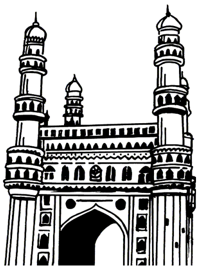
Rahul's ✔
Topper's Voice

NEW SYLLABUS



B.C.A.

II Year IV Sem

Latest 2023 Edition

COMPUTER NETWORKS

- ☞ **Study Manual**
- ☞ **Important Questions**
- ☞ **Lab Programming**
- ☞ **Short Question & Answers**
- ☞ **Choose the correct Answers**
- ☞ **Fill in the blanks**
- ☞ **Solved Model Papers**
- ☞ **Previous Question Paper**

` .199/-

- by -

WELL EXPERIENCED LECTURER

Rahul Publications™
Hyderabad. Cell : 9391018098, 9505799122

All disputes are subjects to Hyderabad Jurisdiction only

B.C.A.

II Year IV Sem

COMPUTER NETWORKS

Inspite of many efforts taken to present this book without errors, some errors might have crept in. Therefore we do not take any legal responsibility for such errors and omissions. However, if they are brought to our notice, they will be corrected in the next edition.

Price ` 199

Sole Distributors : **Cell : 9391018098, 9505799122**

VASU BOOK CENTRE

Shop No. 2, Beside Gokul Chat, Koti, Hyderabad.

Maternity Hospital Opp. Lane, Narayan Naik Complex, Koti, Hyderabad.
Near Andhra Bank, Subway, Sultan Bazar, Koti, Hyderabad -195.

CONTENTS

COMPUTER NETWORKS

STUDY MANUAL

SOLVED MODEL PAPERS

PREVIOUS QUESTION PAPER

SYLLABUS

UNIT - I

Multiple Access: Wired LAN-Ethernet IEEE 802.3LAN, CSMA/CD protocol, Binary exponential backoff algorithm. Comparison of Switched , Fast and Gigabit Ethernet.

Wireless LAN

IEEE 802.11 architecture. CSMA/CA protocol, Bridges and types of bridges. ARP and RARP

UNIT - II

Network Layer: Logical Addressing-IPv4, Subnetting, and supernetting, CIDR, introduction to IPv6 ICMP, IGMP. Routing-Distance Vector Routing Link State Routing, OSPF and BGP.

UNIT - III

Transport Layer: TCP State diagram, Window Management, Congestion Control, Timer Management and UDP protocol

UNIT - IV

Socket Programming

Primitive and Advanced Systems Calls, TCP Iterative and Concurrent programs, UDP systems calls, socket options IO Multiplexing, Asynchronous IO

UNIT - V

Application Layer: Domain Name System, Simple Mail Transfer Protocol (SMTP),

File Transfer Protocol (FTP), Hyper Text Transfer Protocol (HTTP)

Contents

Topic **Page No.**

UNIT - I

UNIT - II

UNIT - III

UNIT - IV

Topic **Page No.**

UNIT - V

Important Questions

UNIT - I

1. What is Ethernet? Explain the types of Ethernet cables.

Ans :

Refer Unit-I, Q.No. 1

2. Explain various types of computer networks?

Ans :

Refer Unit-I, Q.No. 3

3. Write about Ethernet Frame Format with neat diagram.

Ans :

Refer Unit-I, Q.No. 5

4. Explain IEEE 802.11 Architecture?

Ans :

Refer Unit-I, Q.No. 9

5. Explain the frame format of 802.11.

Ans :

Refer Unit-I, Q.No. 10

6. Explain about various types of bridges

Ans :

Refer Unit-I, Q.No. 12

UNIT - II

1. Explain about logical addressing and IP addressing in network layer.

Ans :

Refer Unit-II, Q.No. 1

2. What is subnetting? Explain, How to use subnetting.

Ans :

Refer Unit-II, Q.No. 4

3. What is CIDR (Classless Inter-Domain Routing or supernetting)?

Ans :

Refer Unit-II, Q.No. 7

4. Explain briefly about ICMP protocol.

Ans :

Refer Unit-II, Q.No. 9

5. Explain about IGMP protocol.

Ans :

Refer Unit-II, Q.No. 10

6. Define routing. Explain different types of routings.

Ans :

Refer Unit-II, Q.No. 11

7. Explain about Distance vector routing.

Ans :

Refer Unit-II, Q.No. 12

8. Explain about OSPF routing protocol.

Ans :

Refer Unit-II, Q.No. 14

9. Explain briefly about BGP protocol.

Ans :

Refer Unit-II, Q.No. 15

UNIT - III

1. What is Transmission Control Protocol (TCP)? Explain the working mechanism of TCP.

Ans :

Refer Unit-III, Q.No. 1

2. Explain the transition states of TCP with a neat diagram.

Ans :

Refer Unit-III, Q.No. 4

3. Explain TCP sliding window protocol with neat diagram in detail.

Ans :

Refer Unit-III, Q.No. 5

4. Explain about Congestion control in TCP.

Ans :

Refer Unit-III, Q.No. 6

5. Explain TCP timer management and transaction TCP.

Ans :

Refer Unit-III, Q.No. 8

6. Explain about UDP protocol

Ans :

Refer Unit-III, Q.No. 9

UNIT - IV

1. Explain briefly about Unix Socket structure.

Ans :

Refer Unit-IV, Q.No. 1

2. Explain the concept of Socket Address Structure (SAS).

Ans :

Refer Unit-IV, Q.No. 2

3. Explain Advanced socket system calls in UNIX.

Ans :

Refer Unit-IV, Q.No. 5

4. What is TCP? Explain TCP client Server model.

Ans :

Refer Unit-IV, Q.No. 6

5. Explain TCP client –server implementation with the help of program.

Ans :

Refer Unit-IV, Q.No. 7

6. Explain IPV4 Socket Options

Ans :

Refer Unit-IV, Q.No. 13

UNIT - V

1. Define Application Layer? What are the Services Provided by the Application Layer?

Ans :

Refer Unit-V, Q.No. 1

2. Explain briefly about DNS.

Ans :

Refer Unit-V, Q.No. 2

3. Explain briefly about SMTP protocol.

Ans :

Refer Unit-V, Q.No. 3

4. Explain briefly about FTP Protocol.

Ans :

Refer Unit-V, Q.No. 4

UNIT I

Multiple Access:

Wired LAN-Ethernet IEEE 802.3LAN, CSMA/CD protocol, Binary exponential backoff algorithm. Comparison of Switched , Fast and Gigabit Ethernet.

Wireless LAN

IEEE 802.11 architecture. CSMA/CA protocol, Bridges and types of bridges. ARP and RARP

1.1 Multiple Access

1.1.1 Wired Lan

1.1.1.1 Ethernet

Q1. What is Ethernet? Explain the types of Ethernet cables.

Ans : **(Imp.)**

Meaning

Ethernet, defined under IEEE 802.3, is one of today's most widely used data communications standards, and it finds its major use in Local Area Network (LAN) applications. With versions including 10 Base-T, 100 Base-T and now Gigabit Ethernet, it offers a wide variety of choices of speeds and capability. Ethernet is also cheap and easy to install. Additionally Ethernet, IEEE 802.3 offers a considerable degree of flexibility in terms of the network topologies that are allowed. Furthermore as it is in widespread use in LANs, it has been developed into a robust system that meets the needs to wide number of networking requirements.

Advantages

- It is not much costly to form an Ethernet network. As compared to other systems of connecting computers, it is relatively inexpensive.
- Ethernet network provides high security for data as it uses firewalls in terms of data security.
- Also, the Gigabit network allows the users to transmit data at a speed of 1-100Gbps.
- In this network, the quality of the data transfer does maintain.
- In this network, administration and maintenance are easier.
- The latest version of gigabit ethernet and wireless ethernet have the potential to transmit data at the speed of 1-100Gbps.

Disadvantages of Ethernet

- It needs deterministic service; therefore, it is not considered the best for real-time applications.
- The wired Ethernet network restricts you in terms of distances, and it is best for using in short distances.
- If you create a wired ethernet network that needs cables, hubs, switches, routers, they increase the cost of installation.
- Data needs quick transfer in an interactive application, as well as data is very small.
- In ethernet network, any acknowledge is not sent by receiver after accepting a packet.
- If you are planning to set up a wireless Ethernet network, it can be difficult if you have no experience in the network field.
- Comparing with the wired Ethernet network, wireless network is not more secure.
- The full-duplex data communication mode is not supported by the 100Base-T4 version.
- Additionally, finding a problem is very difficult in an Ethernet network (if has), as it is not easy to determine which node or cable is causing the problem.

Types of Ethernet Cables

Mainly there are three types of ethernet cables used in LANs i.e., Coaxial cables, Twisted Pair cables, and Fiber optic cables.

1. Coaxial Cables

A coaxial cable is used to carry high-frequency electrical signals with low losses. It uses 10Base2 and 10Base5 Ethernet variants. It has a copper conductor in the middle that is surrounded by a dielectric insulator usually made of PVC or Teflon. The dielectric insulator is surrounded by a braided conducting metallic shield which reduces EMI (Electromagnetic Interference) of the metal and outside interference; and finally, the metallic shield is covered by a plastic covering called a sheath usually made of PVC or some other fire-resistant plastic material. Its maximum transmission speed is 10 Mbps. It is usually used in telephone systems, cable TV, etc.

Types of Coaxial Cables

- **Hardline coaxial cable** is used in applications where high signal strength is required; this type is most commonly used. They are used in internet lines and telephone lines.
- **RG-6 Coaxial Cable** is used where better signal quality is required; it has a thicker dielectric insulator, they are used in broadband internet, cable TV, etc.
- **Tri-axial Cable** They offer more bandwidth and interference rejection; they use an additional copper braid shield. Commonly used in connecting cameras and cable TVs. Etc.

Types of Connectors used in Coaxial cable:

- BNC (Bayonet Neil Concelman),
- N series Connectors,
- F Type connectors,
- SMA or Subminiature connector,
- TNC (Threaded Neil Concelman), etc.

2. Twisted Pair Cable

Twisted pair is a copper wire cable in which two insulated copper wires are twisted around each other to reduce interference or crosstalk. It uses 10BASE-T, 100BASE-T, and some other newer ethernet variants. It uses RJ-45 connectors.

Types of Twisted Pair Cable

- **Shielded Twisted Pair (STP) Cable:** In STP the wires are covered by a copper braid covering or a foil shield, this foil shield adds a layer that protects it against interference leaking into and out of the cable. Hence, they are used for longer distances and higher transmission rates.
- **Unshielded Twisted Pair (UTP) Cable:** Unshielded twisted pair cable is one of the most commonly used cables in computer networks at present time. UTP consists of two insulated copper wires twisted around one another, the twisting of wires helps in controlling interference.

Categories of UTP Cables

Category	Bandwidth	Speed	Use
1	1.4 MHz	1 Mbps	Telephone wire
2	4 MHz	4 Mbps	Transmission Lines
3	16 MHz	16 Mbps	10BaseT Ethernet
4	20 MHz	20 Mbps	Used in Token Ring
5	100 MHz	100 Mbps	100BaseT Ethernet
5	100 MHz	1 Gbps	Gigabit Ethernet
5e	100 MHz	1 Gbps	Gigabit Ethernet
6	250 MHz	10 Gbps	Gigabit Ethernet
7	600 MHz	10 Gbps	Gigabit Ethernet
7a	1 GHz	Up to 10 Gbps	Gigabit Ethernet
8	2 GHz	25 Gbps to up to 40 Gbps	Datacenters

3. Fiber Optic Cable

Fiber optic cables use optical fibers which are made of glass cores surrounded by several layers of cladding material usually made of PVC or Teflon, it transmits data in the form of light signals due to which there are no interference issues in fiber optics. Fiber optics can transmit signals over a very long distance as compared to twisted pairs or coaxial cables. It uses 10BaseF, 100BaseFX, 100BaseBX, 100BaseSX, 1000BaseFx, 1000BaseSX, and 1000BaseBx ethernet variants. Hence, it is capable of carrying information at a great speed.

Types of Fiber Optics:

- **SMF (Single-mode fiber) -** it uses one single ray of light to transmit data, it is used for long-distance transmission.
- **MMF (Multi-mode Fiber) -** it uses multiple light rays to transmit data, it is comparatively less expensive.

Types of Connectors Used: Mainly these four connectors are used with fiber optic cable:

- ST (Straight-tip) Connector
- FC (Fiber Channel) Connector
- SC (Subscriber) Connector
- LC (Lucent) Connector

Q2. Explain various types of network elements.

Ans :

In a computer network, various types of network elements are used to perform different functions to ensure the efficient operation of the network. Some of the most common network elements include:

1. **Network Interface Card (NIC):** A network interface card is a hardware component that connects a computer to a network. It is responsible for managing the exchange of data between the computer and the network.
2. **Switch:** A switch is a networking device that connects multiple devices in a local area network (LAN). It forwards data packets between devices based on the Media Access Control (MAC) addresses of the devices.
3. **Router:** A router is a networking device that connects multiple networks and forwards data packets between them. It uses logical addresses (IP addresses) to direct traffic between different networks.
4. **Gateway:** A gateway is a networking device that acts as a bridge between different types of networks, such as between a local area network (LAN) and the internet. It translates data between different protocols and network architectures.
5. **Modem:** A modem is a networking device that converts digital data into analog signals that can be transmitted over telephone lines or other analog networks. It also converts analog signals back into digital data at the receiving end.
6. **Firewall:** A firewall is a security device that monitors and controls the traffic between a network and the internet. It filters incoming and outgoing traffic based on predefined security rules to protect the network from unauthorized access and malicious attacks.
7. **Load Balancer:** A load balancer is a networking device that distributes incoming network traffic across multiple servers or network resources to optimize resource utilization and ensure high availability of the network.
8. **Access Point:** An access point is a networking device that provides wireless connectivity to devices in a local area network (LAN). It acts as a bridge between wireless devices and a wired network.

Q3. Explain various types of computer networks?

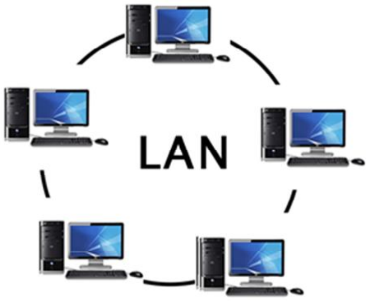
Ans : **(Imp.)**

A computer network is a group of computers linked to each other that enables the computer to communicate with another computer and share their resources, data, and applications.

A computer network can be categorized by their size. A computer network is mainly of four types:

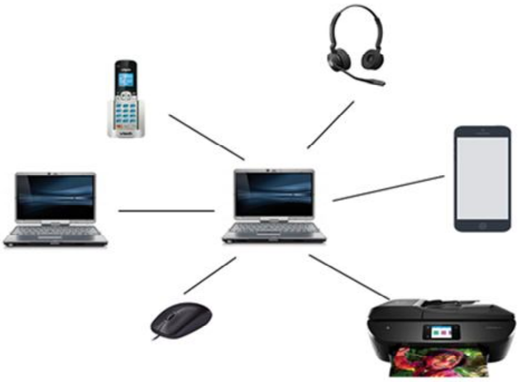
1. LAN (Local Area Network)

- Local Area Network is a group of computers connected to each other in a small area such as building, office.
- LAN is used for connecting two or more personal computers through a communication medium such as twisted pair, coaxial cable, etc.
- It is less costly as it is built with inexpensive hardware such as hubs, network adapters, and ethernet cables.
- The data is transferred at an extremely faster rate in Local Area Network.
- Local Area Network provides higher security.



2. PAN (Personal Area Network)

- Personal Area Network is a network arranged within an individual person, typically within a range of 10 meters.
- Personal Area Network is used for connecting the computer devices of personal use is known as Personal Area Network.
- Thomas Zimmerman was the first research scientist to bring the idea of the Personal Area Network.
- Personal Area Network covers an area of 30 feet.
- Personal computer devices that are used to develop the personal area network are the laptop, mobile phones, media player and play stations.



There are two types of Personal Area Network:

(i) Wired Personal Area Network

(ii) Wireless Personal Area Network

(i) Wireless Personal Area Network: Wireless Personal Area Network is developed by simply using wireless technologies such as WiFi, Bluetooth. It is a low range network.

(ii) Wired Personal Area Network: Wired Personal Area Network is created by using the USB.

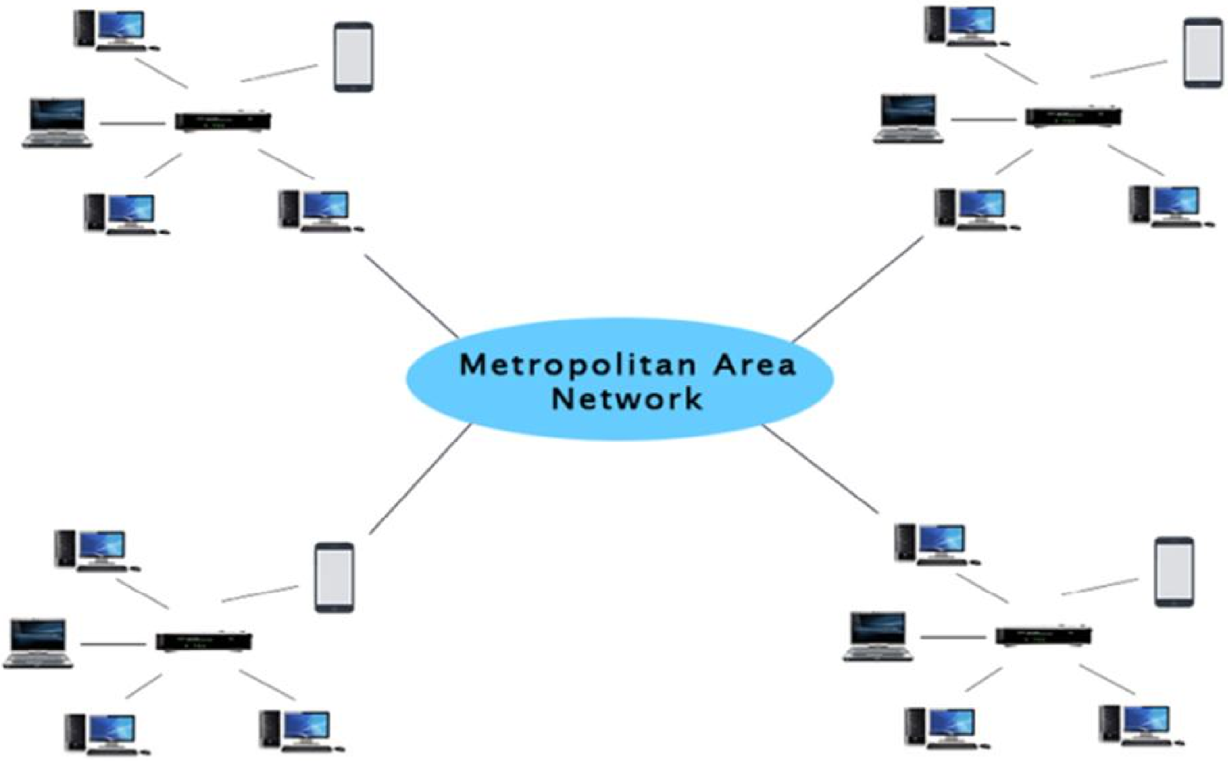
Examples of Personal Area Network:

- **Body Area Network:** Body Area Network is a network that moves with a person. For example, a mobile network moves with a person. Suppose a person establishes a network connection and then creates a connection with another device to share the information.
- **Offline Network:** An offline network can be created inside the home, so it is also known as a home **network**. A home network is designed to integrate the devices such as printers, computer, television but they are not connected to the internet.

- **Small Home Office:** It is used to connect a variety of devices to the internet and to a corporate network using a VPN

3. MAN (Metropolitan Area Network)

- A metropolitan area network is a network that covers a larger geographic area by interconnecting a different LAN to form a larger network.
- Government agencies use MAN to connect to the citizens and private industries.
- In MAN, various LANs are connected to each other through a telephone exchange line.
- The most widely used protocols in MAN are RS-232, Frame Relay, ATM, ISDN, OC-3, ADSL, etc.
- It has a higher range than Local Area Network(LAN).



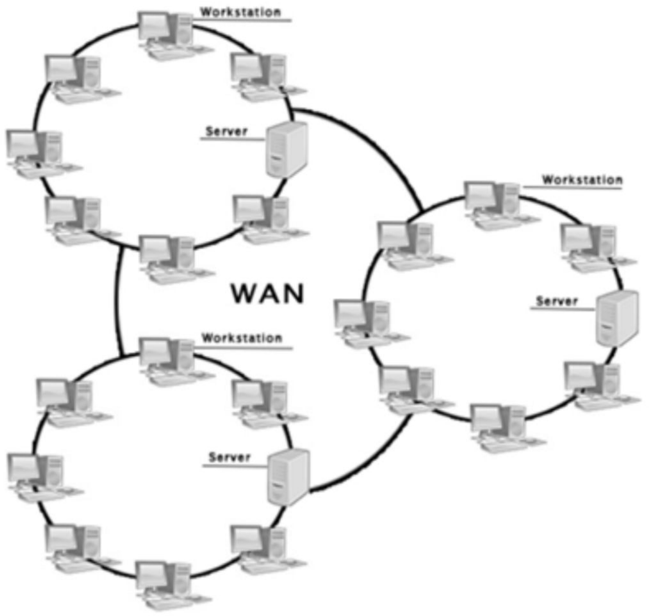
Uses of Metropolitan Area Network

- MAN is used in communication between the banks in a city.
- It can be used in an Airline Reservation.
- It can be used in a college within a city.
- It can also be used for communication in the military.

4. WAN (Wide Area Network)

- A Wide Area Network is a network that extends over a large geographical area such as states or countries.
- A Wide Area Network is quite bigger network than the LAN.
- A Wide Area Network is not limited to a single location, but it spans over a large geographical area through a telephone line, fibre optic cable or satellite links.

- The internet is one of the biggest WAN in the world.
- A Wide Area Network is widely used in the field of Business, government, and education.



Examples of Wide Area Network:

- **Mobile Broadband:** A 4G network is widely used across a region or country.
- **Last mile:** A telecom company is used to provide the internet services to the customers in hundreds of cities by connecting their home with fiber.
- **Private network:** A bank provides a private network that connects the 44 offices. This network is made by using the telephone leased line provided by the telecom company.

Advantages :

Following are the advantages of the Wide Area Network:

- **Geographical area:** A Wide Area Network provides a large geographical area. Suppose if the branch of our office is in a different city then we can connect with them through WAN. The internet provides a leased line through which we can connect with another branch.
- **Centralized data:** In case of WAN network, data is centralized. Therefore, we do not need to buy the emails, files or back up servers.
- **Get updated files:** Software companies work on the live server. Therefore, the programmers get the updated files within seconds.
- **Exchange messages:** In a WAN network, messages are transmitted fast. The web application like Facebook, Whatsapp, Skype allows you to communicate with friends.
- **Sharing of software and resources:** In WAN network, we can share the software and other resources like a hard drive, RAM.
- **Global business:** We can do the business over the internet globally.
- **High bandwidth:** If we use the leased lines for our company then this gives the high bandwidth. The high bandwidth increases the data transfer rate which in turn increases the productivity of our company.

Disadvantages of Wide Area Network:

The following are the disadvantages of the Wide Area Network:

- **Security issue:** A WAN network has more security issues as compared to LAN and MAN network as all the technologies are combined together that creates the security problem.
- **Needs Firewall & antivirus software:** The data is transferred on the internet which can be changed or hacked by the hackers, so the firewall needs to be used. Some people can inject the virus in our system so antivirus is needed to protect from such a virus.
- **High Setup cost:** An installation cost of the WAN network is high as it involves the purchasing of routers, switches.
- **Troubleshooting problems:** It covers a large area so fixing the problem is difficult.

Q4. Explain about various types of topologies in network.

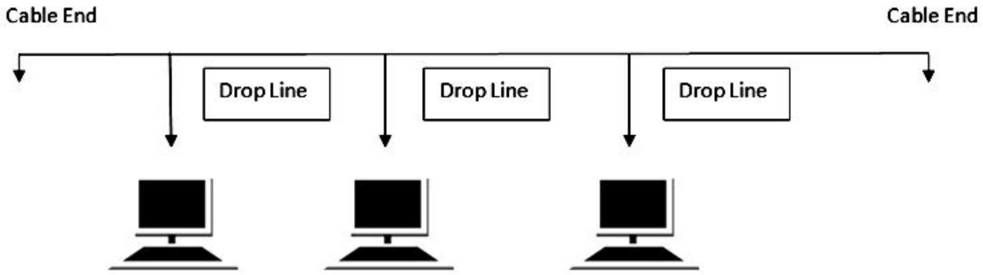
Ans :

Types of Network Topology

Network Topology is the schematic description of a network arrangement, connecting various nodes(sender and receiver) through lines of connection.

1. BUS Topology

Bus topology is a network type in which every computer and network device is connected to single cable. When it has exactly two endpoints, then it is called Linear Bus topology



Features

1. It transmits data only in one direction.
2. Every device is connected to a single cable

Advantages

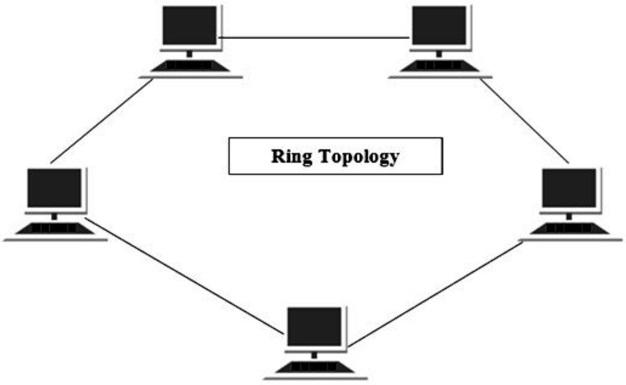
1. It is cost effective.
2. Cable required is least compared to other network topology.
3. Used in small networks.
4. It is easy to understand.
5. Easy to expand joining two cables together.

Disadvantages

1. Cables fails then whole network fails.
2. If network traffic is heavy or nodes are more the performance of the network decreases.
3. Cable has a limited length.
4. It is slower than the ring topology.

2. RING Topology

It is called ring topology because it forms a ring as each computer is connected to another computer, with the last one connected to the first. Exactly two neighbours for each device.



Features

1. A number of repeaters are used for Ring topology with large number of nodes, because if someone wants to send some data to the last node in the ring topology with 100 nodes, then the data will have to pass through 99 nodes to reach the 100th node. Hence to prevent data loss repeaters are used in the network.
2. The transmission is unidirectional, but it can be made bidirectional by having 2 connections between each Network Node, it is called Dual Ring Topology.
3. In Dual Ring Topology, two ring networks are formed, and data flow is in opposite direction in them. Also, if one ring fails, the second ring can act as a backup, to keep the network up.
4. Data is transferred in a sequential manner that is bit by bit. Data transmitted, has to pass through each node of the network, till the destination node.

Advantages

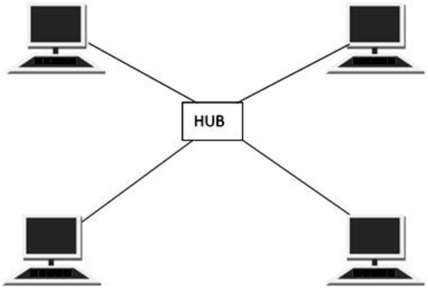
1. Transmitting network is not affected by high traffic or by adding more nodes, as only the nodes having tokens can transmit data.
2. Cheap to install and expand

Disadvantages

1. Troubleshooting is difficult in ring topology.
2. Adding or deleting the computers disturbs the network activity.
3. Failure of one computer disturbs the whole network.

3. STAR Topology

In this type of topology all the computers are connected to a single hub through a cable. This hub is the central node and all others nodes are connected to the central node.



Features

1. Every node has its own dedicated connection to the hub.
2. Hub acts as a repeater for data flow.
3. Can be used with twisted pair, Optical Fibre or coaxial cable.

Advantages

1. Fast performance with few nodes and low network traffic.
2. Hub can be upgraded easily.
3. Easy to troubleshoot.
4. Easy to setup and modify.
5. Only that node is affected which has failed, rest of the nodes can work smoothly.

Disadvantages

1. Cost of installation is high.
2. Expensive to use.
3. If the hub fails then the whole network is stopped because all the nodes depend on the hub.
4. Performance is based on the hub that is it depends on its capacity

4. MESH Topology

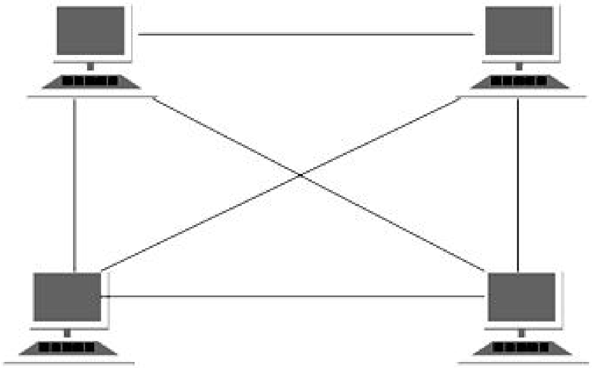
It is a point-to-point connection to other nodes or devices. All the network nodes are connected to each other. Mesh has n(n-1)/2 physical channels to link n devices.

There are two techniques to transmit data over the Mesh topology, they are :

1. Routing
2. Flooding

1. **Routing:** In routing, the nodes have a routing logic, as per the network requirements. Like routing logic to direct the data to reach the destination using the shortest distance. Or, routing logic which has information about the broken links, and it avoids those node etc. We can even have routing logic, to re-configure the failed nodes.

2. **Flooding:** In flooding, the same data is transmitted to all the network nodes, hence no routing logic is required. The network is robust, and the its very unlikely to lose the data. But it leads to unwanted load over the network.



Types of Mesh Topology

1. **Partial Mesh Topology :** In this topology some of the systems are connected in the same fashion as mesh topology but some devices are only connected to two or three devices.
2. **Full Mesh Topology :** Each and every nodes or devices are connected to each other.

Features

1. Fully connected.
2. Robust.
3. Not flexible.

Advantages

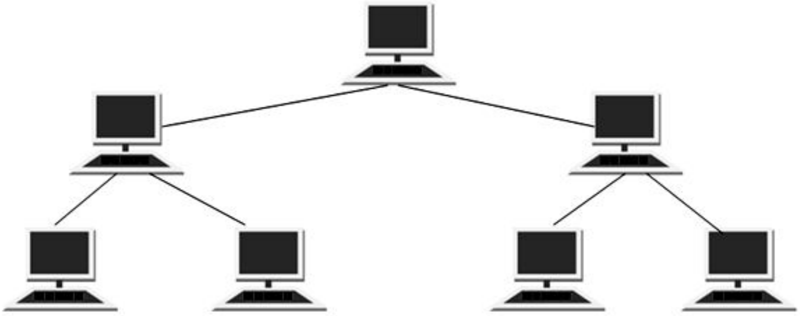
1. Each connection can carry its own data load.
2. It is robust.
3. Fault is diagnosed easily.
4. Provides security and privacy.

Disadvantages

1. Installation and configuration is difficult.
2. Cabling cost is more.
3. Bulk wiring is required.

5. TREE Topology

It has a root node and all other nodes are connected to it forming a hierarchy. It is also called hierarchical topology. It should at least have three levels to the hierarchy.



Features

1. Ideal if workstations are located in groups.
2. Used in Wide Area Network.

Advantages

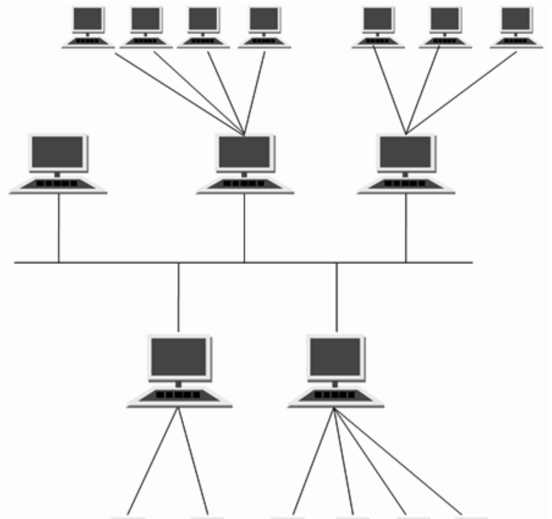
1. Extension of bus and star topologies.
2. Expansion of nodes is possible and easy.
3. Easily managed and maintained.
4. Error detection is easily done.

Disadvantages

1. Heavily cabled.
2. Costly.
3. If more nodes are added maintenance is difficult.
4. Central hub fails, network fails.

6. HYBRID Topology

It is two different types of topologies which is a mixture of two or more topologies. For example if in an office in one department ring topology is used and in another star topology is used, connecting these topologies will result in Hybrid Topology (ring topology and star topology).



Features

1. It is a combination of two or topologies
2. Inherits the advantages and disadvantages of the topologies included

Advantages

1. Reliable as Error detecting and trouble shooting is easy.
2. Effective.
3. Scalable as size can be increased easily.
4. Flexible.

Disadvantages

1. Complex in design.
2. Costly.

1.1.1.2 IEEE 802.3 LAN

Q5. Write about Ethernet Frame Format with neat diagram.

Ans : **(Imp.)**

Ethernet, IEEE 802.3, is one of the most widely used standards for computer networking and general data communications.

It is widely used in all forms of data networking from connecting to a variety of devices in the home including Wi-Fi routers to business data networks and telecommunications networking. In fact everything from small local area networks to mobile phone networks and large business orientated local area networks and wide area networks.

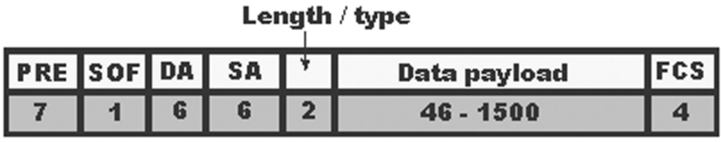
Ethernet IEEE 802.3 Frame Format / Structure

Ethernet, IEEE 802.3 defines the frame formats or frame structures that are developed within the MAC layer of the protocol stack.

Essentially the same frame structure is used for the different variants of Ethernet, although there are some changes to the frame structure to extend the performance of the system should this be needed. With the high speeds and variety of media used, this basic format sometimes needs to be adapted to meet the individual requirements of the transmission system, but this is still specified within the amendment / update for that given Ethernet variant.

10 / 100 Mbps Ethernet MAC data frame format

The basic MAC data frame format for Ethernet, IEEE 802.3 used within the 10 and 100 Mbps systems is given below:



Length / type

PRE	SOF	DA	SA		Data payload	FCS
7	1	6	6	2	46 - 1500	4

The basic frame consists of seven elements split between three main areas:-

Header

- ➢ Preamble (PRE) - This is seven bytes long and it consists of a pattern of alternating ones and zeros, and this informs the receiving stations that a frame is starting as well as enabling synchronisation. (10 Mbps Ethernet)

- Start Of Frame delimiter (SOF) - This consists of one byte and contains an alternating pattern of ones and zeros but ending in two ones.
- Destination Address (DA) - This field contains the address of station for which the data is intended. The left most bit indicates whether the destination is an individual address or a group address. An individual address is denoted by a zero, while a one indicates a group address. The next bit into the DA indicates whether the address is globally administered, or local. If the address is globally administered the bit is a zero, and a one of it is locally administered. There are then 46 remaining bits. These are used for the destination address itself.
- Source Address (SA) - The source address consists of six bytes, and it is used to identify the sending station. As it is always an individual address the left most bit is always a zero.
- Length / Type - This field is two bytes in length. It provides MAC information and indicates the number of client data types that are contained in the data field of the frame. It may also indicate the frame ID type if the frame is assembled using an optional format.(IEEE 802.3 only).

Payload

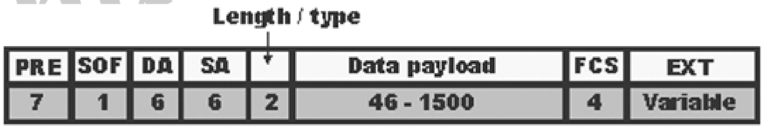
- Data - This block contains the payload data and it may be up to 1500 bytes long. If the length of the field is less than 46 bytes, then padding data is added to bring its length up to the required minimum of 46 bytes.

Trailer

- Frame Check Sequence (FCS) - This field is four bytes long. It contains a 32 bit Cyclic Redundancy Check (CRC) which is generated over the DA, SA, Length / Type and Data fields.

1000 Mbps Ethernet MAC data frame format

The basic MAC data frame format for Ethernet is modified slightly for 1GE, IEEE 802.3z systems. When using the 1000Base-X standard, there is a minimum frame size of 416bytes, and for 1000Base-T there is a minimum frame size of 520bytes. To accommodate this, an extension is added as appropriate. This is a non-data variable extension field to any frames that are shorter than the minimum required length.



Length / type

PRE	SOF	DA	SA		Data payload	FCS	EXT
7	1	6	6	2	46 - 1500	4	Variable

1.1.1.3 CSMA/CD PROTOCOL

Q6. What is CSMA/CD Protocol? Explain.

Ans :

CSMA/CD procedure can be understood as a group discussion, where if the participants speak all at once then it will be very confusing and the communication will not happen.

Instead, for good communication, it is required that the participants speak one after another so that we can clearly understand the contribution of each participant in the discussion.

Once a participant has finished talking, we should wait for a certain time period to see if any other participant is speaking or not. One should start speaking only when no other participant has spoken. If another participant also speaks at the same time, then we should stop, wait, and try again after some time.

Similar is the process of CSMA/CD, where the data packet transmission is only done when the data transmission medium is free. When various network devices try to share a data channel simultaneously, then it will encounter a data collision.

The medium is continuously monitored to detect any data collision. When the medium is detected as free, the station should wait for a certain time period before sending the data packet to avoid any chances of data collision.

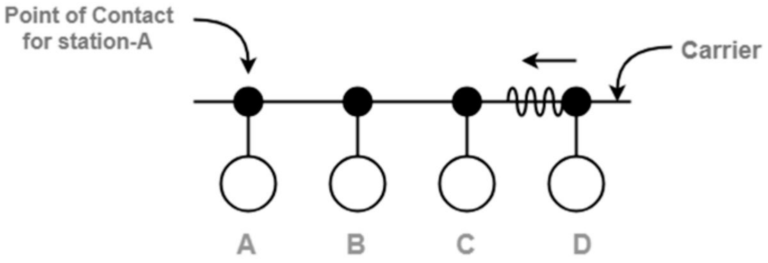
When no other station tries to send the data and there is no data collision detected, then the transmission of data is said to be successful.

This access control method works as follows-

Step-01: Sensing the Carrier

Any station willing to transmit the data senses the carrier.

- If it finds the carrier free, it starts transmitting its data packet otherwise not.
- Each station can sense the carrier only at its point of contact with the carrier.
- It is not possible for any station to sense the entire carrier.
- Thus, there is a huge possibility that a station might sense the carrier free even when it is actually not.



(Bus Topology)

Example

At the current instance,

- If station A senses the carrier at its point of contact, then it will find the carrier free.
- But the carrier is actually not free because station D is already transmitting its data.
- If station A starts transmitting its data now, then it might lead to a collision with the data transmitted by station D.

Step-02: Detecting the Collision

In CSMA / CD,

- It is the responsibility of the transmitting station to detect the collision.
- For detecting the collision, CSMA / CD implements the following condition.
- This condition is followed by each station-

Transmission delay >= 2 x Propagation delay

Meaning

According to this condition,

- Each station must transmit the data packet of size whose transmission delay is at least twice its propagation delay.
- If the size of data packet is smaller, then collision detection would not be possible.

Length of Data Packet

We know

- Transmission delay = Length of data packet (L) / Bandwidth (B)
- Propagation delay = Distance between the two stations (D) / Propagation speed (V)

Substituting values in the above condition, we get-

$L / B > = 2 \times D / V$

Thus,

$L > = 2 \times B \times D / V$

Two cases are possible-

Case-01:

If no collided signal comes back during the transmission,

- It indicates that no collision has occurred.
- The data packet is transmitted successfully.

Case-02:

If the collided signal comes back during the transmission,

- It indicates that the collision has occurred.
- The data packet is not transmitted successfully.
- Step-03 is followed.

Step-03: Releasing Jam Signal

Jam signal is a 48 bit signal.

- It is released by the transmitting stations as soon as they detect a collision.
- It alerts the other stations not to transmit their data immediately after the collision.
- Otherwise, there is a possibility of collision again with the same data packet.
- Ethernet sends the jam signal at a frequency other than the frequency of data signals.
- This ensures that jam signal does not collide with the data signals undergone collision.

Step-04: Waiting For Back Off Time

- After the collision, the transmitting station waits for some random amount of time called as back off time.
- After back off time, it tries transmitting the data packet again.
- If again the collision occurs, then station again waits for some random back off time and then tries again.
- The station keeps trying until the back off time reaches its limit.
- After the limit is reached, station aborts the transmission.
- Back off time is calculated using Back Off Algorithm.

