases, using a secret algorithm is not feasible, hence Kerckhoff principles became essential guidelines for designing algorithms in modern cryptography.

In the present era, not only business but almost all the aspects of human life are driven by information. Hence, it has become imperative to protect useful information from malicious activities such as attacks. Let us consider the types of attacks to which information is typically subjected to. Attacks are typically categorized based on the action performed by the attacker. An attack, thus, can be **passive** or **active**.

Passive Attacks

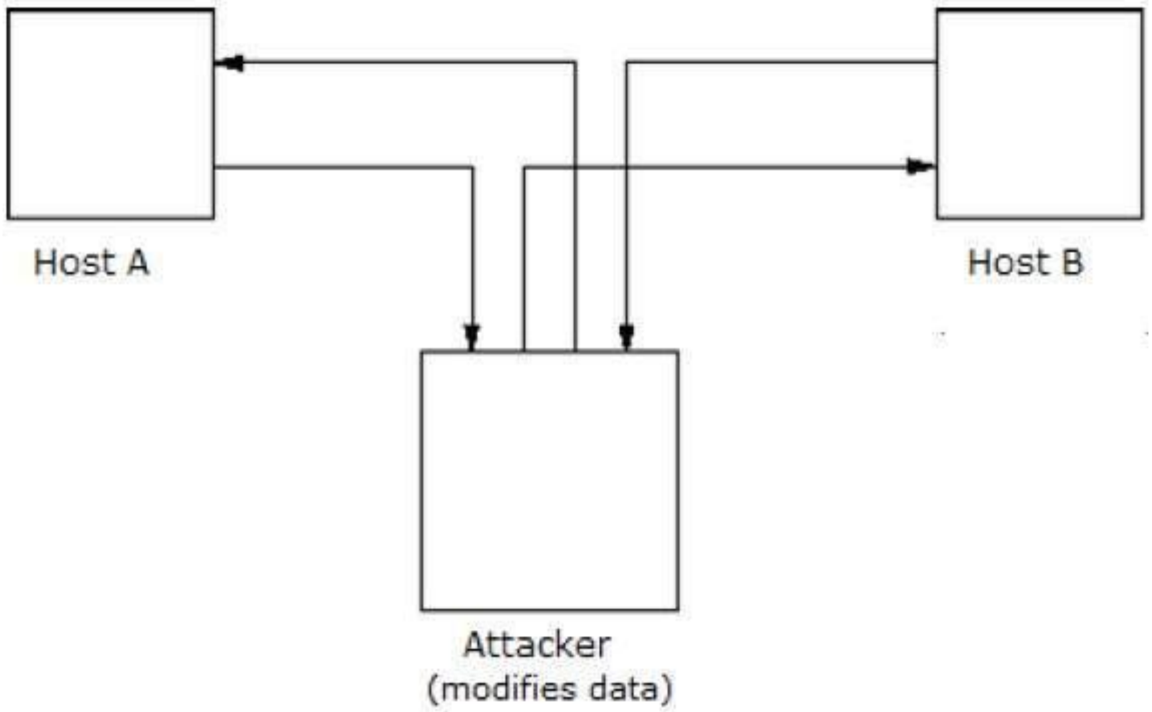
The main goal of a passive attack is to obtain **unauthorized access to the information**. For example, actions such as intercepting and eavesdropping on the communication channel can be regarded as passive attack. These actions are passive in nature, as they neither affect information nor disrupt the communication channel. A passive attack is often seen as *stealing*information. The only difference in stealing physical goods and stealing information is that theft of data still leaves the owner in possession of that data. Passive information attack is thus more dangerous than stealing of goods, as information theft may go unnoticed by the owner.



Active Attacks

An active attack involves changing the information in some way by conducting some process on the information. For example,

- Modifying the information in an unauthorized manner.
- Initiating unintended or unauthorized transmission of information.
- Alteration of authentication data such as originator name or timestamp associated with information
- Unauthorized deletion of data.
- Denial of access to information for legitimate users (denial of service).



Cryptography provides many tools and techniques for implementing cryptosystems capable of preventing most of the attacks described above.

Assumptions of Attacker

Let us see the prevailing environment around cryptosystems followed by the types of attacks employed to break these systems −

Environment around Cryptosystem

While considering possible attacks on the cryptosystem, it is necessary to know the cryptosystems environment. The attacker’s assumptions and knowledge about the environment decides his capabilities.

In cryptography, the following three assumptions are made about the security environment and attacker’s capabilities.

Details of the Encryption Scheme

The design of a cryptosystem is based on the following two cryptography algorithms −

- **Public Algorithms** − With this option, all the details of the algorithm are in the public domain, known to everyone.
- **Proprietary algorithms** − The details of the algorithm are only known by the system designers and users.

In case of proprietary algorithms, security is ensured through obscurity. Private algorithms may not be the strongest algorithms as they are developed in-house and may not be extensively investigated for weakness.

Secondly, they allow communication among closed group only. Hence they are not suitable for modern communication where people communicate with large number of known or unknown entities. Also, according to Kerckhoff’s principle, the algorithm is preferred to be public with strength of encryption lying in the *key*.

Thus, the first assumption about security environment is that the **encryption algorithm is known to the attacker**.

Availability of Ciphertext

We know that once the plaintext is encrypted into ciphertext, it is put on unsecure public channel (say email) for transmission. Thus, the attacker can obviously assume that it has **access to the ciphertext generated by the cryptosystem**.

Availability of Plaintext and Ciphertext

This assumption is not as obvious as other. However, there may be situations where an attacker can have **access to plaintext and corresponding ciphertext**. Some such possible circumstances are −

- The attacker influences the sender to convert plaintext of his choice and obtains the ciphertext.

- The receiver may divulge the plaintext to the attacker inadvertently. The attacker has access to corresponding ciphertext gathered from open channel.
- In a public-key cryptosystem, the encryption key is in open domain and is known to any potential attacker. Using this key, he can generate pairs of corresponding plaintexts and ciphertexts.

Cryptographic Attacks

The basic intention of an attacker is to break a cryptosystem and to find the plaintext from the ciphertext. To obtain the plaintext, the attacker only needs to find out the secret decryption key, as the algorithm is already in public domain.

Hence, he applies maximum effort towards finding out the secret key used in the cryptosystem. Once the attacker is able to determine the key, the attacked system is considered as *broken* or *compromised*.

Based on the methodology used, attacks on cryptosystems are categorized as follows −

- **Ciphertext Only Attacks (COA)** − In this method, the attacker has access to a set of ciphertext(s). He does not have access to corresponding plaintext. COA is said to be successful when the corresponding plaintext can be determined from a given set of ciphertext. Occasionally, the encryption key can be determined from this attack. Modern cryptosystems are guarded against ciphertext-only attacks.
- **Known Plaintext Attack (KPA)** − In this method, the attacker knows the plaintext for some parts of the ciphertext. The task is to decrypt the rest of the ciphertext using this information. This may be done by determining the key or via some other method. The best example of this attack is *linear cryptanalysis* against block ciphers.
- **Chosen Plaintext Attack (CPA)** − In this method, the attacker has the text of his choice encrypted. So he has the ciphertext-plaintext pair of his choice. This simplifies his task of determining the encryption key. An example of this attack is *differential cryptanalysis* applied against block ciphers as well as hash functions. A popular public key cryptosystem, RSA is also vulnerable to chosen-plaintext attacks.
- **Dictionary Attack** − This attack has many variants, all of which involve compiling a ‘dictionary’. In simplest method of this attack, attacker builds a dictionary of ciphertexts and corresponding plaintexts that he has learnt over a period of time. In future, when an attacker gets the ciphertext, he refers the dictionary to find the corresponding plaintext.
- **Brute Force Attack (BFA)** − In this method, the attacker tries to determine the key by attempting all possible keys. If the key is 8 bits long, then the number of possible keys is 2^8 = 256. The attacker knows the ciphertext and the algorithm, now he attempts all the

256 keys one by one for decryption. The time to complete the attack would be very high if the key is long.

- **Birthday Attack** − This attack is a variant of brute-force technique. It is used against the cryptographic hash function. When students in a class are asked about their birthdays, the answer is one of the possible 365 dates. Let us assume the first student's birthdate is 3rd Aug. Then to find the next student whose birthdate is 3rd Aug, we need to enquire 1.25^{*}□$\sqrt{365} \approx 25$ students.

  Similarly, if the hash function produces 64 bit hash values, the possible hash values are 1.8×10^{19}. By repeatedly evaluating the function for different inputs, the same output is expected to be obtained after about 5.1×10^{9} random inputs.

  If the attacker is able to find two different inputs that give the same hash value, it is a **collision** and that hash function is said to be broken.
- **Man in Middle Attack (MIM)** − The targets of this attack are mostly public key cryptosystems where key exchange is involved before communication takes place.
  - Host *A* wants to communicate to host *B*, hence requests public key of *B*.
  - An attacker intercepts this request and sends his public key instead.
  - Thus, whatever host *A* sends to host *B*, the attacker is able to read.
  - In order to maintain communication, the attacker re-encrypts the data after reading with his public key and sends to *B*.
  - The attacker sends his public key as *A*'s public key so that *B*takes it as if it is taking it from *A*.
- **Side Channel Attack (SCA)** − This type of attack is not against any particular type of cryptosystem or algorithm. Instead, it is launched to exploit the weakness in physical implementation of the cryptosystem.
- **Timing Attacks** − They exploit the fact that different computations take different times to compute on processor. By measuring such timings, it is be possible to know about a particular computation the processor is carrying out. For example, if the encryption takes a longer time, it indicates that the secret key is long.
- **Power Analysis Attacks** − These attacks are similar to timing attacks except that the amount of power consumption is used to obtain information about the nature of the underlying computations.
- **Fault analysis Attacks** − In these attacks, errors are induced in the cryptosystem and the attacker studies the resulting output for useful information.

Practicality of Attacks

The attacks on cryptosystems described here are highly academic, as majority of them come from the academic community. In fact, many academic attacks involve quite unrealistic assumptions about environment as well as the capabilities of the attacker. For example, in chosen-ciphertext attack, the attacker requires an impractical number of deliberately chosen plaintext-ciphertext pairs. It may not be practical altogether.

Nonetheless, the fact that any attack exists should be a cause of concern, particularly if the attack technique has the potential for improvement.

In the second chapter, we discussed the fundamentals of modern cryptography. We equated cryptography with a toolkit where various cryptographic techniques are considered as the basic tools. One of these tools is the Symmetric Key Encryption where the key used for encryption and decryption is the same.

In this chapter, we discuss this technique further and its applications to develop various cryptosystems.

Earlier Cryptographic Systems

Before proceeding further, you need to know some facts about historical cryptosystems −

- All of these systems are **based on symmetric key encryption**scheme.
- The only security service these systems provide is confidentiality of information.
- Unlike modern systems which are digital and treat data as binary numbers, the earlier systems worked on alphabets as basic element.

These earlier cryptographic systems are also referred to as Ciphers. In general, a cipher is simply just a set of steps (an algorithm) for performing both an encryption, and the corresponding decryption.

Caesar Cipher

It is a mono-alphabetic cipher wherein each letter of the plaintext is substituted by another letter to form the ciphertext. It is a simplest form of substitution cipher scheme.