Efficiency

Efficiency (η) = Useful Time / Total Time

Before a successful transmission,

- There may occur many number of collisions.
- $2 \times T_p$ time is wasted during each collision.

Thus,

- Useful time = Transmission delay of data packet = T_t
- Useless time = Time wasted during collisions + Propagation delay of data packet = $c \times 2 \times T_p + T_p$
- Here, c = Number of contention slots / collision slots.

Thus,

$$\boxed{\text{Efficiency } (\eta) = \frac{T_t}{c \times 2 \times T_p + T_t + T_p}}$$

Here,

- c is a variable.
- This is because number of collisions that might occur before a successful transmission are variable.

Probabilistic Analysis shows-

Average number of collisions before a successful transmission = e

Substituting c = e in the above relation, we get-

$$\boxed{\text{Efficiency } (\eta) = \frac{T_t}{e \times 2 \times T_p + T_t + T_p}}$$

OR

$$\boxed{\text{Efficiency } (\eta) = \frac{T_t}{T_t + 6.44 \times T_p}}$$

OR

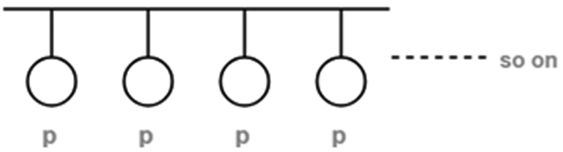
$$\boxed{\text{Efficiency } (\eta) = \frac{1}{1 + 6.44 \times a}, \text{ where } a = T_p / T_t}$$

Probabilistic Analysis

Let us perform the probabilistic analysis to find the average number of collisions before a successful transmission.

Consider

- Number of stations connected to a CSMA / CD network = n
- Probability of each station to transmit the data = p



Transmission will be successful only when-

- One station transmits the data
- Other (n – 1) stations do not transmit the data.

Thus, Probability of successful transmission is given by-

$$\boxed{P_{\text{successful transmission}} = {}^{n}C_1 \times p \times (1 - p)^{n-1}}$$

Now, let us find the maximum value of $P_{\text{successful transmission}}$.

For maximum value, we put-

$$\boxed{\frac{dP_{\text{successful of transmission}}}{dp} = 0}$$

On solving,

At p = 1/n, we get the maximum value of $P_{\text{successful transmission}}$

Thus,

$(P_{\text{successful transmission}})_{\max}$

$= {}^{n}C_1 \times 1/n \times (1 - 1/n)^{n-1}$

$= n \times 1/n \times (1 - 1/n)^{n-1}$

$= (1 - 1/n)^{n-1}$

$(P_{\text{successful transmission}})_{\max} = (1 - 1/n)^{n-1}$

If there are sufficiently large number of stations i.e., n $\to \infty$, then we have-

$$\boxed{\begin{aligned} \lim_{n\to\infty} (P_{\text{successful transmission}})_{\max} &= \lim_{n\to\infty} \left(1 - \frac{1}{n}\right)^{n-1} \\ &= \frac{1}{e} \end{aligned}}$$

Number of times a station must try before successfully transmitting the data packet

$= 1 / P_{\max}$ (Using Poisson's distribution)

$= 1 / (1/e)$

$= e$

From here, we conclude-

Average number of collisions that might occur before a successful transmission = e

1.1.2 Binary Exponential Backoff Algorithm

Q7. Explain about Back off algorithm.

Ans : **(Imp.)**

In CSMA / CD protocol,

- After the occurrence of collision, station waits for some random back off time and then retransmits.
- This waiting time for which the station waits before retransmitting the data is called as back off time.
- Back Off Algorithm is used for calculating the back off time.

Back Off Algorithm-

After undergoing the collision,

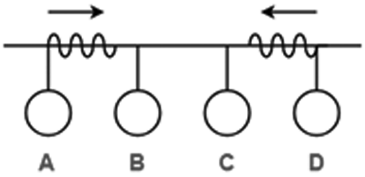
- Transmitting station chooses a random number in the range [0, 2^n-1] if the packet is undergoing collision for the nth time.
- If station chooses a number k, then-

Back off time = k x Time slot

Where value of one time slot = 1 RTT

Example-

Consider the following scenario where stations A and D start transmitting their data simultaneously-



For simplicity,

- We consider the value of time slot = 1 unit.
- Thus, back off time = K units.

Scene-01: For 1st Data Packet Of Both Stations-

- Both the stations start transmitting their 1st data packet simultaneously.
- This leads to a collision.
- Clearly, the collision on both the packets is occurring for the 1st time.
- So, collision number for the 1st data packet of both the stations = 1.

At Station A

After detecting the collision,

- Station A randomly chooses a number in the range $[0, 2^1 - 1] = [0,1]$.
- If station A chooses the number K_A, then back off time = K_A units.

At Station D

After detecting the collision,

- Station D randomly chooses a number in the range $[0, 2^1 - 1] = [0, 1]$.
- If station D chooses the number K_D, then back off time = K_D units.

Following 4 cases are possible-

K_A	K_D	Remarks
0	0	In this case, both the stations start retransmitting their data immediately. This case leads to a collision again.
0	1	In this case, station A starts retransmitting its data immediately while station D waits for 1 unit of time. This case leads to A successfully retransmitting its data after the 1st collision.
1	0	In this case, station A waits for 1 unit of time while station D starts retransmitting its data immediately. This case leads to D successfully retransmitting its data after the 1st collision.
1	1	In this case, both the stations wait for 1 unit of time and then starts retransmitting their data simultaneously. This case leads to a collision again.

From here,

- Probability of station A to successfully retransmit its data after the 1st collision = 1/4
- Probability of station D to successfully retransmit its data after the 1st collision = 1/4
- Probability of occurrence of collision again after the 1st collision = 2/4 = 1/2

Now,

- Consider case-02 occurs.
- This causes station A to successfully retransmit its 1st packet after the 1st collision.

Scene-02: For 2nd Data Packet of Station A And 1st Data Packet of Station D

Consider after some time,

- Station A starts transmitting its 2nd data packet and station D starts retransmitting its 1st data packet simultaneously.
- This leads to a collision.

At Station A

- The 2nd data packet of station A undergoes collision for the 1st time.
- So, collision number for the 2nd data packet of station A = 1.
- Now, station A randomly chooses a number in the range [0, $2^1 – 1$] = [0, 1].
- If station A chooses the number K_A, then back off time = K_A units.

At Station D-

- The 1st data packet of station D undergoes collision for the 2nd time.
- So, collision number for the 1st data packet of station D = 2.
- Now, station D randomly chooses a number in the range [0, $2^2 – 1$] = [0, 3].
- If station D chooses the number K_D, then back off time = K_D units.

Following 8 cases are possible-

K_A	K_D	Remarks
0	0	In this case, both the stations start retransmitting their data immediately. This case leads to a collision again.
0	1	In this case, station A starts retransmitting its data immediately while station D waits for 1 unit of time. This case leads to A successfully retransmitting its data after the 2nd collision.
0	2	In this case, station A starts retransmitting its data immediately while station D waits for 2 unit of time. This case leads to A successfully retransmitting its data after the 2nd collision.
0	3	In this case, station A starts retransmitting its data immediately while station D waits for 3 unit of time. This case leads to A successfully retransmitting its data after the 2nd collision.

1	0	In this case, station A waits for 1 unit of time while station D starts retransmitting its data immediately. This case leads to D successfully retransmitting its data after the 2nd collision.
1	1	In this case, both the stations wait for 1 unit of time and then starts retransmitting their data simultaneously. This case leads to a collision again.
1	2	In this case, station A waits for 1 unit of time while station D waits for 2 unit of time. This case leads to A successfully retransmitting its data after the 2nd collision.
1	3	In this case, station A waits for 1 unit of time while station D waits for 3 unit of time. This case leads to A successfully retransmitting its data after the 2nd collision.

From here,

- Probability of station A to successfully retransmit its data after the 2nd collision = 5 / 8
- Probability of station D to successfully retransmit its data after the 2nd collision = 1 / 8
- Probability of occurrence of collision again after the 2nd collision = 2 / 8 = 1 / 4

Now,

- Consider case-03 occurs.
- This causes station A to successfully retransmit its 2nd packet after the 2nd collision.

Scene-03: For 3rd Data Packet Of Station A And 1st Data Packet Of Station D

Consider after some time,

- Station A starts transmitting its 3rd data packet and station D starts retransmitting its 1st data packet simultaneously.
- This leads to a collision.

At Station A

- The 3rd data packet of station A undergoes collision for the 1st time.
- So, collision number for the 3rd data packet of station A = 1.
- Now, station A randomly chooses a number in the range $[0, 2^1-1] = [0,1]$.
- If station A chooses the number K_A, then back off time = K_A unit.

At Station D-

- The 1st data packet of station D undergoes collision for the 3rd time.
- So, collision number for the 1st data packet of station D = 3.
- Now, station D randomly chooses a number in the range $[0, 2^3-1] = [0,7]$.
- If station D chooses the number K_D, then back off time = K_D unit.

Following 16 cases are possible-

K_A	K_D	Remarks
0	0	In this case, both the stations start retransmitting their data immediately. This case leads to a collision again.
0	1	• In this case, station A starts retransmitting its data immediately while station D waits for 1 unit of time. • This case leads to A successfully retransmitting its data after the 3rd collision.
0	2	In this case, station A starts retransmitting its data immediately while station D waits for 2 unit of time. This case leads to A successfully retransmitting its data after the 3rd collision.
0	3	In this case, station A starts retransmitting its data immediately while station D waits for 3 unit of time. This case leads to A successfully retransmitting its data after the 3rd collision.
0	4	• In this case, station A starts retransmitting its data immediately while station D waits for 4 unit of time. • This case leads to A successfully retransmitting its data after the 3rd collision.
0	5	• In this case, station A starts retransmitting its data immediately while station D waits for 5 unit of time. • This case leads to A successfully retransmitting its data after the 3rd collision.
0	6	In this case, station A starts retransmitting its data immediately while station D waits for 6 unit of time. This case leads to A successfully retransmitting its data after the 3rd collision.
0	7	In this case, station A starts retransmitting its data immediately while station D waits for 7 unit of time. This case leads to A successfully retransmitting its data after the 3rd collision.
1	0	In this case, station A waits for 1 unit of time while station D starts retransmitting its data immediately. This case leads to D successfully retransmitting its data after the 3rd collision.
1	1	In this case, both the stations wait for 1 unit of time and then starts retransmitting their data simultaneously. This case leads to a collision again.
1	2	• In this case, station A waits for 1 unit of time while station D waits for 2 unit of time. • This case leads to A successfully retransmitting its data after the 3rd collision.

1	3	In this case, station A waits for 1 unit of time while station D waits for 3 unit of time. This case leads to A successfully retransmitting its data after the 3rd collision.
1	4	• In this case, station A waits for 1 unit of time while station D waits for 4 unit of time. • This case leads to A successfully retransmitting its data after the 3rd collision.
1	5	In this case, station A waits for 1 unit of time while station D waits for 5 unit of time. This case leads to A successfully retransmitting its data after the 3rd collision.
1	6	• In this case, station A waits for 1 unit of time while station D waits for 6 unit of time. • This case leads to A successfully retransmitting its data after the 3rd collision.
1	7	In this case, station A waits for 1 unit of time while station D waits for 7 unit of time. This case leads to A successfully retransmitting its data after the 3rd collision.

From here,

- Probability of station A to successfully retransmit its data after the 3rd $\text{collision} = 13/16$
- Probability of station D to successfully retransmit its data after the 3rd $\text{collision} = 1/16$
- Probability of occurrence of collision again after the 3rd $\text{collision} = 1/16$

In the similar manner, the procedure continues.

1.1.3 Comparison of Switched , Fast and Gigabit Ethernet

Q8. Compare and contrast Fast Ethernet abd Gigabit Ethernet.

Ans :

S.No.	Key	Fast Ethernet	Gigabit Ethernet
1.	Successor	Fast Ethernet is successor of 10-Base-T-Ethernet	Gigabit Ethernet is successor of Fast Ethernet.
2.	Network speed	Fast Ethernet speed is upto 100 Mbps.	Gigabit Ethernet speed is upto 1 Gbps
3.	Complexity	Fast Ethernet is simple to configure.	Gigabit Ethernet is quiet complex to configure.
4.	Delay	Fast Ethernet generates more delay.	Gigabit Ethernet generates less delay than Fast Ethernet.
5.	Coverage	Fast Ethernet coverage limit is upto 10KM.	Gigabit Ethernet coverage limit is upto 70KM.
6.	Round trip delay	Fast Ethernet round trip delay is 100 to 500 bit times.	Gigabit Ethernet round trip delay is 4000 bit times.

1.2 Wireless LAN

1.2.1 IEEE 802.11 Architecture.

Q9. Explain IEEE 802.11 Architecture?

OR

Expalin Wireless LAN.

Ans : **(Imp.)**

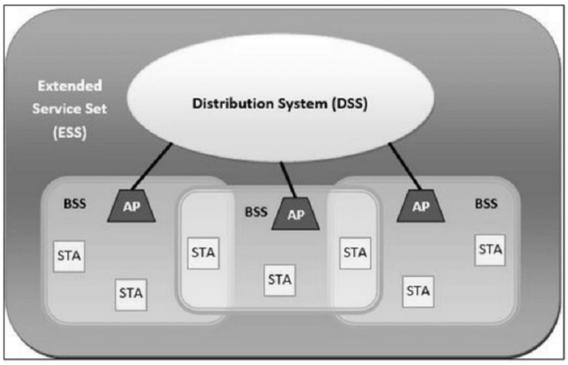
IEEE 802.11 standard, popularly known as WiFi, lays down the architecture and specifications of wireless LANs (WLANs). WiFi or WLAN uses high-frequency radio waves instead of cables for connecting the devices in LAN. Users connected by WLANs can move around within the area of network coverage.

IEEE 802.11 Architecture

The components of an IEEE 802.11 architecture are as follows :

1. **Stations (STA):** Stations comprise all devices and equipments that are connected to the wireless LAN. A station can be of two types:
   - **Wireless Access Point (WAP):** WAPs or simply access points (AP) are generally wireless routers that form the base stations or access.
   - **Client:** Clients are workstations, computers, laptops, printers, smartphones, etc.

   Each station has a wireless network interface controller.
2. **Basic Service Set (BSS):** A basic service set is a group of stations communicating at physical layer level. BSS can be of two categories depending upon mode of operation:
   - **Infrastructure BSS:** Here, the devices communicate with other devices through access points.
   - **Independent BSS:** Here, the devices communicate in peer-to-peer basis in an ad hoc manner.
3. **Extended Service Set (ESS):** It is a set of all connected BSS.
4. **Distribution System (DS):** It connects access points in ESS.



Advantages

- They provide clutter free homes, offices and other networked places.
- The LANs are scalable in nature, i.e. devices may be added or removed from the network at a greater ease than wired LANs.
- The system is portable within the network coverage and access to the network is not bounded by the length of the cables.
- Installation and setup is much easier than wired counterparts.
- The equipment and setup costs are reduced.

Disadvantages

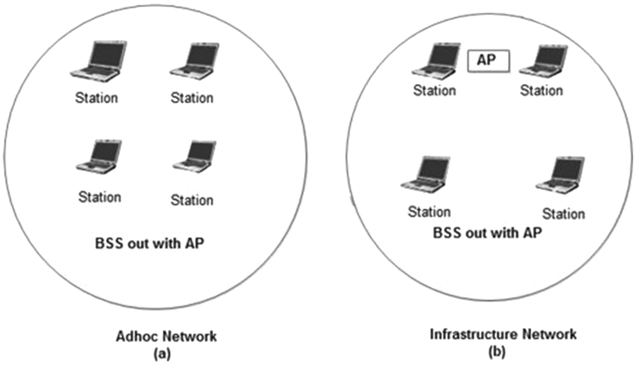
- Since radio waves are used for communications, the signals are noisier with more interference from nearby systems.
- Greater care is needed for encrypting information. Also, they are more prone to errors. So, they require greater bandwidth than the wired LANs.
- WLANs are slower than wired LANs.
- The 802.11architecture defines two types of services and three different types of stations

802.11 Services

- The two types of services are
- 1. Basic services set (BSS)
- 2. Extended Service Set (ESS)

1. **Basic Services Set (BSS)**
   - The basic services set contain stationary or mobile wireless stations and a central base station called access point (AP).
   - The use of access point is optional.

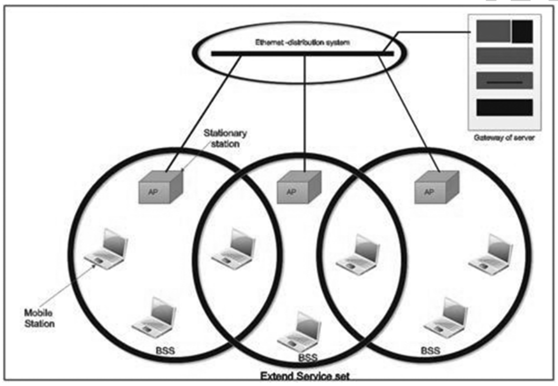
- If the access point is not present, it is known as stand-alone network. Such a
- BSS cannot send data to other BSSs. This type of architecture is known as adhoc architecture.
- The BSS in which an access point is present is known as an infrastructure network.



Basic Service Sets

2. Extend Service Set (ESS)

- An extended service set is created by joining two or more basic service sets (BSS) having access points (APs).



These extended networks are created by joining the access points of basic services sets through a wired LAN known as distribution system.

- The distribution system can be any IEET LAN.
- There are two types of stations in ESS:
  - (i) **Mobile stations**: These are normal stations inside a BSS.
  - (ii) **Stationary stations**: These are AP stations that are part of a wired LAN.
- Communication between two stations in two different BSS usually occurs via two APs.
- A mobile station can belong to more than one BSS at the same time.

802.11 Station Types

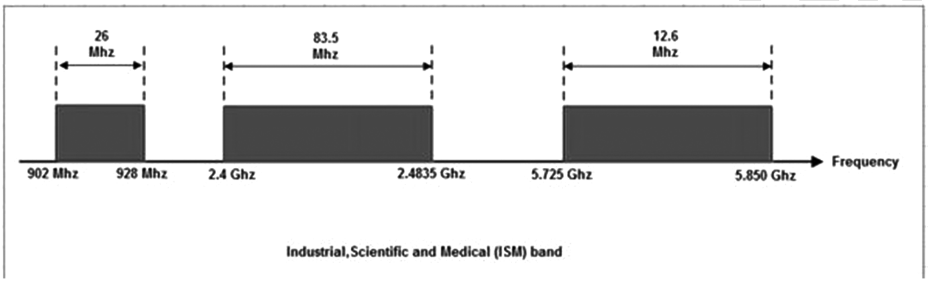
IEEE 802.11 defines three types of stations on the basis of their mobility in wireless LAN. These are:

1. No-transition Mobility
2. BSS-transition Mobility
3. ESS-transition Mobility

1. **No-transition .Mobility**: These types of stations are either stationary *i.e.* immovable or move only inside a BSS.
2. **BSS-transition mobility**: These types of stations can move from one BSS to another but the movement is limited inside an ESS.
3. **ESS-transition mobility**: These types of stations can move from one ESS to another. The communication mayor may not be continuous when a station moves from one ESS to another ESS.

Physical Layer Functions

- As we know that physical layer is responsible for converting data stream into signals, the bits of 802.11 networks can be converted to radio waves or infrared waves.
- These are six different specifications of IEEE 802.11. These implementations, except the first one, operate in *industrial, scientific* and *medical (ISM)* band. These three banks are unlicensed and their ranges are
  1. 902-928 MHz
  2. 2.400-4.835 GHz
  3. 5.725-5.850 GHz



Industrial,Scientific and Medical (ISM) band

- The different implementations of IEE802.11 are given below:

1. **IEEE 802.11 infrared**
   - It uses diffused (not line of sight) infrared light in the range of 800 to 950 nm.
   - It allows two different speeds: I Mbps and 2Mbps.
   - For a I-Mbps data rate, 4 bits of data are encoded into 16 bit code. This 16 bit code contains fifteen as and a single 1.
   - For a 2-Mbps data rate, a 2 bit code is encoded into 4 bit code. This 4 bit code contains three Os and a single 1.
   - The modulation technique used is pulse position modulation (PPM) *i.e.* for converting digital signal to analog.
2. **IEEE 802.11 FHSS**
   - IEEE 802.11 uses Frequency Hoping Spread Spectrum (FHSS) method for signal generation.
   - This method uses 2.4 GHz ISM band. This band is divided into 79 subbands of 1MHz with some guard bands.
   - In this method, at one moment data is sent by using one carrier frequency and then by some other carrier frequency at next moment. After this, an idle time is there in communication. This cycle is repeated after regular intervals.
   - A pseudo random number generator selects the hopping sequence.

- The allowed data rates are 1 or 2 Mbps.
- This method uses frequency shift keying (two level or four level) for modulation *i.e.* for converting digital signal to analogy.

3. IEEE 802.11 DSSS

- This method uses Direct Sequence Spread Spectrum (DSSS) method for signal generation. Each bit is transmitted as 11 chips using a Barker sequence.
- DSSS uses the 2.4-GHz ISM band.
- It also allows the data rates of 1 or 2 Mbps.
- It uses phase shift keying (PSK) technique at 1 M baud for converting digital signal to analog signal.

4. IEEE 802.11a OFDM

- This method uses Orthogonal Frequency Division Multiplexing (OFDM) for signal generation.
- This method is capable of delivering data upto 18 or 54 Mbps.
- In OFDM all the subbands are used by one source at a given time.
- It uses 5 GHz ISM band.
- This band is divided into 52 subbands, with 48 subbands for data and 4 subbands for control information.
- If phase shift keying (PSK) is used for modulation then data rate is 18 Mbps. If quadrature amplitude modulation (QAM) is used, the data rate can be 54 Mbps.

5. IEEE 802.11b HR-OSSS

- It uses High Rate Direct Sequence Spread Spectrum method for signal generation.
- HR-DSSS is similar to DSSS except for encoding method.
- Here, 4 or 8 bits are encoded into a special symbol called complementary code key (CCK).
- It uses 2.4 GHz ISM band.
- It supports four data rates: 1,2,5.5 and 11 Mbps.
- 1 Mbps and 2 Mbps data rates uses phase shift modulation.
- The 5.5. Mbps version uses BPSK and transmits at 1.375 Mbaud/s with 4-bit CCK encoding.
- The 11 Mbps version uses QPSK and transmits at 1.375 Mbps with 8-bit CCK encoding.

6. IEEE 802.11g OFDM

- It uses OFDM modulation technique.
- It uses 2.4 GHz ISM band.
- It supports the data rates of 22 or 54 Mbps.
- It is backward compatible with 802.11 b.

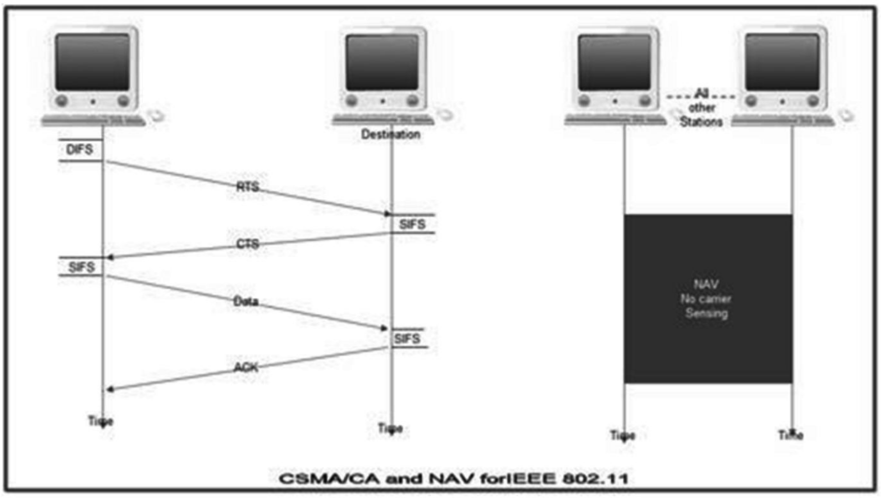
MAC sublayer Functions

802.11 support two different modes of operations. These are:

1. Distributed Coordination Function (DCF)
2. Point Coordination Function (PCF)

1. Distributed Coordination Function

- The DCF is used in BSS having no access point.
- DCF uses CSMA/CA protocol for transmission.
- The following steps are followed in this method.



CSMA/CA and NAV forIEEE 802.11

1. When a station wants to transmit, it senses the channel to see whether it is free or not.
2. If the channel is not free the station waits for back off time.
3. If the station finds a channel to be idle, the station waits for a period of time called distributed interframe space (DIFS).
4. The station then sends control frame called request to send (RTS) as shown in figure.
5. The destination station receives the frame and waits for a short period of time called short interframe space (SIFS).
6. The destination station then sends a control frame called clear to send (CTS) to the source station. This frame indicates that the destination station is ready to receive data.
7. The sender then waits for SIFS time and sends data.
8. The destination waits for SIFS time and sends acknowledgement for the received frame.

Collision avoidance

- 802.11 standard uses Network Allocation Vector (NAV) for collision avoidance.
- The procedure used in NAV is explained below:
  1. Whenever a station sends an RTS frame, it includes the duration of time for which the station will occupy the channel.
  2. All other stations that are affected by the transmission creates a timer caned network allocation vector (NAV).
  3. This NAV (created by other stations) specifies for how much time these stations must not check the channel.

4. Each station before sensing the channel, check its NAV to see if has expired or not.

5. If its NA V has expired, the station can send data, otherwise it has to wait.

- There can also be a collision during handshaking *i.e.* when RTS or CTS control frames are exchanged between the sender and receiver. In this case following procedure is used for collision avoidance:

1. When two or more stations send RTS to a station at same time, their control frames collide.

2. If CTS frame is not received by the sender, it assumes that there has been a collision.

3. In such a case sender, waits for back off time and retransmits RTS.

2. Point Coordination Function

- PCF method is used in infrastructure network. In this Access point is used to control the network activity.
- It is implemented on top of the DCF and IS used for time sensitive transmissions.
- PCF uses centralized, contention free polling access method.
- The AP performs polling for stations that wants to transmit data. The various stations are polled one after the other.
- To give priority to PCF over DCF, another interframe space called PIFS is defined. PIFS (PCF IFS) is shorter than DIFS.
- If at the same time, a station is using DCF and AP is using PCF, then AP is given priority over the station.
- Due to this priority of PCF over DCF, stations that only use DCF may not gain access to the channel.
- To overcome this problem, a repetition interval is defined that is repeated continuously. This repetition interval starts with a special control frame called beacon frame.
- When a station hears beacon frame, it start their NAV for the duration of the period of the repetition interval.

Q10. Explain the frame format of 802.11.

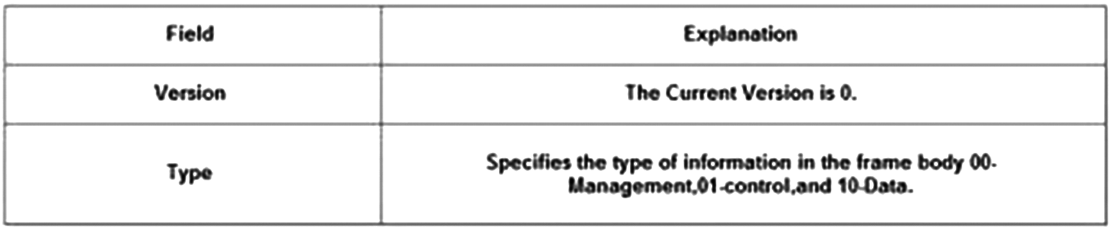
Ans : **(Imp.)**

Frame Format of 802.11

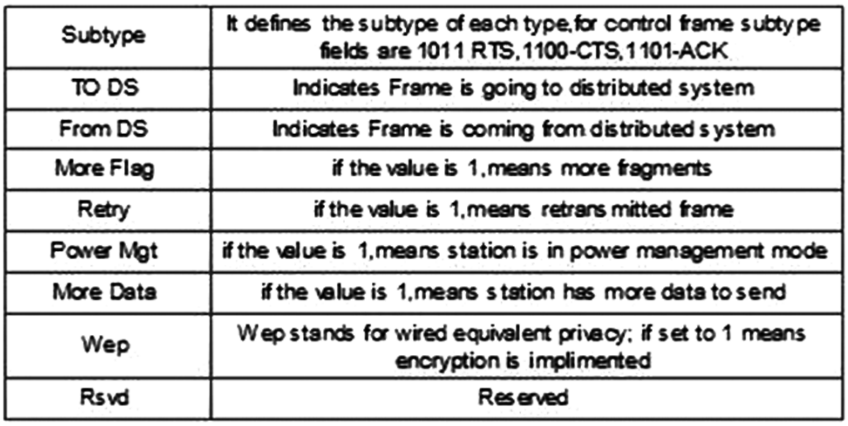
The MAC layer frame consists of nine fields.

1. **Frame Control (FC).** This is 2 byte field and defines the type of frame and some control information. This field contains several different subfields.

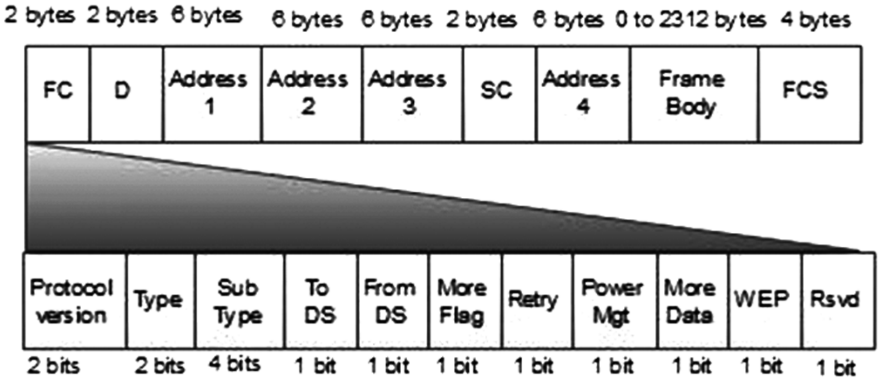
These are listed in the table below:



Field	Explanation
Version	The Current Version is 0.
Type	Specifies the type of information in the frame body 00-Management,01-control,and 10-Data.



Subtype	It defines the subtype of each type,for control frame subtype fields are 1011 RTS,1100-CTS,1101-ACK
TO DS	Indicates Frame is going to distributed system
From DS	Indicates Frame is coming from distributed system
More Flag	if the value is 1,means more fragments
Retry	if the value is 1,means retransmitted frame
Power Mgt	if the value is 1,means station is in power management mode
More Data	if the value is 1,means station has more data to send
Wep	Wep stands for wired equivalent privacy; if set to 1 means encryption is implimented
Rsvd	Reserved



Frame Format of IEEE 802.11

2. **D**. It stands for duration and is of 2 bytes. This field defines the duration for which the frame and its acknowledgement will occupy the channel. It is also used to set the value of NA V for other stations.

3. **Addresses**. There are 4 address fields of 6 bytes length. These four addresses represent source, destination, source base station and destination base station.

4. **Sequence Control (SC).** This 2 byte field defines the sequence number of frame to be used in flow control.

5. **Frame body**. This field can be between 0 and 2312 bytes. It contains the information.

6. **FCS.** This field is 4 bytes long and contains ‘cRC-32 error detection sequence.

IEEE 802.11 Frame types

There are three different types of frames:

1. Management frame
2. Control frame
3. Data frame

1. **Management frame**. These are used for initial communication between stations and access points.

2. **Control frame**. These are used for accessing the channel and acknowledging frames. The control frames are RTS and CTS.

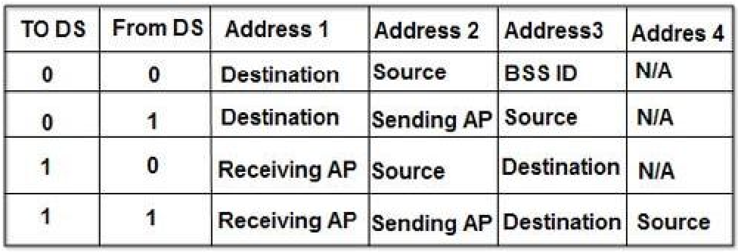
3. **Data frame**. These are used for carrying data and control information.

802.11 Addressing

- There are four different addressing cases depending upon the value of To DS And from DS subfields of FC field.
- Each flag can be 0 or 1, resulting in 4 different situations.

1. If To DS = 0 and From DS = 0, it indicates that frame is not going to distribution system and is not coming from a distribution system. The frame is going from one station in a BSS to another.
2. If To DS = 0 and From DS = 1, it indicates that the frame is coming from a distribution system. The frame is coming from an AP and is going to a station. The address 3 contains original sender of the frame (in another BSS).
3. If To DS = 1 and From DS = 0, it indicates that the frame is going to a distribution system. The frame is going from a station to an AP. The address 3 field contains the final destination of the frame.
4. If To DS = 1 and From DS = 1,it indicates that frame is going from one AP to another AP in a wireless distributed system.

The table below specifies the addresses of all four cases.



TO DS	From DS	Address 1	Address 2	Address3	Addres 4
0	0	Destination	Source	BSS ID	N/A
0	1	Destination	Sending AP	Source	N/A
1	0	Receiving AP	Source	Destination	N/A
1	1	Receiving AP	Sending AP	Destination	Source

1.2.2 CSMA/CA Protocol

Q11. Explain about CSMA/CA protocol.

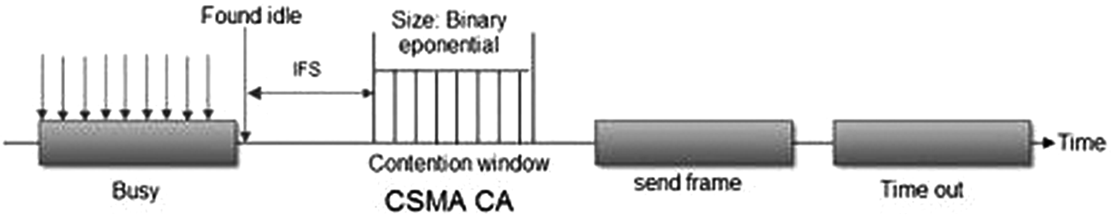
Ans :

- CSMA/CA protocol is used in wireless networks because they cannot detect the collision so the only solution is collision avoidance.
- CSMA/CA avoids the collisions using three basic techniques.

(i) Interframe space

(ii) Contention window

(iii) Acknowledgements



CSMA CA

1. Interframe Space (IFS)

- Whenever the channel is found idle, the station does not transmit immediately. It waits for a period of time called interframe space (IFS).

- When channel is sensed to be idle, it may be possible that same distant station may have already started transmitting and the signal of that distant station has not yet reached other stations.
- Therefore the purpose of IFS time is to allow this transmitted signal to reach other stations.
- If after this IFS time, the channel is still idle, the station can send, but it still needs to wait a time equal to contention time.
- IFS variable can also be used to define the priority of a station or a frame.

2. **Contention Window**

- Contention window is an amount of time divided into slots.
- A station that is ready to send chooses a random number of slots as its wait time.
- The number of slots in the window changes according to the binary exponential back-off strategy. It means that it is set of one slot the first time and then doubles each time the station cannot detect an idle channel after the IFS time.
- This is very similar to the p-persistent method except that a random outcome defines the number of slots taken by the waiting station.
- In contention window the station needs to sense the channel after each time slot.
- If the station finds the channel busy, it does not restart the process. It just stops the timer & restarts it when the channel is sensed as idle.

3. **Acknowledgement**

- Despite all the precautions, collisions may occur and destroy the data.
- The positive acknowledgment and the time-out timer can help guarantee that receiver has received the frame.

CSMA/CA Procedure

- This is the CSMA protocol with collision avoidance.
- The station ready to transmit, senses the line by using one of the persistent strategies.
- As soon as it find the line to be idle, the station waits for an IFG (Interframe gap) amount of time.
- If then waits for some random time and sends the frame.
- After sending the frame, it sets a timer and waits for the acknowledgement from the receiver.
- If the acknowledgement is received before expiry of the timer, then the transmission is successful.
- But if the transmitting station does not receive the expected acknowledgement before the timer expiry then it increm

1.2.3 Bridges and Types of Bridges

Q12. Explain about various types of bridges

Ans : **(Imp.)**

A bridge is a networking device that connects two or more network segments or LANs (Local Area Networks) together. Bridges are used to extend the reach of a network, reduce network congestion, and increase overall network performance. Here are the types of bridges in computer networks, along with their advantages and disadvantages:

1. **Local Bridge:** A local bridge connects two LANs that use the same data link protocol, such as Ethernet. Local bridges can increase the capacity and performance of a network by reducing the number of packets that need to be transmitted across the network. However, they can also increase the complexity of the network and require additional configuration.

Advantages

- Increases the overall network capacity and performance.
- Reduces network congestion by only allowing relevant traffic to pass through the bridge.
- Helps to segment the network and reduce the effects of broadcast storms.

Disadvantages

- Can increase the complexity of the network.
- Requires additional configuration.
- Can introduce additional points of failure.

2. **Remote Bridge:** A remote bridge connects two LANs that use different data link protocols, such as Ethernet and Token Ring. Remote bridges can help to extend the reach of a network and improve overall network performance. However, they can also be more complex to configure and manage than local bridges.

Advantages

- ➢ Helps to extend the reach of a network.
- ➢ Improves overall network performance by reducing network congestion.
- ➢ Can help to integrate different types of LANs into a single network.

Disadvantages

- ➢ Can be more complex to configure and manage than local bridges.
- ➢ Requires additional hardware and software to convert between different data link protocols.
- ➢ Can introduce additional points of failure.

3. **Transparent Bridge:** A transparent bridge connects two LANs that use the same data link protocol and appears to the network as a single network segment. Transparent bridges can help to reduce network congestion and improve overall network performance, but they can also introduce additional complexity to the network.

Advantages

- ➢ Reduces network congestion by only allowing relevant traffic to pass through the bridge.
- ➢ Helps to segment the network and reduce the effects of broadcast storms.
- ➢ Can improve overall network performance.

Disadvantages

- ➢ Can increase the complexity of the network.
- ➢ Requires additional configuration.
- ➢ Can introduce additional points of failure.

4. **Source Route Bridge:** A source route bridge uses information contained in the network packet to determine the path the packet should take through the network. Source route bridges can help to improve network performance and reduce network congestion, but they can also introduce additional complexity to the network.

Advantages

- ➢ Improves network performance by reducing network congestion.
- ➢ Can help to integrate different types of LANs into a single network.
- ➢ Can help to reduce the effects of broadcast storms.

Disadvantages

- ➢ Can be more complex to configure and manage than other types of bridges.
- ➢ Requires additional hardware and software to implement.
- ➢ Can introduce additional points of failure.

5. **Wireless Bridge:** A wireless bridge connects two or more LANs wirelessly, using radio waves to transmit data. Wireless bridges can help to extend the reach of a network and provide greater mobility for users, but they can also be susceptible to interference and security risks.

Advantages

- ➢ Helps to extend the reach of a network.
- ➢ Provides greater mobility for users.
- ➢ Reduces the need for physical cabling.

Disadvantages

- ➢ Can be susceptible to interference and signal degradation.
- ➢ Can introduce additional security risks.
- ➢ May require additional hardware and software to implement.

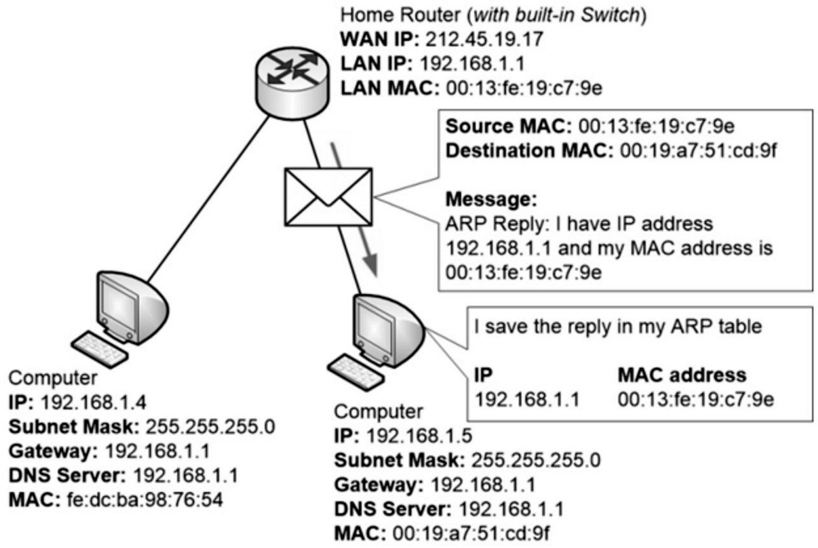
1.2.4 ARP and RARP

Q13. Explain Address Resolution Protocol (ARP) and its types.

Ans :

The address resolution protocol (arp) is a protocol used by the Internet Protocol (IP) [RFC826], specifically IPv4, to map IP network addresses to the hardware addresses used by a data link protocol. The protocol operates below the network layer as a part of the interface between the OSI network and OSI link layer. It is used when IPv4 is used over Ethernet.

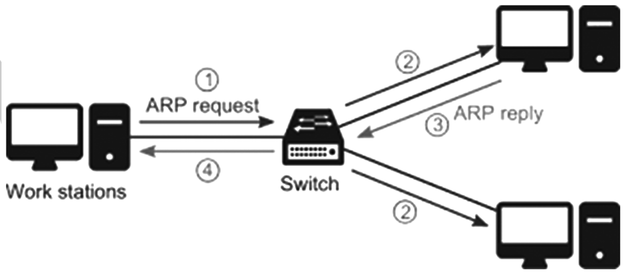
The term address resolution refers to the process of finding an address of a computer in a network.


Home Router (*with built-in Switch*)
WAN IP: 212.45.19.17
LAN IP: 192.168.1.1
LAN MAC: 00:13:fe:19:c7:9e
Source MAC: 00:13:fe:19:c7:9e
Destination MAC: 00:19:a7:51:cd:9f
Message:
ARP Reply: I have IP address 192.168.1.1 and my MAC address is 00:13:fe:19:c7:9e
I save the reply in my ARP table
IP **MAC address**
192.168.1.1 00:13:fe:19:c7:9e
Computer
IP: 192.168.1.4
Subnet Mask: 255.255.255.0
Gateway: 192.168.1.1
DNS Server: 192.168.1.1
MAC: fe:dc:ba:98:76:54
Computer
IP: 192.168.1.5
Subnet Mask: 255.255.255.0
Gateway: 192.168.1.1
DNS Server: 192.168.1.1
MAC: 00:19:a7:51:cd:9f

The address is "resolved" using a protocol in which a piece of information is sent by a client process executing on the local computer to a server process executing on a remote computer.

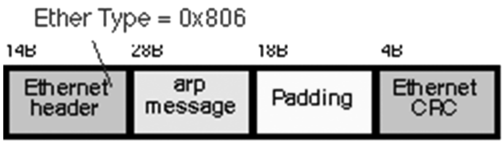
The information received by the server allows the server to uniquely identify the network system for which the address was required and therefore to provide the required address.

The address resolution procedure is completed when the client receives a response from the server containing the required address.



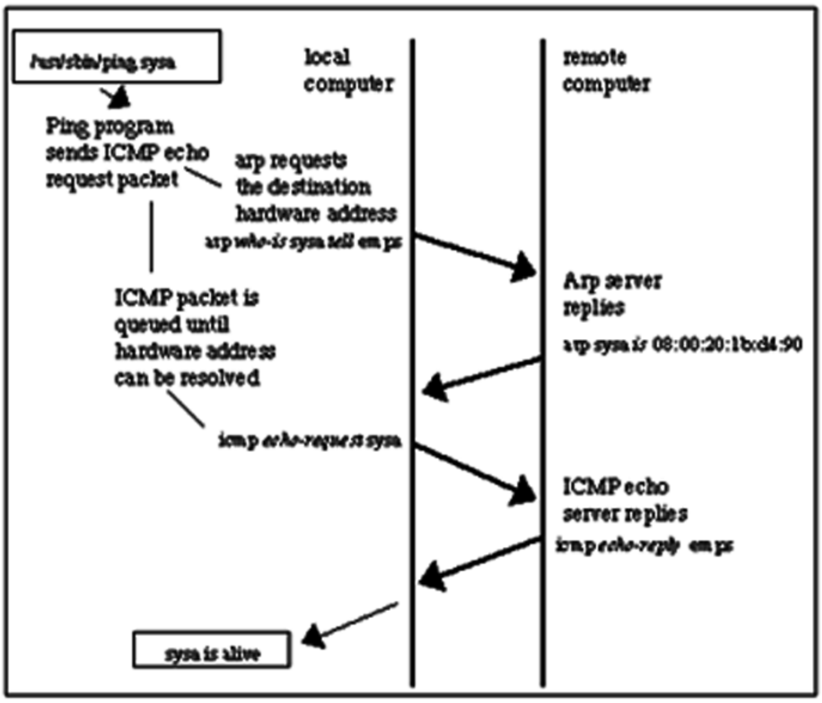
Example of use of the Address Resolution Protocol (arp)

The figure below shows the use of arp when a computer tries to contact a remote computer on the same LAN (known as "sysa") using the "ping" program. It is assumed that no previous IP datagrams have been received form this computer, and therefore arp must first be used to identify the MAC address of the remote computer.



The arp request message is sent using the Ethernet broadcast address, and an Ethernet protocol type of value 0x806. Since it is broadcast, it is received by all systems in the LAN. This ensures that the target of the query is connected to the network, it will receive a copy of the query. Only this system responds. The other systems discard the packet silently.

The target system forms an arp response . This packet is unicast to the address of the computer sending the query . Since the original request also included the hardware address (Ethernet source address) of the requesting computer.


/usr/sbin/ping sysa
local computer
remote computer
Ping program sends ICMP echo request packet
arp requests the destination hardware address
arp who-is sysa tell emps
Arp server replies
arp sysa is 08:00:20:1b:d4:90
ICMP packet is queued until hardware address can be resolved
icmp echo-request sysa
ICMP echo server replies
icmp echo-reply emps
sysa is alive

Q14. Explain about Reverse Address Resolution Protocol.

Ans :

RARP (Reverse Address Resolution Protocol) is a protocol by which a physical machine in a local area network can request to learn its IP address from a gateway server's Address Resolution Protocol (ARP) table or cache.

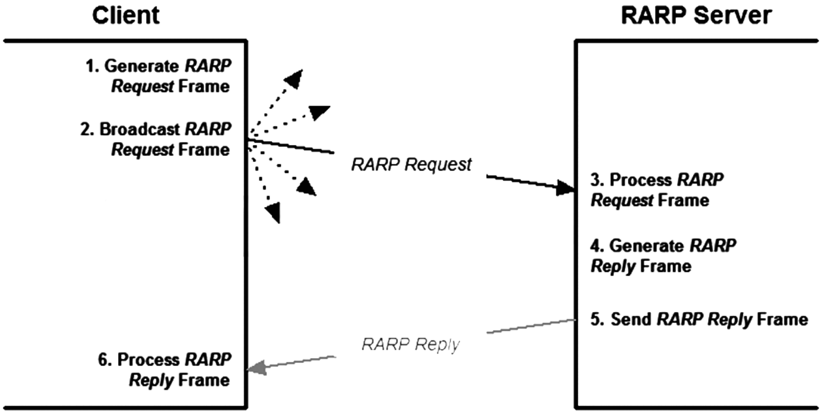
A network administrator creates a table in a local area network's gateway router that maps the physical machine (or Media Access Control - MAC address) addresses to corresponding Internet Protocol addresses.

When a new machine is set up, its RARP client program requests from the RARP server on the router to be sent its IP address. Assuming that an entry has been set up in the router table, the RARP server will return the IP address to the machine which can store it for future use.

RARP is available for Ethernet, Fiber Distributed-Data Interface, and token ring LAN

There are four types of arp messages that may be sent by the arp protocol. These are identified by four values in the "operation" field of an arp message. The types of message are:

1. ARP request
2. ARP reply
3. RARP request
4. RARP reply


Client
RARP Server
1. Generate RARP Request Frame
2. Broadcast RARP Request Frame
RARP Request
3. Process RARP Request Frame
4. Generate RARP Reply Frame
5. Send RARP Reply Frame
RARP Reply
6. Process RARP Reply Frame

1. Source Device Generates RARP Request Message: The source device generates an RARP Request message. Thus, it uses the value 3 for the Opcode in the message. It puts its own data link layer address as both the Sender Hardware Address and also the Target Hardware Address. It leaves both the Sender Protocol Address and the Target Protocol Address blank, since it doesn't know either.
2. Source Device Broadcasts RARP Request Message: The source broadcasts the ARP Request message on the local network.
3. Local Devices Process RARP Request Message: The message is received by each device on the local network and processed. Devices that are not configured to act as RARP servers ignore the message.
4. RARP Server Generates RARP Reply Message: Any device on the network that is set up to act as an RARP server responds to the broadcast from the source device. It generates an RARP Reply using an Opcode value of 4.
5. RARP Server Sends RARP Reply Message: The RARP server sends the RARP Reply message unicast to the device looking to be configured.
6. Source Device Processes RARP Reply Message: The source device processes the reply from the RARP server. It then configures itself using the IP address in the Target Protocol Address supplied by the RARP server. It is possible that more than one RARP server may respond to any request, if two or more are configured on any local network. The source device will typically use the first reply and discard the others.

RARP (Reverse Address Resolution Protocol) is a protocol by which a physical machine in a local area network can request to learn its IP address from a gateway server's Address Resolution Protocol (ARP) table or cache.

A network administrator creates a table in a local area network's gateway router that maps the physical machine (or Media Access Control - MAC address) addresses to corresponding Internet Protocol addresses.

When a new machine is set up, its RARP client program requests from the RARP server on the router to be sent its IP address. Assuming that an entry has been set up in the router table, the RARP server will return the IP address to the machine which can store it for future use.

RARP is available for Ethernet, Fiber Distributed-Data Interface, and token ring LAN

Types

There are four types of arp messages that may be sent by the arp protocol. These are identified by four values in the "operation" field of an arp message. The types of message are:

1. ARP request
2. ARP reply
3. RARP request
4. RARP reply

1. **Source Device Generates RARP Request Message:** The source device generates an RARP Request message. Thus, it uses the value 3 for the Opcode in the message. It puts its own data link layer address as both the Sender Hardware Address and also the Target Hardware Address. It leaves both the Sender Protocol Address and the Target Protocol Address blank, since it doesn't know either.
2. **Source Device Broadcasts RARP Request Message:** The source broadcasts the ARP Request message on the local network.
3. **Local Devices Process RARP Request Message:** The message is received by each device on the local network and processed. Devices that are not configured to act as RARP servers ignore the message.
4. **RARP Server Generates RARP Reply Message:** Any device on the network that is set up to act as an RARP server responds to the broadcast from the source device. It generates an RARP Reply using an Opcode value of 4.
5. **RARP Server Sends RARP Reply Message:** The RARP server sends the RARP Reply message unicast to the device looking to be configured.
6. **Source Device Processes RARP Reply Message:** The source device processes the reply from the RARP server. It then configures itself using the IP address in the Target Protocol Address supplied by the RARP server. It is possible that more than one RARP server may respond to any request, if two or more are configured on any local network. The source device will typically use the first reply and discard the others.

Short Question and Answers

1. What is Ethernet

Ans :

Meaning

Ethernet, defined under IEEE 802.3, is one of today's most widely used data communications standards, and it finds its major use in Local Area Network (LAN) applications. With versions including 10Base-T, 100Base-T and now Gigabit Ethernet, it offers a wide variety of choices of speeds and capability. Ethernet is also cheap and easy to install. Additionally Ethernet, IEEE 802.3 offers a considerable degree of flexibility in terms of the network topologies that are allowed. Furthermore as it is in widespread use in LANs, it has been developed into a robust system that meets the needs to wide number of networking requirements.

2. Various types of network elements.

Ans :

In a computer network, various types of network elements are used to perform different functions to ensure the efficient operation of the network. Some of the most common network elements include:

1. **Network Interface Card (NIC):** A network interface card is a hardware component that connects a computer to a network. It is responsible for managing the exchange of data between the computer and the network.
2. **Switch:** A switch is a networking device that connects multiple devices in a local area network (LAN). It forwards data packets between devices based on the Media Access Control (MAC) addresses of the devices.
3. **Router:** A router is a networking device that connects multiple networks and forwards data packets between them. It uses logical addresses (IP addresses) to direct traffic between different networks.
4. **Gateway:** A gateway is a networking device that acts as a bridge between different types of networks, such as between a local area network (LAN) and the internet. It translates data between different protocols and network architectures.

3. CSMA/CD Protocol

Ans :

CSMA/CD procedure can be understood as a group discussion, where if the participants speak all at once then it will be very confusing and the communication will not happen.

Instead, for good communication, it is required that the participants speak one after another so that we can clearly understand the contribution of each participant in the discussion.

Once a participant has finished talking, we should wait for a certain time period to see if any other participant is speaking or not. One should start speaking only when no other participant has spoken. If another participant also speaks at the same time, then we should stop, wait, and try again after some time.

4. Wireless LAN.

Ans :

IEEE 802.11 standard, popularly known as WiFi, lays down the architecture and specifications of wireless LANs (WLANs). WiFi or WLAN uses high-frequency radio waves instead of cables for connecting the devices in LAN. Users connected by WLANs can move around within the area of network coverage.

5. Bridges

Ans :

A bridge is a networking device that connects two or more network segments or LANs (Local Area Networks) together. Bridges are used to extend the reach of a network, reduce network congestion, and increase overall network performance. Here are the types of bridges in computer networks, along with their advantages and disadvantages:

6. Remote Bridge

Ans :

A remote bridge connects two LANs that use different data link protocols, such as Ethernet and Token Ring. Remote bridges can help to extend the reach of a network and improve overall network performance. However, they can also be more complex to configure and manage than local bridges.

Advantages

- Helps to extend the reach of a network.
- Improves overall network performance by reducing network congestion.
- Can help to integrate different types of LANs into a single network.

Disadvantages

- Can be more complex to configure and manage than local bridges.
- Requires additional hardware and software to convert between different data link protocols.
- Can introduce additional points of failure.

7. Transparent Bridge

Ans :

A transparent bridge connects two LANs that use the same data link protocol and appears to the network as a single network segment. Transparent bridges can help to reduce network congestion and improve overall network performance, but they can also introduce additional complexity to the network.

Advantages

- Reduces network congestion by only allowing relevant traffic to pass through the bridge.
- Helps to segment the network and reduce the effects of broadcast storms.
- Can improve overall network performance.

Disadvantages

- Can increase the complexity of the network.
- Requires additional configuration.
- Can introduce additional points of failure.

8. Wireless Bridge

Ans :

A wireless bridge connects two or more LANs wirelessly, using radio waves to transmit data. Wireless bridges can help to extend the reach of a network and provide greater mobility for users, but they can also be susceptible to interference and security risks.

Advantages

- Helps to extend the reach of a network.
- Provides greater mobility for users.
- Reduces the need for physical cabling.

Disadvantages

- Can be susceptible to interference and signal degradation.
- Can introduce additional security risks.
- May require additional hardware and software to implement.

9. Local Bridge

Ans :

A local bridge connects two LANs that use the same data link protocol, such as Ethernet. Local bridges can increase the capacity and performance of a network by reducing the number of packets that need to be transmitted across the network. However, they can also increase the complexity of the network and require additional configuration.

Advantages

- Increases the overall network capacity and performance.
- Reduces network congestion by only allowing relevant traffic to pass through the bridge.
- Helps to segment the network and reduce the effects of broadcast storms.

Disadvantages

- Can increase the complexity of the network.
- Requires additional configuration.
- Can introduce additional points of failure.

10. Address Resolution Protocol

Ans :

The address resolution protocol (arp) is a protocol used by the Internet Protocol (IP) [RFC826], specifically IPv4, to map IP network addresses to the hardware addresses used by a data link protocol. The protocol operates below the network layer as a part of the interface between the OSI network and OSI link layer. It is used when IPv4 is used over Ethernet.

Choose the Correct Answer

1. FDDI used which type of physical topology? [b]

   (a) Bus (b) Ring

   (c) Star (d) Tree

2. What kind of transmission medium is most appropriate to carry data in a computer network that is exposed to electrical interferences? [b]

   (a) Unshielded twisted pair (b) Optical fiber

   (c) Coaxial cable (d) Microwave

3. Which of the following is the standard for Ethernet? [b]

   (a) 802.11 (b) 802.3

   (c) 802.15 (d) 802.16

4. Which type of cable is used in 802.3 Ethernet? [d]

   (a) Coaxial cable (b) Twisted-pair cable

   (c) Fiber-optic cable (d) All of the above

5. Which Ethernet standard uses the CSMA/CD protocol ? [a]

   (a) 10BASE-T (b) 100BASE-TX

   (c) 1000BASE-T (d) 10GBASE-T

6. Which of the following is the standard for wireless local area network (WLAN)? [c]

   (a) IEEE 802.3 (b) IEEE 802.5

   (c) IEEE 802.11 (d) IEEE 802.16

7. Which of the following modulation techniques is used by IEEE 802.11g standard? [c]

   (a) BPSK (b) QPSK

   (c) OFDM (d) DSSS

8. Which of the following security protocols is used by IEEE 802.11i standard? [d]

   (a) WEP (b) WPA

   (c) TKIP (d) AES

9. What is the maximum range of IEEE 802.11b WLAN? [a]

   (a) 100 meters (b) 200 meters

   (c) 300 meters (d) 400 meters

10. The IEEE 802.11 standard for wireless LANs defines two services: _______ and _______ [c]

    (a) BSS; ASS (b) ESS; SSS

    (c) BSS; ESS (d) BSS; DCF

Fill in the blanks

1. If all devices are connected to a central hub, then topology is called ______
2. The Local Area Network(LAN) is defined by ______
3. The ______ are the points to and from which the communication takes place.
4. A ______ is a security device that monitors and controls the traffic between a network and the internet.
5. The maximum length of a 10BASE-T Ethernet cable is ______
6. What does the "CD" in CSMA/CD refer to______
7. The waiting time for which the station waits before retransmitting the data is called as ______
8. IEEE 802.11 standard, popularly known as ______
9. A ______is a group of stations communicating at physical layer level.
10. FHSS stands for ______

ANSWERS

1. Star topology
2. The geometric size of the network
3. Network nodes
4. Firewall
5. 100 meters
6. Collision detection
7. Back off time.
8. WiFi
9. Basic service set
10. Frequency Hoping Spread Spectrum

UNIT II

Network Layer: Logical Addressing-IPv4, Subnetting, and supernetting, CIDR, introduction to IPv6 ICMP, IGMP. Routing-Distance Vector Routing Link State Routing, OSPF and BGP.

2.1 Network Layer

2.1.1 Logical Addressing

Q1. Explain about logical addressing and IP addressing in network layer.

Ans : **(Imp.)**

In computer networks, logical addressing is a method of assigning addresses to devices at the Network Layer (Layer 3) of the OSI (Open Systems Interconnection) model. Logical addressing is used to identify devices on a network and to enable communication between them.

The most common form of logical addressing is IP (Internet Protocol) addressing. IP addressing is a hierarchical addressing scheme that consists of two parts: the network address and the host address. The network address identifies the network that the device is connected to, while the host address identifies the individual device on that network.

IP addresses are 32-bit binary numbers that are typically represented in decimal form, separated by dots. For example, the IP address 192.168.0.1 is a 32-bit binary number that represents a device on a network.

Logical addressing is important for several reasons:

1. **Routing:** Logical addressing enables routers to forward packets between networks based on the network address in the packet header.
2. **Network Management:** Logical addressing enables network administrators to manage and troubleshoot networks by identifying individual devices and tracking network traffic.
3. **Security:** Logical addressing enables security measures such as firewalls and access control lists to control access to network resources based on IP addresses.
4. **Scalability:** Logical addressing enables networks to scale to large numbers of devices by using a hierarchical addressing scheme that allows for the creation of subnetworks.

Overall, logical addressing is a critical component of computer networking that enables communication between devices on a network and enables network administrators to manage and control network

IP Address

An Internet Protocol address is an IP address. It is a unique address that identifies the device on the network. The Internet Service Provider (ISP) assigns IP addresses to all devices on its network. IP addresses are not generated at random. The Internet Assigned Numbers Authority (IANA), a part of the Internet Corporation for Assigned Names and Numbers (ICANN), generates and assigns them mathematically . IP addresses are used at the network layer. IP Addresses are routable in nature.

IP Version Types

There are 2 different versions of IP as follows.

1. **IPv4 (IP version 4)**

   IPv4 employs a 32-bit address. It is composed of four numbers separated by a ‘dot’ i.e., periods called an octet (byte). Each number in the octet can range from 0 to 255.

   Example: 172.166.3.28

2. **IPv6 (IP version 6)**

   IPv6 is the next generation of Internet Protocol addresses. In comparison to IPV4, IPv6 has a larger address space. IPv6 has a length of 128 bits and is written in hexadecimal. It is composed of eight fields, each of which contains two octets. As a result, IPv6 has 16 octets in total.

   Example:

   3221:1cd7:74b6:6da7:0000:0000:7349:6472

2.1.1.1 IPV4

Q2. Explain about IPv4 addresses.

Ans :

IPV4 addresses

IPv4 (Internet Protocol version 4) is a widely used protocol for sending data across the Internet. It uses a 32-bit address space to uniquely identify devices on a network. IPv4 addresses are represented in dot-decimal notation, where each 8-bit segment is separated by a period. For example, 192.168.1.1 is a typical IPv4 address.

IPv4 addresses are divided into two parts: the network portion and the host portion. The network portion identifies the network on which the device is located, while the host portion identifies the specific device on the network.

Address Space

The IPv4 (Internet Protocol version 4) address space is a 32-bit address space, which means there are 2 ^ 32 possible unique IPv4 addresses. IPv4 addresses are represented in dotted decimal notation, which consists of four 8-bit numbers separated by periods (dots). Each 8-bit number is also called an octet, and can have a value from 0 to 255.

For example, the IP address 192.168.0.1 is represented as four decimal numbers 192, 168, 0, and 1, where each number represents an octet of the address. This notation is also known as dotted quad notation.

IPv4 addresses are typically divided into two parts: the network address and the host address. The network address is used to identify the network to which the device belongs, while the host address identifies the specific device on the network.

To determine the network and host portions of an IPv4 address, a subnet mask is used. The subnet mask is a 32-bit number that specifies which bits of the IPv4 address represent the network portion and which bits represent the host portion.

For example, a subnet mask of 255.255.255.0 specifies that the first 24 bits (or the first three octets) of the IPv4 address represent the network portion, while the last 8 bits (or the last octet) represent the host portion.

Another notation used for IPv4 addresses is CIDR (Classless Inter-Domain Routing) notation, which consists of the IP address followed by a forward slash and a number indicating the number of bits in the subnet mask. For example, the IP address 192.168.0.1 with a subnet mask of 255.255.255.0 can be represented in CIDR notation as 192.168.0.1/24.

Hierarchical Addressing Scheme

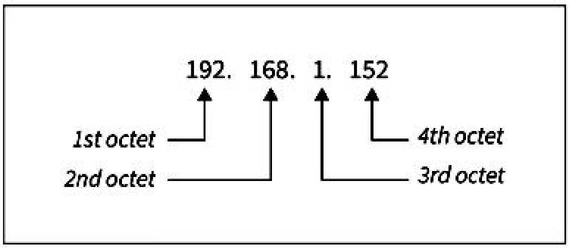
IPv4 uses hierarchical addressing scheme. An IP address, which is 32-bits in length, is divided into two or three parts as depicted

8 bits	8 bits	8 bits	8 bits
Network	Network	Sub-Network	Host

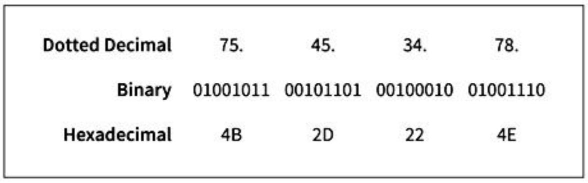
A single IP address can contain information about the network and its sub-network and ultimately the host. This scheme enables the IP Address to be hierarchical where a network can have many sub-networks which in turn can have many hosts.

IPv4 addresses are 32 bit-addresses and are divided into 4 octets(1 octet = 8 bits). They are usually represented in dotted decimals, but binary representation is another way to represent them.

Example : 192.168.1.152192.168.1.152



Let us see the different notations of an IPv4 address :



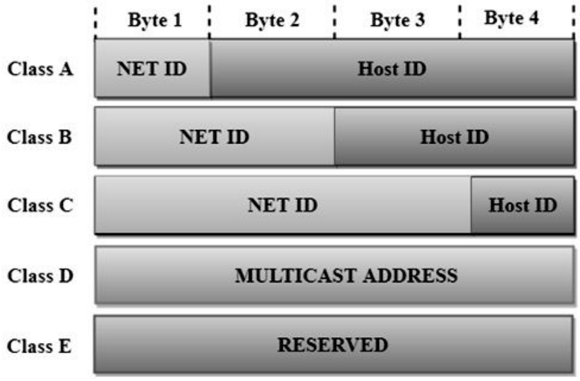
Dotted Decimal	75.	45.	34.	78.
Binary	01001011	00101101	00100010	01001110
Hexadecimal	4B	2D	22	4E

Q3. Explain about the various classes of IPv4.

Ans :

Classes of IPv4

IPv4 (Internet Protocol version 4) addresses are divided into five classes based on the first few bits of the address:



In the above diagram, we observe that each class have a specific range of IP addresses. The class of IP address is used to determine the number of bits used in a class and number of networks and hosts available in the class.

Class A

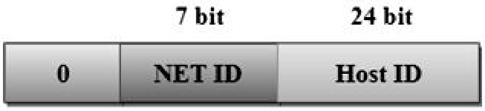
In Class A, an IP address is assigned to those networks that contain a large number of hosts.

- The network ID is 8 bits long.
- The host ID is 24 bits long.

In Class A, the first bit in higher order bits of the first octet is always set to 0 and the remaining 7 bits determine the network ID. The 24 bits determine the host ID in any network.

The total number of networks in Class A = 2^7 = 128 network address

The total number of hosts in Class A = $2^{24}-2$ = 16,777,214 host address



Class B

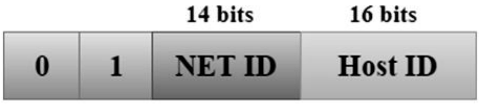
In Class B, an IP address is assigned to those networks that range from small-sized to large-sized networks.

- The Network ID is 16 bits long.
- The Host ID is 16 bits long.

In Class B, the higher order bits of the first octet is always set to 10, and the remaining14 bits determine the network ID. The other 16 bits determine the Host ID.

The total number of networks in Class B = 2^{14} = 16384 network address

The total number of hosts in Class B = 2^{16} - 2 = 65534 host address



Class C

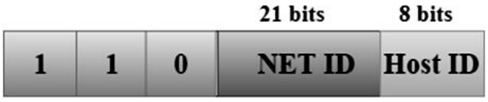
In Class C, an IP address is assigned to only small-sized networks.

- The Network ID is 24 bits long.
- The host ID is 8 bits long.

In Class C, the higher order bits of the first octet is always set to 110, and the remaining 21 bits determine the network ID. The 8 bits of the host ID determine the host in a network.

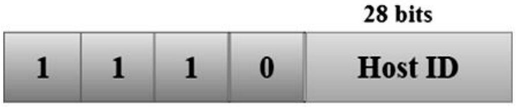
The total number of networks = 2^{21} = 2097152 network address

The total number of hosts = 2^8 - 2 = 254 host address



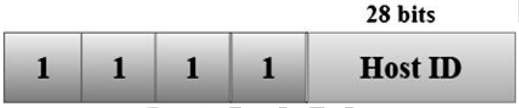
Class D

In Class D, an IP address is reserved for multicast addresses. It does not possess subnetting. The higher order bits of the first octet is always set to 1110, and the remaining bits determines the host ID in any network.



Class E

In Class E, an IP address is used for the future use or for the research and development purposes. It does not possess any subnetting. The higher order bits of the first octet is always set to 1111, and the remaining bits determines the host ID in any network.



Rules for assigning Host ID

The Host ID is used to determine the host within any network. The Host ID is assigned based on the following rules:

- The Host ID must be unique within any network.
- The Host ID in which all the bits are set to 0 cannot be assigned as it is used to represent the network ID of the IP address.
- The Host ID in which all the bits are set to 1 cannot be assigned as it is reserved for the multicast address.

Rules for assigning Network ID

If the hosts are located within the same local network, then they are assigned with the same network ID. The following are the rules for assigning Network ID:

- The network ID cannot start with 127 as 127 is used by Class A.
- The Network ID in which all the bits are set to 0 cannot be assigned as it is used to specify a particular host on the local network.
- The Network ID in which all the bits are set to 1 cannot be assigned as it is reserved for the multicast address.

Classful Network Architecture

Class	Higher bits	NET ID bits	HOST ID bits	No.of networks	No.of hosts per network	Range
A	0	8	24	2^7	2^{24}	0.0.0.0 to 127.255.255.255
B	10	16	16	2^{14}	2^{16}	128.0.0.0 to 191.255.255.255
C	110	24	8	2^{21}	2^8	192.0.0.0 to 223.255.255.255
D	1110	Not Defined	Not Defined	Not Defined	Not Defined	224.0.0.0 to 239.255.255.255
E	1111	Not Defined	Not Defined	Not Defined	Not Defined	240.0.0.0 to 255.255.255.255

2.1.1.2 Subnetting

Q4. What is subnetting? Explain, How to use subnetting.

Ans : **(Imp.)**

Subnetting is a technique used in computer networking to divide a larger network into smaller subnetworks, or subnets. This allows for more efficient use of IP addresses and better management of network traffic.

When subnetting, the network administrator divides the original network into smaller networks, each with its own unique network address. The subnet mask is used to divide the IP address space into two parts: the network part and the host part. The network part identifies the network, while the host part identifies the specific device on the network. The subnet mask determines the number of bits used for the network part and the number of bits used for the host part.

For example, consider an IP address of 192.168.1.1 with a subnet mask of 255.255.255.0. In this case, the first three octets (192.168.1) represent the network address, while the last octet (1) represents the host address. The subnet mask of 255.255.255.0 means that the first 24 bits (3 octets) are used for the network address, and the remaining 8 bits (1 octet) are used for the host address.

To subnet a network, the administrator can use various subnetting techniques to determine the number of subnets and the number of hosts per subnet. One common technique is the use of the subnet mask, which can be modified to create more subnets or more hosts per subnet. Another technique is variable-length subnet masking (VLSM), which allows for the creation of subnets with varying numbers of hosts.

To use subnetting, you need to have a good understanding of IP addressing and subnetting concepts. You can use subnetting calculators and tools to simplify the process of subnetting, but it's important to understand the underlying principles to ensure that your network is properly configured and optimized for performance. It's also important to carefully plan your subnetting scheme to ensure that it meets your network requirements and allows for future growth and scalability.

How to Subnet (or the Concept of Subnetting Explained)

To calculate subnets, you need to know the network address, subnet mask, and number of bits used for the network and host addresses. Here's an example of how to calculate subnets:

Let's say we have been assigned the IP address range 192.168.0.0/24 and we need to divide it into subnets with at least 30 hosts per subnet.

Step 1: Determine the number of bits required for the host portion of the address. In this case, we need at least 30 hosts per subnet, which requires 5 host bits (2 ^ 5 = 32).

Step 2: Determine the number of subnets required. To do this, we need to know the number of hosts required per subnet and then determine how many subnets are needed to meet that requirement. In this case, we need at least 30 hosts per subnet, which means we need to reserve 5 bits for the host portion of the address. This leaves us with 3 bits for the network portion of the address. 2 ^ 3 = 8, so we need at least 8 subnets to meet our requirements.

Step 3: Determine the new subnet mask. In this case, we are using 3 bits for the network portion of the address and 5 bits for the host portion of the address. The new subnet mask is 255.255.255.224 (binary: 11111111.11111111.11111111.11100000), which reserves the first 27 bits for the network portion of the address and the remaining 5 bits for the host portion of the address.

Step 4: Determine the subnet addresses. To determine the subnet addresses, we increment the network portion of the address by the number of subnets required. In this case, we need 8 subnets, so we increment the network portion of the address by 32 (2 ^ 5 = 32). The subnet addresses are:

- Subnet 1: 192.168.0.0/27
- Subnet 2: 192.168.0.32/27
- Subnet 3: 192.168.0.64/27
- Subnet 4: 192.168.0.96/27
- Subnet 5: 192.168.0.128/27
- Subnet 6: 192.168.0.160/27
- Subnet 7: 192.168.0.192/27
- Subnet 8: 192.168.0.224/27

Each of these subnets has a maximum of 30 hosts (2 ^ 5 - 2 = 30), with the first and last addresses reserved for the network address and broadcast address, respectively.

By subnetting our network in this way, we can more efficiently use our IP address space and better manage network traffic.

Q5. How to Subnet a Class C Address Using the Binary Method ? Explain.

Ans :

To subnet a Class C address using the binary method, follow these steps:

Step 1: Write down the Class C IP address in binary form

For example, let’s use the Class C IP address 192.168.1.0. In binary form, this is:

11000000.10101000.00000001.00000000

Step 2: Determine the subnet mask

The default subnet mask for a Class C address is 255.255.255.0. In binary form, this is:

11111111.11111111.11111111.00000000

Step 3: Determine the number of subnets required

Determine the number of subnets required based on your network requirements. For example, if you need 4 subnets, you need to use 2 bits for the subnet ID, as 2 ^ 2 = 4. This means that you will borrow 2 bits from the host ID portion of the IP address.

Step 4: Determine the new subnet mask

To determine the new subnet mask, you need to set the borrowed bits to 1 and the remaining bits to 0. In this example, we are borrowing 2 bits, so our new subnet mask will be:

11111111.11111111.11111111.11000000

In decimal form, this is 255.255.255.192.

Step 5: Determine the subnet IDs

To determine the subnet IDs, you need to divide the remaining bits in the host ID portion of the IP address into the number of subnets required. In this example, we are borrowing 2 bits, so we have 6 bits remaining in the host ID portion. This gives us 2 ^ 6 = 64 possible host addresses per subnet.

The subnet IDs can be determined by setting the borrowed bits to 0 and incrementing the remaining bits by the number of addresses per subnet. In this example, the subnet IDs would be:

- Subnet 1: 192.168.1.0
- Subnet 2: 192.168.1.64
- Subnet 3: 192.168.1.128
- Subnet 4: 192.168.1.192

Step 6: Determine the range of IP addresses for each subnet

To determine the range of IP addresses for each subnet, you need to use the subnet ID and the number of addresses per subnet. In this example, the range of IP addresses for each subnet would be:

- Subnet 1: 192.168.1.0 to 192.168.1.63
- Subnet 2: 192.168.1.64 to 192.168.1.127
- Subnet 3: 192.168.1.128 to 192.168.1.191
- Subnet 4: 192.168.1.192 to 192.168.1.255

By subnetting a Class C address in this way, you can more efficiently use your IP address space and better manage network traffic.

2.1.1.3 Supernetting

Q6. What is supernetting? Explain.

Ans :

Supernetting is the opposite of Subnetting. In subnetting, a single big network is divided into multiple smaller subnetworks. In Supernetting, multiple networks are combined into a bigger network termed as a Supernetwork or Supernet.

Supernetting is mainly used in Route Summarization, where routes to multiple networks with similar network prefixes are combined into a single routing entry, with the routing entry pointing to a Super network, encompassing all the networks. This in turn significantly reduces the size of routing tables and also the size of routing updates exchanged by routing protocols.

More specifically,

- When multiple networks are combined to form a bigger network, it is termed super-netting
- Super netting is used in route aggregation to reduce the size of routing tables and routing table updates

There are some points which should be kept in mind while supernetting:

1. All the Networks should be contiguous.
2. The block size of every network should be equal and must be in form of 2^n.
3. First Network id should be exactly divisible by whole size of supernet.

Example:

Suppose 4 small networks of class C:

200.1.0.0,

200.1.1.0,

200.1.2.0,

200.1.3.0

Build a bigger network that has a single Network Id.

Explanation:

Before Supernetting routing table will look like as:

Network Id	Subnet Mask	Interface
200.1.0.0	255.255.255.0	A
200.1.1.0	255.255.255.0	B
200.1.2.0	255.255.255.0	C
200.1.3.0	255.255.255.0	D

First, let's check whether three conditions are satisfied or not:

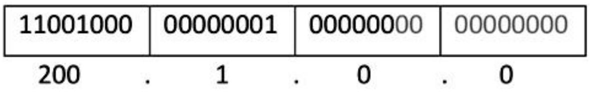
1. **Contiguous:** You can easily see that all networks are contiguous all having size 256 hosts.

   Range of first Network from 200.1.0.0 to 200.1.0.255. If you add 1 in last IP address of first network that is 200.1.0.255 + 0.0.0.1, you will get the next network id which is 200.1.1.0. Similarly, check that all network are contiguous.

2. **Equal size of all network:** As all networks are of class C, so all of them have a size of 256 which is in turn equal to 2^8.

3. **First IP address exactly divisible by total size:** When a binary number is divided by 2^n then last n bits are the remainder. Hence in order to prove that first IP address is exactly divisible by while size of Supernet Network. You can check that if last n v=bits are 0 or not.

   In the given example first IP is 200.1.0.0 and whole size of supernet is $4*2^8 = 2^{10}$. If last 10 bits of first IP address are zero then IP will be divisible.



11001000	00000001	00000000	00000000
200 .	1 .	0 .	0

Last 10 bits of first IP address are zero (highlighted by green color). So 3rd condition is also satisfied.

1. Control and reduce network traffic

   Helpful to solve the problem of lacking IP addresses

2. Minimizes the routing table

   - It cannot cover a different area of the network when combined.
   - All the networks should be in the same class and all IP should be contiguous.

2.1.2 CIDR

Q7. What is CIDR (Classless Inter-Domain Routing or supernetting)?

Ans : **(Imp.)**

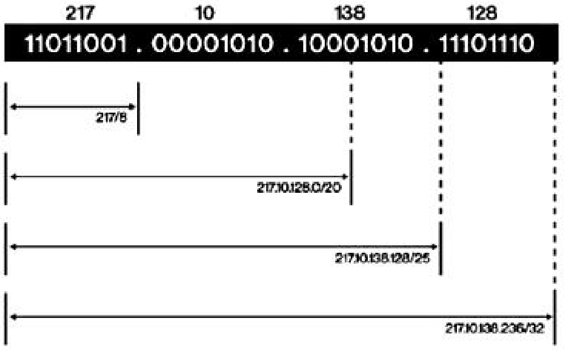
CIDR stands for Classless Inter-Domain Routing, which is a method used to allocate and manage IP addresses in a more flexible and efficient way than the traditional IP address classes. With CIDR, IP addresses are not bound to any predefined class boundaries, as was the case with the original Classful addressing scheme, and instead, can be divided into smaller subnets of various sizes.

CIDR is based on the use of a subnet mask, which is used to define the boundaries between the network portion and the host portion of an IP address. The subnet mask is expressed in the same dotted decimal format as IP addresses, but instead of representing the individual octets of an IP address, it represents the number of bits that are used for the network portion of the address. For example, a subnet mask of 255.255.255.0 would mean that the first 24 bits of an IP address are used for the network portion, and the remaining 8 bits are used for the host portion.

CIDR notation is a shorthand way of representing the subnet mask and the network address of an IP address using a forward slash and a number that represents the number of bits used for the network portion. For example, the IP address 192.168.1.0 with a subnet mask of 255.255.255.0 can be represented in CIDR notation as 192.168.1.0/24, where the 24 represents the number of bits used for the network portion of the address.

CIDR allows for the allocation of IP addresses in a more efficient way by allowing network administrators to allocate smaller subnets than would be possible with the traditional IP address classes. For example, instead of allocating a Class C network, which would provide 254 usable host addresses, a network administrator could allocate a /26 subnet, which would provide 62 usable host addresses, allowing for more efficient use of the available IP address space.

CIDR also allows for more efficient routing of IP traffic by reducing the size of routing tables in routers, as it allows for more specific routes to be defined for smaller subnets. This can help to reduce the amount of network traffic and improve network performance.



An example of CIDR

Example

Let's take the example of an organization that has been allocated the IP address range 192.168.0.0/16. This means that the organization has been allocated a Class B network with 16 bits allocated for the network portion of the IP address and 16 bits allocated for the host portion. The subnet mask for this network would be 255.255.0.0.

Now, the organization wants to create four subnets within their network, with each subnet having 200 host addresses. In order to do this, they would need to allocate a /23 subnet to each of the four subnets. A /23 subnet uses 23 bits for the network portion of the IP address and 9 bits for the host portion.

To calculate the network addresses for each of the four subnets, we would take the original network address of 192.168.0.0 and increment the third octet by 1 for each subnet. This would give us the following network addresses:

- Subnet 1: 192.168.1.0/23
- Subnet 2: 192.168.3.0/23
- Subnet 3: 192.168.5.0/23
- Subnet 4: 192.168.7.0/23

Each of these subnets would have a total of 512 addresses (2 ^ 9 - 2), with 510 of those addresses being usable for hosts. The organization would then be able to allocate IP addresses from each of these subnets to devices on their network, and routers would be able to route traffic to these subnets more efficiently, as each subnet would have a more specific route defined in the routing table.

2.1.3 Introduction To Ipv6

Q8. Explain IPV6 addressing with an example.

Ans :

IPv6 is the latest version of the Internet Protocol that is designed to replace the aging IPv4 protocol. IPv6 addresses are 128 bits long, which provides for a much larger address space than IPv4's 32 bits, allowing for virtually unlimited addresses.

IPv6 addresses are typically represented using hexadecimal notation, which uses digits 0-9 and letters A-F. The address is divided into eight 16-bit blocks separated by colons, for a total of 32 hexadecimal characters. Leading zeroes can be omitted in each block, and consecutive blocks of zeroes can be compressed with a double colon. For example, the loopback address in IPv6 is represented as ::1, which represents eight blocks of zeroes followed by a single block of 1.