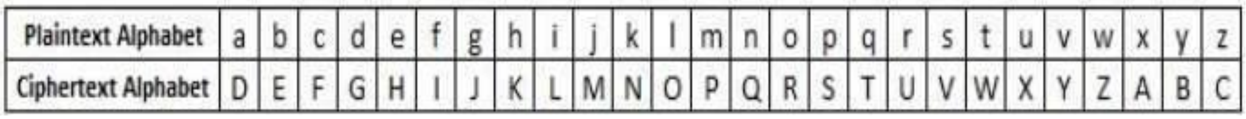
This cryptosystem is generally referred to as the **Shift Cipher**. The concept is to replace each alphabet by another alphabet which is 'shifted' by some fixed number between 0 and 25.

For this type of scheme, both sender and receiver agree on a 'secret shift number' for shifting the alphabet. This number which is between 0 and 25 becomes the key of encryption.

The name 'Caesar Cipher' is occasionally used to describe the Shift Cipher when the 'shift of three' is used.

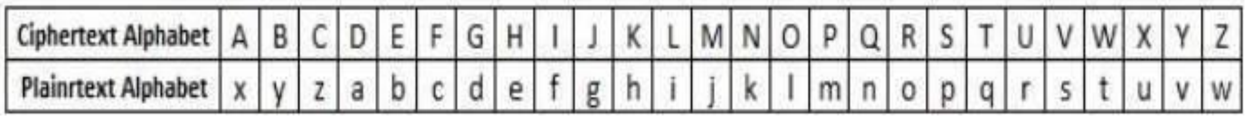
Process of Shift Cipher

- In order to encrypt a plaintext letter, the sender positions the sliding ruler underneath the first set of plaintext letters and slides it to LEFT by the number of positions of the secret shift.
- The plaintext letter is then encrypted to the ciphertext letter on the sliding ruler underneath. The result of this process is depicted in the following illustration for an agreed shift of three positions. In this case, the plaintext ‘tutorial’ is encrypted to the ciphertext ‘WXWRULDO’. Here is the ciphertext alphabet for a Shift of 3 −



Plaintext Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext Alphabet	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

- On receiving the ciphertext, the receiver who also knows the secret shift, positions his sliding ruler underneath the ciphertext alphabet and slides it to RIGHT by the agreed shift number, 3 in this case.
- He then replaces the ciphertext letter by the plaintext letter on the sliding ruler underneath. Hence the ciphertext ‘WXWRULDO’ is decrypted to ‘tutorial’. To decrypt a message encoded with a Shift of 3, generate the plaintext alphabet using a shift of ‘-3’ as shown below −



Ciphertext Alphabet	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Plainrtext Alphabet	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Security Value

Caesar Cipher is **not a secure** cryptosystem because there are only 26 possible keys to try out. An attacker can carry out an exhaustive key search with available limited computing resources.

Simple Substitution Cipher

It is an improvement to the Caesar Cipher. Instead of shifting the alphabets by some number, this scheme uses some permutation of the letters in alphabet.

For example, A.B…..Y.Z and Z.Y……B.A are two obvious permutation of all the letters in alphabet. Permutation is nothing but a jumbled up set of alphabets.

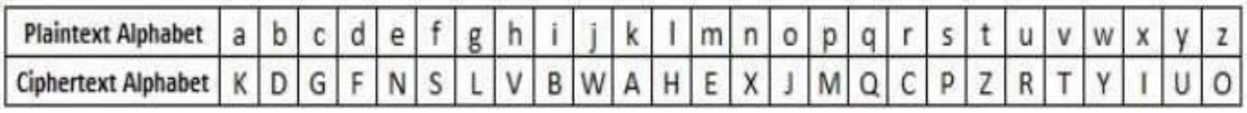
With 26 letters in alphabet, the possible permutations are 26! (Factorial of 26) which is equal to $4x10^{26}$. The sender and the receiver may choose any one of these possible permutation as a ciphertext alphabet. This permutation is the secret key of the scheme.

Process of Simple Substitution Cipher

- Write the alphabets A, B, C,...,Z in the natural order.
- The sender and the receiver decide on a randomly selected permutation of the letters of the alphabet.

- Underneath the natural order alphabets, write out the chosen permutation of the letters of the alphabet. For encryption, sender replaces each plaintext letters by substituting the permutation letter that is directly beneath it in the table. This process is shown in the following illustration. In this example, the chosen permutation is K,D, G, ..., O. The plaintext ‘point’ is encrypted to ‘MJBXZ’.

Here is a jumbled Ciphertext alphabet, where the order of the ciphertext letters is a key.



Plaintext Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext Alphabet	K	D	G	F	N	S	L	V	B	W	A	H	E	X	J	M	Q	C	P	Z	R	T	Y	I	U	O

- On receiving the ciphertext, the receiver, who also knows the randomly chosen permutation, replaces each ciphertext letter on the bottom row with the corresponding plaintext letter in the top row. The ciphertext ‘MJBXZ’ is decrypted to ‘point’.

Security Value

Simple Substitution Cipher is a considerable improvement over the Caesar Cipher. The possible number of keys is large (26!) and even the modern computing systems are not yet powerful enough to comfortably launch a brute force attack to break the system. However, the Simple Substitution Cipher has a simple design and it is prone to design flaws, say choosing obvious permutation, this cryptosystem can be easily broken.

Monoalphabetic and Polyalphabetic Cipher

Monoalphabetic cipher is a substitution cipher in which for a given key, the cipher alphabet for each plain alphabet is fixed throughout the encryption process. For example, if ‘A’ is encrypted as ‘D’, for any number of occurrence in that plaintext, ‘A’ will always get encrypted to ‘D’.

All of the substitution ciphers we have discussed earlier in this chapter are monoalphabetic; these ciphers are highly susceptible to cryptanalysis.

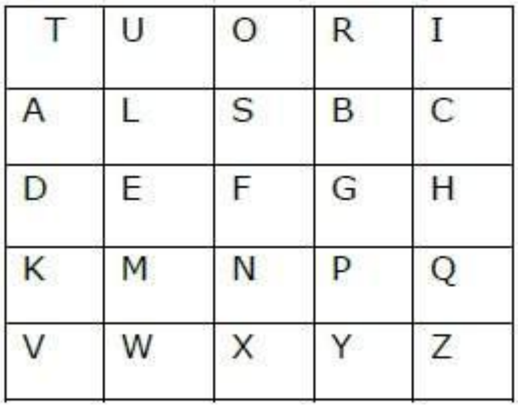
Polyalphabetic Cipher is a substitution cipher in which the cipher alphabet for the plain alphabet may be different at different places during the encryption process. The next two examples, **playfair and Vigenere Cipher are polyalphabetic ciphers**.

Playfair Cipher

In this scheme, pairs of letters are encrypted, instead of single letters as in the case of simple substitution cipher.

In playfair cipher, initially a key table is created. The key table is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table as we need only 25 alphabets instead of 26. If the plaintext contains J, then it is replaced by I.

The sender and the receiver deicide on a particular key, say ‘tutorials’. In a key table, the first characters (going left to right) in the table is the phrase, excluding the duplicate letters. The rest of the table will be filled with the remaining letters of the alphabet, in natural order. The key table works out to be −



T	U	O	R	I
A	L	S	B	C
D	E	F	G	H
K	M	N	P	Q
V	W	X	Y	Z

Process of Playfair Cipher

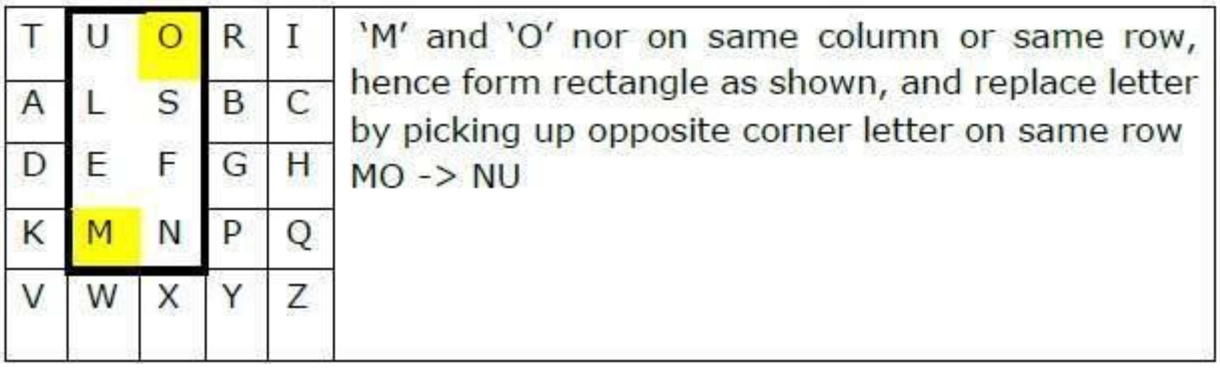
- First, a plaintext message is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter. Let us say we want to encrypt the message “hide money”. It will be written as −
  HI DE MO NE YZ
- The rules of encryption are −
  - If both the letters are in the same column, take the letter below each one (going back to the top if at the bottom)

T	U	O	R	I	
A	L	S	B	C	
D	E	F	G	H	‘H’ and ‘I’ are in same column, hence take letter below them to replace. HI → QC
K	M	N	P	Q	
V	W	X	Y	Z	

- If both letters are in the same row, take the letter to the right of each one (going back to the left if at the farthest right)

T	U	O	R	I	
A	L	S	B	C	
D	E	F	G	H	‘D’ and ‘E’ are in same row, hence take letter to the right of them to replace. DE → EF
K	M	N	P	Q	
V	W	X	Y	Z	

- If neither of the preceding two rules are true, form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.



T	U	O	R	I	‘M’ and ‘O’ nor on same column or same row, hence form rectangle as shown, and replace letter by picking up opposite corner letter on same row MO -> NU
A	L	S	B	C	
D	E	F	G	H	
K	M	N	P	Q	
V	W	X	Y	Z	

Using these rules, the result of the encryption of ‘hide money’ with the key of ‘tutorials’ would be −
QC EF NU MF ZV
Decrypting the Playfair cipher is as simple as doing the same process in reverse. Receiver has the same key and can create the same key table, and then decrypt any messages made using that key.
Security Value
It is also a substitution cipher and is difficult to break compared to the simple substitution cipher. As in case of substitution cipher, cryptanalysis is possible on the Playfair cipher as well, however it would be against 625 possible pairs of letters (25x25 alphabets) instead of 26 different possible alphabets.

The Playfair cipher was used mainly to protect important, yet non-critical secrets, as it is quick to use and requires no special equipment.

Vigenere Cipher

This scheme of cipher uses a text string (say, a word) as a key, which is then used for doing a number of shifts on the plaintext.

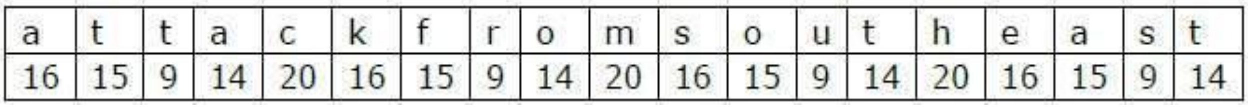
For example, let's assume the key is 'point'. Each alphabet of the key is converted to its respective numeric value: In this case,

p → 16, o → 15, i → 9, n → 14, and t → 20.