Here's an example of an IPv6 address in full notation:

2001:0db8:85a3:0000:0000:8a2e:0370:7334

n this example, each block is represented by four hexadecimal characters, separated by colons. Note that leading zeroes are included in each block.

IPv6 also supports a shorthand notation where the leading zeroes can be omitted from each block and consecutive blocks of zeroes can be compressed with a double colon. For example, the same address as above can be represented in shorthand notation as:

2001:db8:85a3::8a2e:370:7334

In this example, the two blocks of consecutive zeroes are compressed with a double colon.

IPv6 addresses are divided into two main parts: the network prefix and the interface identifier. The network prefix is used to identify the network portion of the address, while the interface identifier identifies the individual interface of the device on the network.

In IPv6, the network prefix is often represented using CIDR notation, where the number after the slash (/) represents the number of bits used for the network prefix. For example, the address 2001:db8:abcd:1234::/64 indicates that the first 64 bits of the address are used for the network prefix, and the remaining 64 bits are used for the interface identifier.

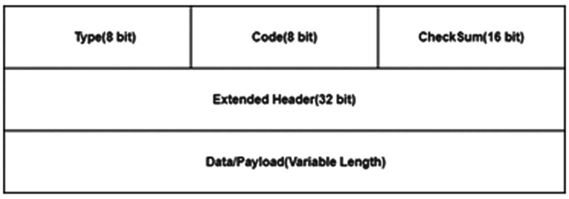
2.1.4 ICMP

Q9. Explain briefly about ICMP protocol.

Ans : **(Imp.)**

Since IP does not have an inbuilt mechanism for sending error and control messages. It depends on Internet Control Message Protocol(ICMP) to provide an error control. It is used for reporting errors and management queries. It is a supporting protocol and is used by networks devices like routers for sending error messages and operations information., e.g. the requested service is not available or that a host or router could not be reached.

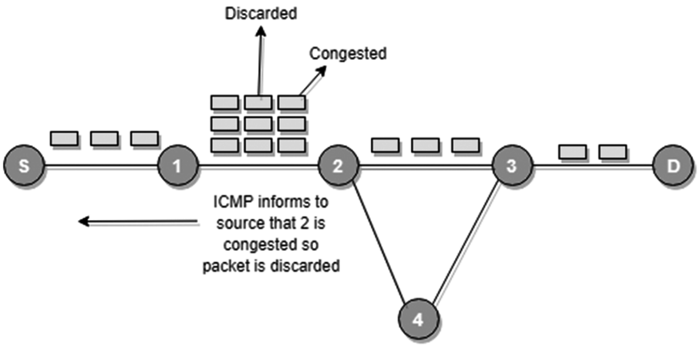
ICMPv4 Packet Format



Type(8 bit)	Code(8 bit)	CheckSum(16 bit)
Extended Header(32 bit)		
Data/Payload(Variable Length)		

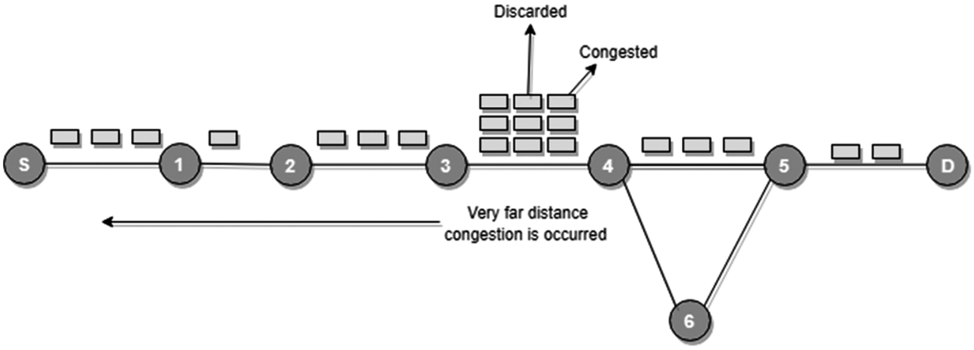
Source Quench Message

Source quench message is a request to decrease the traffic rate for messages sending to the host(destination). Or we can say when receiving host detects that the rate of sending packets (traffic rate) to it is too fast it sends the source quench message to the source to slow the pace down so that no packet can be lost.


Discarded
Congested
S
1
2
3
D
4
ICMP informs to source that 2 is congested so packet is discarded

ICMP will take the source IP from the discarded packet and informs the source by sending a source quench message.

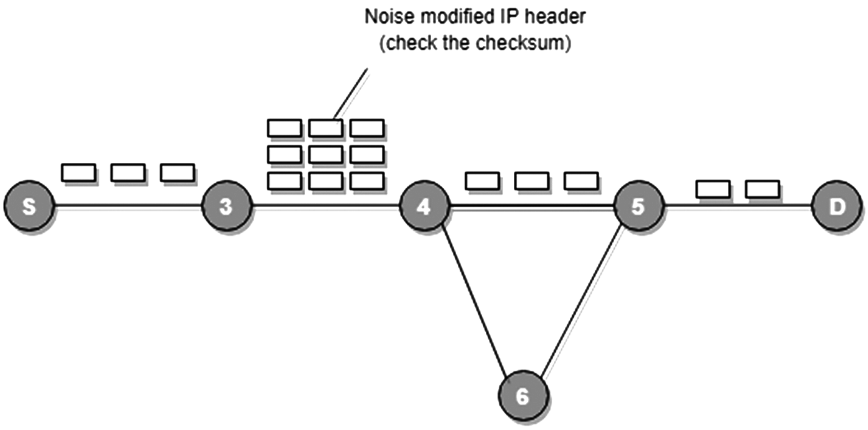
Then source will reduce the speed of transmission so that router will be free from congestion.



When the congestion router is far away from the source the ICMP will send hop by hop source quench message so that every router will reduce the speed of transmission.

Parameter problem

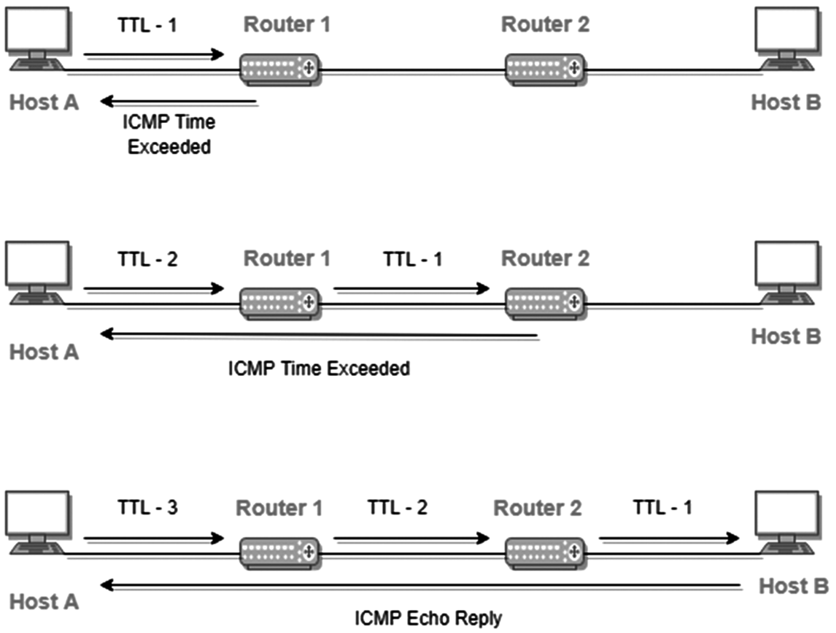
Whenever packets come to the router then the calculated header checksum should be equal to the received header checksum then the only the packet is accepted by the router.



If there is a mismatch packet will be dropped by the router.

ICMP will take the source IP from the discarded packet and informs to the source by sending a parameter problem message.

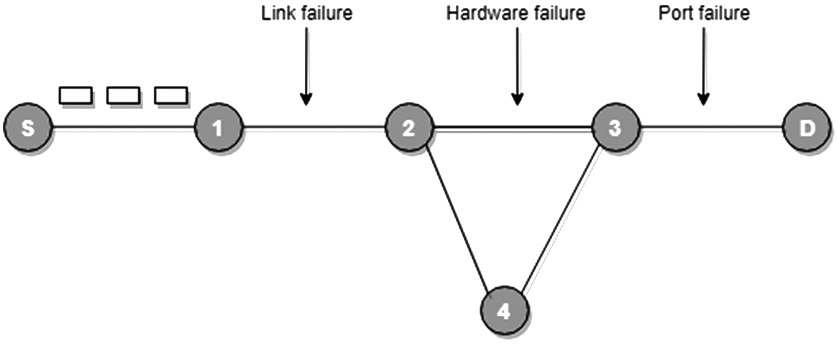
Time Exceeded Message



When some fragments are lost in a network then the holding fragment by the router will be dropped then ICMP will take the source IP from the discarded packet and informs the source, of discarded datagram due to time to live field reaches zero, by sending time exceeded message.

Destination un-reachable

Destination unreachable is generated by the host or its inbound gateway to inform the client that the destination is unreachable for some reason.



There is no necessary condition that the only the router gives the ICMP error message some time the destination host sends an ICMP error message when any type of failure (link failure, hardware failure, port failure, etc) happens in the network.

Redirection Message

Redirect requests data packets are sent on an alternate route. The message informs a host to update its routing information (to send packets on an alternate route).

Ex. If the host tries to send data through a router R1 and R1 sends data on a router R2 and there is a direct way from the host to R2. Then R1 will send a redirect message to inform the host that there is the best way to the destination directly through R2 available. The host then sends data packets for the destination directly to R2.

The router R2 will send the original datagram to the intended destination.

But if the datagram contains routing information then this message will not be sent even if a better route is available as redirects should only be sent by gateways and should not be sent by Internet hosts.

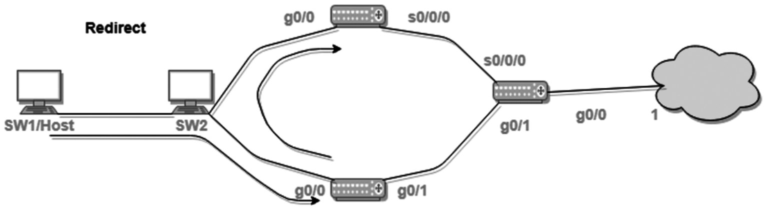


Figure - ICMP redirect Verification CCNP 2.0 100 - 101 (v - 71)

- ✓ ICMP Redirect
- ✓ ICMP Redirect for host
- ✓ ICMP Redirect for network
- ✓ How ICMP redirect work
- ✓ ICMP Redirect verification step by step

Whenever a packet is forwarded in a wrong direction later it is re-directed in a current direction then ICMP will send a re-directed message.

Types of ICMP (Internet Control Message Protocol) Messages

Internet Control Message Protocol (ICMP) is a protocol used for error handling and debugging in the network layer. It is mainly used in network devices such as routers.

ICMP messages are mainly divided into two categories :

1. Error Reporting Messages

Error Reporting Messages are used to report problems encountered by the router/host while processing the IP packets. These messages are always sent to the source because the datagram only contains the source and destination IP address.

Following are the types of error reporting messages :

- **Destination Unreachable Message:** The host/router send this message if it is not able to route the IP packet to its destination.

  For example, sender A wants to send the datagram to receiver B but it is not received by B then the intermediate router will discard the datagram and send the destination unreachable message to A.

- **Source Quench Message:** Host/router send this message if there is congestion in the network or the source is sending packets at a higher rate which the router can't handle.

  For example, if sender A is sending the data packets at a high data rate which the router is unable to handle then it will discard the packet and send a source quench message to A to tell it to send the packets at a lower rate. Now, after receiving the message A will either stop or slow down sending of the packets.

- **Time Exceeded Message:** The host/router sends this message if it decrements the time to live value of the datagram to zero or the destination address does not receive all the packets in the specified time interval.

  For example, a packet is sent from a layer having 1000 units to the layer having 200 units, then the packet is divided into five fragments. If all the fragments don't reach the destination in a set time, all fragments are discarded and the time-exceeded message is sent to the original source.

- **Parameter Problem Message:** The host/ router sends this message if some parameter is not properly set in the datagram. It is used to indicate errors in the header field of the datagram.

- **Redirection Message:** The host/router sends this message to update the routing table of the host.

  For example, sender A wants to send the message to receiver B and there is a router between them. Then, A sends the data to the router and the router sends the message to B and redirection message to A so that A can update its routing table.

2. Query Messages

Query Messages are used for error handling and debugging network problems. These messages help the host to get specific information about another host or router.

Following are the types of query messages :

- **Echo-Request and Reply Message:** It is used by the network managers to check the operations of the IP protocol and host's reachability. The host/router sends the echo request message, while the echo reply message is sent by the host/router that receives an echo request message.

  For example, host A wants to check whether it can communicate with another host B so it will send an echo-request to B and if there is a link between A and B and B is active then it will send echo-reply to A on receiving the request.

- **Timestamp Request and Reply Message:** It is used to determine the round trip time taken by each IP datagram to travel from one host to another host. It can also synchronize the clocks between hosts if transit time is known.

  For example, host A wants to synchronize its clock with B then it will ask time on B's clock by sending timestamp request and B will reply with the timestamp to A. Then A will add the time and propagation delay to synchronize the time on its system.

- **Address Mask Request and Reply Message:** It is used to determine the subnet mask used in the local network.

  For example, host A wants to know the mask of the corresponding router then it will send an address mask request to the router if the address of the router is known else it will broadcast the request. The router that receives the request responds with an address mask reply and provides the mask to A.

- **Router Solicitation and Advertisement Message:** It is used to confirm the presence of a router on the local area network. It is done by broadcasting a router solicitation message and the router that receives the message broadcasts the routing information using a router advertisement message.

  For example, if host A wants to get the information of routers present on the network. It will broadcast a router solicitation message to request routers to advertise their presence and in response router on the network will announce its IP address available for routing through advertisement message.

2.1.5 IGMP

Q10. Explain about IGMP protocol.

Ans : **(Imp.)**

IGMP is acronym for Internet Group Management Protocol. IGMP is a communication protocol used by hosts and adjacent routers for multicasting communication with IP networks and uses the resources efficiently to transmit the message/data packets. Multicast communication can have single or multiple senders and receivers and thus, IGMP can be used in streaming videos, gaming or web conferencing tools. This protocol is used on IPv4 networks and for using this on IPv6, multicasting is managed by Multicast Listener Discovery (MLD). Like other network protocols, IGMP is used on network layer. MLDv1 is almost same in functioning as IGMPv2 and MLDv2 is almost similar to IGMPv3. The communication protocol, IGMPv1 was developed in 1989 at Stanford University. IGMPv1 was updated to IGMPv2 in year 1997 and again updated to IGMPv3 in year 2002.

Applications:

- **Streaming:** Multicast routing protocol are used for audio and video streaming over the network i.e., either one-to-many or many-to-many.

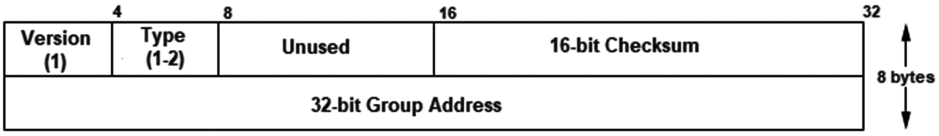
- **Gaming:** Internet group management protocol is often used in simulation games which has multiple users over the network such as online games.
- **Web Conferencing Tools:** Video conferencing is a new method to meet people from your own convenience and IGMP connects to the users for conferencing and transfers the message/data packets efficiently.

Types: There are 3 versions of IGMP. These versions are backward compatible. Following are the versions of IGMP:

1. IGMPv1

The version of IGMP communication protocol allows all the supporting hosts to join the multicast groups using membership request and include some basic features. But, host cannot leave the group on their own and have to wait for a timeout to leave the group. The message packet format in IGMPv1:

IGMPv1 Packet Format

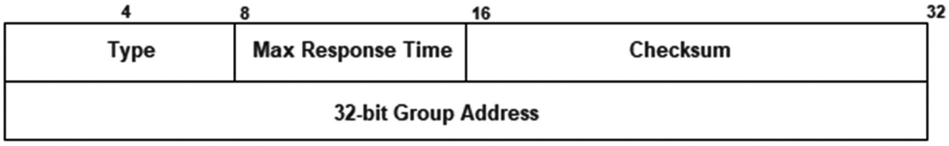


- **Version:** Set to 1.
- **Type:** 1 for Host Membership Query and Host Membership Report.
- **Unused:** 8-bits of zero which are of no use.
- **Checksum:** It is the one's complement of the sum of IGMP messages.
- **Group Address:** The group address field is zero when sent and ignored when received in membership query message. In a membership report message, the group address field takes the IP host group address of the group being reported.

2. IGMPv2

IGMPv2 is the revised version of IGMPv1 communication protocol. It has added functionality of leaving the multicast group using group membership. The message packet format in IGMPv2:

IGMPv2 Packet Format



Type:

0x11 for Membership Query

0x12 for IGMPv1 Membership Report

0x16 for IGMPv2 Membership Report

0x22 for IGMPv3 Membership Report

0x17 for Leave Group

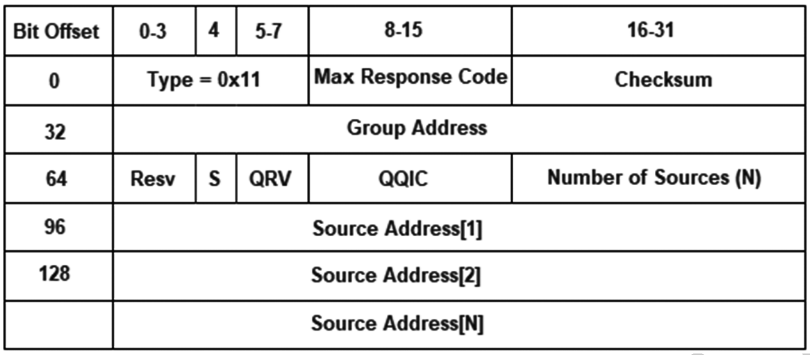
- **Max Response Time:** This field is ignored for message types other than membership query. For membership query type, it is the maximum time allowed before sending a response report. The value is in units of 0.1 seconds.
- **Checksum:** It is the one's complement of the sum of IGMP message.

- **Group Address:** It is set as 0 when sending a general query. Otherwise, multicast address for group-specific or source-specific queries.

3. IGMPv3

IGMPv2 was revised to IGMPv3 and added source-specific multicast and membership report aggregation. These reports are sent to 224.0.0.22. The message packet format in IGMPv3:

IGMPv3 Packet Format

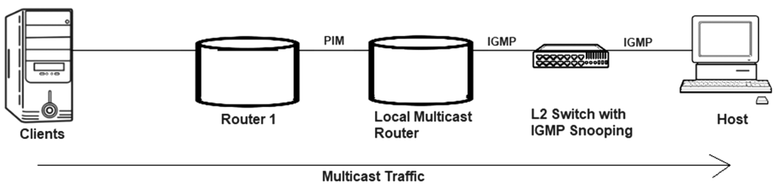


Bit Offset	0-3	4	5-7	8-15	16-31
0	Type = 0x11			Max Response Code	Checksum
32	Group Address				
64	Resv	S	QRV	QQIC	Number of Sources (N)
96	Source Address[1]				
128	Source Address[2]				
	Source Address[N]				

- **Max Response Time:** This field is ignored for message types other than membership query. For membership query type, it is the maximum time allowed before sending a response report. The value is in units of 0.1 seconds.
- **Checksum:** It is the one's complement of the one's complement of the sum of IGMP message.
- **Group Address:** It is set as 0 when sending a general query. Otherwise, multicast address for group-specific or source-specific queries.
- **Resv:** It is set zero of sent and ignored when received.
- **S flag:** It represents Suppress Router-side Processing flag. When the flag is set, it indicates to suppress the timer updates that multicast routers perform upon receiving any query.
- **QRV:** It represents Querier's Robustness Variable. Routers keeps on retrieving the QRV value from the most recently received query as their own value until the most recently received QRV is zero.
- **QQIC:** It represents Querier's Query Interval Code.
- **Number of Sources:** It represents the number of source addresses present in the query. For general query or group-specific query, this field is zero and for group-and-source-specific query, this field is non-zero.
- **Source Address[i]:** It represents the IP unicast address for N fields.

Working: IGMP works on devices that are capable of handling multicast groups and dynamic multicasting. These devices allows the host to join or leave the membership in the multicast group. These devices also allows to add and remove clients from the group. This communication protocol is operated between host and local multicast router. When a multicast group is created, the multicast group address is in range of class D (224-239) IP addresses and is forwarded as destination IP address in the packet.

Working of IGMP



L2 or Level-2 devices such as switches are used in between host and multicast router for IGMP snooping. IGMP snooping is a process to listen to the IGMP network traffic in controlled manner. Switch receives the message from host and forwards the membership report to the local multicast router. The multicast traffic is further forwarded to remote routers from local multicast routers using PIM (Protocol Independent Multicast) so that clients can receive the message/data packets. Clients wishing to join the network sends join message in the query and switch intercepts the message and adds the ports of clients to its multicast routing table.

Advantages

- IGMP communication protocol efficiently transmits the multicast data to the receivers and so, no junk packets are transmitted to the host which shows optimized performance.
- Bandwidth is consumed totally as all the shared links are connected.
- Hosts can leave a multicast group and join another.

Disadvantages

- It does not provide good efficiency in filtering and security.
- Due to lack of TCP, network congestion can occur.
- IGMP is vulnerable to some attacks such as DOS attack (Denial-Of-Service).

2.2 Routing

Q11. Define routing. Explain different types of routings.

Ans : **(Imp.)**

Routing is a process that is performed by layer 3 (or network layer) devices in order to deliver the packet by choosing an optimal path from one network to another.

There are 3 types of routing:

1. Static Routing

Static routing is a process in which we have to manually add routes to the routing table.

Advantages

- No routing overhead for router CPU which means a cheaper router can be used to do routing.
- It adds security because an only administrator can allow routing to particular networks only.
- No bandwidth usage between routers.

Disadvantage

- For a large network, it is a hectic task for administrators to manually add each route for the network in the routing table on each router.
- The administrator should have good knowledge of the topology. If a new administrator comes, then he has to manually add each route so he should have very good knowledge of the routes of the topology.

2. Default Routing

This is the method where the router is configured to send all packets towards a single router (next hop). It doesn't matter to which network the packet belongs, it is forwarded out to the router which is configured for default routing. It is generally used with stub routers. A stub router is a router that has only one route to reach all other networks.

3. Dynamic Routing

Dynamic routing makes automatic adjustments of the routes according to the current state of the route in the routing table. Dynamic routing uses protocols to discover network destinations and the routes to reach them. RIP and OSPF are the best examples of dynamic routing protocols. Automatic adjustments will be made to reach the network destination if one route goes down.

A dynamic protocol has the following features:

1. The routers should have the same dynamic protocol running in order to exchange routes.
2. When a router finds a change in the topology then the router advertises it to all other routers.

Advantages

- Easy to configure.
- More effective at selecting the best route to a destination remote network and also for discovering remote network.

Disadvantage

- Consumes more bandwidth for communicating with other neighbors.
- Less secure than static routing

2.2.1 Distance Vector Routing

Q12. Explain about Distance vector routing.

Ans : **(Imp.)**

A distance-vector routing (DVR) protocol requires that a router inform its neighbors of topology changes periodically. Historically known as the old ARPANET routing algorithm (or known as Bellman-Ford algorithm).

It works in the following steps :

Step-01:

Each router prepares its routing table. By their local knowledge. each router knows about-

- All the routers present in the network
- Distance to its neighboring routers

Step-02:

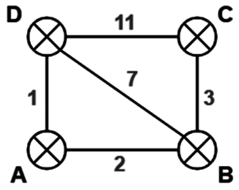
- Each router exchanges its distance vector with its neighboring routers.
- Each router prepares a new routing table using the distance vectors it has obtained from its neighbors.
- This step is repeated for (n-2) times if there are n routers in the network.
- After this, routing tables converge / become stable.

Distance Vector Routing Example

Consider-

There is a network consisting of 4 routers.

- The weights are mentioned on the edges.
- Weights could be distances or costs or delays.



Step-01

Each router prepares its routing table using its local knowledge.

Routing table prepared by each router is shown below-

At Router A

Destination	Distance	Next Hop
A	0	A
B	2	B
C	∞	–
D	1	D

At Router B

Destination	Distance	Next Hop
A	2	A
B	0	B
C	3	C
D	7	D

At Router C

Destination	Distance	Next Hop
A	∞	–
B	3	B
C	0	C
D	11	D

At Router D

Destination	Distance	Next Hop
A	1	A
B	7	B
C	11	C
D	0	D

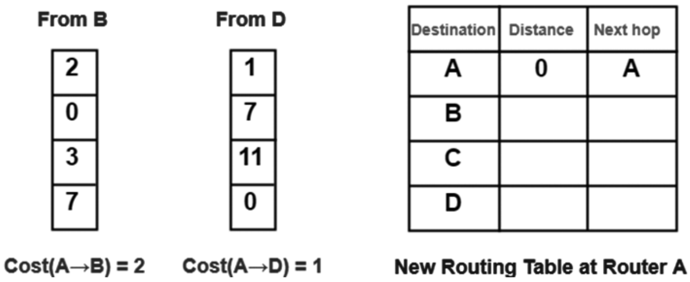
Step-02:

- ➢ Each router exchanges its distance vector obtained in Step-01 with its neighbors.
- ➢ After exchanging the distance vectors, each router prepares a new routing table.

This is shown below

At Router A

- ➢ Router A receives distance vectors from its neighbors B and D.
- ➢ Router A prepares a new routing table as-



From B	From D
2	1
0	7
3	11
7	0

Cost(A→B) = 2 Cost(A→D) = 1

Destination	Distance	Next hop
A	0	A
B		
C		
D		

New Routing Table at Router A

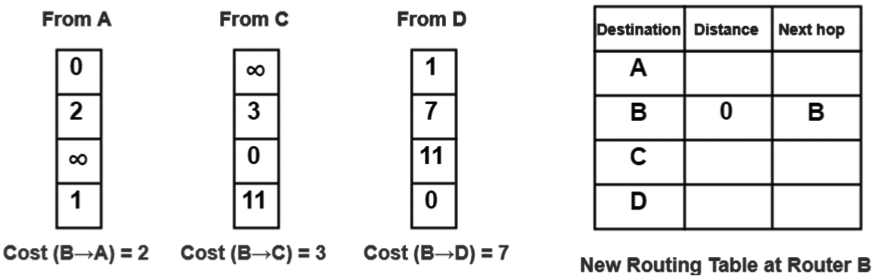
- ➢ Cost of reaching destination B from router A = min {2 + 0, 1 + 7} = 2 via B.Cost of reaching destination C from router A = min { 2+3 , 1+11 } = 5 via B.
- ➢ Cost of reaching destination D from router A = min {2 + 7, 1 + 0} = 1 via D.

Thus, the new routing table at router A is-

Destination	Distance	Next Hop
A	0	A
B	2	B
C	5	B
D	1	D

At Router B

- ➢ Router B receives distance vectors from its neighbors A, C and D.
- ➢ Router B prepares a new routing table as-



From A	From C	From D
0	∞	1
2	3	7
∞	0	11
1	11	0

Cost (B→A) = 2 Cost (B→C) = 3 Cost (B→D) = 7

Destination	Distance	Next hop
A		
B	0	B
C		
D		

New Routing Table at Router B

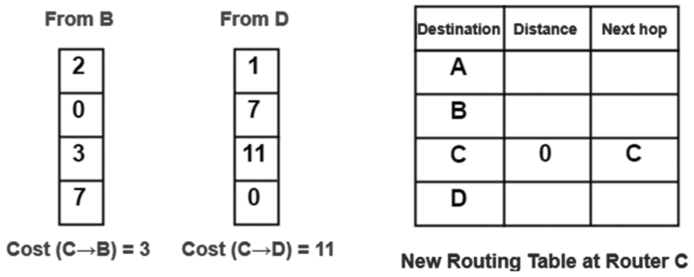
- ➢ Cost of reaching destination A from router B = min {2+0 , 3+ ∞ , 7+1 } = 2 via A.
- ➢ Cost of reaching destination C from router B = min {2+ ∞ , 3+0 , 7+11 } = 3 via C.
- ➢ Cost of reaching destination D from router B = min {2+1 , 3+11 , 7+0 } = 3 via A.

Thus, the new routing table at router B is

Destination	Distance	Next Hop
A	2	A
B	0	B
C	3	C
D	3	A

At Router C

- Router C receives distance vectors from its neighbors B and D.
- Router C prepares a new routing table as-



From B
2
0
3
7

Cost (C→B) = 3

From D
1
7
11
0

Cost (C→D) = 11

Destination	Distance	Next hop
A		
B		
C	0	C
D		

New Routing Table at Router C

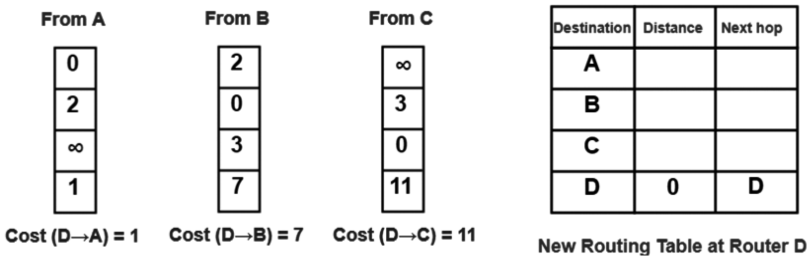
- Cost of reaching destination A from router C = min { 3+2 , 11+1 } = 5 via B.
- Cost of reaching destination B from router C = min { 3+0 , 11+7 } = 3 via B.
- Cost of reaching destination D from router C = min { 3+7 , 11+0 } = 10 via B.

Thus, the new routing table at router C is-

Destination	Distance	Next Hop
A	5	B
B	3	B
C	0	C
D	10	B

At Router D-

- Router D receives distance vectors from its neighbors A, B and C.
- Router D prepares a new routing table as-



From A
0
2
∞
1

Cost (D→A) = 1

From B
2
0
3
7

Cost (D→B) = 7

From C
∞
3
0
11

Cost (D→C) = 11

Destination	Distance	Next hop
A		
B		
C		
D	0	D

New Routing Table at Router D

- ➢ Cost of reaching destination A from router D = min { 1+0 , 7+2 , 11+ ∞ } = 1 via A.
- ➢ Cost of reaching destination B from router D = min { 1+2 , 7+0 , 11+3 } = 3 via A.
- ➢ Cost of reaching destination C from router D = min { 1+ ∞ , 7+3 , 11+0 } = 10 via B.

Thus, the new routing table at router D is-

Destination	Distance	Next Hop
A	1	A
B	3	A
C	10	B
D	0	D

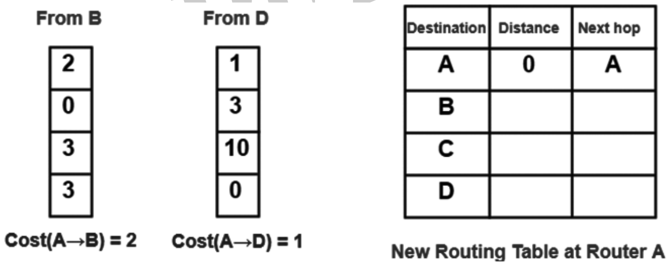
Step-03:

- ➢ Each router exchanges its distance vector obtained in Step-02 with its neighboring routers.
- ➢ After exchanging the distance vectors, each router prepares a new routing table.

This is shown below-

At Router A

- ➢ Router A receives distance vectors from its neighbors B and D.
- ➢ Router A prepares a new routing table as-



Destination	Distance	Next hop
A	0	A
B		
C		
D		

New Routing Table at Router A

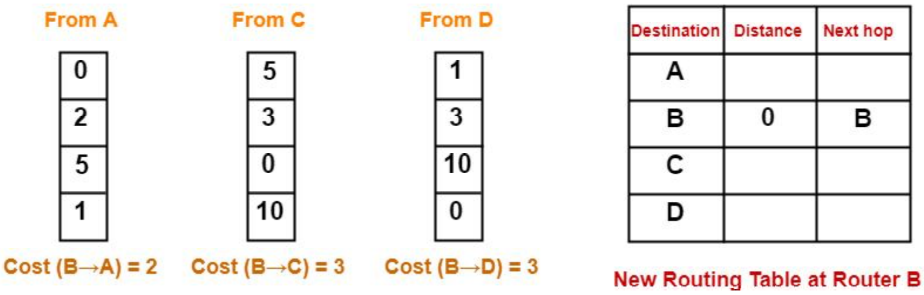
- ➢ Cost of reaching destination B from router A = min { 2+0 , 1+3 } = 2 via B.
- ➢ Cost of reaching destination C from router A = min { 2+3 , 1+10 } = 5 via B.
- ➢ Cost of reaching destination D from router A = min { 2+3 , 1+0 } = 1 via D.

Thus, the new routing table at router A is-

Destination	Distance	Next Hop
A	0	A
B	2	B
C	5	B
D	1	D

At Router B-

- Router B receives distance vectors from its neighbors A, C and D.
- Router B prepares a new routing table as-



From A	From C	From D
0	5	1
2	3	3
5	0	10
1	10	0
Cost (B→A) = 2	Cost (B→C) = 3	Cost (B→D) = 3

Destination	Distance	Next hop
A		
B	0	B
C		
D		

New Routing Table at Router B

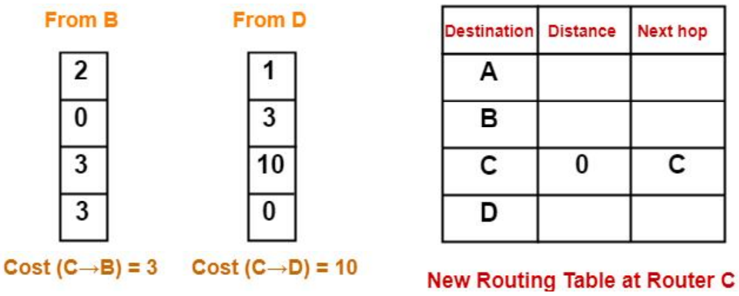
- Cost of reaching destination A from router B = min { 2+0 , 3+5 , 3+1 } = 2 via A.
- Cost of reaching destination C from router B = min { 2+5 , 3+0 , 3+10 } = 3 via C.
- Cost of reaching destination D from router B = min { 2+1 , 3+10 , 3+0 } = 3 via A.

Thus, the new routing table at router B is-

Destination	Distance	Next Hop
A	2	A
B	0	B
C	3	C
D	3	A

At Router C-

- Router C receives distance vectors from its neighbors B and D.
- Router C prepares a new routing table as-



From B	From D
2	1
0	3
3	10
3	0
Cost (C→B) = 3	Cost (C→D) = 10

Destination	Distance	Next hop
A		
B		
C	0	C
D		

New Routing Table at Router C

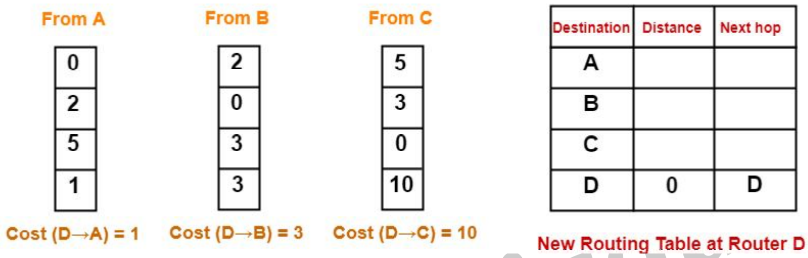
- Cost of reaching destination A from router C = min { 3+2 , 10+1 } = 5 via B.
- Cost of reaching destination B from router C = min { 3+0 , 10+3 } = 3 via B.
- Cost of reaching destination D from router C = min { 3+3 , 10+0 } = 6 via B.

Thus, the new routing table at router C is-

Destination	Distance	Next Hop
A	5	B
B	3	B
C	0	C
D	6	B

At Router D-

- Router D receives distance vectors from its neighbors A, B and C.
- Router D prepares a new routing table as-



Destination	Distance	Next hop
A		
B		
C		
D	0	D

New Routing Table at Router D

- Cost of reaching destination A from router D = min { 1+0 , 3+2 , 10+5 } = 1 via A.
- Cost of reaching destination B from router D = min { 1+2 , 3+0 , 10+3 } = 3 via A.
- Cost of reaching destination C from router D = min { 1+5 , 3+3 , 10+0 } = 6 via A.

Thus, the new routing table at router D is-

Destination	Distance	Next Hop
A	1	A
B	3	A
C	6	A
D	0	D

These will be the final routing tables at each router.

Identifying Unused Links

After routing tables converge (becomes stable),

- Some of the links connecting the routers may never be used.
- In the above example, we can identify the unused links as-

We have-

- The value of next hop in the final routing table of router A suggests that only edges AB and AD are used.
- The value of next hop in the final routing table of router B suggests that only edges BA and BC are used.

- The value of next hop in the final routing table of router C suggests that only edge CB is used.
- The value of next hop in the final routing table of router D suggests that only edge DA is used.

Thus, edges BD and CD are never used.

2.2.2 Link State Routing

Q13. Explain about Link State Routing with example.

Ans : **(Imp.)**

Link state routing is a technique in which each router shares the knowledge of its neighborhood with every other router in the internetwork.

The three keys to understand the Link State Routing algorithm:

- **Knowledge about the neighborhood:** Instead of sending its routing table, a router sends the information about its neighborhood only. A router broadcast its identities and cost of the directly attached links to other routers.
- **Flooding:** Each router sends the information to every other router on the internetwork except its neighbors. This process is known as Flooding. Every router that receives the packet sends the copies to all its neighbors. Finally, each and every router receives a copy of the same information.
- **Information sharing:** A router sends the information to every other router only when the change occurs in the information.

Link State Routing has two phases:

Reliable Flooding

- **Initial state:** Each node knows the cost of its neighbors.

o **Final state:** Each node knows the entire graph.

Route Calculation

Each node uses Dijkstra's algorithm on the graph to calculate the optimal routes to all nodes.

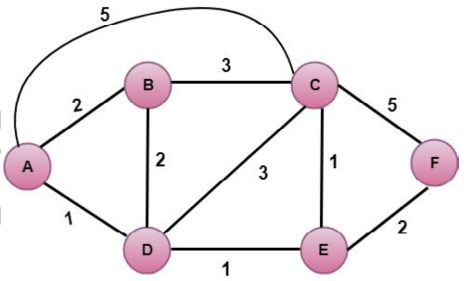
- The Link state routing algorithm is also known as Dijkstra's algorithm which is used to find the shortest path from one node to every other node in the network.
- The Dijkstra's algorithm is an iterative, and it has the property that after kth iteration of the algorithm, the least cost paths are well known for k destination nodes.

Let's describe some notations:

- **c(i , j):** Link cost from node i to node j. If i and j nodes are not directly linked, then c(i , j) = ".
- **D(v):** It defines the cost of the path from source code to destination v that has the least cost currently.
- **P(v):** It defines the previous node (neighbor of v) along with current least cost path from source to v.
- **N:** It is the total number of nodes available in the network.

In the above algorithm, an initialization step is followed by the loop. The number of times the loop is executed is equal to the total number of nodes available in the network.

Let's understand through an example:



In the above figure, source vertex is A.

Step 1:

The first step is an initialization step. The currently known least cost path from A to its directly attached neighbors, B, C, D are 2,5,1 respectively. The cost from A to B is set to 2, from A to D is set to 1 and from A to C is set to 5. The cost from A to E and F are set to infinity as they are not directly linked to A.

Step	N	D(B),P(B)	D(C),P(C)	D(D),P(D)	D(E),P(E)	D(F),P(F)
1	A	2,A	5,A	1,A	∞	∞

Step 2:

In the above table, we observe that vertex D contains the least cost path in step 1. Therefore, it is added in N. Now, we need to determine a least-cost path through D vertex.

(a) Calculating shortest path from A to B

$v = B, w = D$

$D(B) = \min(D(B) , D(D) + c(D,B))$

$= \min(2, 1+2)>$

$= \min(2, 3)$

The minimum value is 2. Therefore, the currently shortest path from A to B is 2.

(b) Calculating shortest path from A to C

v = C, w = D

D(B) = min(D(C) , D(D) + c(D,C))

= min(5, 1+3)

= min(5, 4)

The minimum value is 4. Therefore, the currently shortest path from A to C is 4.</p>

(c) Calculating shortest path from A to E

v = E, w = D

D(B) = min(D(E) , D(D) + c(D,E))

= min(∞ , 1+1)

= min(∞ , 2)

The minimum value is 2. Therefore, the currently shortest path from A to E is 2.

Step 3:

In the above table, we observe that both E and B have the least cost path in step 2. Let's consider the E vertex. Now, we determine the least cost path of remaining vertices through E.

(a) Calculating the shortest path from A to B.

v = B, w = E

D(B) = min(D(B) , D(E) + c(E,B))

= min(2 , 2+ ∞)

= min(2, ∞)

The minimum value is 2. Therefore, the currently shortest path from A to B is 2.