Thus, the key is: 16 15 9 14 20.

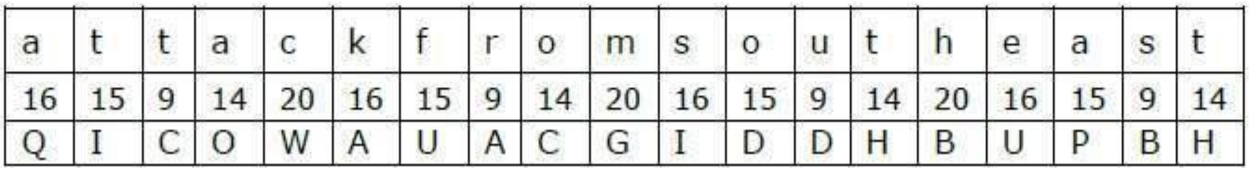
Process of Vigenere Cipher

- The sender and the receiver decide on a key. Say 'point' is the key. Numeric representation of this key is '16 15 9 14 20'.
- The sender wants to encrypt the message, say 'attack from south east'. He will arrange plaintext and numeric key as follows −



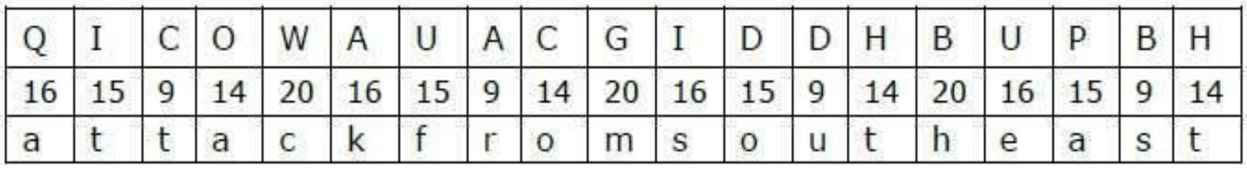
a	t	t	a	c	k	f	r	o	m	s	o	u	t	h	e	a	s	t
16	15	9	14	20	16	15	9	14	20	16	15	9	14	20	16	15	9	14

- He now shifts each plaintext alphabet by the number written below it to create ciphertext as shown below −



a	t	t	a	c	k	f	r	o	m	s	o	u	t	h	e	a	s	t
16	15	9	14	20	16	15	9	14	20	16	15	9	14	20	16	15	9	14
Q	I	C	O	W	A	U	A	C	G	I	D	D	H	B	U	P	B	H

- Here, each plaintext character has been shifted by a different amount – and that amount is determined by the key. The key must be less than or equal to the size of the message.
- For decryption, the receiver uses the same key and shifts received ciphertext in reverse order to obtain the plaintext.



Q	I	C	O	W	A	U	A	C	G	I	D	D	H	B	U	P	B	H
16	15	9	14	20	16	15	9	14	20	16	15	9	14	20	16	15	9	14
a	t	t	a	c	k	f	r	o	m	s	o	u	t	h	e	a	s	t

Security Value

Vigenere Cipher was designed by tweaking the standard Caesar cipher to reduce the effectiveness of cryptanalysis on the ciphertext and make a cryptosystem more robust. It is significantly **more secure than a regular Caesar Cipher**.

In the history, it was regularly used for protecting sensitive political and military information. It was referred to as the **unbreakable cipher** due to the difficulty it posed to the cryptanalysis.

Variants of Vigenere Cipher

There are two special cases of Vigenere cipher −

- The keyword length is same as plaintect message. This case is called **Vernam Cipher**. It is more secure than typical Vigenere cipher.
- Vigenere cipher becomes a cryptosystem with perfect secrecy, which is called **One-time pad**.

One-Time Pad

The circumstances are −

- The length of the keyword is same as the length of the plaintext.
- The keyword is a randomly generated string of alphabets.
- The keyword is used only once.

Security Value

Let us compare Shift cipher with one-time pad.

Shift Cipher − Easy to Break

In case of Shift cipher, the entire message could have had a shift between 1 and 25. This is a very small size, and very easy to brute force. However, with each character now having its own individual shift between 1 and 26, the possible keys grow exponentially for the message.

One-time Pad − Impossible to Break

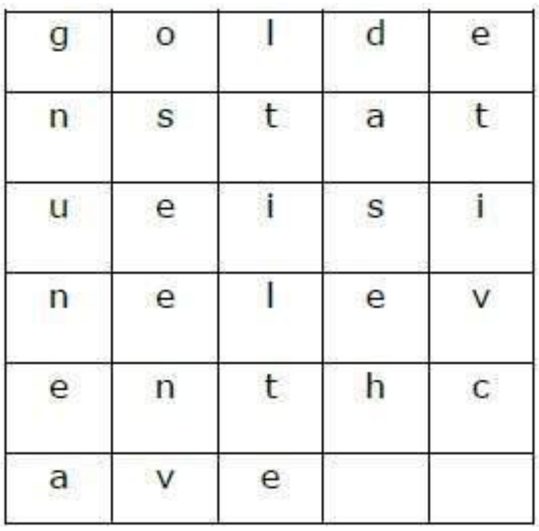
Let us say, we encrypt the name "point" with a one-time pad. It is a 5 letter text. To break the ciphertext by brute force, you need to try all possibilities of keys and conduct computation for (26 x 26 x 26 x 26 x 26) = 26^5 = 11881376 times. That's for a message with 5 alphabets. Thus, for a longer message, the computation grows exponentially with every additional alphabet. This makes it computationally impossible to break the ciphertext by brute force.

Transposition Cipher

It is another type of cipher where the order of the alphabets in the plaintext is rearranged to create the ciphertext. The actual plaintext alphabets are not replaced.

An example is a 'simple columnar transposition' cipher where the plaintext is written horizontally with a certain alphabet width. Then the ciphertext is read vertically as shown.

For example, the plaintext is "golden statue is in eleventh cave" and the secret random key chosen is "five". We arrange this text horizontally in table with number of column equal to key value. The resulting text is shown below.



g	o	l	d	e
n	s	t	a	t
u	e	i	s	i
n	e	l	e	v
e	n	t	h	c
a	v	e		

The ciphertext is obtained by reading column vertically downward from first to last column. The ciphertext is 'gnuneaoseenvltiltedasehetivc'.

To decrypt, the receiver prepares similar table. The number of columns is equal to key number. The number of rows is obtained by dividing number of total ciphertext alphabets by key value and rounding of the quotient to next integer value.
The receiver then writes the received ciphertext vertically down and from left to right column. To obtain the text, he reads horizontally left to right and from top to bottom row.

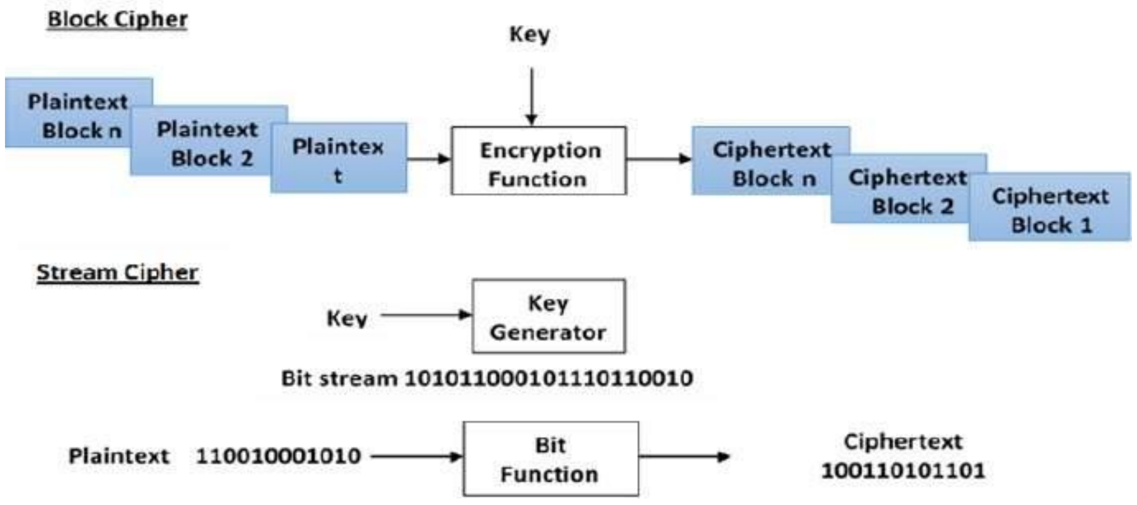
Digital data is represented in strings of binary digits (bits) unlike alphabets. Modern cryptosystems need to process this binary strings to convert in to another binary string. Based on how these binary strings are processed, a symmetric encryption schemes can be classified in to −

Block Ciphers

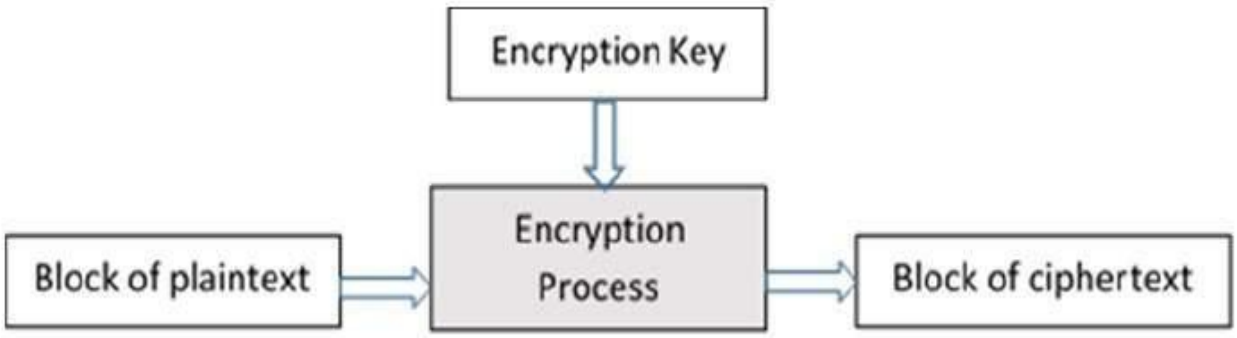
In this scheme, the plain binary text is processed in blocks (groups) of bits at a time; i.e. a block of plaintext bits is selected, a series of operations is performed on this block to generate a block of ciphertext bits. The number of bits in a block is fixed. For example, the schemes DES and AES have block sizes of 64 and 128, respectively.

Stream Ciphers

In this scheme, the plaintext is processed one bit at a time i.e. one bit of plaintext is taken, and a series of operations is performed on it to generate one bit of ciphertext. Technically, stream ciphers are block ciphers with a block size of one bit.



The basic scheme of a block cipher is depicted as follows −



A block cipher takes a block of plaintext bits and generates a block of ciphertext bits, generally of same size. The size of block is fixed in the given scheme. The choice of block size does not directly affect to the strength of encryption scheme. The strength of cipher depends up on the key length.

Block Size

Though any size of block is acceptable, following aspects are borne in mind while selecting a size of a block.

- **Avoid very small block size** − Say a block size is m bits. Then the possible plaintext bits combinations are then 2^m. If the attacker discovers the plain text blocks corresponding to some previously sent ciphertext blocks, then the attacker can launch a type of 'dictionary attack' by building up a dictionary of plaintext/ciphertext pairs sent using that encryption key. A larger block size makes attack harder as the dictionary needs to be larger.
- **Do not have very large block size** − With very large block size, the cipher becomes inefficient to operate. Such plaintexts will need to be padded before being encrypted.
- **Multiples of 8 bit** − A preferred block size is a multiple of 8 as it is easy for implementation as most computer processor handle data in multiple of 8 bits.

Padding in Block Cipher

Block ciphers process blocks of fixed sizes (say 64 bits). The length of plaintexts is mostly not a multiple of the block size. For example, a 150-bit plaintext provides two blocks of 64 bits each with third block of balance 22 bits. The last block of bits needs to be padded up with redundant information so that the length of the final block equal to block size of the scheme. In our example, the remaining 22 bits need to have additional 42 redundant bits added to provide a complete block. The process of adding bits to the last block is referred to as **padding**.

Too much padding makes the system inefficient. Also, padding may render the system insecure at times, if the padding is done with same bits always.

Block Cipher Schemes

There is a vast number of block ciphers schemes that are in use. Many of them are publically known. Most popular and prominent block ciphers are listed below.

- **Digital Encryption Standard (DES)** − The popular block cipher of the 1990s. It is now considered as a 'broken' block cipher, due primarily to its small key size.
- **Triple DES** − It is a variant scheme based on repeated DES applications. It is still a respected block ciphers but inefficient compared to the new faster block ciphers available.
- **Advanced Encryption Standard (AES)** − It is a relatively new block cipher based on the encryption algorithm **Rijndael** that won the AES design competition.
- **IDEA** − It is a sufficiently strong block cipher with a block size of 64 and a key size of 128 bits. A number of applications use IDEA encryption, including early versions of

Pretty Good Privacy (PGP) protocol. The use of IDEA scheme has a restricted adoption due to patent issues.

- **Twofish** − This scheme of block cipher uses block size of 128 bits and a key of variable length. It was one of the AES finalists. It is based on the earlier block cipher Blowfish with a block size of 64 bits.
- **Serpent** − A block cipher with a block size of 128 bits and key lengths of 128, 192, or 256 bits, which was also an AES competition finalist. It is a slower but has more secure design than other block cipher.

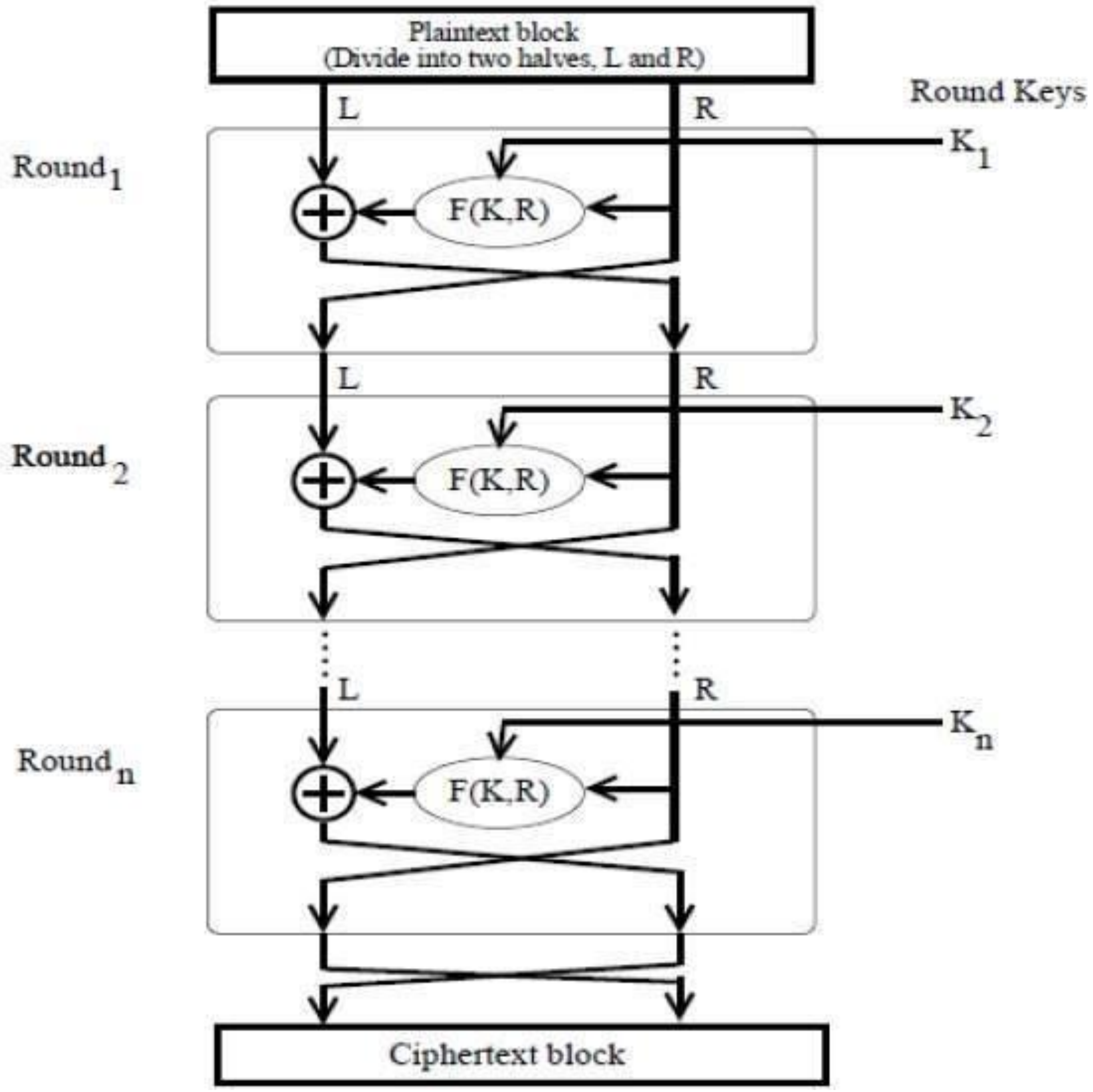
In the next sections, we will first discuss the model of block cipher followed by DES and AES, two of the most influential modern block ciphers.

Feistel Cipher is not a specific scheme of block cipher. It is a design model from which many different block ciphers are derived. DES is just one example of a Feistel Cipher. A cryptographic system based on Feistel cipher structure uses the same algorithm for both encryption and decryption.

Encryption Process

The encryption process uses the Feistel structure consisting multiple rounds of processing of the plaintext, each round consisting of a "substitution" step followed by a permutation step.

Feistel Structure is shown in the following illustration −



- The input block to each round is divided into two halves that can be denoted as L and R for the left half and the right half.
- In each round, the right half of the block, R, goes through unchanged. But the left half, L, goes through an operation that depends on R and the encryption key. First, we apply an encrypting function ‘f’ that takes two input − the key K and R. The function produces the output f(R,K). Then, we XOR the output of the mathematical function with L.
- In real implementation of the Feistel Cipher, such as DES, instead of using the whole encryption key during each round, a round-dependent key (a subkey) is derived from the encryption key. This means that each round uses a different key, although all these subkeys are related to the original key.
- The permutation step at the end of each round swaps the modified L and unmodified R. Therefore, the L for the next round would be R of the current round. And R for the next round be the output L of the current round.
- Above substitution and permutation steps form a ‘round’. The number of rounds are specified by the algorithm design.

- Once the last round is completed then the two sub blocks, 'R' and 'L' are concatenated in this order to form the ciphertext block.

The difficult part of designing a Feistel Cipher is selection of round function 'f'. In order to be unbreakable scheme, this function needs to have several important properties that are beyond the scope of our discussion.

Decryption Process

The process of decryption in Feistel cipher is almost similar. Instead of starting with a block of plaintext, the ciphertext block is fed into the start of the Feistel structure and then the process thereafter is exactly the same as described in the given illustration.

The process is said to be almost similar and not exactly same. In the case of decryption, the only difference is that the subkeys used in encryption are used in the reverse order.

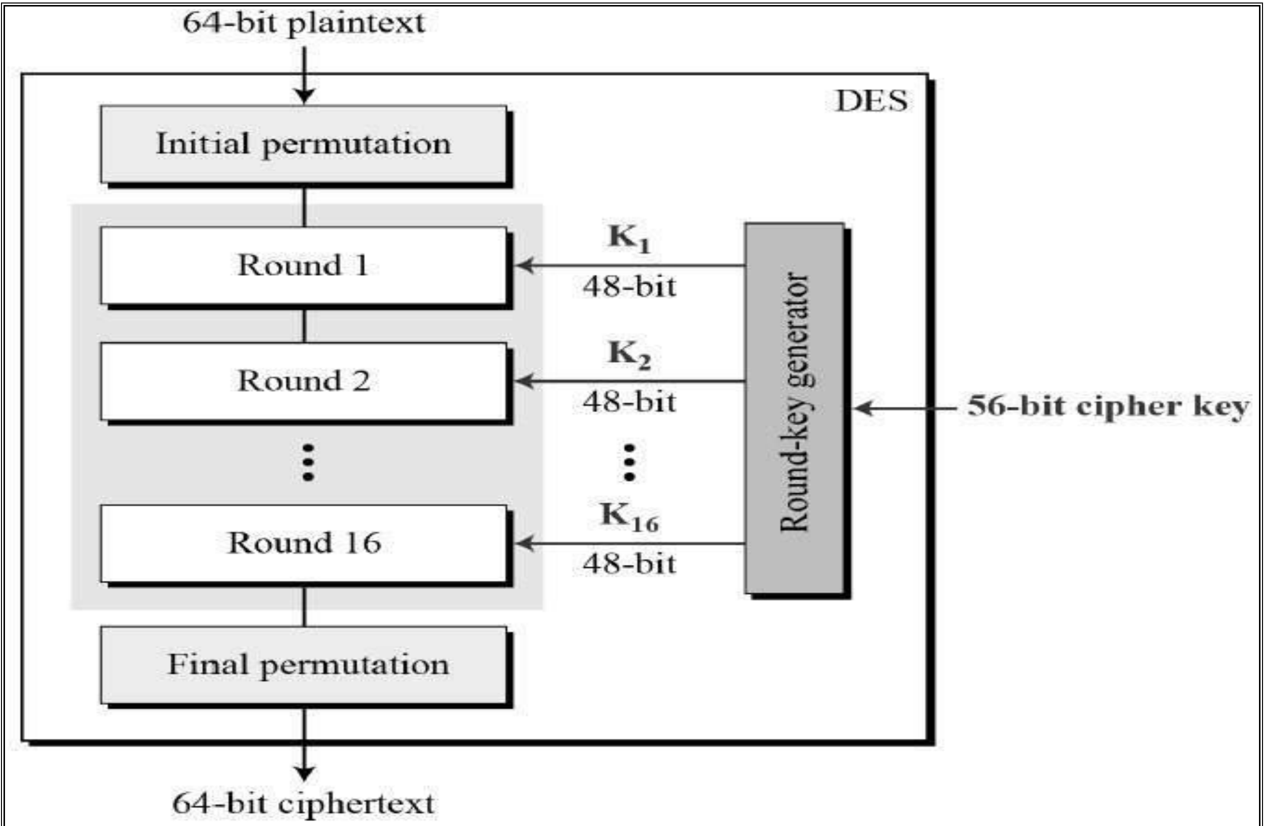
The final swapping of 'L' and 'R' in last step of the Feistel Cipher is essential. If these are not swapped then the resulting ciphertext could not be decrypted using the same algorithm.