(b) Calculating the shortest path from A to C.

v = C, w = E

D(B) = min(D(C) , D(E) + c(E,C))

= min(4 , 2+1)

= min(4,3)

The minimum value is 3. Therefore, the currently shortest path from A to C is 3.

(c) Calculating the shortest path from A to F.

v = F, w = E

D(B) = min(D(F) , D(E) + c(E,F))

= min(∞ , 2+2)

= min(∞ , 4)

The minimum value is 4. Therefore, the currently shortest path from A to F is 4.

Step	N	D(B),P(B)	D(C),P(C)	D(D),P(D)	D(E),P(E)	D(F),P(F)
1	A	2,A	5,A	1,A	∞	∞
2	AD	2,A	4,D		2,D	∞
3	ADE	2,A	3,E			4,E

Step 4:

In the above table, we observe that B vertex has the least cost path in step 3. Therefore, it is added in N. Now, we determine the least cost path of remaining vertices through B.

(a) Calculating the shortest path from A to C.

$v = C, w = B$

$D(B) = \min(D(C), D(B) + c(B,C))$

$= \min(3, 2+3)$

$= \min(3,5)$

The minimum value is 3. Therefore, the currently shortest path from A to C is 3.

(b) Calculating the shortest path from A to F.

$v = F, w = B$

$D(B) = \min(D(F), D(B) + c(B,F))$

$= \min(4, \infty)$

$= \min(4, \infty)$

The minimum value is 4. Therefore, the currently shortest path from A to F is 4.

Step	N	D(B),P(B)	D(C),P(C)	D(D),P(D)	D(E),P(E)	D(F),P(F)
1	A	2,A	5,A	1,A	∞	∞
2	AD	2,A	4,D		2,D	∞
3	ADE	2,A	3,E			4,E
4	ADEB		3,E			4,E

Step 5:

In the above table, we observe that C vertex has the least cost path in step 4. Therefore, it is added in N. Now, we determine the least cost path of remaining vertices through C.

(a) Calculating the shortest path from A to F.

$v = F, w = C$

$D(B) = \min(D(F), D(C) + c(C,F))$

$= \min(4, 3+5)$

$= \min(4,8)$

The minimum value is 4. Therefore, the currently shortest path from A to F is 4.

Step	N	D(B),P(B)	D(C),P(C)	D(D),P(D)	D(E),P(E)	D(F),P(F)
1	A	2,A	5,A	1,A	∞	∞
2	AD	2,A	4,D		2,D	∞
3	ADE	2,A	3,E			4,E
4	ADEB		3,E			4,E
5	ADEBC					4,E

Final table:

Step	N	D(B),P(B)	D(C),P(C)	D(D),P(D)	D(E),P(E)	D(F),P(F)
1	A	2,A	5,A	1,A	∞	∞
2	AD	2,A	4,D		2,D	∞
3	ADE	2,A	3,E			4,E
4	ADEB		3,E			4,E
5	ADEBC					4,E
6	ADEBCF					

Disadvantage:

Heavy traffic is created in Line state routing due to Flooding. Flooding can cause an infinite looping, this problem can be solved by using Time-to-leave field

2.2.3 OSPF

Q14. Explain about OSPF routing protocol.

Ans : **(Imp.)**

Open Shortest Path First (OSPF) is a link-state routing protocol that is used to find the best path between the source and the destination router using its own Shortest Path First). OSPF is developed by Internet Engineering Task Force (IETF) as one of the Interior Gateway Protocol (IGP), i.e, the protocol which aims at moving the packet within a large autonomous system or routing domain. It is a network layer protocol which works on protocol number 89 and uses AD value 110. OSPF uses multicast address 224.0.0.5 for normal communication and 224.0.0.6 for update to designated router(DR)/Backup Designated Router (BDR).

OSPF Terms

1. **Router I'd:** It is the highest active IP address present on the router. First, the highest loopback address is considered. If no loopback is configured then the highest active IP address on the interface of the router is considered.
2. **Router priority:** It is an 8-bit value assigned to a router operating OSPF, used to elect DR and BDR in a broadcast network.
3. **Designated Router (DR):** It is elected to minimize the number of adjacencies formed. DR distributes the LSAs to all the other routers. DR is elected in a broadcast network to which all the other routers share their DBD. In a broadcast network, the router requests for an update to DR, and DR will respond to that request with an update.
4. **Backup Designated Router (BDR):** BDR is a backup to DR in a broadcast network. When DR goes down, BDR becomes DR and performs its functions.

DR and BDR election: DR and BDR election takes place in the broadcast network or multi-access network. Here are the criteria for the election:

1. Router having the highest router priority will be declared as DR.
2. If there is a tie in router priority then the highest router I'd be considered. First, the highest loopback address is considered. If no loopback is configured then the highest active IP address on the interface of the router is considered.

OSPF states : The device operating OSPF goes through certain states. These states are:

1. **Down:** In this state, no hello packets have been received on the interface.

   Note: The Downstate doesn't mean that the interface is physically down. Here, it means that the OSPF adjacency process has not started yet.

2. **INIT:** In this state, the hello packets have been received from the other router.

3. **2WAY:** In the 2WAY state, both the routers have received the hello packets from other routers. Bidirectional connectivity has been established.

   Note: In between the 2WAY state and Exstart state, the DR and BDR election takes place.

4. **Exstart:** In this state, NULL DBD are exchanged. In this state, the master and slave elections take place. The router having the higher router I'd become the master while the other becomes the slave. This election decides Which router will send its DBD first (routers who have formed neighbourship will take part in this election).

5. **Exchange:** In this state, the actual DBDs are exchanged.

6. **Loading:** In this state, LSR, LSU, and LSA (Link State Acknowledgement) are exchanged.

   Important : When a router receives DBD from other router, it compares its own DBD with the other router DBD. If the received DBD is more updated than its own DBD then the router will send LSR to the other router stating what links are needed. The other router replies with the LSU containing the updates that are needed. In return to this, the router replies with the Link State Acknowledgement.

7. **Full:** In this state, synchronization of all the information takes place. OSPF routing can begin only after the Full state.

2.2.4. BGP

Q15. Explain briefly about BGP protocol.

Ans : **(Imp.)**

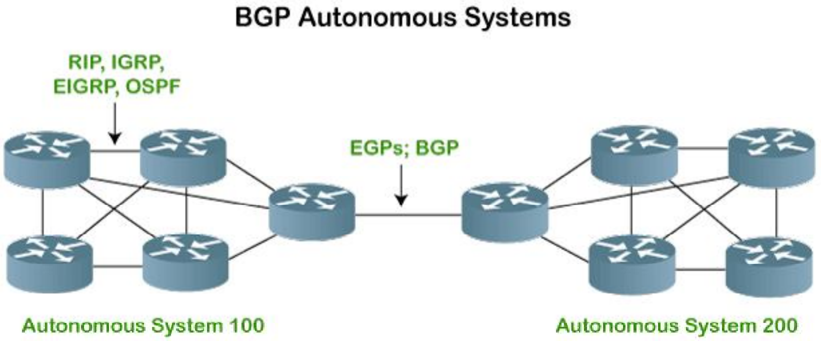
Border Gateway Protocol (BGP) is used to Exchange routing information for the internet and is the protocol used between ISP which are different ASes.

The protocol can connect together any internetwork of autonomous system using an arbitrary topology. The only requirement is that each AS have at least one router that is able to run BGP and that is router connect to at least one other AS's BGP router. BGP's main function is to exchange network reach-ability information with other BGP systems. Border Gateway Protocol constructs an autonomous systems' graph based on the information exchanged between BGP routers.

Characteristics of Border Gateway Protocol (BGP)

- **Inter-Autonomous System Configuration:** The main role of BGP is to provide communication between two autonomous systems.
- BGP supports Next-Hop Paradigm.
- Coordination among multiple BGP speakers within the AS (Autonomous System).
- **Path Information:** BGP advertisement also include path information, along with the reachable destination and next destination pair.
- **Policy Support:** BGP can implement policies that can be configured by the administrator. For ex:- a router running BGP can be configured to distinguish between the routes that are known within the AS and that which are known from outside the AS.
- Runs Over TCP.
- BGP conserve network Bandwidth.
- BGP supports CIDR.
- BGP also supports Security.

BGP Autonomous Systems



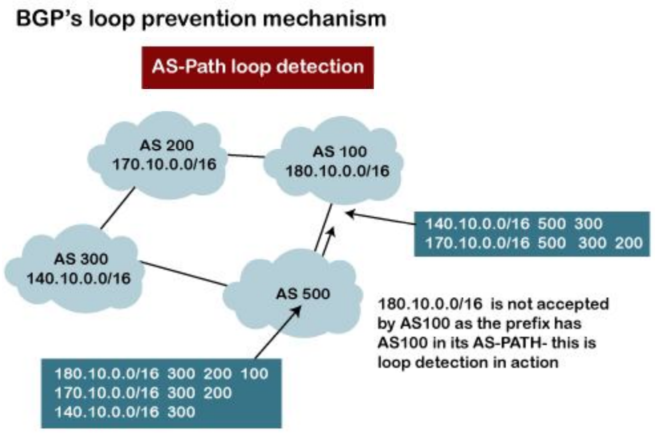
An autonomous system is a collection of networks that comes under the single common administrative domain. Or we can say that it is a collection of routers under the single administrative domain. For example, an organization can contain multiple routers having different locations, but the single autonomous number system will recognize them. Within the same autonomous system or same organization, we generally use IGP (Interior Gateway Protocol) protocols like RIP, IGRP, EIGRP, OSPF. Suppose we want to communicate between two autonomous systems. In that case, we use EGP (Exterior Gateway Protocols). The protocol that is running on the internet or used to communicate between two different autonomous number systems is known as BGP (Border Gateway Protocol). The BGP is the only protocol that is running on the internet backbone or used to exchange the routes between two different autonomous number systems. Internet service providers use the BGP protocol to control all the routing information.

BGP Features

The following are the features of a BGP protocol:

- **Open standard -**It is a standard protocol which can run on any window device.
- **Exterior Gateway Protocol -**It is an exterior gateway protocol that is used to exchange the routing information between two or more autonomous system numbers.
- **InterAS-domain routing -**It is specially designed for inter-domain routing, where interAS-domain routing means exchanging the routing information between two or more autonomous number system.
- **Supports internet -**It is the only protocol that operates on the internet backbone.
- **Classless -**It is a classless protocol.
- **Incremental and trigger updates -**Like IGP, BGP also supports incremental and trigger updates.
- **Path vector protocol -**The BGP is a path vector protocol. Here, path vector is a method of sending the routes along with routing information.
- **Configure neighborhood relationship -**It sends updates to configure the neighborhood relationship manually. Suppose there are two routers R1 and R2. Then, R1 has to send the configure command saying that you are my neighbor. On the other side, R2 also has to send the configure command to R1, saying that R1 is a neighbor of R1. If both the configure commands match, then the neighborhood relationship will get developed between these two routers.
- **Application layer protocol -**It is an application layer protocol and uses TCP protocol for reliability.
- **Metric -**It has lots of attributes like weight attribute, origin, etc. BGP supports a very rich number of attributes that can affect the path manipulation process.
- **Administrative distance -**If the information is coming from the external autonomous system, then it uses 20 administrative distance. If the information is coming from the same autonomous system, then it uses 200 administrative distance.

BGP’s Loop prevention mechanism



There is a possibility that when you are connecting to the internet, then you may be advertising route 10.0.0.0 to some autonomous system, then it is advertised to some other autonomous system. Then there is a possibility that the same route is coming back again. This creates a loop. But, in BGP, there is a rule that when the router sees its own AS number for example, as shown in the above figure, the network 180.10.0.0/16 is originating from the AS 100, and when it sends to the AS 200, it is going to carry its path information, i.e., 180.10.0.0/16 and AS 100. When AS 200 sends to the AS 300, AS 200 will send its path information 180.10.0.0/16 and AS path is 100 and then 200, which means that the route originates from AS 100, then reaches 200 and finally reaches to 300. When AS 300 sends to the AS 500, it will send the network information 180.10.0.0/16, and AS path is 100, 200, and then 300. If AS 500 sends to the AS 100, and AS 100 sees its own autonomous number inside the update, it will not accept it. In this way, BGP prevents the loop creation.

Path Attributes

The BGP chooses the best route based on the attributes of the path.

As we know that path-vector routing is used in the border gateway routing protocol, which contains the routing table that shows the path information. The path attributes provide the path information. The attributes that show or store the path information are known as path attributes. This list of attributes helps the receiving router to make a better decision while applying any policy. Let’s see the different types of attributes. The path attribute is broadly classified into two categories:

1. **Well-known attribute:** It is an attribute that should be recognized by every BGP router.

   The well-known attribute is further classified into two categories:

   - **Well-known mandatory:** When BGP is going to advertise any network, but it also advertises extra information, and that information with path attributes information. The information includes AS path information, origin information, next-hop information. Here, mandatory means that it has to be present in all the BGP routing updates.
   - **Well-known discretionary:** It is recognized by all the BGP routers and passed on to other BGP routers, but it is not mandatory to be present in an update.

2. **Optional attribute:** It is an attribute that is not necessarily to be recognized by every BGP router. In short, we can say that it is not a mandatory attribute.

   The optional attribute is further classified into two categories:

   - **Optional transitive:** BGP may or may not recognize this attribute, but it is passed on to the other BGP neighbors. Here, transitive means that if the attribute is not recognized, then it is marked as a partial.
   - **Optional non-transitive:** If the BGP cannot recognize the attribute, it ignores the update and does not advertise to another BGP router.

BGP Sessions

When we talk about the BGP, which means that the communication between the autonomous systems. Let’s consider two autonomous systems having five nodes each.

BGP sessions are classified into two categories:

1. **Internal BGP session**

   The internal BGP session is used to exchange information between the routers inside an autonomous system. In short, we can say that the routing information is exchanged between the routers of the same autonomous system.

2. **External BGP session**

   The external BGP session is a session in which nodes or routers of different autonomous systems communicate with each other.

Types of Packets

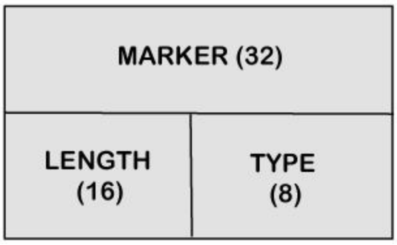
There are four different types of packets exist in BGP:

- **Open:** When the router wants to create a neighborhood relation with another router, it sends the Open packet.
- **Update:** The update packet can be used in either of the two cases:
  1. It can be used to withdraw the destination, which has been advertised previously.
  2. It can also be used to announce the route to the new destination.
- **Keep Alive:** The keep alive packet is exchanged regularly to tell other routers whether they are alive or not. For example, there are two routers, i.e., R1 and R2. The R1 sends the keep alive packet to R2 while R2 sends the keep alive packet to R1 so that R1 can get to know that R2 is alive, and R2 can get to know that R1 is alive.
- **Notification:** The notification packet is sent when the router detects the error condition or close the connection.

BGP Packet Format

Now we will see the format in which the packet travels. The following are the fields in a BGP packet format:

BGP Packet Format



1. **Marker:** It is a 32-bit field which is used for the authentication purpose.
2. **Length:** It is a 16-bit field that defines the total length of the message, including the header.
3. **Type:** It is an 8-bit field that defines the type of the packet.

Short Question and Answers

1. Logical addressing

Ans :

In computer networks, logical addressing is a method of assigning addresses to devices at the Network Layer (Layer 3) of the OSI (Open Systems Interconnection) model. Logical addressing is used to identify devices on a network and to enable communication between them.

The most common form of logical addressing is IP (Internet Protocol) addressing. IP addressing is a hierarchical addressing scheme that consists of two parts: the network address and the host address. The network address identifies the network that the device is connected to, while the host address identifies the individual device on that network.

2. IP Address

Ans :

An Internet Protocol address is an IP address. It is a unique address that identifies the device on the network. The Internet Service Provider (ISP) assigns IP addresses to all devices on its network. IP addresses are not generated at random. The Internet Assigned Numbers Authority (IANA), a part of the Internet Corporation for Assigned Names and Numbers (ICANN), generates and assigns them mathematically . IP addresses are used at the network layer. IP Addresses are routable in nature.

3. IP Version Types

Ans :

There are 2 different versions of IP as follows.

i) IPv4 (IP version 4)

IPv4 employs a 32-bit address. It is composed of four numbers separated by a 'dot' i.e., periods called an octet (byte). Each number in the octet can range from 0 to 255.

Example: 172.166.3.28

ii) IPv6 (IP version 6)

IPv6 is the next generation of Internet Protocol addresses. In comparison to IPV4, IPv6 has a larger address space. IPv6 has a length of 128 bits and is written in hexadecimal. It is composed of eight fields, each of which contains two octets. As a result, IPv6 has 16 octets in total.

Example:

3221:1cd7:74b6:6da7:0000:0000:7349:6472

4. IPV4 addresses

Ans :

IPv4 (Internet Protocol version 4) is a widely used protocol for sending data across the Internet. It uses a 32-bit address space to uniquely identify devices on a network. IPv4 addresses are represented in dot-decimal notation, where each 8-bit segment is separated by a period. For example, 192.168.1.1 is a typical IPv4 address.

IPv4 addresses are divided into two parts: the network portion and the host portion. The network portion identifies the network on which the device is located, while the host portion identifies the specific device on the network.

5. What is subnetting?

Ans :

Subnetting is a technique used in computer networking to divide a larger network into smaller subnetworks, or subnets. This allows for more efficient use of IP addresses and better management of network traffic.

When subnetting, the network administrator divides the original network into smaller networks, each with its own unique network address. The subnet mask is used to divide the IP address space into two parts: the network part and the host part. The network part identifies the network, while the host part identifies the specific device on the network. The subnet mask determines the number of bits used for the network part and the number of bits used for the host part.

6. Supernetting

Ans :

Supernetting is the opposite of Subnetting. In subnetting, a single big network is divided into multiple smaller subnetworks. In Supernetting, multiple networks are combined into a bigger network termed as a Supernetwork or Supernet.

Supernetting is mainly used in Route Summari-zation, where routes to multiple networks with similar network prefixes are combined into a single routing entry, with the routing entry pointing to a Super network, encompassing all the networks. This in turn significantly reduces the size of routing tables and also the size of routing updates exchanged by routing protocols.

More specifically,

- When multiple networks are combined to form a bigger network, it is termed super-netting
- Super netting is used in route aggregation to reduce the size of routing tables and routing table updates

There are some points which should be kept in mind while supernetting:

1. All the Networks should be contiguous.
2. The block size of every network should be equal and must be in form of 2^n.
3. First Network id should be exactly divisible by whole size of supernet.

7. CIDR

Ans :

CIDR stands for Classless Inter-Domain Routing, which is a method used to allocate and manage IP addresses in a more flexible and efficient way than the traditional IP address classes. With CIDR, IP addresses are not bound to any predefined class boundaries, as was the case with the original Classful addressing scheme, and instead, can be divided into smaller subnets of various sizes.

CIDR is based on the use of a subnet mask, which is used to define the boundaries between the network portion and the host portion of an IP address. The subnet mask is expressed in the same dotted decimal format as IP addresses, but instead of representing the individual octets of an IP address, it represents the number of bits that are used for the network portion of the address. For example, a subnet mask of 255.255.255.0 would mean that the first 24 bits of an IP address are used for the network portion, and the remaining 8 bits are used for the host portion.

CIDR notation is a shorthand way of representing the subnet mask and the network address of an IP address using a forward slash and a number that represents the number of bits used for the network portion. For example, the IP address 192.168.1.0 with a subnet mask of 255.255.255.0 can be represented in CIDR notation as 192.168.1.0/24, where the 24 represents the number of bits used for the network portion of the address.

8. IPV6

Ans :

IPv6 is the latest version of the Internet Protocol that is designed to replace the aging IPv4 protocol. IPv6 addresses are 128 bits long, which provides for a much larger address space than IPv4's 32 bits, allowing for virtually unlimited addresses.

IPv6 addresses are typically represented using hexadecimal notation, which uses digits 0-9 and letters A-F. The address is divided into eight 16-bit blocks separated by colons, for a total of 32 hexadecimal characters. Leading zeroes can be omitted in each block, and consecutive blocks of zeroes can be compressed with a double colon. For example, the loopback address in IPv6 is represented as ::1, which represents eight blocks of zeroes followed by a single block of 1.

Here's an example of an IPv6 address in full notation:

2001:0db8:85a3:0000:0000:8a2e:0370:7334

n this example, each block is represented by four hexadecimal characters, separated by colons. Note that leading zeroes are included in each block.

9. Types of ICMP

Ans :

Internet Control Message Protocol (ICMP) is a protocol used for error handling and debugging in the network layer. It is mainly used in network devices such as routers.

ICMP messages are mainly divided into two categories :

1. Error Reporting Messages

Error Reporting Messages are used to report problems encountered by the router/host while processing the IP packets. These messages are always sent to the source because the datagram only contains the source and destination IP address.

Following are the types of error reporting messages :

- **Destination Unreachable Message:** The host/router send this message if it is not able to route the IP packet to its destination.

  For example, sender A wants to send the datagram to receiver B but it is not received by B then the intermediate router will discard the datagram and send the destination unreachable message to A.

- **Source Quench Message:** Host/router send this message if there is congestion in the network or the source is sending packets at a higher rate which the router can't handle.

  For example, if sender A is sending the data packets at a high data rate which the router is unable to handle then it will discard the packet and send a source quench message to A to tell it to send the packets at a lower rate. Now, after receiving the message A will either stop or slow down sending of the packets.

- **Time Exceeded Message:** The host/router sends this message if it decrements the time to live value of the datagram to zero or the destination address does not receive all the packets in the specified time interval.

  For example, a packet is sent from a layer having 1000 units to the layer having 200 units, then the packet is divided into five fragments. If all the fragments don't reach the destination in a set time, all fragments are discarded and the time-exceeded message is sent to the original source.

- **Parameter Problem Message:** The host/router sends this message if some parameter is not properly set in the datagram. It is used to indicate errors in the header field of the datagram.

- **Redirection Message:** The host/router sends this message to update the routing table of the host.

  For example, sender A wants to send the message to receiver B and there is a router between them. Then, A sends the data to the router and the router sends the message to B and redirection message to A so that A can update its routing table.

2. Query Messages

Query Messages are used for error handling and debugging network problems. These messages help the host to get specific information about another host or router.

Following are the types of query messages :

- **Echo-Request and Reply Message:** It is used by the network managers to check the operations of the IP protocol and host's reachability. The host/router sends the echo request message, while the echo reply message is sent by the host/router that receives an echo request message.

  For example, host A wants to check whether it can communicate with another host B so it will send an echo-request to B and if there is a link between A and B and B is active then it will send echo-reply to A on receiving the request.

10. IGMP protocol.

Ans :

IGMP is acronym for Internet Group Management Protocol. IGMP is a communication protocol used by hosts and adjacent routers for multicasting communication with IP networks and uses the resources efficiently to transmit the message/data packets. Multicast communication can have single or multiple senders and receivers and thus, IGMP can be used in streaming videos, gaming or web conferencing tools. This protocol is used on IPv4 networks and for using this on IPv6, multicasting is managed by Multicast Listener Discovery (MLD). Like other network protocols, IGMP is used on network layer. MLDv1 is almost same in functioning as IGMPv2 and MLDv2 is almost similar to IGMPv3. The communication protocol, IGMPv1 was developed in 1989 at Stanford University. IGMPv1 was updated to IGMPv2 in year 1997 and again updated to IGMPv3 in year 2002.

11. Routings.

Ans :

Routing is a process that is performed by layer 3 (or network layer) devices in order to deliver the packet by choosing an optimal path from one network to another.

There are 3 types of routing:

1. Static Routing

Static routing is a process in which we have to manually add routes to the routing table.

Advantages

- No routing overhead for router CPU which means a cheaper router can be used to do routing.
- It adds security because an only administrator can allow routing to particular networks only.
- No bandwidth usage between routers.

Disadvantage

- For a large network, it is a hectic task for administrators to manually add each route for the network in the routing table on each router.
- The administrator should have good knowledge of the topology. If a new administrator comes, then he has to manually add each route so he should have very good knowledge of the routes of the topology.

2. Default Routing

This is the method where the router is configured to send all packets towards a single router (next hop). It doesn't matter to which network the packet belongs, it is forwarded out to the router which is configured for default routing. It is generally used with stub routers. A stub router is a router that has only one route to reach all other networks.

3. Dynamic Routing

Dynamic routing makes automatic adjustments of the routes according to the current state of the route in the routing table. Dynamic routing uses protocols to discover network destinations and the routes to reach them. RIP and OSPF are the best examples of dynamic routing protocols. Automatic adjustments will be made to reach the network destination if one route goes down.

A dynamic protocol has the following features:

1. The routers should have the same dynamic protocol running in order to exchange routes.
2. When a router finds a change in the topology then the router advertises it to all other routers.

Advantages

- Easy to configure.
- More effective at selecting the best route to a destination remote network and also for discovering remote network.

Disadvantage

- Consumes more bandwidth for communi-cating with other neighbors.
- Less secure than static routing

12. Expalin about Distance vector routing.

Ans :

A distance-vector routing (DVR) protocol requires that a router inform its neighbors of topology changes periodically. Historically known as the old ARPANET routing algorithm (or known as Bellman-Ford algorithm).

13. Explain about Link State Routing with example.

Ans :

Link state routing is a technique in which each router shares the knowledge of its neighborhood with every other router in the internetwork.

The three keys to understand the Link State Routing algorithm:

- **Knowledge about the neighborhood:** Instead of sending its routing table, a router sends the information about its neighborhood only. A router broadcast its identities and cost of the directly attached links to other routers.
- **Flooding:** Each router sends the information to every other router on the internetwork except its neighbors. This process is known as Flooding. Every router that receives the packet sends the copies to all its neighbors. Finally, each and every router receives a copy of the same information.
- **Information sharing:** A router sends the information to every other router only when the change occurs in the information.

14. OSPF

Ans :

Open Shortest Path First (OSPF) is a link-state routing protocol that is used to find the best path between the source and the destination router using its own Shortest Path First). OSPF is developed by Internet Engineering Task Force (IETF) as one of the Interior Gateway Protocol (IGP), i.e, the protocol which aims at moving the packet within a large autonomous system or routing domain. It is a network layer protocol which works on protocol number 89 and uses AD value 110. OSPF uses multicast address 224.0.0.5 for normal communication and 224.0.0.6 for update to designated router(DR)/Backup Designated Router (BDR).

15. BGP protocol.

Ans :

Border Gateway Protocol (BGP) is used to Exchange routing information for the internet and is the protocol used between ISP which are different ASes.

The protocol can connect together any internetwork of autonomous system using an arbitrary topology. The only requirement is that each AS have at least one router that is able to run BGP and that is router connect to at least one other AS's BGP router. BGP's main function is to exchange network reach-ability information with other BGP systems. Border Gateway Protocol constructs an autonomous systems' graph based on the information exchanged between BGP routers.

16. Features of a BGP protocol

Ans :

- **Open standard:** It is a standard protocol which can run on any window device.
- **Exterior Gateway Protocol:** It is an exterior gateway protocol that is used to exchange the routing information between two or more autonomous system numbers.
- **InterAS-domain routing:** It is specially designed for inter-domain routing, where interAS-domain routing means exchanging the routing information between two or more autonomous number system.
- **Supports internet:** It is the only protocol that operates on the internet backbone.
- **Classless:** It is a classless protocol.
- **Incremental and trigger updates:** Like IGP, BGP also supports incremental and trigger updates.
- **Path vector protocol:** The BGP is a path vector protocol. Here, path vector is a method of sending the routes along with routing information.
- **Configure neighborhood relationship:** It sends updates to configure the neighborhood relationship manually. Suppose there are two routers R1 and R2. Then, R1 has to send the configure command saying that you are my neighbor. On the other side, R2 also has to send the configure command to R1, saying that R1 is a neighbor of R1. If both the configure commands match, then the neighborhood relationship will get developed between these two routers.

Choose the Correct Answer

1. Which function removes a set's first and the last element from a list? [a]

   (a) pop (b) remove

   (c) dispose (d) discard

2. The output of this Python code would be: [c]

   sum(1,2,3)

   sum([2,4,6])

   (a) 6, 12 (b) Error, Error

   (c) Error, 12 (d) 6, Error

3. Which function doesn't accept any argument? [d]

   (a) re.compile (b) re.findall

   (c) re.match (d) re.purge

4. Which of the following functions is a built-in function in python language? [b]

   (a) val() (b) print()

   (c) print () (d) None of these

5. Which one of the following is a valid Python if statement. [a]

   (a) if a>=2 : (b) if (a >= 2)

   (c) if (a => 22) (d) if a >= 22

6. Which of the following is not used as loop in Python? [c]

   (a) for loop (b) while loop

   (c) do-while loop (d) None of the above

7. Which one of the following is a valid Python if statement : [a]

   (a) if a>=2 : (b) if (a >= 2)

   (c) if (a => 22) (d) if a >= 22

8. What keyword would you use to add an alternative condition to an if statement? [c]

   (a) else if (b) elseif

   (c) elif (d) None of the above

9. Which statement will check if a is equal to b? [b]

   (a) if a = b: (b) if a == b:

   (c) if a === c: (d) if a == b

10. Which of the following is a valid for loop in Python? [b]

    (a) for(i=0; i < n; i++) (b) for i in range(0,5):

    (c) for i in range(0,5) (d) for i in range(5)

Fill in the blanks

1. ________ are used to indicate variables that must not be accessed from outside the class.
2. ________ keyword would you use to add an alternative condition to an if statement?
3. ________ loop is used when multiple statements are to executed repeatedly until the given condition becomes False.
4. ________ determine which statements in the program will be executed and in what order, allowing for statements to be skipped over or executed repeatedly.
5. In Python When a statement occurs on a line which is indented less than the previous one, it indicates ________.
6. ________ Keyword can be used to bring control out of the current loop statement is True regarding loops in Python.
7. Else statement after loop will be executed only when the loop condition becomes ________.
8. A loop becomes ________ loop if a condition never becomes FALSE.
9. If the else statement is used with a while loop, the else statement is executed when the condition becomes ________.
10. Python programming language allows to use one loop inside another loop known as ________

ANSWERS

1. Leading underscores
2. Elif
3. While
4. Control structures
5. The end of a block
6. Break
7. False
8. Infinite
9. False
10. Nested

UNIT III

Transport Layer:

TCP State diagram, Window Management, Congestion Control, Timer Management and UDP protocol

3.1 Transport Layer

3.1.1 TCP State Diagram

Q1. What is Transmission Control Protocol (TCP)? Explain the working mechanism of TCP.

Ans : **(Imp.)**

Meaning

The Transmission Control Protocol (TCP) is a transport protocol that is used on top of IP to ensure reliable transmission of packets.

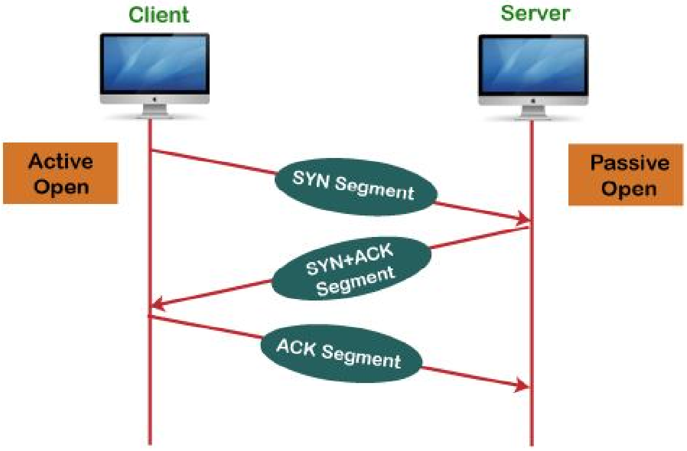
TCP includes mechanisms to solve many of the problems that arise from packet-based messaging, such as lost packets, out of order packets, duplicate packets, and corrupted packets.

Since TCP is the protocol used most commonly on top of IP, the Internet protocol stack is sometimes referred to as TCP/IP.

Working of TCP

In TCP, the connection is established by using three-way handshaking. The client sends the segment with its sequence number. The server, in return, sends its segment with its own sequence number as well as the acknowledgement sequence, which is one more than the client sequence number. When the client receives the acknowledgment of its segment, then it sends the acknowledgment to the server. In this way, the connection is established between the client and the server.

Working of the TCP Protocol



Q2. State the Advantages and Disadvantages of TCP.

Ans :

Advantages

- It provides a connection-oriented reliable service, which means that it guarantees the delivery of data packets. If the data packet is lost across the network, then the TCP will resend the lost packets.
- It provides a flow control mechanism using a sliding window protocol.
- It provides error detection by using checksum and error control by using Go Back or ARP protocol.
- It eliminates the congestion by using a network congestion avoidance algorithm that includes various schemes such as additive increase/ multiplicative decrease (AIMD), slow start, and congestion window.

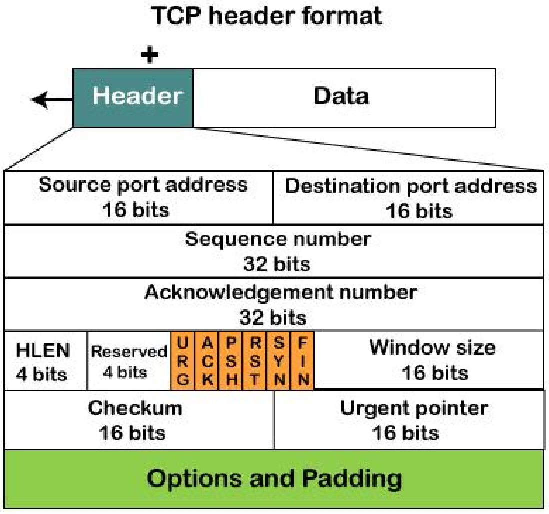
Disadvantage

- It increases a large amount of overhead as each segment gets its own TCP header, so fragmentation by the router increases the overhead.

Q3. Explain briefly about TCP header Format.

Ans :

TCP Header Format



- **Source port:** It defines the port of the application, which is sending the data. So, this field contains the source port address, which is 16 bits.
- **Destination port:** It defines the port of the application on the receiving side. So, this field contains the destination port address, which is 16 bits.
- **Sequence number:** This field contains the sequence number of data bytes in a particular session.
- **Acknowledgment number:** When the ACK flag is set, then this contains the next sequence number of the data byte and works as an acknowledgment for the previous data received. For example, if the receiver receives the segment number 'x', then it responds 'x+1' as an acknowledgment number.
- **HLEN:** It specifies the length of the header indicated by the 4-byte words in the header. The size of the header lies between 20 and 60 bytes. Therefore, the value of this field would lie between 5 and 15.
- **Reserved:** It is a 4-bit field reserved for future use, and by default, all are set to zero.

Flags

There are six control bits or flags:

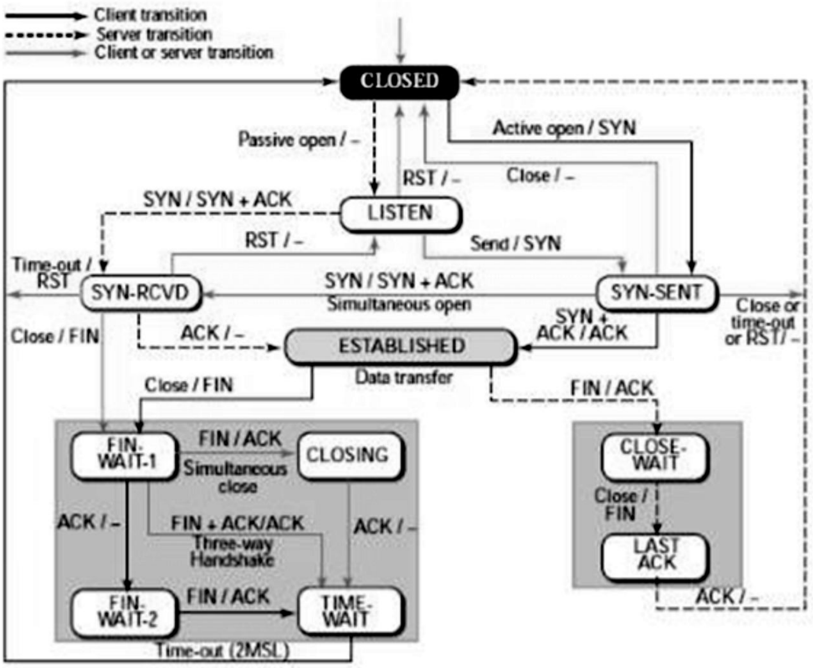
1. **URG:** It represents an urgent pointer. If it is set, then the data is processed urgently.
2. **ACK:** If the ACK is set to 0, then it means that the data packet does not contain an acknowledgment.
3. **PSH:** If this field is set, then it requests the receiving device to push the data to the receiving application without buffering it.
4. **RST:** If it is set, then it requests to restart a connection.
5. **SYN:** It is used to establish a connection between the hosts.
6. **FIN:** It is used to release a connection, and no further data exchange will happen.
   - **Window size:** It is a 16-bit field. It contains the size of data that the receiver can accept. This field is used for the flow control between the sender and receiver and also determines the amount of buffer allocated by the receiver for a segment. The value of this field is determined by the receiver.
   - **Checksum:** It is a 16-bit field. This field is optional in UDP, but in the case of TCP/IP, this field is mandatory.
   - **Urgent pointer:** It is a pointer that points to the urgent data byte if the URG flag is set to 1. It defines a value that will be added to the sequence number to get the sequence number of the last urgent byte.

- **Options:** It provides additional options. The optional field is represented in 32-bits. If this field contains the data less than 32-bit, then padding is required to obtain the remaining bits.

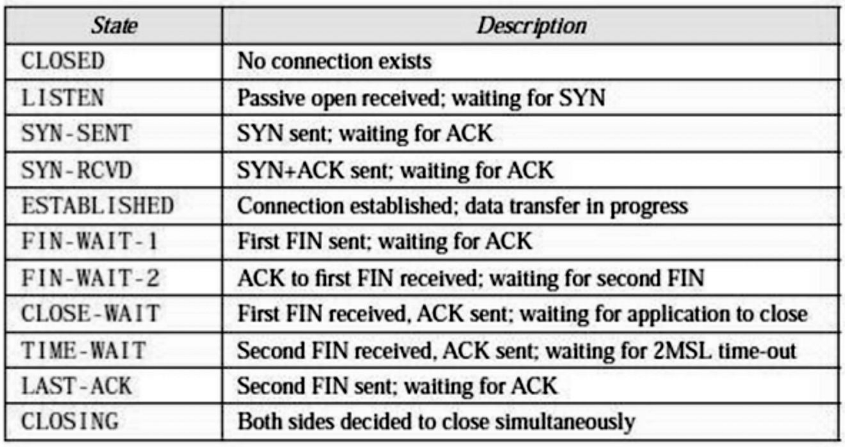
Q4. Explain the transition states of TCP with a neat diagram.

Ans : **(Imp.)**

To keep track of all the different events happening during connection establishment, connection termination, and data transfer, TCP is specified as the finite state machine shown in Figure.



The states for TCP are as follows:



State	Description
CLOSED	No connection exists
LISTEN	Passive open received; waiting for SYN
SYN-SENT	SYN sent; waiting for ACK
SYN-RCVD	SYN+ACK sent; waiting for ACK
ESTABLISHED	Connection established; data transfer in progress
FIN-WAIT-1	First FIN sent; waiting for ACK
FIN-WAIT-2	ACK to first FIN received; waiting for second FIN
CLOSE-WAIT	First FIN received, ACK sent; waiting for application to close
TIME-WAIT	Second FIN received, ACK sent; waiting for 2MSL time-out
LAST-ACK	Second FIN sent; waiting for ACK
CLOSING	Both sides decided to close simultaneously

Here are the steps of connecting and libration of a TCP connection.

The client application opens a connection to the server by sending a TCP segment which only the header is present (no data). This header contains a flag SYN stands for "Synchronize" and the TCP port number the server (application). The client is in SYN_SENT state (SYN sent).

The server (application) is listening (listen) and on receipt of the SYN from the client, it changes of state and responds with a SYN and ACK flag. The server is then able SYN_RCVD (SYN received).

The client receives the server's TCP segment with SYN ACK indicators and move in status ESTABLISHED. He also sent a response ACK to the server that also passes in status ESTABLISHED.

This exchange in three phases (three-way handshake) complete the establishment of the TCP connection can now be used to exchange data between the client and server.

The client side of the connection is responsible for the connection performs an active connection (active open) while the server performs a passive connection (passive open).

In the event that a connection request arrives on the server and that no application is listening on the requested port, a segment with flag RST (reset) is sent to the client by the server, the connection attempt is immediately terminated.

Principle Termination of a TCP Connection State Diagram

The termination of a TCP connection requires four exchanges of TCP segments.