Number of Rounds

The number of rounds used in a Feistel Cipher depends on desired security from the system. More number of rounds provide more secure system. But at the same time, more rounds mean the inefficient slow encryption and decryption processes. Number of rounds in the systems thus depend upon efficiency–security tradeoff.

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). General Structure of DES is depicted in the following illustration −

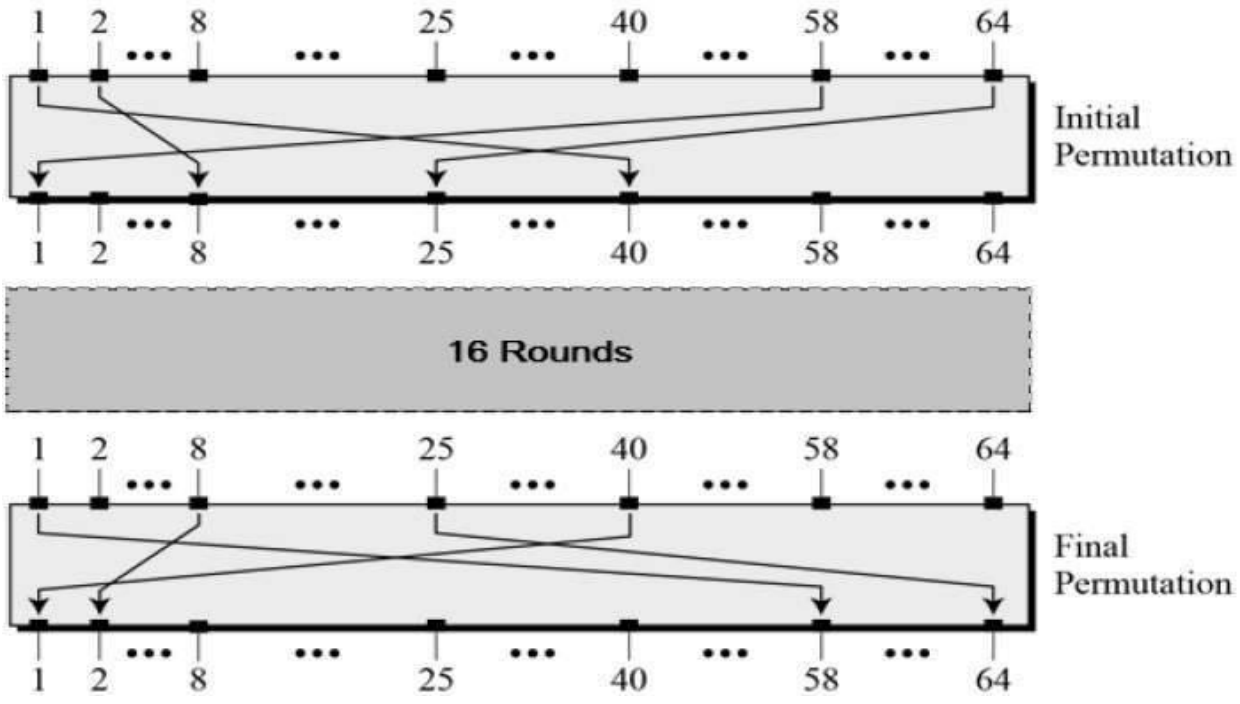


Since DES is based on the Feistel Cipher, all that is required to specify DES is −

- Round function
- Key schedule
- Any additional processing − Initial and final permutation

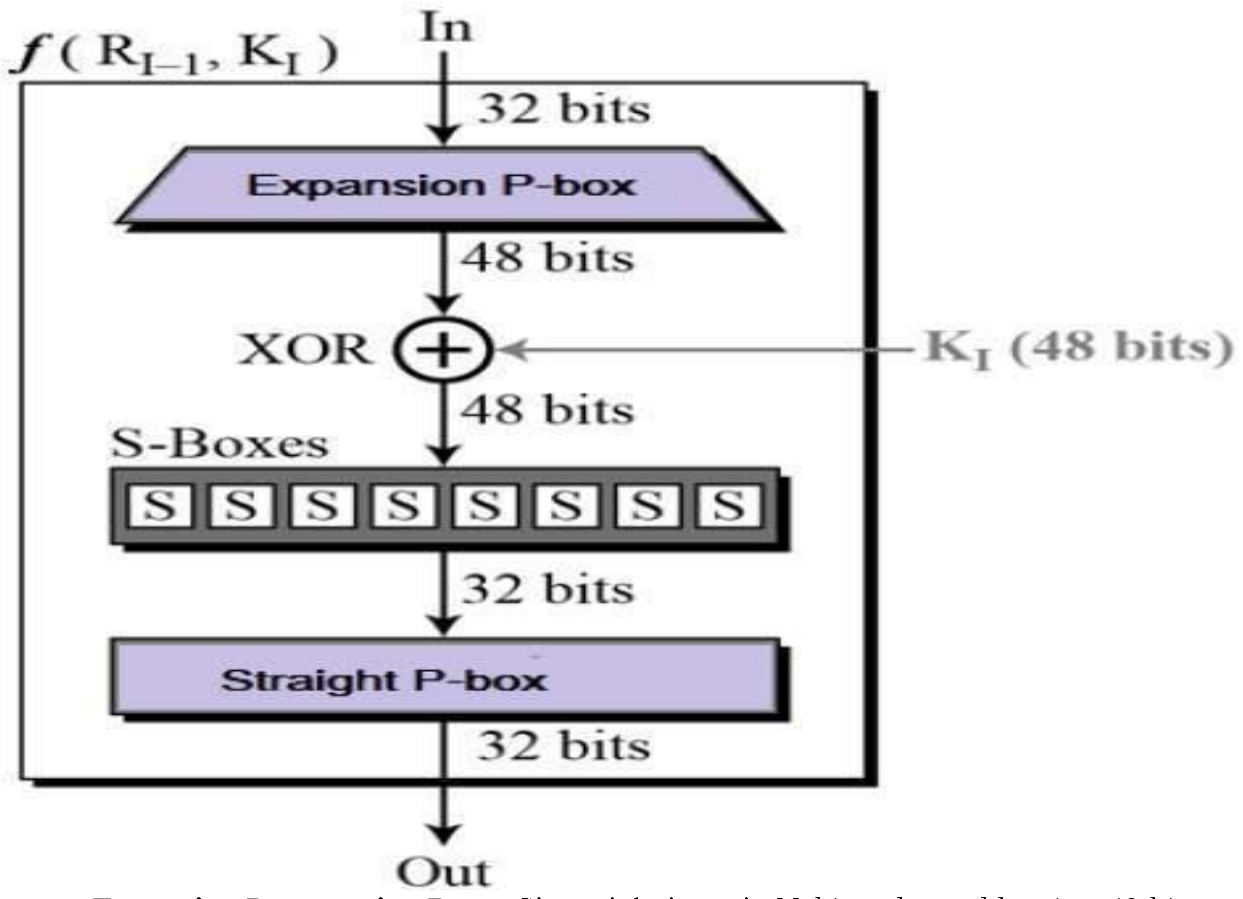
Initial and Final Permutation

The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES. The initial and final permutations are shown as follows −

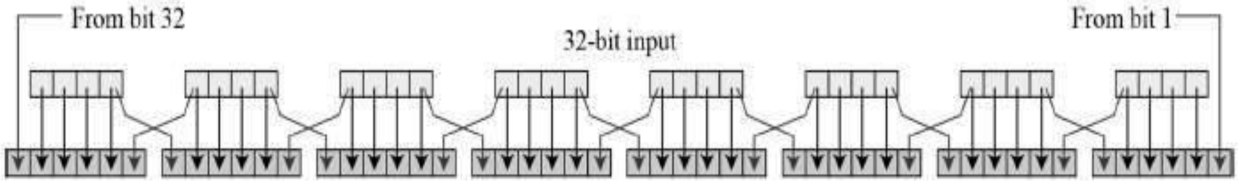


Round Function

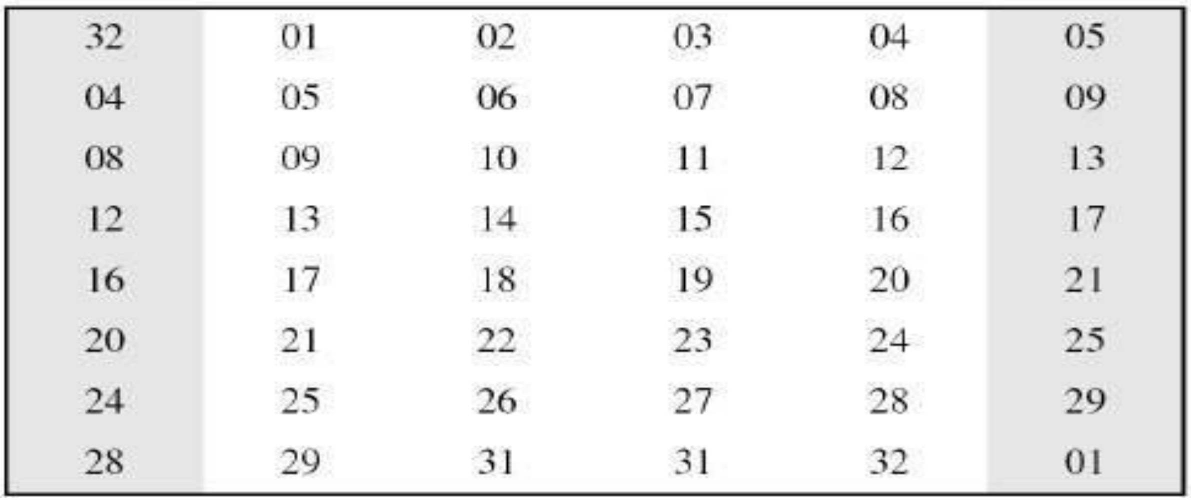
The heart of this cipher is the DES function, *f*. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.



- **Expansion Permutation Box** − Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the following illustration −

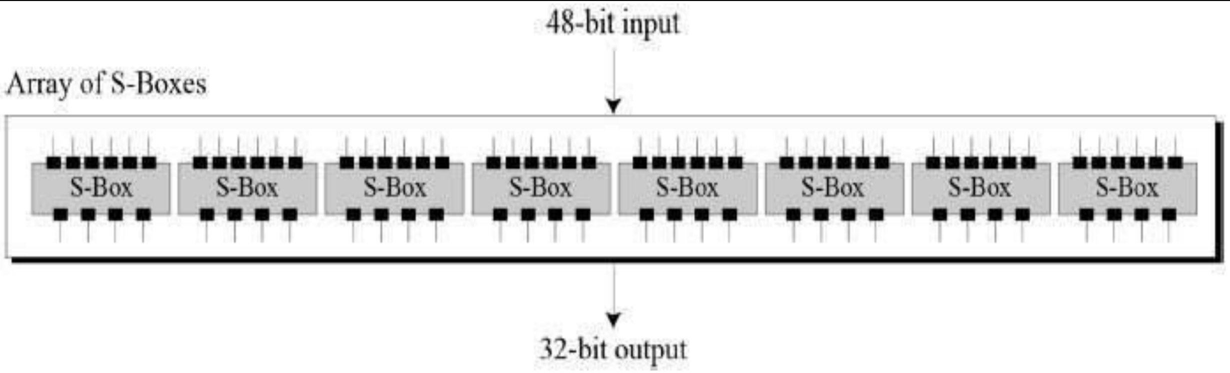


- The graphically depicted permutation logic is generally described as table in DES specification illustrated as shown −

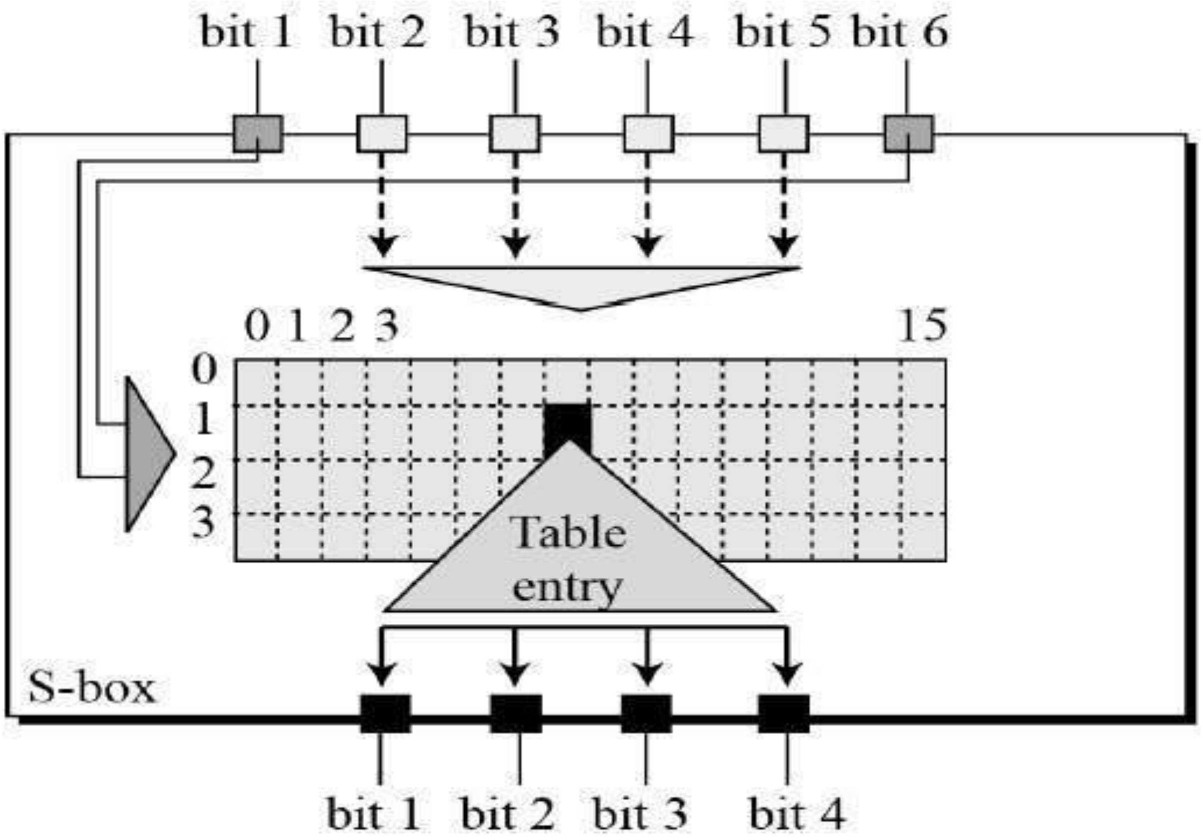


32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

- **XOR (Whitener).** − After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.
- **Substitution Boxes.** − The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration −



- The S-box rule is illustrated below −

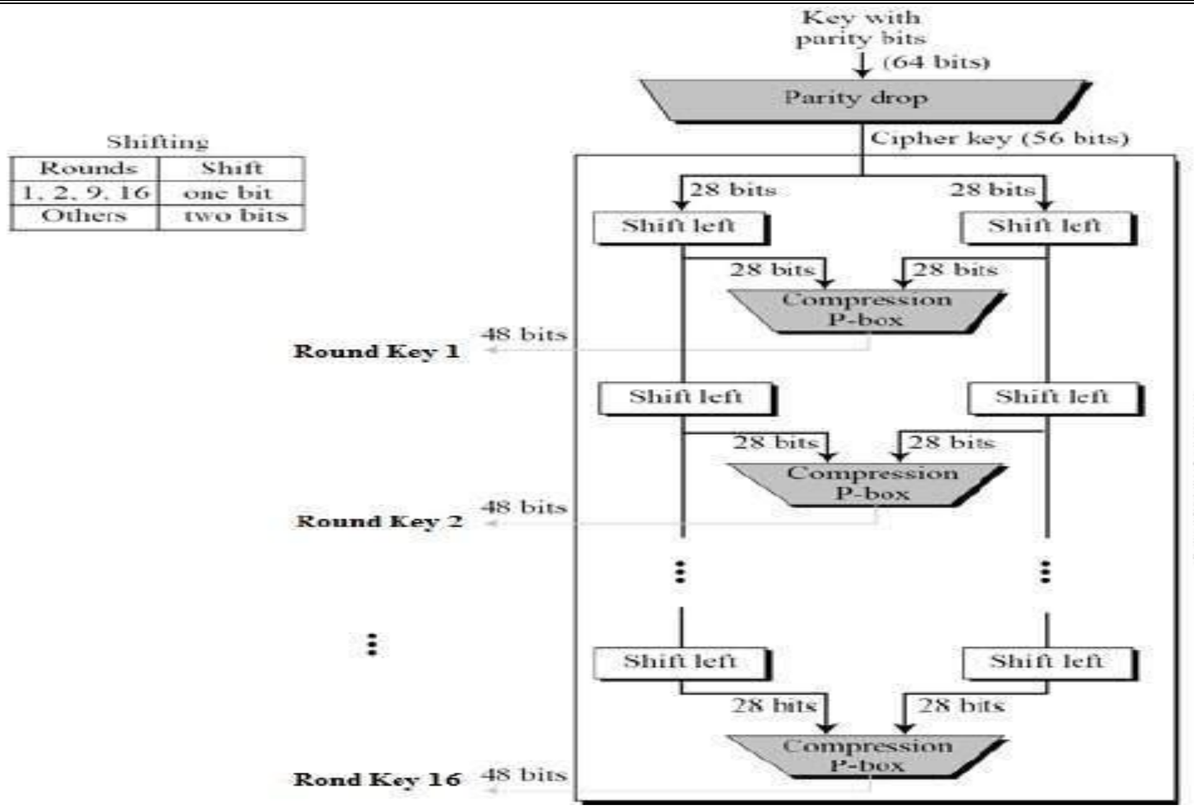


- There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32 bit section.
- **Straight Permutation** − The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Key Generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration −



The logic for Parity drop, shifting, and Compression P-box is given in the DES description.

DES Analysis

The DES satisfies both the desired properties of block cipher. These two properties make cipher very strong.

- **Avalanche effect** − A small change in plaintext results in the very great change in the ciphertext.
- **Completeness** − Each bit of ciphertext depends on many bits of plaintext.

During the last few years, cryptanalysis have found some weaknesses in DES when key selected are weak keys. These keys shall be avoided.

DES has proved to be a very well designed block cipher. There have been no significant cryptanalytic attacks on DES other than exhaustive key search.

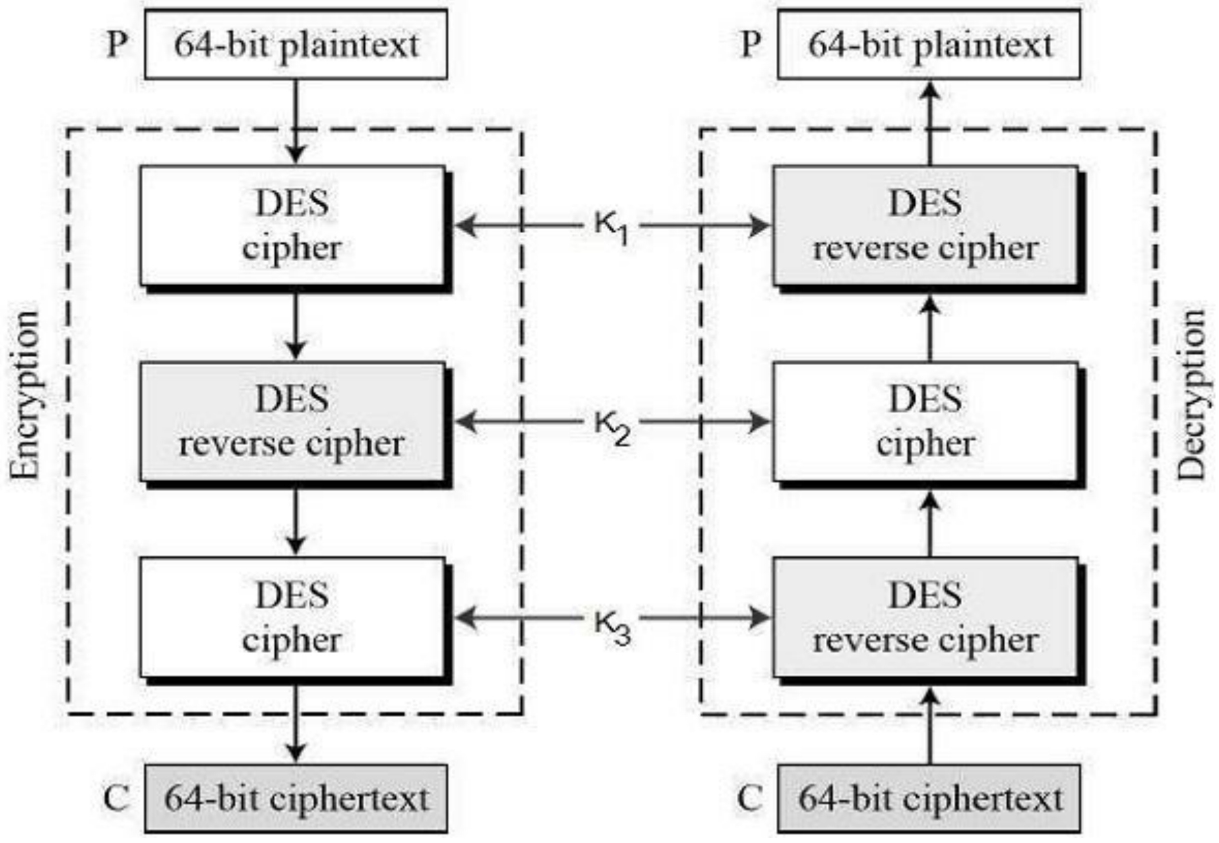
The speed of exhaustive key searches against DES after 1990 began to cause discomfort amongst users of DES. However, users did not want to replace DES as it takes an enormous amount of time and money to change encryption algorithms that are widely adopted and embedded in large security architectures.