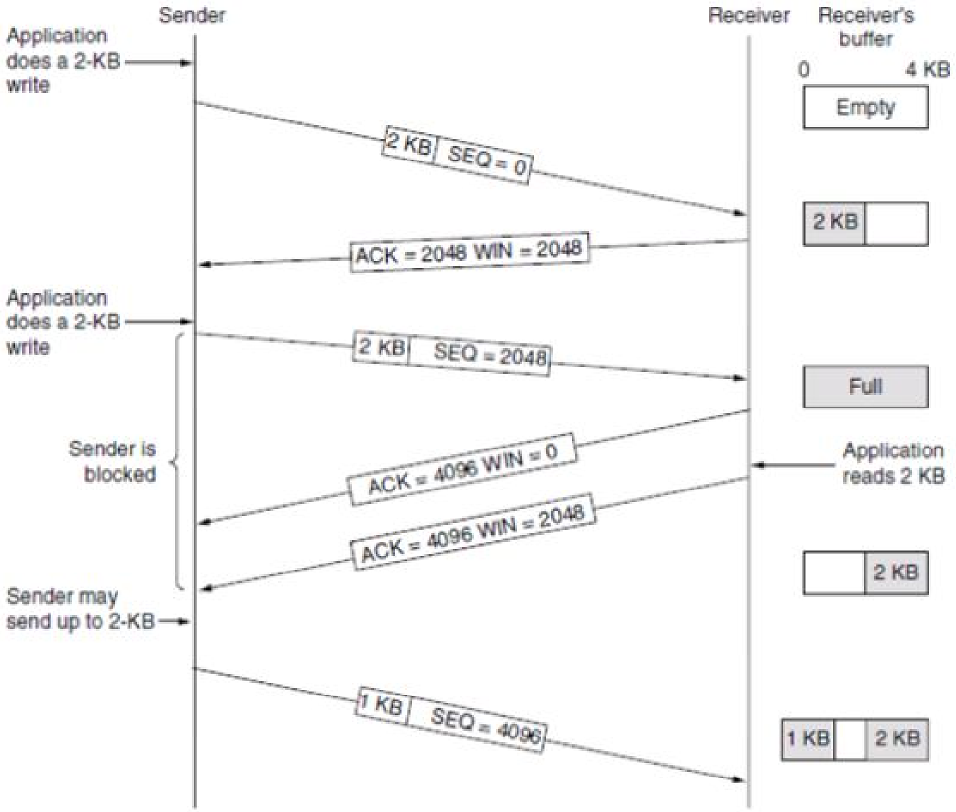
- As a TCP connection is bidirectional (full duplex) , connection termination process should be made in both directions of the communication. The client as the server can send a segment with FIN flag, this mean an end to sending data. Receiving a segment with FIN indicates that the other end will not send more data. The term used then is half closed, the connection is half closed.
- Typically, the client generates the transmission segment with a FIN flag, he made a closing force (active close), the server receives the segment realizes a passive close. The server acknowledge the FIN with ACK, informs the application for the release of this connection and when done then sends a FIN segment to the customer which in turn, acknowledge it with an ACK flag.
- All segments exchanged between client and server are numbered sequentially. The sequence number is an integer of 32 bits. Each side of the connection initiates and maintains the sequence of segments sent.
- The header segment always contains two fields, one for the sequence number of the sent segment and one for the sequence number of the acknowledged segment . The sequence number of the acknowledged segment is actually the number of the next segment expected. In other words the acquittal as "waiting segment number 10 on your part" means that the 9 was received correctly.
- The tests performed LoriotPro within its various modules to check TCP connection and measure TCP response time are based mainly on the time between the request from the client (initial segment SIN) and the server acknowledgement by SIN/ACK.

3.1.2 Window Management

Q5. Explain TCP sliding window protocol with neat diagram in detail.

Ans : **(Imp.)**

- Window management in TCP decouples the issues of acknowledgement of the correct receipt of segments and receiver buffer allocation.
- For example, suppose the receiver has a 4096-byte buffer, as shown in Fig. below. If the sender transmits a 2048-byte segment that is correctly received, the receiver will acknowledge the segment. However, since it now has only 2048 bytes of buffer space (until the application removes some data from the buffer), it will advertise a window of 2048 starting at the next byte expected.



Window management in TCP

- Now the sender transmits another 2048 bytes, which are acknowledged, but the advertised window is of size 0. The sender must stop until the application process on the receiving host has removed some data from the buffer, at which time TCP can advertise a larger window and more data can be sent.
- When the window is 0, the sender may not normally send segments, with two exceptions. First, urgent data may be sent, for example, to allow the user to kill the process running on the remote machine. Second, the sender may send a 1-byte segment to force the receiver to reannounce the next byte expected and the window size. This packet is called a window probe.
- The TCP standard explicitly provides this option to prevent deadlock if a window update ever gets lost.
- Senders are not required to transmit data as soon as they come in from the application.
- Neither are receivers required to send acknowledgements as soon as possible. For example, in Fig. above, when the first 2 KB of data came in, TCP, knowing that it had a 4-KB window, would have been completely correct in just buffering the data until another 2 KB came in, to be able to transmit a segment with a 4-KB payload. This freedom can be used to improve performance.
- Consider a connection to a remote terminal, for example using SSH or telnet, that reacts on every keystroke. In the worst case, whenever a character arrives at the sending TCP entity, TCP creates a 21-byte TCP segment, which it gives to IP to send as a 41-byte IP datagram.
- At the receiving side, TCP immediately sends a 40-byte acknowledgement (20 bytes of TCP header and 20 bytes of IP header).
- Later, when the remote terminal has read the byte, TCP sends a window update, moving the window 1 byte to the right. This packet is also 40 bytes. Finally, when the remote terminal has processed the character, it echoes the character for local display using a 41-byte packet.

- In all, 162 bytes of bandwidth are used and four segments are sent for each character typed. When bandwidth is scarce, this method of doing business is not desirable.

3.1.3 Congestion Control

Q6. Expalin about Congestion control in TCP.

Ans : **(Imp.)**

Congestion control refers to techniques and mechanisms that can-

- Either prevent congestion before it happens
- Or remove congestion after it has happened

TCP Congestion Control

1. Receiver window size
2. Congestion window size

1. Receiver Window Size

Receiver window size is an advertisement of-

"How much data (in bytes) the receiver can receive without acknowledgement?"

- Sender should not send data greater than receiver window size.
- Otherwise, it leads to dropping the TCP segments which causes TCP Retransmission.
- So, sender should always send data less than or equal to receiver window size.
- Receiver dictates its window size to the sender through TCP Header.

2. Congestion Window-

- Sender should not send data greater than congestion window size.
- Otherwise, it leads to dropping the TCP segments which causes TCP Retransmission.
- So, sender should always send data less than or equal to congestion window size.
- Different variants of TCP use different approaches to calculate the size of congestion window.
- Congestion window is known only to the sender and is not sent over the links.

So, always-

Sender window size = Minimum (Receiver window size, Congestion window size)

TCP Congestion Policy

TCP's general policy for handling congestion consists of following three phases

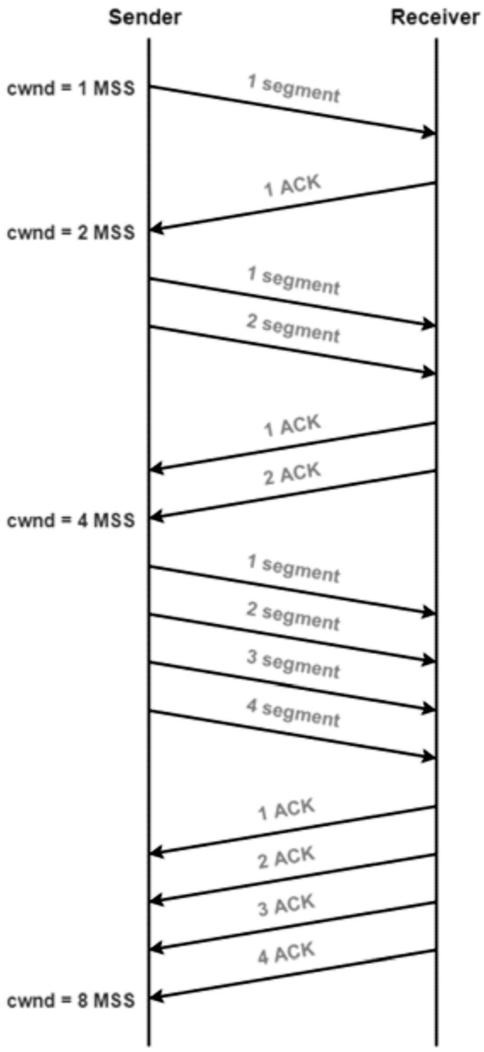
1. Slow Start Phase

- Initially, sender sets congestion window size = Maximum Segment Size (1 MSS).
- After receiving each acknowledgment, sender increases the congestion window size by 1 MSS.
- In this phase, the size of congestion window increases exponentially.

The followed formula is-

Congestion window size = Congestion window size + Maximum segment size

This is shown below-



(cwnd = congestion window size)

- After 1 round trip time, congestion window size = $(2)^1$ = 2 MSS
- After 2 round trip time, congestion window size = $(2)^2$ = 4 MSS
- After 3 round trip time, congestion window size = $(2)^3$ = 8 MSS and so on.

This phase continues until the congestion window size reaches the slow start threshold.

Threshold

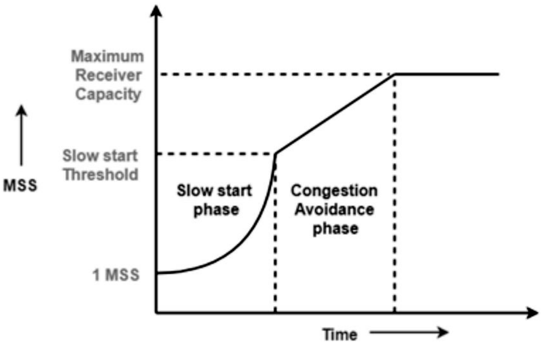
= Maximum number of TCP segments that receiver window can accommodate / 2= (Receiver window size / Maximum Segment Size) / 2

2. Congestion Avoidance Phase-

After reaching the threshold,

- Sender increases the congestion window size linearly to avoid the congestion.
- On receiving each acknowledgement, sender increments the congestion window size by 1.
- The followed formula is-

  Congestion window size = Congestion window size + 1
- This phase continues until the congestion window size becomes equal to the receiver window size.



3. Congestion Detection Phase-

When sender detects the loss of segments, it reacts in different ways depending on how the loss is detected-

Case-01: Detection On Time Out-

- Time Out Timer expires before receiving the acknowledgement for a segment.
- This case suggests the stronger possibility of congestion in the network.
- There are chances that a segment has been dropped in the network

Reaction

In this case, sender reacts by-

Setting the slow start threshold to half of the current congestion window size.

- Decreasing the congestion window size to 1 MSS.
- Resuming the slow start phase.

Case-02: Detection On Receiving 3 Duplicate Acknowledgements-

- Sender receives 3 duplicate acknowledge-ments for a segment.
- This case suggests the weaker possibility of congestion in the network.
- There are chances that a segment has been dropped but few segments sent later may have reached.

Reaction

In this case, sender reacts by-

- Setting the slow start threshold to half of the current congestion window size.
- Decreasing the congestion window size to slow start threshold.
- Resuming the congestion avoidance phase.

3.1.4 Timer Management

Q7. What are timers in TCP? Explain.

Ans :

Timers used by TCP to avoid excessive delays during communication are called as TCP Timers.

The 4 important timers used by a TCP implementation are-

1. Time Out Timer
2. Time Wait Timer
3. Keep Alive Timer
4. Persistent Timer

Time Out Timer-

TCP uses a time out timer for retransmission of lost segments.

- Sender starts a time out timer after transmitting a TCP segment to the receiver.
- If sender receives an acknowledgement before the timer goes off, it stops the timer.
- If sender does not receives any acknowledgement and the timer goes off, then TCP Retransmission occurs.
- Sender retransmits the same segment and resets the timer.
- The value of time out timer is dynamic and changes with the amount of traffic in the network.
- Time out timer is also called as Retransmission Timer.

Time Wait Timer

TCP uses a time wait timer during connection termination.

- Sender starts the time wait timer after sending the ACK for the second FIN segment.
- It allows to resend the final acknowledgement if it gets lost.
- It prevents the just closed port from reopening again quickly to some other application.
- It ensures that all the segments heading towards the just closed port are discarded.
- The value of time wait timer is usually set to twice the lifetime of a TCP segment.

Keep Alive Timer

TCP uses a keep alive timer to prevent long idle TCP connections.

- Each time server hears from the client, it resets the keep alive timer to 2 hours.

- If server does not hear from the client for 2 hours, it sends 10 probe segments to the client.
- These probe segments are sent at a gap of 75 seconds.
- If server receives no response after sending 10 probe segments, it assumes that the client is down.
- Then, server terminates the connection automatically.

Persistent Timer

- TCP uses a persistent timer to deal with a zero-widow-size deadlock situation.
- It keeps the window size information flowing even if the other end closes its receiver window.

Q8. Explain TCP timer management and transaction TCP.

Ans : **(Imp.)**

- TCP uses multiple timers (at least conceptually) to do its work. The most important of these is the RTO (Retransmission TimeOut). When a segment is sent, a retransmission timer is started. If the segment is acknowledged before the timer expires, the timer is stopped. If, on the other hand, the timer goes off before the acknowledgement comes in, the segment is retransmitted (and the timer OS started again). It is difficult to predict how long should the timeout be.
- This problem is much more difficult in the transport layer than in data link protocols such as 802.11. In the latter case, the expected delay is measured in microseconds and is highly predictable (i.e., has a low variance), so the timer can be set to go off just slightly after the acknowledgement is expected, as shown in Fig below. Since acknowledgements are rarely delayed in the data link layer (due to lack of congestion), the absence of an acknowledgement at the expected time generally means either the frame or the acknowledgement has been lost.

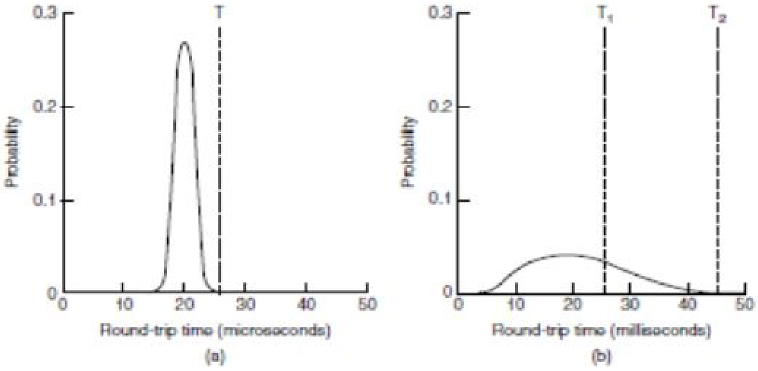


Fig : (a) Probability density of acknowledgement arrival times in the data link layer

(b) Probability density of acknowledgement arrival times for TCP

- TCP is faced with a radically different environment. The probability density function for the time it takes for a TCP acknowledgement to come back looks more like Fig. (b) than Fig. (a) as shown above. It is larger and more variable. Determining the round-trip time to the destination is tricky. Even when it is known, deciding on the timeout interval is also difficult. If the timeout is set too short, say, T1 in Fig. (b), unnecessary retransmissions will occur, clogging the Internet with useless packets. If it is set too long (e.g., T2), performance will suffer due to the long retransmission delay whenever a packet is lost. Furthermore, the mean and variance of the acknowledgement arrival distribution can change rapidly within a few seconds as congestion builds up or is resolved.
- The solution is to use a dynamic algorithm that constantly adapts the timeout interval, based on continuous measurements of network performance. The algorithm generally used by TCP is due to Jacobson (1988) and works as follows. For each connection, TCP maintains a variable, SRTT (Smoothed Round-Trip Time) that is the best current estimate of the round-trip time to the destination in question.

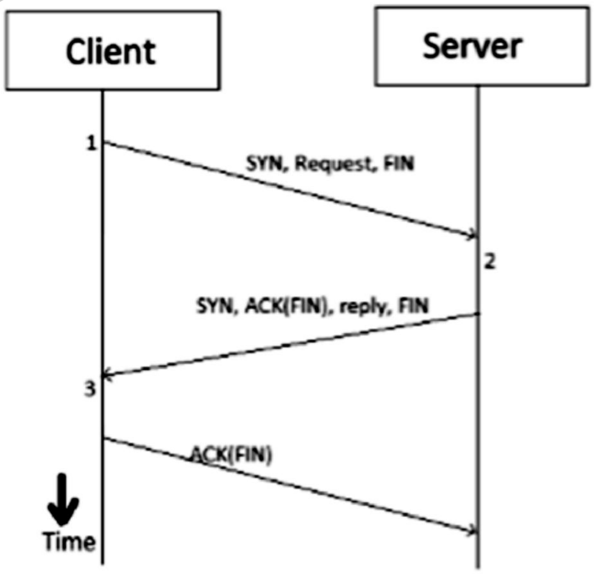
- When a segment is sent, a timer is started, both to see how long the acknowledgement takes and also to trigger a retransmission if it takes too long. If the acknowledgement gets back before the timer expires, TCP measures how long the acknowledgement took, say, R. It then updates SRTT according to the formula.

  SRTT = áSRTT + (1 á)RSRTT = áSRTT + (1 á)R

  where á is a smoothing factor that determines how quickly the old values are forgotten. Typically, á = 7/8. This kind of formula is an EWMA (Exponentially Weighted Moving Average) or low-pass filter that discards noise in the samples.

Transactional TCP:

- One of the ways to implement a client server system is to use the remote procedure call (RPC). We could also use UDP if both the request and reply are small enough to fit into single packets. But if these conditions are not satisfied, then UDP is no more an attractive option. The use of conventional TCP also does not prove to be a practical option due to its low efficiency.
- The disadvantage of using RPC is that even under the best working conditions we need nine packets. This number will increase in the worse cases.
- In order to overcome these problems, an experimental TCP variant was developed known as transactional TCP or T/TCP
- T/TCP is very efficient and has only two messages. T/TCP is described in RFCs 1379 and 1644. The T/TCP protocol is illustrated below:
- The first packet from the client consists of a SYN bit, the request and FIN(finished). This means that the client says I want to establish a connection, here is the data and I have finished.
- On receiving the request, the server prepares the reply and chooses how to respond to the clients request. If the reply fits in one packet then its reply is as shown in fig above which says “I acknowledge your FIN, this is my reply and I have finished”.
- The client then acknowledges server’s FIN and terminates the conversation. If reply is longer than 1 packet then the server can send more than one packet before sending the FIN bit.
- T/TCP has a major disadvantage of security problems due to which it has not gained a widespread popularity.



RCP using T/TCP

3.1.5 UDP PROTOCOL

Q9. Explain about UDP protocol

Ans : **(Imp.)**

Meaning

UDP (User Datagram Protocol) is a transport layer protocol in the TCP/IP protocol suite. It is a connectionless protocol, which means it does not establish a dedicated end-to-end connection before sending data packets. UDP is designed for applications that require fast, efficient, and low-latency data transmission.

Unlike TCP, UDP does not provide any reliability or flow control mechanisms. It simply sends datagrams (data packets) to the destination without checking if they arrive or not. This makes UDP very efficient and suitable for applications where speed is more important than reliability, such as video streaming, online gaming, and DNS (Domain Name System) lookups.

UDP datagrams consist of a header and a payload. The header contains information such as source and destination ports, length of the datagram, and a checksum for error detection. The payload is the actual data being transmitted.

UDP supports both unicast (one-to-one) and multicast (one-to-many) communication. In unicast communication, a single datagram is sent from the source to the destination IP address and port number. In multicast communication, a single datagram is sent to a group of IP addresses.

UDP is also used in conjunction with other protocols, such as TFTP (Trivial File Transfer Protocol) for transferring files, SNMP (Simple Network Management Protocol) for network management, and DHCP (Dynamic Host Configuration Protocol) for assigning IP addresses to devices.

- UDP is short for User Datagram Protocol.
- It is the simplest transport layer protocol.
- It has been designed to send data packets over the Internet.
- It simply takes the datagram from the network layer, attaches its header and sends it to the user.

Characteristics of UDP

- It is a connectionless protocol.
- It is a stateless protocol.
- It is an unreliable protocol.
- It is a fast protocol.
- It offers the minimal transport service.
- It is almost a null protocol.
- It does not guarantee in order delivery.
- It does not provide congestion control mechanism.
- It is a good protocol for data flowing in one direction.

Applications of UDP

Following applications use UDP-

- Applications which require one response for one request use UDP. Example- DNS.
- Routing Protocols like RIP and OSPF use UDP because they have very small amount of data to be transmitted.
- Trivial File Transfer Protocol (TFTP) uses UDP to send very small sized files.
- Broadcasting and multicasting applications use UDP.
- Streaming applications like multimedia, video conferencing etc use UDP since they require speed over reliability.
- Real time applications like chatting and online games use UDP.
- Management protocols like SNMP (Simple Network Management Protocol) use UDP.
- Bootp / DHCP uses UDP.
- Other protocols that use UDP are- Kerberos, Network Time Protocol (NTP), Network News Protocol (NNP), Quote of the day protocol etc.

Limitations of UDP

- It provides an unreliable connection delivery service. It does not provide any services of IP except that it provides process-to-process communication.
- The UDP message can be lost, delayed, duplicated, or can be out of order.
- It does not provide a reliable transport delivery service. It does not provide any acknowledgment or flow control mechanism. However, it does provide error control to some extent.

Q10. Explain briefly about UDP header format.

Ans :

UDP Header

The UDP header is a fixed-length header of 8 bytes. It consists of four fields:

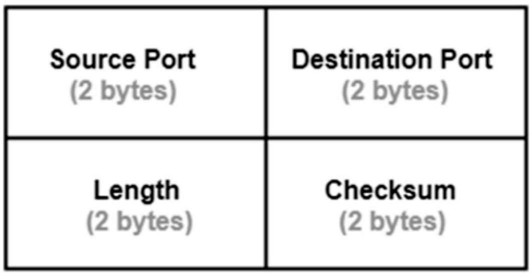
Source Port: This 16-bit field identifies the sending application's port number. A port is a logical address that identifies a specific process or service running on a device. The source port number helps the receiving application determine the source of the incoming datagram.

Destination Port: This 16-bit field identifies the destination application's port number. Like the source port, the destination port number helps the receiving application determine which application should receive the incoming datagram.

Length: This 16-bit field specifies the total length of the UDP datagram in bytes, including the header and the data. The minimum value of the length field is 8 bytes (for an empty datagram with no data). The maximum value of the length field is 65,535 bytes, which is the maximum size of a UDP datagram.

Checksum: The checksum field is a 16-bit field used to detect errors in the UDP datagram. The checksum is calculated by the sending application based on the contents of the datagram, including the header and data. The receiving application also calculates the checksum and compares it to the value in the datagram's header. If the checksum values do not match, the datagram is considered corrupted and is discarded.

Here's a visual representation of the UDP header format:



Source Port (2 bytes)	**Destination Port** (2 bytes)
Length (2 bytes)	**Checksum** (2 bytes)

UDP Header

The UDP header is very simple compared to other transport protocols, such as TCP. This simplicity makes UDP faster and more efficient than TCP, but also less reliable. Because UDP does not provide any error recovery or flow control mechanisms, it is up to the application using UDP to implement its own error detection and recovery mechanisms if necessary.

Short Question and Answers

1. Transmission Control Protocol

Ans :

The Transmission Control Protocol (TCP) is a transport protocol that is used on top of IP to ensure reliable transmission of packets.

TCP includes mechanisms to solve many of the problems that arise from packet-based messaging, such as lost packets, out of order packets, duplicate packets, and corrupted packets.

Since TCP is the protocol used most commonly on top of IP, the Internet protocol stack is sometimes referred to as TCP/IP.

2. State the Advantages and Disadvantages of TCP.

Ans :

Advantages.

- It provides a connection-oriented reliable service, which means that it guarantees the delivery of data packets. If the data packet is lost across the network, then the TCP will resend the lost packets.
- It provides a flow control mechanism using a sliding window protocol.
- It provides error detection by using checksum and error control by using Go Back or ARP protocol.
- It eliminates the congestion by using a network congestion avoidance algorithm that includes various schemes such as additive increase/multiplicative decrease (AIMD), slow start, and congestion window.

Disadvantage

- It increases a large amount of overhead as each segment gets its own TCP header, so fragmentation by the router increases the overhead.

3. Principle Termination of a TCP

Ans :

The termination of a TCP connection requires four exchanges of TCP segments.

- As a TCP connection is bidirectional (full duplex) connection termination process should be made in both directions of the communication. The client as the server can send a segment with FIN flag, this mean an end to sending data. Receiving a segment with FIN indicates that the other end will not send more data. The term used then is half closed, the connection is half closed.
- Typically, the client generates the transmission segment with a FIN flag, he made a closing force (active close), the server receives the segment realizes a passive close. The server acknowledge the FIN with ACK, informs the application for the release of this connection and when done then sends a FIN segment to the customer which in turn, acknowledge it with an ACK flag.

- All segments exchanged between client and server are numbered sequentially. The sequence number is an integer of 32 bits. Each side of the connection initiates and maintains the sequence of segments sent.
- The header segment always contains two fields, one for the sequence number of the sent segment and one for the sequence number of the acknowledged segment . The sequence number of the acknowledged segment is actually the number of the next segment expected. In other words the acquittal as "waiting segment number 10 on your part" means that the 9 was received correctly.

4. Congestion control in TCP.

Ans :

TCP Congestion Control

1. Receiver window size
2. Congestion window size

1. Receiver Window Size

Receiver window size is an advertisement of-

"How much data (in bytes) the receiver can receive without acknowledgement?"

- Sender should not send data greater than receiver window size.
- Otherwise, it leads to dropping the TCP segments which causes TCP Retransmission.
- So, sender should always send data less than or equal to receiver window size.
- Receiver dictates its window size to the sender through TCP Header.

2. Congestion Window-

- Sender should not send data greater than congestion window size.
- Otherwise, it leads to dropping the TCP segments which causes TCP Retransmission.
- So, sender should always send data less than or equal to congestion window size.
- Different variants of TCP use different approaches to calculate the size of congestion window.
- Congestion window is known only to the sender and is not sent over the links.

5. What are timers in TCP

Ans :

Timers used by TCP to avoid excessive delays during communication are called as TCP Timers.

The 4 important timers used by a TCP implementation are-

1. Time Out Timer
2. Time Wait Timer
3. Keep Alive Timer
4. Persistent Timer

6. UDP protocol

Ans :

Meaning

UDP (User Datagram Protocol) is a transport layer protocol in the TCP/IP protocol suite. It is a connectionless protocol, which means it does not establish a dedicated end-to-end connection before sending data packets. UDP is designed for applications that require fast, efficient, and low-latency data transmission.

Unlike TCP, UDP does not provide any reliability or flow control mechanisms. It simply sends datagrams (data packets) to the destination without checking if they arrive or not. This makes UDP very efficient and suitable for applications where speed is more important than reliability, such as video streaming, online gaming, and DNS (Domain Name System) lookups.

UDP datagrams consist of a header and a payload. The header contains information such as source and destination ports, length of the datagram, and a checksum for error detection. The payload is the actual data being transmitted.

7. Characteristics of UDP

Ans :

- It is a connectionless protocol.
- It is a stateless protocol.
- It is an unreliable protocol.
- It is a fast protocol.
- It offers the minimal transport service.
- It is almost a null protocol.
- It does not guarantee in order delivery.
- It does not provide congestion control mechanism.
- It is a good protocol for data flowing in one direction.

8. Applications of UDP

Ans :

Following applications use UDP-

- Applications which require one response for one request use UDP. Example- DNS.
- Routing Protocols like RIP and OSPF use UDP because they have very small amount of data to be transmitted.
- Trivial File Transfer Protocol (TFTP) uses UDP to send very small sized files.
- Broadcasting and multicasting applications use UDP.
- Streaming applications like multimedia, video conferencing etc use UDP since they require speed over reliability.
- Real time applications like chatting and online games use UDP.

9. Limitations of UDP

Ans :

- It provides an unreliable connection delivery service. It does not provide any services of IP except that it provides process-to-process communication.
- The UDP message can be lost, delayed, duplicated, or can be out of order.
- It does not provide a reliable transport delivery service. It does not provide any acknowledgment or flow control mechanism. However, it does provide error control to some extent.

10. TCP sliding window protocol.

Ans :

- Window management in TCP decouples the issues of acknowledgement of the correct receipt of segments and receiver buffer allocation.
- For example, suppose the receiver has a 4096-byte buffer, as shown in Fig. below. If the sender transmits a 2048-byte segment that is correctly received, the receiver will acknowledge the segment. However, since it now has only 2048 bytes of buffer space (until the application removes some data from the buffer), it will advertise a window of 2048 starting at the next byte expected.

Choose the Correct Answer

1. Which of the Following Protocol is a Connection-Oriented Protocol? [a]

   (a) TCP (b) ARP

   (c) UDP (d) ICMP

2. Which of the following protocols is used for reliable data transmission in a network? [b]

   (a) UDP (b) TCP

   (c) FTP (d) DNS

3. Which of the following TCP flags is used to initiate a connection? [a]

   (a) SYN (b) ACK

   (c) FIN (d) RST

4. What is the purpose of the window size field in TCP? [c]

   (a) To indicate the number of bytes in a segment

   (b) To identify the destination port number

   (c) To control the flow of data

   (d) To ensure that segments are received in the correct order

5. Among the following which timer is used during connection termination [b]

   (a) Time Out Timer (b) Time Wait Timer

   (c) Keep Alive Timer (d) Persistent Timer

6. Among the following which timer is used to deal with a zero-widow-size deadlock situation [d]

   (a) Time Out Timer (b) Time Wait Timer

   (c) Keep Alive Timer (d) Persistent Timer

7. Which of the following is NOT a common congestion control algorithm? [d]

   (a) Slow Start (b) Fast Retransmit

   (c) Random Early Detection (RED) (d) Round Trip Time (RTT)

8. What is the size of the UDP header? [c]

   (a) 4 bytes (b) 6 bytes

   (c) 8 bytes (d) 10 bytes

9. What is the maximum value for the port numbers in the UDP header? [a]

   (a) 16 bits (b) 24 bits

   (c) 32 bits (d) There is no maximum value.

10. Which of the Following Uses UDP as its Transport Protocol? [d]

    (a) Telnet (b) DNS

    (c) SMTP (d) HTTP

Fill in the blanks

1. UDP packets are called ________
2. TCP groups the number of the number of bytes together into a packet called a ________
3. The connection establishment procedure in TCP is susceptible to a serious security problem called the ________ attack.
4. In TCP to avoid excessive delays during communication are called as ________
5. TCP uses a ________ timer to prevent long idle TCP connections.
6. A ________ is a mechanism to increase the congestion window size gradually until congestion is detected
7. In TCP, a FIN segment consumes ________ sequence numbers if it does not carry data
8. TCP uses a ________ timer for retransmission of lost segments.
9. UDP datagrams consist of a ________ and ________
10. The ________ field is a 16-bit field used to detect errors in the UDP datagram.

Answers

1. Datagrams
2. Segement
3. SYN flooding
4. TCP timers
5. Keep alive
6. Slow start
7. One
8. Time out
9. Header and a payload.
10. Checksum

UNIT IV

Socket Programming

Primitive and Advanced Systems Calls, TCP Iterative and Concurrent programs, UDP systems calls, socket options IO Multiplexing, Asynchronous IO

4.1 Socket Programming

4.1.1 Primitive and Advanced Systems Calls

Q1. Explain briefly about Unix Socket structure.

Ans : **(Imp.)**

Meaning

Various structures are used in Unix Socket Programming to hold information about the address and port, and other information. Most socket functions require a pointer to a socket address structure as an argument. Structures defined in this chapter are related to Internet Protocol Family.

sockaddr

The first structure is *sockaddr* that holds the socket information -

```
struct sockaddr {
unsignedshort  sa_family;
char           sa_data[14];
};
```

This is a generic socket address structure, which will be passed in most of the socket function calls. The following table provides a description of the member fields “

Attribute	Values	Description
sa_family	AF_INET AF_UNIX AF_NS AF_IMPLINK	It represents an address family. In most of the Internet-based applications, we use AF_INET.
sa_data	Protocol-specific Address	The content of the 14 bytes of protocol specific address are interpreted according to the type of address. For the Internet family, we will use port number IP address, which is represented by *sockaddr_in* structure defined below.

sockaddr in

The second structure that helps you to reference to the socket's elements is as follows "

```
struct sockaddr_in {
shortint          sin_family;
unsignedshortint  sin_port;
struct in_addr    sin_addr;
unsignedchar      sin_zero[8];
};
```

Here is the description of the member fields -

Attribute	Values	Description
sa_family	AF_INET AF_UNIX AF_NS AF_IMPLINK	It represents an address family. In most of the Internet-based applications, we use AF_INET.
sin_port	Service Port	A 16-bit port number in Network Byte Order.
sin_addr	IP Address	A 32-bit IP address in Network Byte Order.
sin_zero	Not Used	You just set this value to NULL as this is not being used.

in addr

This structure is used only in the above structure as a structure field and holds 32 bit netid/hostid.

```
struct in_addr {
unsignedlong s_addr;
};
```

Here is the description of the member fields -

Attribute	Values	Description
s_addr	service port	A 32-bit IP address in Network Byte Order.

hostent

This structure is used to keep information related to host.

```
struct hostent {
char*h_name;
char**h_aliases;
int h_addrtype;
int h_length;
char**h_addr_list

#define h_addr  h_addr_list[0]
};
```

Here is the description of the member fields -

Attribute	Values	Description
h_name	ti.com etc.	It is the official name of the host. For example, tutorialspoint.com, google.com, etc.
h_aliases	TI	It holds a list of host name aliases.
h_addrtype	AF_INET	It contains the address family and in case of Internet based application, it will always be AF_INET.
h_length	4	It holds the length of the IP address, which is 4 for Internet Address.
h_addr_list	in_addr	For Internet addresses, the array of pointers h_addr_list[0], h_addr_list[1], and so on, are points to structure in_addr.

Note: h_addr is defined as h_addr_list[0] to keep backward compatibility.

servent

This particular structure is used to keep information related to service and associated ports.

```
struct servent {
char*s_name;
char**s_aliases;
int   s_port;
char*s_proto;
};
```

Here is the description of the member fields -

Attribute	Values	Description
s_name	http	This is the official name of the service. For example, SMTP, FTP POP3, etc.
s_aliases	ALIAS	It holds the list of service aliases. Most of the time this will be set to NULL.
s_port	80	It will have associated port number. For example, for HTTP, this will be 80.
s_proto	TCP UDP	It is set to the protocol used. Internet services are provided using either TCP or UDP.

Q2. Explain the concept of Socket Address Structure (SAS).

Ans : **(Imp.)**

Socket Address Structure

This SAS is between application and kernal. An address conversion function translates between text representation of an address and binary value that makes up SAS. IPv4 uses inet_addr and inet_ntoa. But inet_pton and inet_ntop handle IPv4 and IPv6. Above functions are protocol dependent. However the functions starting with sock are protocol independent. Most socket functions require a pointer to a socket address structure as an argument.

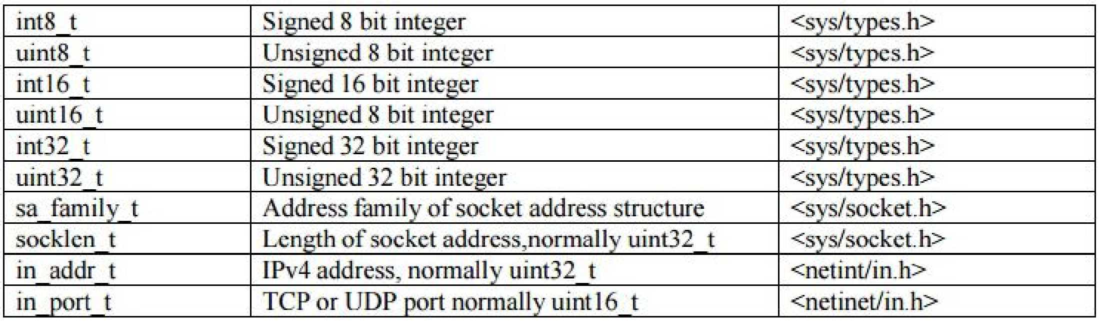
IPv4 Socket Address Structure

An IPv4 socket address structure, commonly called an "Internet socket address structure", is named sockaddr_in and is defined by including the <netinet/in.h> header.

```
structin_addr{
in_addr_ts_addr;/* 32-bit IPv4 address */
/* network byte ordered */
};

structsockaddr_in{
uint8_tsin_len;/* length of structure (16) */
sa_family_tsin_family;/* AF_INET */
in_port_tsin_port;/* 16-bit TCP or UDP port number */
/* network byte ordered */
structin_addrsin_addr;/* 32-bit IPv4 address */
/* network byte ordered */
charsin_zero[8];/* unused */
};
```

sin_len, added in 4.3 BSD, is not normally supported by many vendors. It facilitates handling of variable length socket address structures.

Various data types that are commonly used are listed below:



int8_t	Signed 8 bit integer	<sys/types.h>
uint8_t	Unsigned 8 bit integer	<sys/types.h>
int16_t	Signed 16 bit integer	<sys/types.h>
uint16_t	Unsigned 8 bit integer	<sys/types.h>
int32_t	Signed 32 bit integer	<sys/types.h>
uint32_t	Unsigned 32 bit integer	<sys/types.h>
sa_family_t	Address family of socket address structure	<sys/socket.h>
socklen_t	Length of socket address,normally uint32_t	<sys/socket.h>
in_addr_t	IPv4 address, normally uint32_t	<netint/in.h>
in_port_t	TCP or UDP port normally uint16_t	<netinet/in.h>

Length field is never used and set. It is used within the kernal before routines that deal with socket address structures from various protocol families.

Four socket functions – bind(), connect(), sendto(), sendmsg() -pass socket address structures from application to kernal. All invoke sockargs() in Berkley derived implementation. This function copies socket address structures and explicitly set the sin_len member to the size of the structure that was passed. The other socket functions that pass socket address to the application from kernal accept(), recvfrom(), recvmsg(), getpeername() and getsockname() all set the sin_len member before returning to the process.

sin_port, sin_family and sin_addr are the only required for Posix.1g. sin_zero is implemented to keep the structure length to 16 byte.

Generic Socket Address Structure

Socket address structures are always passed by reference when passed as an arguments to any of the socket functions.

int bind (int sockfd, struct sockaddr *, socklen_t);

But the socket function that accept these address structures as pointers must deal with any of these supported protocols. This calls for the any functions must cast the pointer to the protocol specific socket address structure to be a pointer to a generic socket address structure. For example

```
struct sockaddr_in serv;
  bind (sockfd, (struct sockaddr *) &serv, sizeof(serv));
```

If we omit the cast, the C compiler generates a warning of the form incompatible pointer type.

Different socket address structures are :

IPv4 (24 bytes), IPv6 (24 bytes), Unix variable length and Data link variable length.

IPv6 SAS:

Defined by # include<netinet/in.h> header. The structure is shown below:

```
struct in6_addr {
  uint8_t  s6_addr[16];         /* 128-bit IPv6 address */
                                /* network byte ordered */
};

#define SIN6_LEN     /* required for compile-time tests */

struct sockaddr_in6 {
  uint8_t        sin6_len;      /* length of this struct (28) */
  sa_family_t    sin6_family;   /* AF_INET6 */
  in_port_t      sin6_port;     /* transport layer port# */
                                /* network byte ordered */
  uint32_t       sin6_flowinfo; /* flow information, undefined */
  struct in6_addr sin6_addr;    /* IPv6 address */
                                /* network byte ordered */
  uint32_t       sin6_scope_id; /* set of interfaces for a scope */
};
```

Important points to note are:

The SIN_LEN constant must be defined if the system supports the length members for socket address structures.

The IPv6 family is AF_INET6

The members in this structure are ordered so that if the sockaddr_in6 structure is 64 bit aligned, so is the 128 bit sin6_addr member.

The sin6_flowinfo member is divided into three fields o The low order 24 bits are the flow label

The next 4 bits are the priority o The next 4 bits are reserved.

3. Write the comparison of four socket structures.

Ans :

Following figure shows the comparison of the four socket address structures that are encountered.

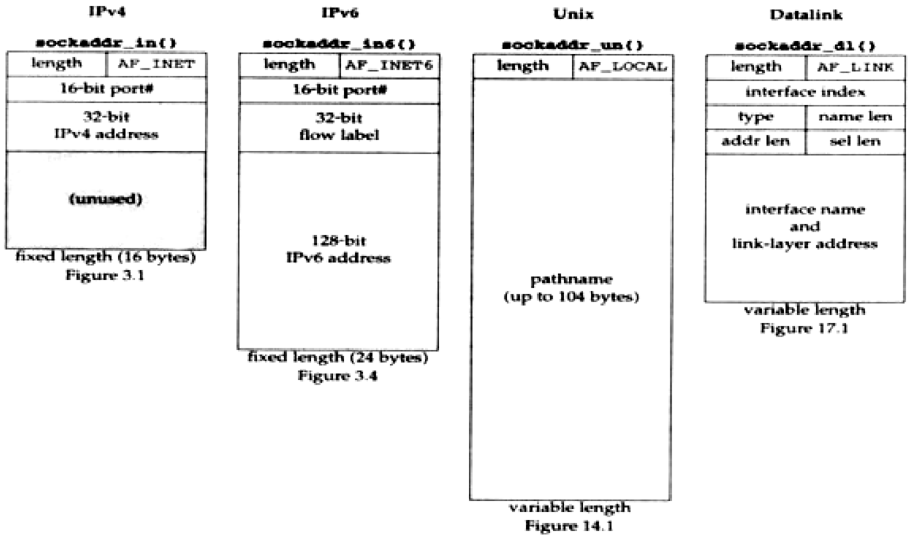


Fig. : Comparison of various socket address structures.