The pragmatic approach was not to abandon the DES completely, but to change the manner in which DES is used. This led to the modified schemes of Triple DES (sometimes known as 3DES).

Incidentally, there are two variants of Triple DES known as 3-key Triple DES (3TDES) and 2-key Triple DES (2TDES).

3-KEY Triple DES

Before using 3TDES, user first generate and distribute a 3TDES key K, which consists of three different DES keys K_1, K_2 and K_3. This means that the actual 3TDES key has length 3×56 = 168 bits. The encryption scheme is illustrated as follows −



The encryption-decryption process is as follows −

- Encrypt the plaintext blocks using single DES with key K_1.
- Now decrypt the output of step 1 using single DES with key K_2.
- Finally, encrypt the output of step 2 using single DES with key K_3.
- The output of step 3 is the ciphertext.
- Decryption of a ciphertext is a reverse process. User first decrypt using K_3, then encrypt with K_2, and finally decrypt with K_1.

Due to this design of Triple DES as an encrypt–decrypt–encrypt process, it is possible to use a 3TDES (hardware) implementation for single DES by setting K_1, K_2, and K_3 to be the same value. This provides backwards compatibility with DES.
Second variant of Triple DES (2TDES) is identical to 3TDES except that $K_3$is replaced by K_1. In other words, user encrypt plaintext blocks with key K_1, then decrypt with key K_2, and finally encrypt with K_1 again. Therefore, 2TDES has a key length of 112 bits.
Triple DES systems are significantly more secure than single DES, but these are clearly a much slower process than encryption using single DES.

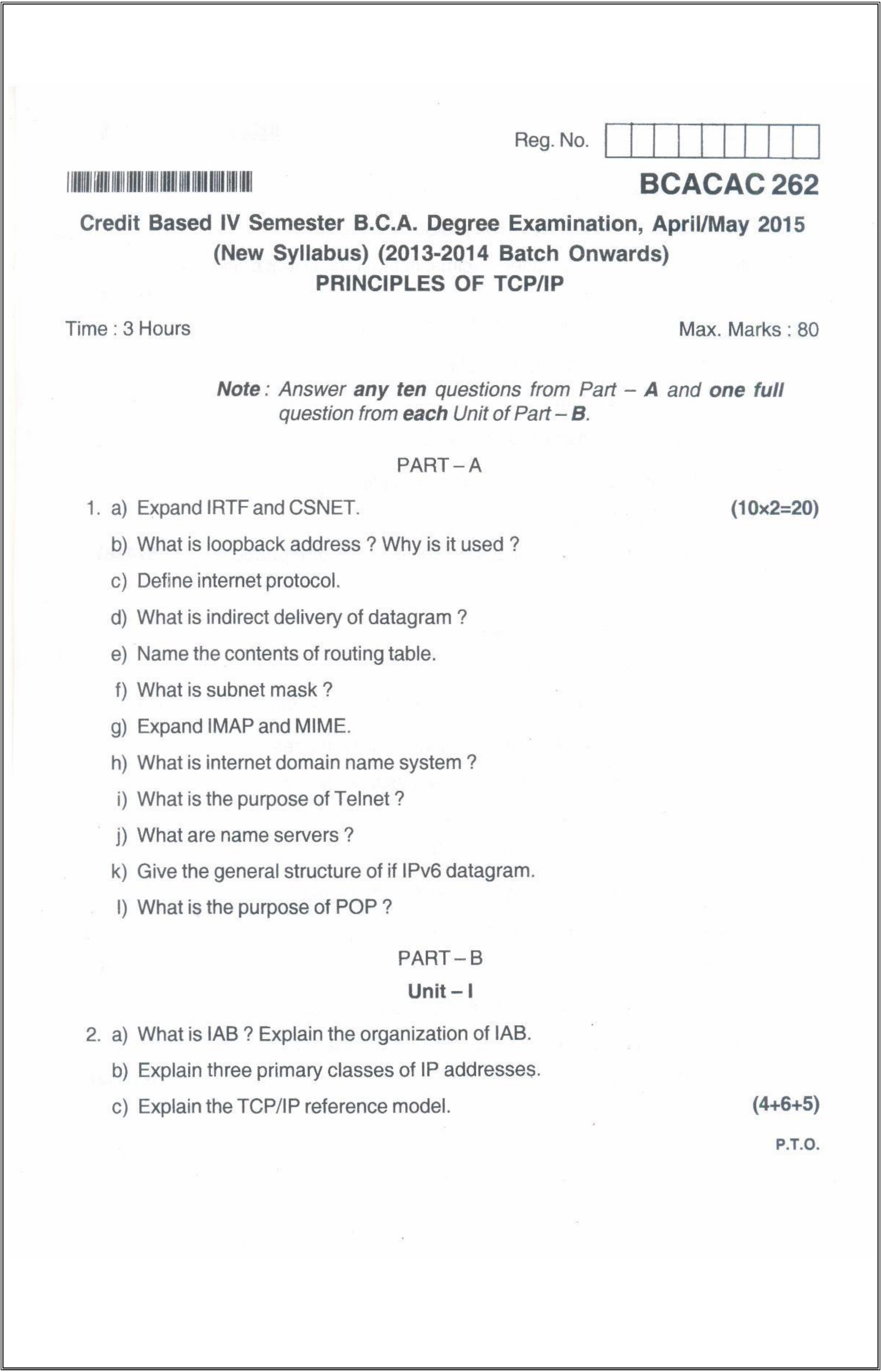
14.6 Assignment Questions

Answer the following questions each carries 2 marks.

1. What do you mean by internet security?
2. What are firewalls?

Essay type questions.

1. List and explain the various resources which need protection.
2. What is the information policy?
3. What are firewalls? Explain .



Reg. No. [| | | | | | | |]

BCACAC 262

Credit Based IV Semester B.C.A. Degree Examination, April/May 2015
(New Syllabus) (2013-2014 Batch Onwards)
PRINCIPLES OF TCP/IP

Time : 3 Hours Max. Marks : 80

***Note** : Answer **any ten** questions from Part – **A** and **one full** question from **each** Unit of Part – **B**.*

PART – A

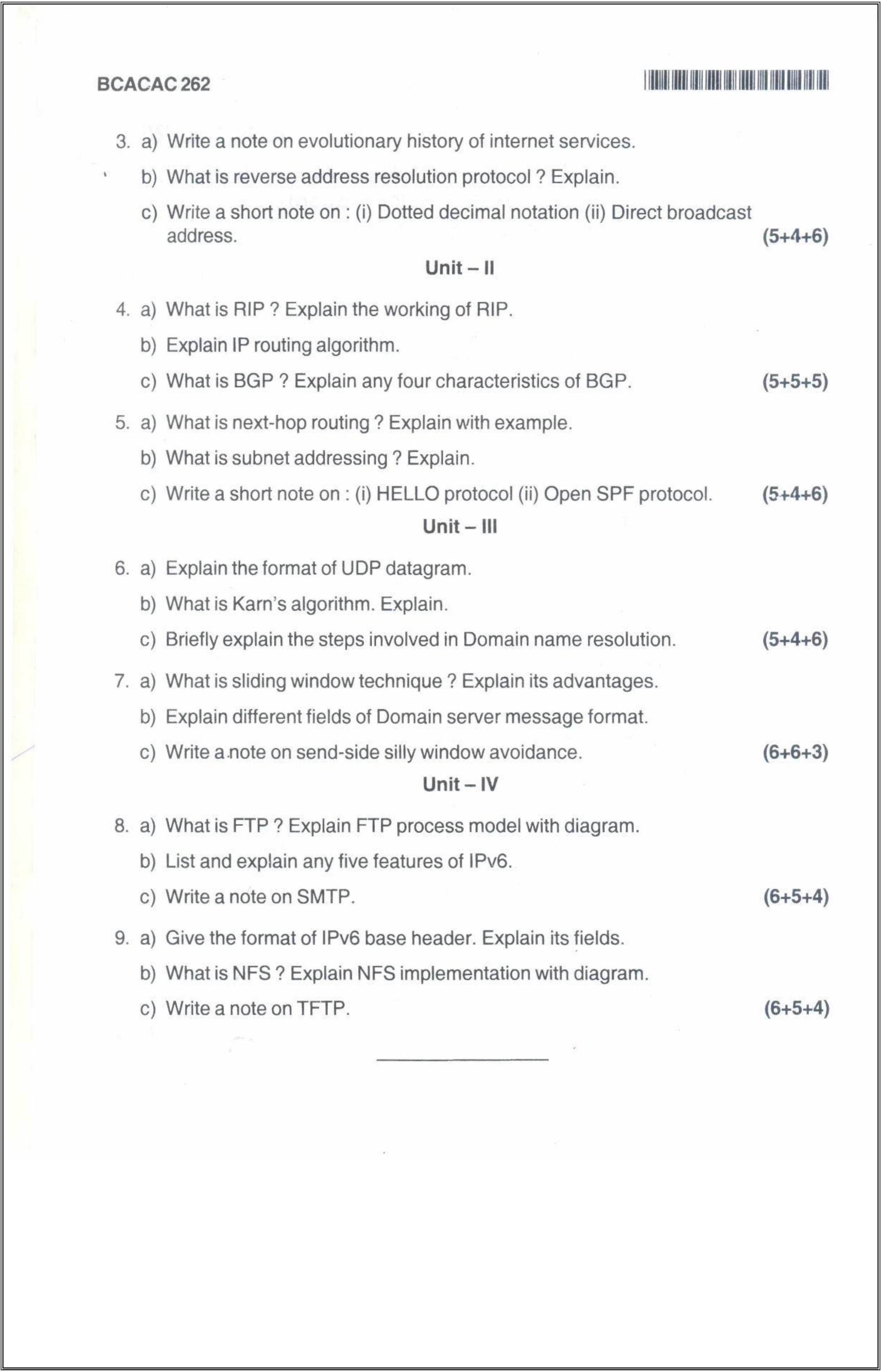
1. a) Expand IRTF and CSNET. **(10×2=20)**
   b) What is loopback address ? Why is it used ?
   c) Define internet protocol.
   d) What is indirect delivery of datagram ?
   e) Name the contents of routing table.
   f) What is subnet mask ?
   g) Expand IMAP and MIME.
   h) What is internet domain name system ?
   i) What is the purpose of Telnet ?
   j) What are name servers ?
   k) Give the general structure of if IPv6 datagram.
   l) What is the purpose of POP ?

PART – B

Unit – I

2. a) What is IAB ? Explain the organization of IAB.
   b) Explain three primary classes of IP addresses.
   c) Explain the TCP/IP reference model. **(4+6+5)**

P.T.O.



BCACAC 262

3. a) Write a note on evolutionary history of internet services.

   b) What is reverse address resolution protocol ? Explain.

   c) Write a short note on : (i) Dotted decimal notation (ii) Direct broadcast address. **(5+4+6)**

Unit – II

4. a) What is RIP ? Explain the working of RIP.

   b) Explain IP routing algorithm.

   c) What is BGP ? Explain any four characteristics of BGP. **(5+5+5)**

5. a) What is next-hop routing ? Explain with example.

   b) What is subnet addressing ? Explain.

   c) Write a short note on : (i) HELLO protocol (ii) Open SPF protocol. **(5+4+6)**

Unit – III

6. a) Explain the format of UDP datagram.

   b) What is Karn's algorithm. Explain.

   c) Briefly explain the steps involved in Domain name resolution. **(5+4+6)**

7. a) What is sliding window technique ? Explain its advantages.

   b) Explain different fields of Domain server message format.

   c) Write a note on send-side silly window avoidance. **(6+6+3)**

Unit – IV

8. a) What is FTP ? Explain FTP process model with diagram.

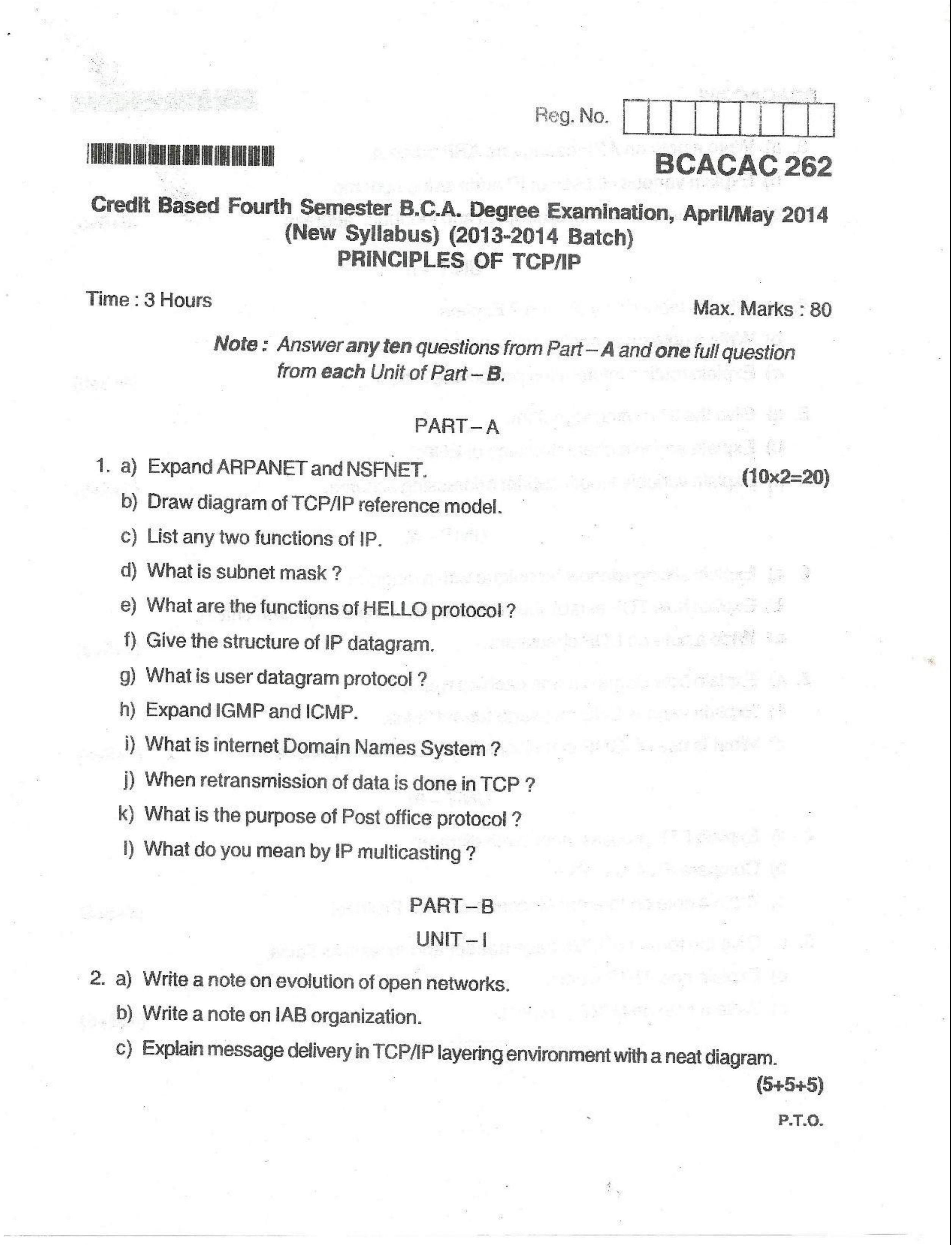
   b) List and explain any five features of IPv6.

   c) Write a note on SMTP. **(6+5+4)**

9. a) Give the format of IPv6 base header. Explain its fields.

   b) What is NFS ? Explain NFS implementation with diagram.

   c) Write a note on TFTP. **(6+5+4)**



Reg. No.

BCACAC 262

Credit Based Fourth Semester B.C.A. Degree Examination, April/May 2014
(New Syllabus) (2013-2014 Batch)
PRINCIPLES OF TCP/IP

Time : 3 Hours Max. Marks : 80

Note : *Answer* ***any ten*** *questions from Part –* ***A*** *and* ***one*** *full question from* ***each*** *Unit of Part –* ***B.***

PART – A

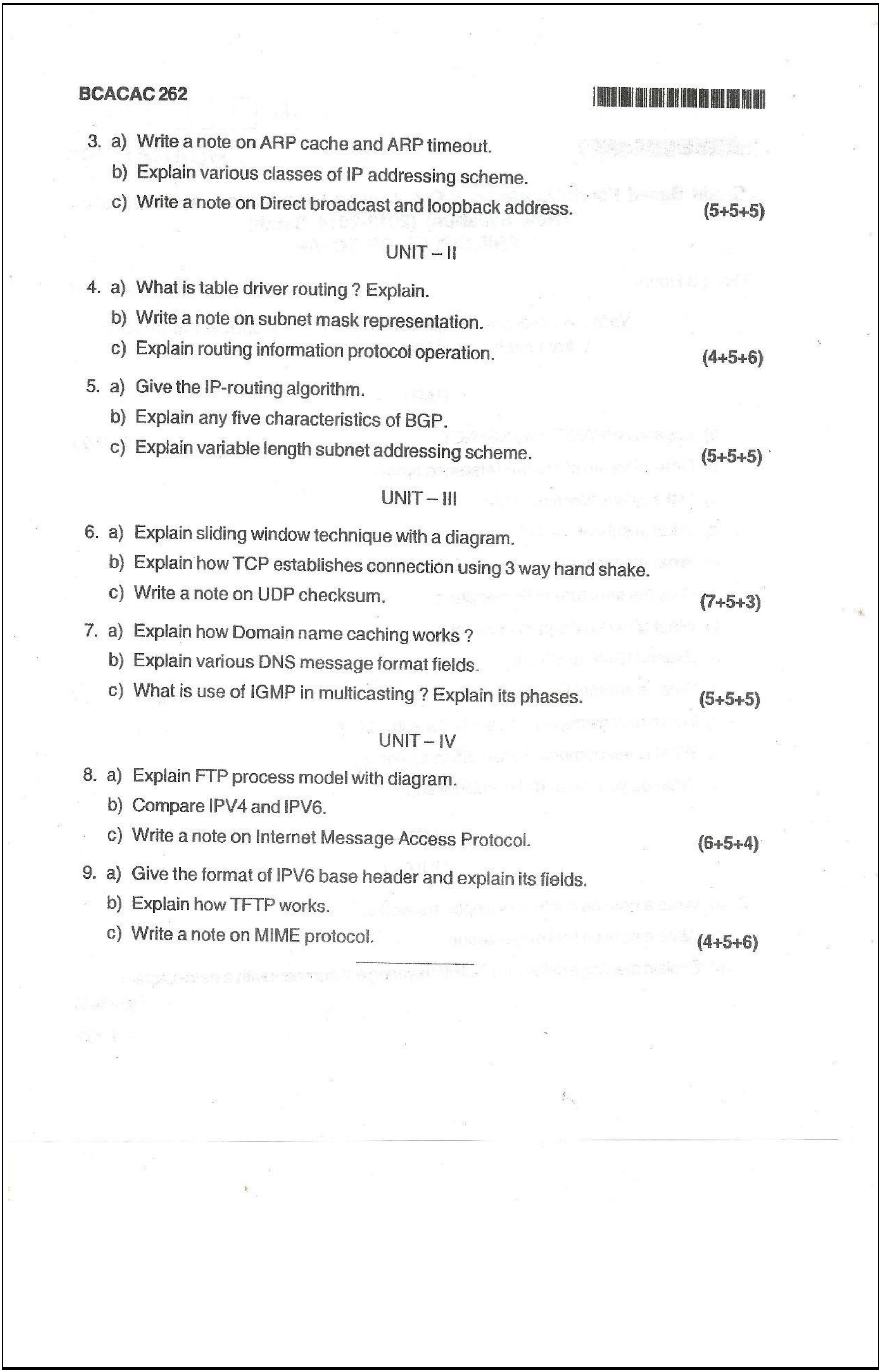
1. a) Expand ARPANET and NSFNET. **(10×2=20)**
   b) Draw diagram of TCP/IP reference model.
   c) List any two functions of IP.
   d) What is subnet mask ?
   e) What are the functions of HELLO protocol ?
   f) Give the structure of IP datagram.
   g) What is user datagram protocol ?
   h) Expand IGMP and ICMP.
   i) What is internet Domain Names System ?
   j) When retransmission of data is done in TCP ?
   k) What is the purpose of Post office protocol ?
   l) What do you mean by IP multicasting ?

PART – B

UNIT – I

2. a) Write a note on evolution of open networks.
   b) Write a note on IAB organization.
   c) Explain message delivery in TCP/IP layering environment with a neat diagram.
   (5+5+5)

P.T.O.



BCACAC 262

3. a) Write a note on ARP cache and ARP timeout.
   b) Explain various classes of IP addressing scheme.
   c) Write a note on Direct broadcast and loopback address. **(5+5+5)**

UNIT – II

4. a) What is table driver routing ? Explain.
   b) Write a note on subnet mask representation.
   c) Explain routing information protocol operation. **(4+5+6)**

5. a) Give the IP-routing algorithm.
   b) Explain any five characteristics of BGP.
   c) Explain variable length subnet addressing scheme. **(5+5+5)**

UNIT – III

6. a) Explain sliding window technique with a diagram.
   b) Explain how TCP establishes connection using 3 way hand shake.
   c) Write a note on UDP checksum. **(7+5+3)**

7. a) Explain how Domain name caching works ?
   b) Explain various DNS message format fields.
   c) What is use of IGMP in multicasting ? Explain its phases. **(5+5+5)**

UNIT – IV

8. a) Explain FTP process model with diagram.
   b) Compare IPV4 and IPV6.
   c) Write a note on Internet Message Access Protocol. **(6+5+4)**

9. a) Give the format of IPV6 base header and explain its fields.
   b) Explain how TFTP works.
   c) Write a note on MIME protocol. **(4+5+6)**