Value Result Arguments: The socket address structure is passed to any of the socket function by reference. The length of the structure is also passed as argument to the function. But the way the length is passed depends on which direction it is being passed. From the process to the kernal or kernal to the process.

The three functions bind(), connect() and sendto() pass a socket address structure from the process to the kernal. One argument to these three function is the pointer to the socket address structure and another argument is the integer size of the structure.

struct sockaddr_in serv;

connect (sockfd, (SA *) & serv, sizeof(serv));

Since the kernal is passed both pointer and the size of what the pointer points to, it knows exactly how much data to copy from the process into the kernal. Following figures shows this scenario:

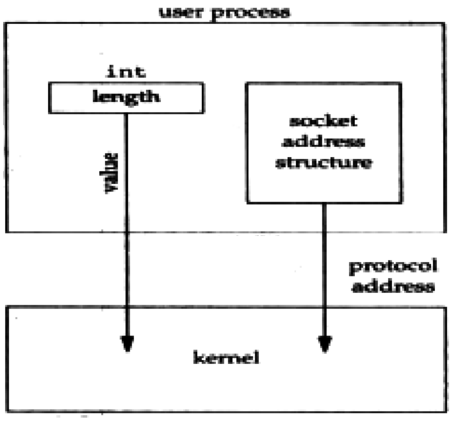


Fig. : Socket address structure passed from process to kernel.

The four functions accept(), recvmsg(), getsockname() and getpeername() pass a socket address structure from kernal to the process, the reverse direction form the precious scenario. In this case the length is passed as pointer to an integer containing the size of structure as in

struct sockaddr_un cli; // Unix domain// socklen_t len;

```
    len = sizeof (cli);
    getpeername (unixfd, (SA *) &cli, & len );
```

The reason that the size changes from an integer to be a pointer to an integer is because the size is both value when the function is called (it tells the kernal the size of the structure so that the kernal does not write past the end of the structure when filling it) and it is the result when the function results (It tells the process how much information the kernal actually stored in the structure). This type of argument is called value – result arguments.

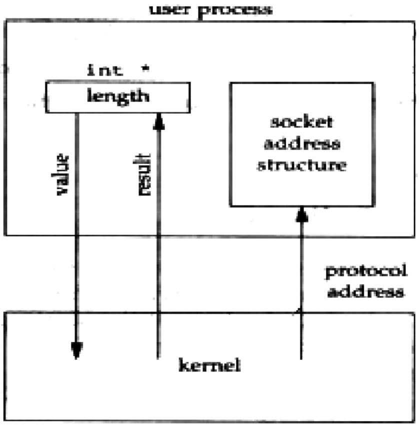


Fig. : Socket address structure passed from kernel to process

With variable length socket address structure, the value returned can be less than the maximum size of the structure.

Q4. What are elementary socket system calls? Explain.

Ans :

socket

To do network I/O, the first thing a process must do is to call the **socket** system call, specifying the type of communication protocol desired.

```
#include      <sys/types.h>
#include      <sys/socket.h>
int socket(int family, int type, int protocol);
```

The *family* is one of

AF_UNIX	— Unix internal protocols
AF_INET	— Internet protocols
AF_NS	— Xerox NS Protocols
AF_IMPLINK	— IMP link layer

The AF_ prefix stands for “address family.” In the first project, we are going to use AF_INET.

The socket *type* is one of the following:

SOCK_STREAM	stream socket
SOCK_DGRAM	datagram socket
SOCK_RAW	raw socket
SOCK_SEQPACKET	sequenced packet socket
SOCK_RDM	reliably delivered message socket (not implemented yet)

The protocol argument to the socket system call is typically set to 0 for most user applications. The valid combinations are shown as follows.

family	type	protocol	Actual protocol
AF_INET	SOCK_DGRAM	IPPROTO_UDP	UDP
AF_INET	SOCK_STREAM	IPPROTO_TCP	TCP
AF_INET	SOCK_RAW	IPPROTO_ICMP	ICMP
AF_INET	SOCK_RAW	IPPROTO_RAW	(raw)

bind

The **bind** system call assigns a name to an unnamed socket.

```
#include    <sys/types.h>
#include    <sys/socket.h>

int bind(int sockfd, struct sockaddr *myaddr, int addrlen);
```

The first argument is the socket descriptor returned from socket system call. The second argument is a pointer to a protocol-specific address and the third argument is the size of this address. There are three uses of bind.

Servers register their well-known address with the system. It tells the system "this is my address and any messages received for this address are to be given to me." Both connection-oriented and connectionless servers need to do this before accepting client requests.

1. A client can register a specific address for itself.
2. A connectionless client needs to assure that the system assigns it some unique address, so that the other end (the server) has a valid return address to send its responses to. This corresponds to making certain an envelope has a valid return address, if we expect to get a reply from the person we sent the letter to.

connect

A client process connects a socket descriptor following the socket system call to establish a connection with a server.

```
#include    <sys/types.h>
#include    <sys/socket.h>
int connect(int sockfd, struct sockaddr *servaddr, int addrlen);
```

The sockfd is a socket descriptor that was returned by the socket system call. The second and third arguments are a pointer to a socket address, and its size, as described earlier.

For most connection-oriented protocols (TCP, for example), the connect system call results in the actual establishment of a connection between the local system and the foreign system.

The connect system call does not return until the connection is established, or an error is returned to the process.

The client does not have to bind a local address before calling connect. The connection typically causes these four elements of the association 5-tuple to be assigned: local-addr, local-process,foreign-addr, and foreign-process. In all the connection-oriented clients, we will let connect assign the local address.

listen

This system call is used by a connection-oriented server to indicate that it is willing to receive connections.

```
#include    <sys/types.h>
#include    <sys/socket.h>
int listen(int sockfd, int backlog);
```

It is usually executed after both the socket and bind system calls, and immediately before the accept system call. The backlog argument specifies how many connection requests can be queued by the system while it waits for the server to execute the accept system call. This argument is usually specified as 5, the maximum value currently allowed.

accept

After a connection-oriented server executes the listen system call described above, an actual connection from some client process is waited for by having the server execute the accept system call.

```
#include    <sys/types.h>
#include    <sys/socket.h>
int accept(int sockfd, struct sockaddr *peer, int *addrlen);
```

accept takes the first connection request on the queue and creates another socket with the same properties as sockfd. If there are no connection requests pending, this call blocks the caller until one arrives.

The peer and addrlen arguments are used to return the address of the connected peer process (the client). addrlen is called a value-result argument: the caller sets its value before the system call, and the system call stores a result in the variable. For this system call the caller sets addrlen to the size of the sockaddr structure whose address is passed as the peer argument.

send, sendto, recv and recvfrom

These system calls are similar to the standard read and write system calls, but additional arguments are required.

```
#include    <sys/types.h>
#include    <sys/socket.h>
int send(int sockfd, char *buff, int nbytes, int flags);
int sendto(int sockfd, char *buff, int nbytes, int flags, struct sockaddr *to, int addrlen);
int recv(int sockfd, char *buff, int nbytes, int flags);
int recvfrom(int sockfd, char *buff, int nbytes, int flags, struct sockaddr *from, int *addrlen);
```

The first three arguments, sockfd, buff, and nbytes, to the four system calls are similar to the first three arguments for read and write. The flags argument can be safely set to zero ignoring the details for it. The to argument for sendto specifies the protocol-specific address of where the data is to be sent. Since this address is protocol-specific, its length must be specified by addrlen. There cv from system call fills in the protocol-specific address of who sent the data into from. The length of this address is also returned to the caller in addrlen. Note that the final argument to send to is an integer value, while the final argument to recvfrom is a pointer to an integer value.

close

The normal Unix close system call is also used to close a socket.

```
int close(int fd);
```

If the socket being closed is associated with a protocol that promises reliable delivery (e.g., TCP or SPP), the system must assure that any data within the kernel that still has to be transmitted or acknowledged, is sent. Normally, the system returns from the close immediately, but the kernel still tries to send any data already queued.

Q5. Explain Advanced socket system calls in UNIX.

Ans : **(Imp.)**

A stream socket must be in a connected state before any data may be sent or received on it. A connection to another socket is created with a connect (2) call. Once connected, data may be transferred using read (2) and write (2) calls or some variant of the send (2) and recv (2) calls. When a session has been completed a close (2) may be performed.

The socket and bind system calls are called in the same way as in the connection-oriented case. Again the bind call is optional at the client side.

The bind system call assigns a name to an unnamed socket. #include <sys/types.h> #include <sys/socket.h> int bind(int sockfd, struct sockaddr *myaddr, int addrlen); The first argument is the socket descriptor returned from socket system call. The second argument is a pointer to a protocol-specific address and the third argument is the size of this address.

1. readv and writev system calls:

These two functions are similar to read and write , but readv and writev let us read into or write from one or more buffers with a single function call. These operations are called scatter read (since the input data is scattered into multiple application buffers) and gather write (since multiple buffers are gathered for a single output operation).

```
#include <sys/uio.h>

int readv(int fd , struct iovec iov[] , int iovcount) ;

int writev(int fd , struct iovec iov[] , int iovcount) ;
```

These two system calls use the following structure that is defined in <syst/uio.h>:

```
struct iovec{

caddr_t iov_base; /*strating address of buffer*/

int iov_len; /*size of buffer in size*/
```

- The writev system call write the buffers specified by iov[0], iov[1], through iov[iovcount-1].
- The readv system call does the input equivalent. It always fills one buffer (as specified but the iov_len value) before proceeding to the next buffer in the iov array.
- Both system calls return the total number of bytes read and written.

2. getpeername - get the name of the peer socket

```
#include <sys/socket.h>

int getpeername(int socket, struct sockaddr *address,

socklen_t *address_len);
```

The getpeername() function retrieves the peer address of the specified socket, stores this address in the sockaddr structure pointed to by the address argument, and stores the length of this address in the object pointed to by the address_len argument.

- If the actual length of the address is greater than the length of the supplied sockaddr structure, the stored address will be truncated.
- If the protocol permits connections by unbound clients, and the peer is not bound, then the value stored in the object pointed to by *address* is unspecified.

3. **getsockname** - get the socket name

#include <sys/socket.h>
int getsockname(int *socket*, struct sockaddr **address*,
socklen_t **address_len*);

- The getsockname() function retrieves the locally-bound name of the specified socket,stores this address in the sockaddr structure pointed to by the address argument, and stores the length of this address in the object pointed to by the address_len argument.
- If the actual length of the address is greater than the length of the supplied sockaddr structure, the stored address will be truncated.
- If the socket has not been bound to a local name, the value stored in the object pointed to by address is unspecified.

4. **getsockopt and setsockopt** allow socket options values to be queried and set, respectively.

int getsockopt (sockid, level, optName, optVal, optLen);

- **sockid**: integer, socket descriptor
- **level**: integer, the layers of the protocol stack (socket, TCP, IP)
- **optName**: integer, option
- **optVal**: pointer to a buffer; upon return it contains the value of the specified option
- **optLen**: integer, in-out parameter it returns -1 if an error occurred int setsockopt (sockid, level, optName, optVal, optLen);
- **optLen** is now only an input parameter

5. **shutdown system call:** this system call terminates the network connect and provide more control on the full-duplex connection.

int shutdown(int sockfd,int howto);

i) if howto argument is 0: no more data can be received on the socket.

ii) if howto argument is 1: no more output to be allowed on the socket.

iii) if howto argument is 2: both send and receive to be disallowed

- shutdown allows either directions to be closed independent of the other direction

6. **select ():**

When a server (or client) has multiple connections, it can be difficult to guess which clients (or servers) have written data on a socket. One approach, called polling, is to use non blocking recv( ) and loop through all the connections. This is inefficient. Another approach, using fork( ), is to fork a child process for each connections. This is also inefficient. A better option is to wait on all the connections simultaneously. This can be done using select( ) function.

#include<sys/select.h>
#include<sys/time.h>
int select (int maxfdp1, fd_set *readset, fd_set *writeset, fd_set *exceptset, const strut timeval *timeout);
-- returns # of ready descriptions, 0 if timeout occurs, -1 on error.
maxfdp1 – the maximum descriptor to test +1, the possible number of descriptors to test, <256.
readset – used to check which connections have data read.
writeset – used to check which connections have space for more output.
exceptset – used to check which connections have exceptions, such as OOB data.
timeout – specifies how long to block waiting for ready connection.

There are three options;

=0 means the call is non blocking. Used for polling connections.

>0 means the call times out after this amount of time if there are no ready connection during this time, NULL means the call blocks until a connection is ready for I/O.

The format of the timeval structure is:

```
struct timeval {
        long tv_sec;        /*seconds*/
        long tv_usec;       /*microseconds*/
};
```

select( ) is used to determine which socket are ready for reading, writing, or exception handling. Use NULL for any fd_set that doesn't ned to be checked.

The fd_set detatype typically uses one bit per socket fd. The appropriate method for using fd_set is to zero out all the bits and then set each one that is to be tested. The select( ) call modifies the readset, writeset, and exceptset variables by clearing the bits that are not ready for I/O. The user then tests each bit to see which are set and processes the corresponding sockets.

Operations on fd_sets should be performed using the following macros:

```
void FD_ZERO(fd_set *fdset);                /* clear all bits in fdset**/
void FD_SET(int fd, fd_set *dset);          /* turn on the bit for f in fdset */
void FD_CLR(int fd, fd_set *fdset);         /* clear off the bits in fdset */
int  FD_ISSET(int fd, fd_set *fdset);       /* test the bit for fd in fdset */
```

7. recvmsg and sendmsg Functions

14.5 recvmsg and sendmsg Functions

These two functions are the most general of all the I/O functions, Indeed, we could replace all calls to read, ready, recv, and recgfrom with calls to recvmsg. Similarly all calls to the various output functions could be replaced with calls to sendmsg.

```
#include <sys/socket.h>
ssize_t recvmsg (intsockfd, struct msghdr *msg, intflags);
ssize_t sendmsg (intsockfd, struct msghdr *msg, intflags);
```

Both return: number of bytes read or written if OK, –1 on error

Both functions package most arguments into a msghdr structure.

```
struct msghdr  {
        void          *msg_name;        /* protocol address */
        socklen_t      msg_namelen;     /* size of protocol address */
        struct lovec  *msg_iov;         /* scatter/gather array */
        int            msg_iovlen;      /* # elements in msg_iov */
        void          *msg_control;     /* ancillary data (cmsghdr struct) */
        socklen_t     msg_controllen;   /* length of ancillary data */
        int           msg_flags;        /* flags returned by recvmsg( ) */
};
```

The msghdr structure that we show is the one specified in POSIX. Some systems still use an older msghdr structure that originated with 4.2BSC. This older structure does not have the msg_flags member, and the msg_control and msg_controllen members are named msg_accrights and msg_accrightslen. The newer form of the msghdr structure is often available using conditional compilation flags. The only form of ancillary data supported by the older structure is the passing of file descriptors (called access rights).

4.1.2 TCP Iterative and Concurrent Programs

Q6. What is TCP? Explain TCP client Server model.

Ans : **(Imp.)**

Meaning

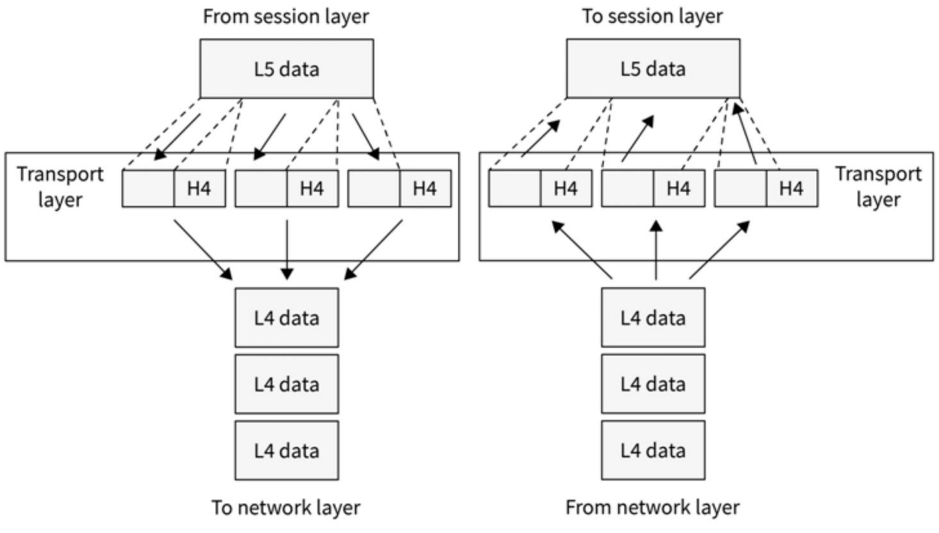
The OSI model stands for Open Systems Interconnection model. The OSI model is also known as the ISO-OSI model as it was developed by ISO (International Organization for Standardization). It is a conceptual reference model that describes the entire flow of information from one computer to the other computer. The OSI model is a 7-layered model so it is also known as a 7-layered architecture model. The basic idea behind layered architecture is to divide the design into smaller pieces. To reduce the design complexity, most networks are organized in a series of layers. The transport layer' is one of the seven layers of the OSI model.

The transport layer is the fourth layer of the OSI model which is responsible for the process to process delivery of data. The main aim of the transport layer is to maintain order so that the data must be received in the same sequence as it was sent by the sender. The transport layer provides two types of services namely - connection-oriented and connectionless.

The functions provided by the transport layer are as follows:

- The transport layer maintains the order of data.
- It receives the data from the upper layer and converts it into smaller parts known as segments.
- One of the major tasks of the transport layer is to add the port addressing (addition of a port number to the header of the data). The port number is added so that the data can be sent to the respective process only.
- The transport layer on the receiver's end reassembles the segments to form the actual data.
- The transport layer also deals with flow control and error control (discussed in the physical layer section).

Refer to the image below to see the basic transmission of data and working of the transport layer.



The transmission control protocol is a transport layer connection-oriented protocol that defined the standard of establishing and maintaining the conversation (or connection) that will be used by the applications to exchange the data. The transmission control protocol is one of the most important and widely used protocols of the IP suite. The IP suite or the What is meant by Internet Protocol Suite? The Internet Protocol Suite is the standard network model and stack of communication protocols that are used on the Internet. Hence, for the data transmission in the communication network, we use the transmission control protocol.

Client-server Architecture

A Client is a computer system that accesses the services provided by a server. On the other hand, a server is a powerful centralized hub that stores various information and handles the requests of the client(s). A server can be on the same machine or can be a remote machine as well.

Let us now briefly discuss the client and server in brief before getting into the steps and TCP client-server implementation.

The client-server network model or architecture is one of the most widely used networking models. In the client-server network, the files are not stored on the hard drive of each computer system. Instead, the files are centrally stored and backed up on a specialized computer known as a server. Here, a server is designed to efficiently provide data to a remote client. On a large-scale network, there can be more than one server. Let us discuss the various type of servers:

- **File Server**: A file server is used to transfer files to the client(s). A file server is a computer attached to a network that provides a location for shared disk access.
- **Email Server**: An email server is used to deal with the internal email system. STMP deals with File Servers.
- **Web Server**: A web server is used to control access to the internet and block any unsuitable websites. It accepts requests via HTTP or its secure variant HTTPS.
- **Print Server**: A print server is used to deal with all of the printing requests from the client(s). It connects printers to client computers over a network.

In a client-server network, there is a specific server and specific clients connected to the server. Refer to the diagram below to see the basic overview of a client-server network system architecture.

Q7. Explain TCP client –server implementation with the help of program.

Ans : **(Imp.)**

Implementation

Let us now move into the TCP client-server implementation.

Steps

First lets us discuss the steps involved in the TCP client-server implementation.

Server-side implementation steps

1. First we will create a socket with the help of a pre-defined socket() system call.
2. After creating the socket() system call, we will initialize the address structure of the socket and then bind the address using a pre-defined system call namely - the bind() system call so that the socket is attached to the proper port number of the desired process.
3. After making the connection, we need to listen to the connection. Now, for listening, we use another pre-defined system call i.e. listen().
4. After listening, we need to accept the connection to the client and here we use the pre-defined system called accept() for the same. The main advantage of the accept() system call is that this system call blocks the line until a client connects to the server. If we want multiple connections then one connection is handled after the other.
5. Now, we can easily send and receive data. For sending the data, we use the send() system call and for receiving the system call, the recv() system is used.

6. At last, we end the connection from the server side using the pre-defined function i.e. close() function.

Client-side implementation steps

1. First we will create a socket with the help of a pre-defined socket() system call so that the socket is attached to the proper port number of the desired process.
2. After creating the socket() system call, we will initialize the address structure of the socket according to the server and then connect the newly created socket to the server's address using another pre-defined system call namely - connect().
3. Now, we can easily send and receive data. For sending the data to the server, we use the send() system call and for receiving the system call from the server, the recv() system is used.
4. As we know that in all the connections, we must end the connection so we must end the connection from the client side as well using the pre-defined function i.e. close() function.

Let us now move into the implementation in C language.

Server Side Code

```
// adding the necessary libraries.
#include<stdio.h>
#include<netdb.h>
#include<netinet/in.h>
#include<stdlib.h>
#include<string.h>
#include<sys/socket.h>
#include<sys/types.h>
#define MAX 80
#define PORT 8080
#define SA struct sockaddr

// A function created for chatting between client and server.
void func(int connectID)
{
// defining a buffer of MAX size
   char buffer[MAX];
   int n;

// infinite loop for chat
   for (;;)
   {
      bzero(buffer, MAX);

// reading the message from the client and then copying it into the buffer.
      read(connectID, buffer, sizeof(buffer));
// printing the buffer which contains the client response
```

```
        printf("From the client: %s\t To the client : ", buffer);
        bzero(buffer, MAX);

         n = 0;
// copying the server message into the buffer using the while loop
        while ((buffer[n++] = getchar()) != '\n')
              ;

// sending the copied buffer message to the client.
       write(connectID, buffer, sizeof(buffer));

// Ending the chat if the message contains "Exit".
        if (strncmp("exit", buffer, 4) == 0)
         {
            printf("Closing Server\n");
             break;
         }
    }
}

// Defining the main function.
int main()
{
   int sockfd, connectID, length;
struct socketAddress_in serverAddress, client;

// creating a socket and verifying it.
   sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd == -1)
    {
       printf("Socket not created!\n");
        exit(0);
    }
    else
       printf("Socket successfully created!\n");
   bzero(&serverAddress, sizeof(serverAddress));

// assigning IP address, and PORT address.
```

```
    serverAddress.sin_family = AF_INET;
    serverAddress.sin_addr.s_addr = htonl(INADDR_ANY);
    serverAddress.sin_port = htons(PORT);
// Binding newly created socket to given IP
    if ((bind(sockfd, (SA *)&serverAddress, sizeof(serverAddress))) != 0)
     {
        printf(“Socket binding failed!\n”);
         exit(0);
     }
     else
        printf(“Socket binding done successfully!\n”);

// Now the server is ready to listen and verification
    if ((listen(sockfd, 5)) != 0)
     {
        printf(“Listen failed!\n”);
         exit(0);
     }
     else
        printf(“Server listening!\n”);
    length = sizeof(client);

// Accept the data packet from the client and verification
    connectID = accept(sockfd, (SA *)&client, &length);
     if (connectID < 0)
     {
        printf(“Server accept failed!\n”);
         exit(0);
     }
     else
        printf(“Server accept the client!\n”);

    func(connectID);

// finally closing the connection.
    close(sockfd);
}
```

Output:

```
Socket successfully created!
Socket binding is done successfully!
Server listening!
The server accepts the client!
```

```
From the client: hi
    To the client: hello
From the client: exit
    To the client: exit
Closing Server
```

- In the above server-side code, we first included the various necessary header files particularly socket programming necessary header files.
- After that, we defined the PORT number of the server and the maximum buffer size that stores messages from the client and the server.
- We then created a function that takes the connection ID and then runs an infinite loop and in each iteration, it reads the message from the client (using the predefined read() function), copied it to the buffer, and then prints it on the console. The Buffer is a region of memory used to temporarily hold data while it is being moved from one place to another. A buffer is used when moving data between processes within a computer.
- Again we empty the buffer using a predefined function bzero().
- We then check if the client wants to close the connection, if the client enters exit then we close the connection and break else the iteration continues.

Now, in the main() function,

- We first create a socket using the pre-defined function socket() and then check if the connection is successfully created or not. If the connection is successfully created then we proceed further else we exit the program.
- We then bind the newly created socket to the given IP. Similar to the socket definition, if the binding is done successfully then we proceed further else we exit the program.
- Now, after the binding, we start listening to the port and verify the connection. After successful listening, we accept the connection using the pre-defined function accept() and then call the function func() by providing the connection ID.
- At last, we close the connection using the close() function.

Client-side Code

```
// adding the necessary libraries.
#include<arpa/inet.h>
#include<netdb.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<strings.h>
#include<sys/socket.h>
#include<unistd.h>

#define MAX 80
#define PORT 8080
#define SA struct socketAddress
```

```
// A function created for chatting between client and server.
void func(int sockfd)
{
// defining a buffer of MAX size
   char buffer[MAX];
    int n;
// infinite loop for chat
    for (;;)
    {
      bzero(buffer, sizeof(buffer));

// Getting the message.
       printf("Enter the string: ");
        n = 0;

// copying the server message into the buffer using the while loop
       while ((buffer[n++] = getchar()) != '\n')
            ;

// sending the copied buffer message.
       write(sockfd, buffer, sizeof(buffer));
       bzero(buffer, sizeof(buffer));

// reading the message.
       read(sockfd, buffer, sizeof(buffer));
       printf("From the server : %s", buffer);

// Ending the chat if the message contains "Exit".
        if ((strncmp(buffer, "exit", 4)) == 0)
         {
           printf("Closing Client\n");
            break;
         }
    }
}

// Defining the main function.
int main()
{
   int sockfd, connfd;
struct socketAddress_in socketAddress, cli;
// socket creation and verification
   sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd = = -1)
     {
        printf("Socket not created!\n");
         exit(0);
     }
    else
        printf("Socket successfully created!\n");
   bzero(&socketAddress, sizeof(socketAddress));

// assigning IP address, and PORT address.
   socketAddress.sin_family = AF_INET;
   socketAddress.sin_addr.s_addr = inet_addr("127.0.0.1");
   socketAddress.sin_port = htons(PORT);

// connecting the client socket to the server socket.
   if (connect(sockfd, (SA *)&socketAddress, sizeof(socketAddress)) != 0)
     {
       printf("Connection with the server failed!\n");
         exit(0);
     }
    else
       printf("Connected to the server!\n");

    func(sockfd);

// finally closing the connection.
    close(sockfd);
}
```

Output:

```
Socket successfully created!
Connected to the server!
Enter the string: hi
From the server: hello
Enter the string: exit
From the server: exit
Closing Client
```

- In the above client-side code, we first included the various necessary header files particularly socket programming necessary header files.
- After that, we defined the PORT number of the client and the maximum buffer size that stores messages from the client and the server.

- We then created a function that runs an infinite loop and in each iteration, it accepts the message from the server and copies it to the buffer, and then writes the message on the buffer defined previously.
- Again we empty the buffer using a predefined function bzero() so that a fresh new request can be dealt with.
- We then read the message of the buffer and then print it on the console.
- We then check if the server wants to close the connection, if the server enters exit then we close the connection and break else the iteration continues.

Now, in the main() function,

- We first create a socket using the pre-defined function socket() and then check if the connection is successfully created or not. If the connection is successfully created then we proceed further else we exit the program.
- We then assign the IP address, PORT number, etc to the socket and then try to connect to the server using the pre-defined function connect(). Similar to the socket one, if the connection is successful then we continue to proceed further, else we break from the program because without a proper connection we cannot request the data from the server.
- At last, we close the connection using the close() function.

Q8. Describe the implementation of TCP concurrent client and server.

Ans :

TCP Concurrent Server

Implement TCP client and server (concurrent) where the client gets input from the user and sends it to the server. The server displays it on the screen. The server then gets another input from the user and sends it to the client. The client displays it on the screen. The process continues till server or client sends "bye" to the other party.

The server processes the multiple client parallel.

TCP Concurrent Server

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define PORT 12345
#define BUFFER_SIZE 1024
#define MAX_CLIENTS 5
void handle_client(int client_socket) {
  char buffer[BUFFER_SIZE];
```

```
    int bytes_received;
    while ((bytes_received = recv(client_socket, buffer, BUFFER_SIZE, 0)) > 0) {
        buffer[bytes_received] = '\0';
        printf("Received message from client: %s\n", buffer);
        send(client_socket, buffer, bytes_received, 0);
    }

    close(client_socket);
}

int main() {
    int server_socket, client_socket, max_clients;
    struct sockaddr_in server_address, client_address;
    socklen_t client_address_len;
     pid_t pid;

    // Create server socket
    server_socket = socket(AF_INET, SOCK_STREAM, 0);
    if (server_socket < 0) {
        perror("socket() failed");
         exit(EXIT_FAILURE);
    }

    // Bind socket to port
    memset(&server_address, 0, sizeof(server_address));
    server_address.sin_family = AF_INET;
    server_address.sin_addr.s_addr = INADDR_ANY;
    server_address.sin_port = htons(PORT);

    if (bind(server_socket, (struct sockaddr *)&server_address, sizeof(server_address)) < 0) {
        perror("bind() failed");
         exit(EXIT_FAILURE);
    }

    // Listen for incoming connections
    if (listen(server_socket, MAX_CLIENTS) < 0) {
        perror("listen() failed");
         exit(EXIT_FAILURE);
    }

    // Accept incoming connections
    client_address_len = sizeof(client_address);
    max_clients = 0;

    while (1) {
      client_socket = accept(server_socket, (struct sockaddr *)&client_address, &client_address_len);
        if (client_socket < 0) {
            perror("accept() failed");
            exit(EXIT_FAILURE);
        }

        if (max_clients == MAX_CLIENTS) {
          printf("Refusing new connection from %s:%d - maximum number of clients reached\n",
                 inet_ntoa(client_address.sin_addr), ntohs(client_address.sin_port));
            close(client_socket);
        } else {
            pid = fork();
            if (pid < 0) {
                perror("fork() failed");
                exit(EXIT_FAILURE);
            }

            if (pid == 0) {
                // Child process
                close(server_socket);
                handle_client(client_socket);
                exit(EXIT_SUCCESS);
            } else {
                // Parent process
                close(client_socket);
                max_clients++;
            }
        }
    }
    return 0;
}
```

TCP Concurrent Client

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
```

```
#include <netinet/in.h>
#include <arpa/inet.h>

#define SERVER_ADDRESS “127.0.0.1”
#define PORT 12345
#define BUFFER_SIZE 1024

int main() {
  int client_socket, bytes_received;
  struct sockaddr_in server_address;
  char buffer[BUFFER_SIZE];

  // Create client socket
  client_socket = socket(AF_INET, SOCK_STREAM, 0);
  if (client_socket < 0) {
    perror(“socket() failed”);
     exit(EXIT_FAILURE);
  }

  // Connect to server
  memset(&server_address, 0, sizeof(server_address));
  server_address.sin_family = AF_INET;
  server_address.sin_addr.s_addr = inet_addr(SERVER_ADDRESS);
  server_address.sin_port = htons(PORT);

  if (connect(client_socket, (struct sockaddr *)&server_address, sizeof(server_address)) < 0) {
    perror(“connect() failed”);
     exit(EXIT_FAILURE);
  }

  // Send messages to the server
  for (int i = 0; i < 5; i++) {
    sprintf(buffer, “Hello from client %d”, i);
    if (send(client_socket, buffer, strlen(buffer), 0) < 0) {
       perror(“send() failed”);
        exit(EXIT_FAILURE);
     }

    bytes_received = recv(client_socket, buffer, BUFFER_SIZE, 0);
    if (bytes_received < 0) {
       perror(“recv() failed”);
        exit(EXIT_FAILURE);
     }

    buffer[bytes_received] = ‘\0’;
```

```
     printf("Received from server: %s\n", buffer);
    }

   // Close the socket
   close(client_socket);

   return 0;
}
```

Q9. Design a concurrent server for handling multiple clients using fork()

Ans :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define MAX_CLIENTS 10
#define BUFFER_SIZE 1024
void handle_client(int client_sock) {
   char buffer[BUFFER_SIZE];
    int num_bytes;
   while ((num_bytes = recv(client_sock, buffer, BUFFER_SIZE, 0)) > 0) {
       buffer[num_bytes] = '\0';
      printf("Received message from client: %s\n", buffer);
       if (send(client_sock, buffer, num_bytes, 0) = = -1) {
           perror("send");
            exit(1);
        }
    }
    if (num_bytes = = -1) {
       perror("recv");
        exit(1);
    }
   close(client_sock);
}
int main(int argc, char *argv[]) {
   int server_sock, client_sock;
   struct sockaddr_in server_addr, client_addr;
```

```
socklen_t client_addr_size;
int port = 1234;
int num_clients = 0;
pid_t pid;
int i;
if ((server_sock = socket(AF_INET, SOCK_STREAM, 0)) = = -1) {
    perror("socket");
    exit(1);
}

memset(&server_addr, 0, sizeof(server_addr));
server_addr.sin_family = AF_INET;
server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
server_addr.sin_port = htons(port);

if (bind(server_sock, (struct sockaddr *)&server_addr, sizeof(server_addr)) = = -1) {
    perror("bind");
    exit(1);
}

if (listen(server_sock, MAX_CLIENTS) = = -1) {
    perror("listen");
    exit(1);
}

printf("Server listening on port %d...\n", port);

while (1) {
   client_addr_size = sizeof(client_addr);
   client_sock = accept(server_sock, (struct sockaddr *)&client_addr, &client_addr_size);
    if (client_sock = = -1) {
        perror("accept");
        continue;
    }
   printf("Client connected: %s\n", inet_ntoa(client_addr.sin_addr));
    num_clients+ +;
    if (num_clients > MAX_CLIENTS) {
      printf("Maximum number of clients exceeded. Connection refused.\n");
        close(client_sock);
        num_clients—;
        continue;
    }
    pid = fork();
```

```
        if (pid = = -1) {
          perror(“fork”);
           exit(1);
         }
        else if (pid = = 0) { // child process
          close(server_sock);
          handle_client(client_sock);
           exit(0);
         }
        else { // parent process
           close(client_sock);
         }
      }

      return 0;
    }
```

The above program is a simple example of a TCP concurrent server that uses fork() to handle multiple clients simultaneously. Here’s a brief explanation of how it works:

- The program starts by defining some constants, including the maximum number of clients (MAX_CLIENTS) and the size of the buffer used for sending and receiving messages (BUFFER_SIZE).
- The handle_client() function is defined to handle incoming client connections. It takes the client socket as an argument and reads data from the socket until the client disconnects or an error occurs. For each message received, it prints the message to the console and sends the message back to the client. When the client disconnects, the function closes the client socket and exits.
- In the main() function, the server socket is created using the socket() system call. Then, the server address is initialized and the socket is bound to the specified port using the bind() system call. The socket is then put into listening mode using the listen() system call.
- The program enters a loop that accepts incoming client connections using the accept() system call. For each connection, a new client socket is created, and the client address is printed to the console. If the maximum number of clients has been reached, the connection is refused, and the socket is closed. Otherwise, a new child process is created using fork(). In the child process, the server socket is closed, and the handle_client() function is called with the client socket as an argument. In the parent process, the client socket is closed, and the loop continues to accept new connections.
- If an error occurs at any point during the program execution, the error message is printed to the console, and the program exits.

Overall, the program demonstrates how to use fork() to handle multiple client connections in a TCP server. The child processes created by fork() allow each client connection to be handled independently, without blocking the main server loop.

4.2 UDP Systems Calls

Q10. What is UDP? Explain the steps in doing UDP programming.

Ans :

Meaning

User Datagram Protocol (UDP) is a Transport Layer protocol. UDP is a part of the Internet Protocol suite, referred to as UDP/IP suite. Unlike TCP, it is an unreliable and connectionless protocol. So, there is no need to establish a connection prior to data transfer. The UDP helps to establish low-latency and loss-tolerating connections establish over the network.The UDP enables process to process communication.

Though Transmission Control Protocol (TCP) is the dominant transport layer protocol used with most of the Internet services; provides assured delivery, reliability, and much more but all these services cost us additional overhead and latency. Here, UDP comes into the picture. For real-time services like computer gaming, voice or video communication, live conferences; we need UDP. Since high performance is needed, UDP permits packets to be dropped instead of processing delayed packets. There is no error checking in UDP, so it also saves bandwidth.

Programming with UDP/IP sockets

There are a few steps involved in using sockets:

1. Include necessary libraries and header files: The first step is to include the necessary libraries and header files for socket programming in C/C++. This includes the <sys/socket.h> header file for socket operations, <netinet/in.h> for internet addresses, <arpa/inet.h> for IP address conversion functions, and <unistd.h> for close() function.

```
#include <netinet/in.h>
#include <sys/socket.h>
#include <unistd.h>
```

2. Create a socket: The next step is to create a socket using the socket() system call. This system call takes three arguments: the address domain (AF_INET for IPv4, AF_INET6 for IPv6), the socket type (SOCK_DGRAM for UDP), and the protocol (usually set to 0)

```
. int socket(int domain, int type, int protocol);
```

The socket() function creates a new socket and returns a socket descriptor (integer). It takes three arguments:

- **domain** specifies the protocol family used for the socket (e.g. AF_INET for IPv4, AF_INET6 for IPv6).
- **type** specifies the type of socket (e.g. SOCK_DGRAM for UDP, SOCK_STREAM for TCP).
- **protocol** specifies the protocol to be used (usually set to 0).

3. Bind the socket: If the program is a server, it must bind the socket to a specific IP address and port number using the bind() system call. This allows the socket to listen for incoming connections. If the program is a client, it can skip this step.

```
int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
```

The bind() function binds a socket to a specific address and port. It takes three arguments:

- sockfd is the socket descriptor returned by the socket() function.
- addr is a pointer to a struct sockaddr structure that contains the address to bind to.
- addrlen is the length of the address structure.

4. Send data: To send data, the program uses the sendto() function to send data to a specific IP address and port number. This function takes several arguments, including the socket descriptor, the data to send, the size of the data, and the address of the receiver.

```
ssize_t sendto(int sockfd, const void *buf, size_t len, int flags,
          const struct sockaddr *dest_addr, socklen_t addrlen);
```

The sendto() function sends data over a UDP socket. It takes six arguments:

- sockfd is the socket descriptor returned by the socket() function.
- buf is a pointer to the data to be sent.
- len is the length of the data to be sent.
- flags specifies the type of message transmission (usually set to 0).
- dest_addr is a pointer to a struct sockaddr structure that contains the destination address and port.
- addrlen is the length of the address structure.

5. Receive data: To receive data, the program uses the recvfrom() function to receive data from a specific IP address and port number. This function takes several arguments, including the socket descriptor, a buffer to store the received data, the maximum size of the buffer, and the address of the sender.

```
ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags,
          struct sockaddr *src_addr, socklen_t *addrlen);
```

The recvfrom() function receives data from a UDP socket. It takes five arguments:

- sockfd is the socket descriptor returned by the socket() function.
- buf is a pointer to the buffer where the received data will be stored.
- len is the maximum length of the data to be received.
- flags specifies the type of message transmission (usually set to 0).
- src_addr is a pointer to a struct sockaddr structure that will be filled with the source address and port.
- addrlen is a pointer to the length of the address structure.

6. Close the socket: When the program is finished using the socket, it should close the socket using the close() system call.

    int close(int sockfd);

    The close() function closes the socket connection. It takes one argument:

    sockfd is the socket descriptor returned by the socket() function.

7. Error handling: It is important to handle errors in the socket programming, as network errors can occur at any time. Error handling may include checking return values of functions, displaying error messages, and closing sockets if necessary.

Q11. Demonstrate the concept of UDP sockets.

Ans :

Example code of using UDP sockets

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#define PORT 8080
#define MAX_BUFFER_SIZE 1024
int main() {
    int sockfd;
   struct sockaddr_in server_addr, client_addr;
   char buffer[MAX_BUFFER_SIZE] = {0};
   int bytes_received;
   socklen_t client_len;
   // Create socket
   if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
      perror("socket creation failed");
      exit(EXIT_FAILURE);
    }
   // Bind socket to a specific IP address and port
   memset(&server_addr, 0, sizeof(server_addr));
   server_addr.sin_family = AF_INET;
```

```
        server_addr.sin_addr.s_addr = INADDR_ANY;
        server_addr.sin_port = htons(PORT);
        if (bind(sockfd, (const struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
            perror("bind failed");
            exit(EXIT_FAILURE);
         }
        // Wait for incoming data
        while (1) {
           memset(&client_addr, 0, sizeof(client_addr));
            client_len = sizeof(client_addr);
           bytes_received = recvfrom(sockfd, buffer, MAX_BUFFER_SIZE, MSG_WAITALL,
                                    (struct sockaddr *)&client_addr, &client_len);
           buffer[bytes_received] = '\0';
           printf("Received message: %s\n", buffer);
            // Send a response back to the client
            char *response = "Hello from server!";
           sendto(sockfd, response, strlen(response), MSG_CONFIRM,
                  (const struct sockaddr *)&client_addr, client_len);
         }
        // Close the socket
        close(sockfd);
        return 0;
      }
```

4.3 Socket Options

Q12. Explain about socket options.

Ans :

In addition to binding a socket to a local address or connecting it to a destination address, application programs need a method to control the socket. For example, when using protocols that use time out and retransmission, the application program may want to obtain or set the time-out parameters.

There are various ways to get and set the options that affect a socket:

- The getsockopt and setsockopt functions.
- The fcntl function, which is the POSIX way to set a socket for nonblocking I/O, signal-driven I/O, and to set the owner of a socket.
- The ioctl function.

getsockopt and setsockopt Functions

These two functions apply only to sockets:

#include<sys/socket.h>

```
intgetsockopt(intsockfd,intlevel,intoptname,void*optval,socklen_t*optlen);

intsetsockopt(intsockfd,intlevel,intoptname,constvoid*optvalsocklen_toptlen);

/* Both return: 0 if OK,–1 on error */
```

Arguments:

- sockfd must refer to an open socket descriptor.
- level specifies the code in the system that interprets the option: the general socket code or some protocol-specific code (e.g., IPv4, IPv6, TCP, or SCTP).
- optval is a pointer to a variable from which the new value of the option is fetched by setsockopt, or into which the current value of the option is stored by getsockopt. The size of this variable is specified by the final argument optlen, as a value for setsockopt and as a value-result for getsockopt.

level	*optname*	get	se t	Description	Flag	Datat ype
SOL_SOC KET	SO_BROADCAST	x	x	Permit sending of broadcast datagrams	x	int
	SO_DEBUG	x	x	Enable debug tracing	x	int
	SO_DONTROUTE	x	x	Bypass routing table lookup	x	int
	SO_ERROR	x		Get pending error and clear		int
	SO_KEEPALIVE	x	x	Periodically test if connection still alive	x	int
	SO_LINGER	x	x	Linger on close if data to send		linger{ }
	SO_OOBINLINE	x	x	Leave received out-of-band data inline	x	int
	SO_RCVBUF	x	x	Receive buffer size		int
	SO_SNDBUF	x	x	Send buffer size		int
	SO_RCVLOWAT	x	x	Receive buffer low-water mark		int
	SO_SNDLOWAT	x	x	Send buffer low-water mark		int
	SO_RCVTIMEO	x	x	Receive timeout		timev al{}
	SO_SNDTIMEO	x	x	Send timeout		timev al{}
	SO_REUSEADDR	x	x	Allow local address reuse	x	int
	SO_REUSEPORT	x	x	Allow local port reuse	x	int
	SO_TYPE	x		Get socket type		int
	SO_USELOOPBA CK	x	x	Routing socket gets copy of what it sends	x	int

There are two basic types of options:

- **Flags**: binary options that enable or disable a certain feature (flags)
- **Values**: options that fetch and return specific values that we can either set or examine.

The column labeled “Flag” specifies a flag option:

- getsockopt: *optval is an integer. The value returned in *optval is zero if the option is disabled, or nonzero if the option is enabled.
- setsockopt: it requires a nonzero *optval to turn the option on, and a zero value to turn the option off.

If the “Flag” column does not contain a block dot, then the option is used to pass a value of the specified datatype between the user process and the system.

Q13. Explain IPV4 Socket Options

Ans : **(Imp.)**

IPv4 Socket Options

Here are some common socket options related to IPv4:

1. IP_TTL: This option sets the time-to-live (TTL) field in IP packets. This determines how many hops a packet can take before it is discarded.
2. IP_HDRINCL: This option allows the application to specify its own IP header instead of having the system generate one. This can be used for creating custom IP packets.
3. IP_OPTIONS: This option allows the application to set custom IP options in the IP header, such as the record route option or the source route option.
4. IP_MULTICAST_TTL: This option sets the TTL for multicast packets sent by the socket. This determines how many hops the packets can take before they are discarded.
5. IP_MULTICAST_LOOP: This option enables or disables loopback of multicast packets on the local machine. If loopback is enabled, multicast packets sent by the socket will be received by the same socket on the local machine.
6. IP_ADD_MEMBERSHIP/IP_DROP_MEMBERSHIP: These options allow a socket to join or leave a multicast group.

To set these options, you can use the **`setsockopt()`** function with the appropriate level and option name. For example:

Example:

```
int sockfd = socket(AF_INET, SOCK_STREAM, 0);

int ttl = 64;

setsockopt(sockfd, IPPROTO_IP, IP_TTL, &ttl, sizeof(ttl));
```

In this example, the IP_TTL option is set to 64 for the socket sockfd.

The SO socket options are used to set or retrieve socket-level properties. Here are some common SO socket options related to IPv4:

1. SO_REUSEADDR: This option allows a socket to bind to a port that is already in use by another socket. This is useful when a server process needs to be restarted while there are still active client connections.
2. SO_SNDBUF/SO_RCVBUF: These options allow you to set the size of the send and receive buffers, respectively. Increasing the buffer size can improve network performance in some cases.

3. SO_KEEPALIVE: This option enables TCP keepalive packets, which are used to check if the other end of the connection is still alive.

4. SO_LINGER: This option determines what happens when the socket is closed. If the option is set to 0 (the default), the socket is closed immediately and any unsent data is discarded. If the option is set to a non-zero value, the socket will wait for the specified number of seconds to send any remaining data before closing.

5. SO_BROADCAST: This option allows a socket to send broadcast packets.

6. SO_ERROR: This option retrieves the last error that occurred on the socket.

To set these options, you can use the setsockopt() function with the appropriate level and option name. For example:

Example:

```
int sockfd = socket(AF_INET, SOCK_STREAM, 0);
int reuse = 1;
setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &reuse, sizeof(reuse));
```

In this example, the SO_REUSEADDR option is set to 1 for the socket sockfd.

4.4 I/O MULTIPLEXING

Q14. What is I/O Multiplexing? Explain various I/O models.

Ans : **(Imp.)**

When the TCP client is handling two inputs at the same time: standard input and a TCP socket, we encountered a problem when the client was blocked in a call to fgets (on standard input) and the server process was killed. The server TCP correctly sent a FIN to the client TCP, but since the client process was blocked reading from standard input, it never saw the EOF until it read from the socket (possibly much later).

We want to be notified if one or more I/O conditions are ready (i.e., input is ready to be read, or the descriptor is capable of taking more output). This capability is called I/O multiplexing and is provided by the select and poll functions, as well as a newer POSIX variation of the former, called pselect.

I/O multiplexing is typically used in networking applications in the following scenarios:

- When a client is handling multiple descriptors (normally interactive input and a network socket)
- When a client to handle multiple sockets at the same time (this is possible, but rare)
- If a TCP server handles both a listening socket and its connected sockets
- If a server handles both TCP and UDP
- If a server handles multiple services and perhaps multiple protocols

I/O multiplexing is not limited to network programming. Many nontrivial applications find a need for these techniques.

I/O Models

We first examine the basic differences in the five I/O models that are available to us under Unix:

- blocking I/O
- nonblocking I/O
- I/O multiplexing (select and poll)

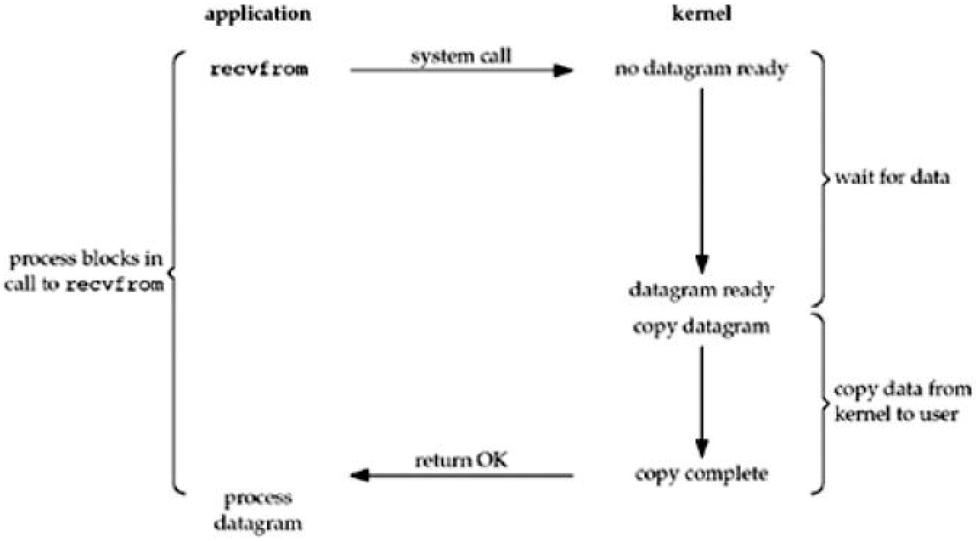
- signal driven I/O (SIGIO)
- asynchronous I/O (the POSIX aio_ functions)

There are normally two distinct phases for an input operation:

1. Waiting for the data to be ready. This involves waiting for data to arrive on the network. When the packet arrives, it is copied into a buffer within the kernel.
2. Copying the data from the kernel to the process. This means copying the (ready) data from the kernel's buffer into our application buffer

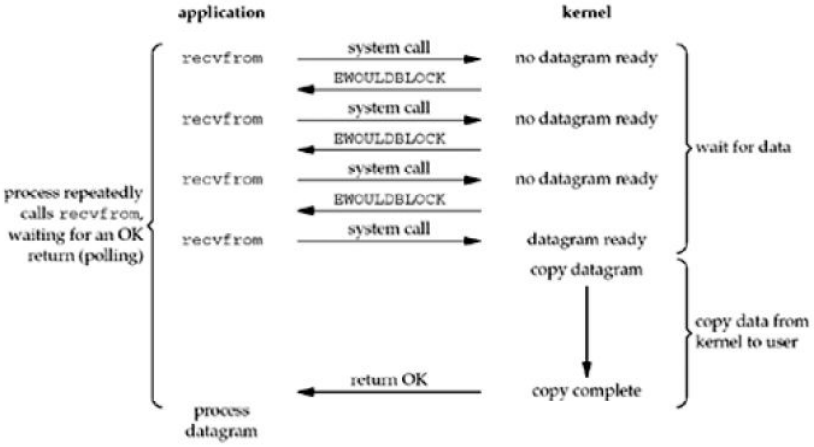
Blocking I/O Model

The most prevalent model for I/O is the blocking I/O model (which we have used for all our examples in the previous sections). By default, all sockets are blocking. The scenario is shown in the figure below:



Nonblocking I/O Model

When a socket is set to be nonblocking, we are telling the kernel "when an I/O operation that I request cannot be completed without putting the process to sleep, do not put the process to sleep, but return an error instead". The figure is below:

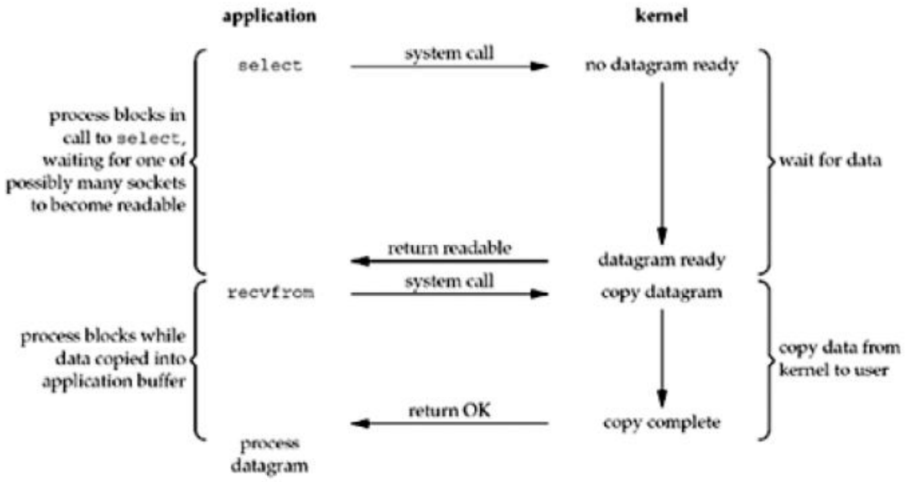


- For the first three recvfrom, there is no data to return and the kernel immediately returns an error of EWOULDBLOCK.
- For the fourth time we call recvfrom, a datagram is ready, it is copied into our application buffer, and recvfrom returns successfully. We then process the data.
- When an application sits in a loop calling recvfrom on a nonblocking descriptor like this, it is called polling.

The application is continually polling the kernel to see if some operation is ready. This is often a waste of CPU time, but this model is occasionally encountered, normally on systemsdedicated to one function.

I/O Multiplexing Model

With I/O multiplexing, we call select or poll and block in one of these two system calls, instead of blocking in the actual I/O system call. The figure is a summary of the I/O multiplexing model:


application
kernel
select
system call
no datagram ready
process blocks in call to select, waiting for one of possibly many sockets to become readable
wait for data
return readable
datagram ready
recvfrom
system call
copy datagram
process blocks while data copied into application buffer
copy data from kernel to user
return OK
copy complete
process datagram

We block in a call to select, waiting for the datagram socket to be readable. When select returns that the socket is readable, we then call recvfrom to copy the datagram into our application buffer.

The select function allows the process to instruct the kernel to either:

Wait for any one of multiple events to occur and to wake up the process only when one or more of these events occurs, or When a specified amount of time has passed.

This means that we tell the kernel what descriptors we are interested in (for reading, writing, or an exception condition) and how long to wait. The descriptors in which we are interested are not restricted to sockets; any descriptor can be tested using select.

```
#include <sys/select.h>
#include <sys/time.h>
int select(int maxfdp1, fd_set *readset, fd_set *writeset, fd_set *exceptset,
const struct timeval *timeout);
/* Returns: positive count of ready descriptors, 0 on timeout, –1 on error */
```

The timeout argument *

The timeout argument tells the kernel how long to wait for one of the specified descriptors to become ready. A timeval structure specifies the number of seconds and microseconds.

```
struct timeval {
```

```
long tv_sec; /* seconds */
long tv_usec; /* microseconds */
};
```

There are three possibilities for the *timeout*:

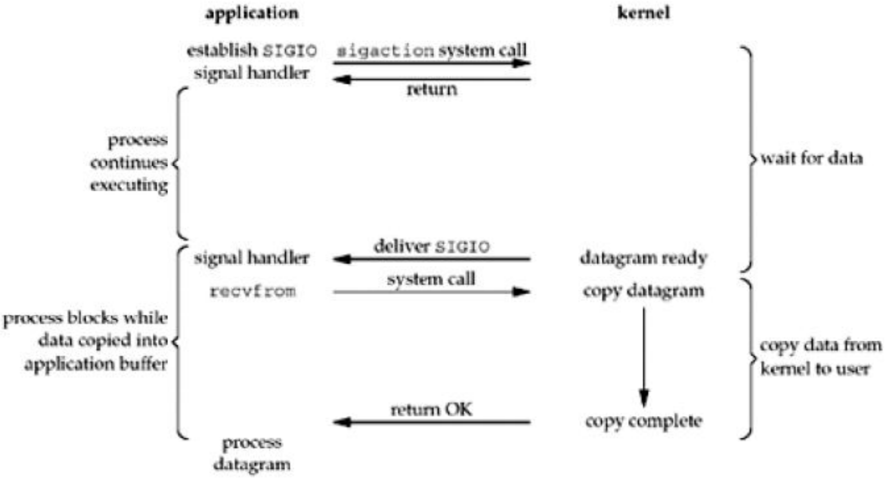
1. **Wait forever** (timeout is specified as a null pointer). Return only when one of the specified descriptors is ready for I/O.
2. **Wait up to a fixed amount of time** (timeout points to a timeval structure). Return when one of the specified descriptors is ready for I/O, but do not wait beyond the number of seconds and microseconds specified in the timeval structure.
3. **Do not wait at all** (timeout points to a timeval structure and the timer value is 0, i.e. the number of seconds and microseconds specified by the structure are 0). Return immediately after checking the descriptors. This is called polling.

Multithreading with blocking I/O *

Another closely related I/O model is to use multithreading with blocking I/O. That model very closely resembles the model described above, except that instead of using select to block on multiple file descriptors, the program uses multiple threads (one per file descriptor), and each thread is then free to call blocking system calls like recvfrom.

Signal-Driven I/O Model

The signal-driven I/O model uses signals, telling the kernel to notify us with the SIGIO signal when the descriptor is ready. The figure is below:



We first enable the socket for signal-driven I/O and install a signal handler using the sigaction system call. The return from this system call is immediate and our process continues; it is not blocked.

When the datagram is ready to be read, the SIGIO signal is generated for our process. We can either:

- read the datagram from the signal handler by calling recvfrom and then notify the main loop that the data is ready to be processed
- notify the main loop and let it read the datagram.

The advantage to this model is that we are not blocked while waiting for the datagram to arrive. The main loop can continue executing and just wait to be notified by the signal handler that either the data is ready to process or the datagram is ready to be read.

Socket-related Signals:

1. **SIGIO:** indicates that a socket is ready for asynchronous I/O as we have discussed. need to specify process ID or process group ID to receive the signal.

   Need to enable asynchronous I/O.

2. **SIGURG:** indicates urgent data is coming due to 1)OOB data or 2) control status information.

   ➢ need to specify process group ID to receive the signal,e.g., fcntl(sd,F_SETOWN, - getpgid( )). Use flag=MSG_URG to send and receive the OOB data.

   ➢ If O_OOBINLINE is set, we must use STOCATMARK ioctl to read OOB

   ➢ data. setsockopt(sd, SOL_SOCKET, SO_OOBINLINE, &seton, sizeof(seton)); /*let seton=1*/

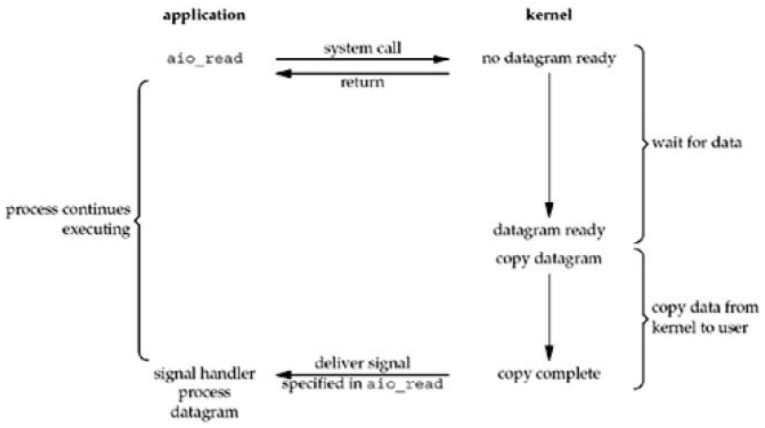
   ➢ If ((n=ioctl(sd,STOCATMARK, &start)>0) read(sd, buf, n); /*OOB data is in buf with n bytes.*/

3. **SIGPIPE:** indicates socket, pipe, or FIFO can never be written to. Sent only to the associated process,

Asynchronous I/O Model

Asynchronous I/O is defined by the POSIX specification, and various differences in the *realtime* functions that appeared in the various standards which came together to form the current POSIX specification have been reconciled.

These functions work by telling the kernel to start the operation and to notify us when the entire operation (including the copy of the data from the kernel to our buffer) is complete. The main difference between this model and the signal-driven I/O model is that with signal-driven I/O, the kernel tells us when an I/O operation can be initiated, but with asynchronous I/O, the kernel tells us when an I/O operation is complete.



Process can wait for the kernel to send signal SIGIO when a specified descriptor is ready for I/O. 3 things to do:

1. Establish a handler for SIGIO by calling signal(SIGIO, ???);
2. Set PID or PGID for the descriptor to receive SIGIO by calling fcntl(fd, F_SETOWN, getpid());
3, Enable asynchronous I/O by calling fcntl(fd, F_SETFL,FASYNC).

```
/* Copy standard input to standard output. */
#define BUFFSIZE 4096
main()
{ int n;
char buff[BUFFSIZE];
} while ( (n = read(0, buff, BUFFSIZE)) > 0) write(1, buff, n);
```

4.5 Asynchronous I/O

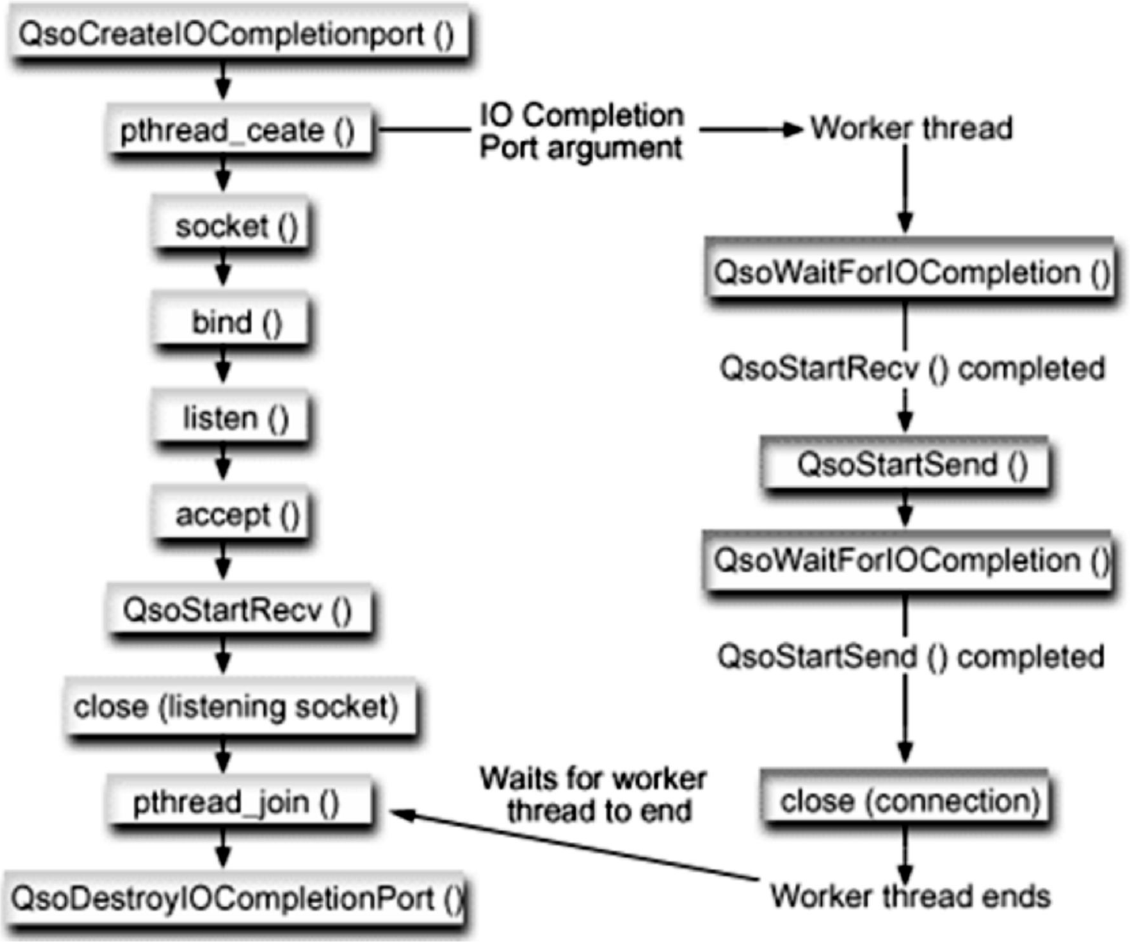
Q15. Explain briefly about Asynchronous I/O.

Ans :

The application starts an input or an output function, specifying an I/O completion port handle. When the I/O is completed, status information and an application-defined handle are posted to the specified I/O completion port. The post to the I/O completion port wakes up exactly one of possibly many threads that are waiting. The application receives the following items:

- A buffer that was supplied on the original request
- The length of data that was processed to or from that buffer
- A indication of what type of I/O operation has been completed
- Application-defined handle that was passed on the initial I/O request

This application handle can be the socket descriptor identifying the client connection, or a pointer to storage that contains extensive information about the state of the client connection. Since the operation was completed and the application handle was passed, the worker thread determines the next step to complete the client connection.


QsoCreateIOCompletionport ()
pthread_ceate ()
IO Completion Port argument
Worker thread
socket ()
bind ()
listen ()
accept ()
QsoStartRecv ()
close (listening socket)
pthread_join ()
QsoDestroyIOCompletionPort ()
QsoWaitForIOCompletion ()
QsoStartRecv () completed
QsoStartSend ()
QsoWaitForIOCompletion ()
QsoStartSend () completed
close (connection)
Worker thread ends
Waits for worker thread to end

Flow of Socket Events: Asynchronous I/O Server

The following sequence of the socket calls provides a description of the graphic. It also describes the relationship between the server and worker examples. Each set of flows contain links to usage notes on specific APIs. If you need more details on the use of a particular API, you can use these links. This flow describes the socket calls in the following sample application. Use this server example with the generic client example.

1. Master thread creates I/O completion port by calling QsoCreateIOCompletionPort()
2. Master thread creates pool of worker thread(s) to process any I/O completion port requests with the pthread_create function.
3. Worker thread(s) call QsoWaitForIOCompletionPort() which waits for client requests to process.
4. The master thread accepts a client connection and proceeds to issue a QsoStartRecv() which specifies the I/O completion port upon which the worker threads are waiting.
5. At some point, a client request arrives asynchronous to the server process. The sockets operating system loads the supplied user buffer and sends the completed QsoStartRecv() request to the specified I/O completion port. One worker thread is awoken and proceeds to process this request.
6. The worker thread extracts the client socket descriptor from the application-defined handle and proceeds to echo the received data back to the client by performing a QsoStartSend() operation.
7. If the data can be immediately sent, then the QsoStartSend() API returns indication of the fact; otherwise, the sockets operating system sends the data as soon as possible and posts indication of the fact to the specified I/O completion port. The worker thread gets indication of data sent and can wait on the I/O completion port for another request or end if instructed to do so. The QsoPostIOCompletion() API can be used by the master thread to post a worker thread end event.
8. Master thread waits for worker thread to finish and then destroys the I/O completion port by calling the QsoDestroyIOCompletionPort() API.

Short Question and Answers

1. Unix Socket structure.

Ans :

Meaning

Unix Socket Programming to hold information about the address and port, and other information. Most socket functions require a pointer to a socket address structure as an argument. Structures defined in this chapter are related to Internet Protocol Family.

2. Socket Address Structure

Ans :

This SAS is between application and kernal. An address conversion function translates between text representation of an address and binary value that makes up SAS. IPv4 uses inet_addr and inet_ntoa. But inet_pton and inet_ntop handle IPv4 and IPv6. Above functions are protocol dependent. However the functions starting with sock are protocol independent. Most socket functions require a pointer to a socket address structure as an argument.

3. Advanced socket system

Ans :

A stream socket must be in a connected state before any data may be sent or received on it. A connection to another socket is created with a connect (2) call. Once connected, data may be transferred using read (2) and write (2) calls or some variant of the send (2) and recv (2) calls. When a session has been completed a close (2) may be performed.

The socket and bind system calls are called in the same way as in the connection-oriented case. Again the bind call is optional at the client side.

The bind system call assigns a name to an unnamed socket. #include <sys/types.h> #include <sys/socket.h> int bind(int sockfd, struct sockaddr *myaddr, int addrlen); The first argument is the socket descriptor returned from socket system call.

4. What is TCP

Ans :

Meaning

The OSI model stands for Open Systems Interconnection model. The OSI model is also known as the ISO-OSI model as it was developed by ISO (International Organization for Standardization). It is a conceptual reference model that describes the entire flow of information from one computer to the other computer. The OSI model is a 7-layered model so it is also known as a 7-layered architecture model. The basic idea behind layered architecture is to divide the design into smaller pieces. To reduce the design complexity, most networks are organized in a series of layers. The transport layer' is one of the seven layers of the OSI model.

5. Functions of Transport Layers.

Ans :

➢ The transport layer maintains the order of data.

➢ It receives the data from the upper layer and converts it into smaller parts known as segments.

- One of the major tasks of the transport layer is to add the port addressing (addition of a port number to the header of the data). The port number is added so that the data can be sent to the respective process only.
- The transport layer on the receiver's end reassembles the segments to form the actual data.
- The transport layer also deals with flow control and error control (discussed in the physical layer section).

6. Client-server Architecture

Ans :

A Client is a computer system that accesses the services provided by a server. On the other hand, a server is a powerful centralized hub that stores various information and handles the requests of the client(s). A server can be on the same machine or can be a remote machine as well.

Let us now briefly discuss the client and server in brief before getting into the steps and TCP client-server implementation.

The client-server network model or architecture is one of the most widely used networking models. In the client-server network, the files are not stored on the hard drive of each computer system. Instead, the files are centrally stored and backed up on a specialized computer known as a server. Here, a server is designed to efficiently provide data to a remote client. On a large-scale network, there can be more than one server. Let us discuss the various type of servers:

- **File Server**: A file server is used to transfer files to the client(s). A file server is a computer attached to a network that provides a location for shared disk access.
- **Email Server**: An email server is used to deal with the internal email system. STMP deals with File Servers.
- **Web Server**: A web server is used to control access to the internet and block any unsuitable websites. It accepts requests via HTTP or its secure variant HTTPS.
- **Print Server**: A print server is used to deal with all of the printing requests from the client(s). It connects printers to client computers over a network.

7. TCP Concurrent Server

Ans :

Implement TCP client and server (concurrent) where the client gets input from the user and sends it to the server. The server displays it on the screen. The server then gets another input from the user and sends it to the client. The client displays it on the screen. The process continues till server or client sends "bye" to the other party.

8. What is UDP

Ans :

Meaning

User Datagram Protocol (UDP) is a Transport Layer protocol. UDP is a part of the Internet Protocol suite, referred to as UDP/IP suite. Unlike TCP, it is an unreliable and connectionless protocol. So, there is no need to establish a connection prior to data transfer. The UDP helps to establish low-latency and loss-tolerating connections establish over the network.The UDP enables process to process communication.

Though Transmission Control Protocol (TCP) is the dominant transport layer protocol used with most of the Internet services; provides assured delivery, reliability, and much more but all these services cost us additional overhead and latency. Here, UDP comes into the picture. For real-time services like computer gaming, voice or video communication, live conferences; we need UDP. Since high performance is needed, UDP permits packets to be dropped instead of processing delayed packets. There is no error checking in UDP, so it also saves bandwidth.

9. IPv4 Socket Options

Ans :

Here are some common socket options related to IPv4:

1. IP_TTL: This option sets the time-to-live (TTL) field in IP packets. This determines how many hops a packet can take before it is discarded.
2. IP_HDRINCL: This option allows the application to specify its own IP header instead of having the system generate one. This can be used for creating custom IP packets.
3. IP_OPTIONS: This option allows the application to set custom IP options in the IP header, such as the record route option or the source route option.
4. IP_MULTICAST_TTL: This option sets the TTL for multicast packets sent by the socket. This determines how many hops the packets can take before they are discarded.
5. IP_MULTICAST_LOOP: This option enables or disables loopback of multicast packets on the local machine. If loopback is enabled, multicast packets sent by the socket will be received by the same socket on the local machine.
6. IP_ADD_MEMBERSHIP/IP_DROP_MEMBERSHIP: These options allow a socket to join or leave a multicast group.

10. What is I/O Multiplexing

Ans :

When the TCP client is handling two inputs at the same time: standard input and a TCP socket, we encountered a problem when the client was blocked in a call to fgets (on standard input) and the server process was killed. The server TCP correctly sent a FIN to the client TCP, but since the client process was blocked reading from standard input, it never saw the EOF until it read from the socket (possibly much later).

We want to be notified if one or more I/O conditions are ready (i.e., input is ready to be read, or the descriptor is capable of taking more output). This capability is called I/O multiplexing and is provided by the select and poll functions, as well as a newer POSIX variation of the former, called pselect.

I/O multiplexing is typically used in networking applications in the following scenarios:

- When a client is handling multiple descriptors (normally interactive input and a network socket)
- When a client to handle multiple sockets at the same time (this is possible, but rare)
- If a TCP server handles both a listening socket and its connected sockets
- If a server handles both TCP and UDP
- If a server handles multiple services and perhaps multiple protocols

Choose the Correct Answer

1. Which system call is used to create a new socket in TCP? [a]

   (a) socket() (b) connect()

   (c) bind() (d) listen()

2. Which system call is used to connect to a remote server in TCP? [b]

   (a) socket() (b) connect()

   (c) bind() (d) listen()

3. Which system call is used to accept a new incoming connection in TCP? [d]

   (a) socket() (b) connect()

   (c) bind() (d) accept()

4. Which methods are commonly used in Server Socket class? [b]

   (a) Public Output Stream get Output Stream ()

   (b) Public Socket accept ()

   (c) Public synchronized void close ()

   (d) Public void connect ()

5. The client in socket programming must know which information? [c]

   (a) IP address of Server (b) Port number

   (c) Both IP address of Server & Port number (d) Only its own IP address

6. Which system call is used to close a socket? [a]

   (a) close( ) (b) shutdown( )

   (c) exit ( ) (d) terminate( )

7. Which header file is required for TCP socket programming in C? [b]

   (a) <stdio.h> (b) <sys/socket.h>

   (c) <stdlib.h> (d) <string.h>

8. Which function is used to convert the host byte order to network byte order? [a]

   (a) htons( ) (b) ntohs()

   (c) htonl() (d) ntohl()

9. Which socket option is used to enable/disable the reuse of local addresses? [c]

   (a) SO_SNDBUF (b) SO_RCVBUF

   (c) SO_REUSEADDR (d) SO_BACKLOG

10. Which socket option is used to set the timeout value for socket operations? [a]

    (a) SO_TIMEOUT (b) TCP_KEEPALIVE

    (c) SO_REUSEADDR (d) SO_BACKLOG

Fill in the blanks

1. In most of the Internet-based applications ______ address family is used
2. ______ system call is used to bind a socket to a specific IP address and port number?
3. ______ system call is used to set socket options in TCP?
4. ______ and ______ system callaare used to send data over a TCP connection?
5. The ______ is a socket descriptor that was returned by the socket system call
6. Transport Layer receives the data from the upper layer and converts it into smaller parts known as ______
7. A ______ address is a combination of logical address and physical address
8. ______ function is used to convert the network byte order to host byte order?
9. ______ function is used to send data over a UDP socket?
10. ______ socket option is used to set the size of the receive buffer

ANSWERS

1. AF_INET
2. bind()
3. setsockopt()
4. send() and write()
5. sockfd
6. segments
7. socket
8. ntohs()
9. sendto()
10. SO_RCVBUF

UNIT V

Application Layer: Domain Name System, Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP), Hyper Text Transfer Protocol (HTTP)

5.1 Application Layer

Q1. Define Application Layer? What are the Services Provided by the Application Layer?

Ans : **(Imp.)**

Meaning

The application layer is the topmost layer of the OSI model and the TCP/IP model. In TCP/IP model, the application layer is formed by combining the top three layers, i.e., the application layer, the presentation layer, and the session layer. An application layer is an abstraction layer that specifies the shared communications protocols and interface methods used by hosts in a communications network. It is the layer closest to the end-user, implying that the application layer and the end-user can interact directly with the software application.

The application provides the following services.

1. The application layer guarantees that the receiver is recognized, accessible, and ready to receive data from the sender.
2. It enables authentication between devices for an extra layer of network security.
3. It determines the protocol and data syntax rules at the application level.
4. The protocols of the application layer also define the basic syntax of the message being forwarded or retrieved.
5. It also checks whether the sender's computer has the necessary communication interfaces, such as an Ethernet or Wi-Fi interface.
6. Finally, the data on the receiving end is presented to the user application.

Functions of the Application Layer

The application layer provides the following functions.

1. The Application Layer provides protocols that allow the software to communicate and receive data and finally present it to users in a meaningful way.
2. This layer allows users to log on as a remote host.
3. The Application Layer provides various facilities for users to forward multiple emails and a storage facility.
4. This layer acts as a window via which users and application processes can access network resources.
5. This layer provides services such as email, file transfer, results distribution, directory services, network resources, etc.
6. The application layer communicates with the operating system and guarantees that data is properly saved.
7. This layer allows users to interact with other software applications.
8. This application layer generally performs host initialization followed by remote login to hosts.
9. This layer visualizes data, allowing individuals to grasp it rather than memorize it or see it in binary format (1s or 0s).

Application Layer Protocols

The various protocols used in the application layer make the communication between the sender and receiver faster, efficient, reliable, and safe. These protocols are discussed below.

1. **HTTP**

Hypertext transfer protocol enables us to access data via the internet. It sends data in plain text, audio, and video formats. Client and servers exchange resources over the internet using the HTTP protocol. Client devices request servers for the resources required to load a web page, and the servers respond by sending responses to the client.

2. **SMTP**

The SMTP (Simple Mail Transfer Protocol) is the TCP/IP protocol that handles email. The data is sent to another email address using this protocol. SMTP uses a procedure known as "store and forward" to transmit user emails on and across networks. It works with the Mail Transfer Agent to ensure that your message is sent to the correct computer and email mailbox. The port number for SMTP is 25.

3. **TELNET**

It's a tool that allows you to save and manage files over the internet. This protocol can be used to set up equipment for the first time, such as switches. The Telnet protoco is used by telnet command to communicate with a remote device or system. The port number for Telnet is 23.

4. **FTP**

The FTP (File Transfer Protocol) is a standard internet protocol for transferring data from one computer to another. FTP uses TCP to transmit data because TCP provides reliability and error-free data transmission. It facilitates file sharing via remote computer devices while ensuring dependable and efficient data delivery. For data control, FTP utilizes port 21, and for data access, it uses port 20.

5. **TFTP**

The TFTP (Trivial File Transfer Protocol) is a simple file transfer protocol. The TFTP uses User Datagram Protocol (UDP) to transmit data from one end to the other. The most common usage of TFTP is to read and write files/mail to and from a distant server. Compared to File Transfer Protocol, Trivial File Transfer Protocol is straightforward in design and has limited functionalities. While transmitting files, TFTP does not provide any authentication or security. As a result, it's commonly used to transfer boot files or configuration information between workstations in a local setup. Users in a computer network rarely use it due to its simplistic architecture. It is also unsafe to use the internet due to its lack of security. The TFTP port number is 69.

6. **NFS**

It's known as a NFS (network file system). It enables remote computers to mount file systems over a network and interact with them as if they were mounted locally. System administrators can combine resources on the network's centralized servers as a result of this. Port number for the NFS is 2049.

7. **DNS**

DNS stand for (Domain Name System). Similar to how a phone's contacts list matches names to numbers, the domain name system is a naming database that locates and translates internet domain names to their unique IP addresses. DNS was created because it is more difficult for humans to recall numerical numbers than alphabetic names. DNS is used in a variety of internet activities to swiftly discover an IP address to connect to and access content.

8. **DHCP**

The Dynamic Host Configuration Protocol is a network management protocol that dynamically allocates a unique IP address to any device or node on a network so that they can communicate using IP. DHCP is used to automate and maintain these setups from a central location. There is no need to manually assign IP addresses to new devices. As a result, connecting to a DHCP-based network requires no user configuration.

Examples of Application Layer Protocols

Some of the examples of the application layer protocols are

- X.400 Message Handling Service Protocol facilitates email transfer among compatible systems.
- Simple Network Management Protocol (SNTP) provides remote host management.
- Bitcoin is for digital currency.
- Hypertext Transfer Protocol(HTTP) for the message or text communications.
- H.323 for voice-over IP. It is a packet-based communication.
- A mail server uses POP (Post Office Protocol) protocol in conjunction with SMTP to accept and store mail for hosts.

5.2 Domain Name System

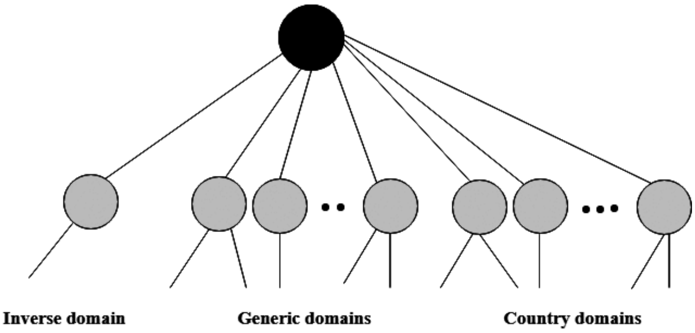
Q2. Explain briefly about DNS.

Ans : **(Imp.)**

An application layer protocol defines how the application processes running on different systems, pass the messages to each other.

- DNS stands for Domain Name System.
- DNS is a directory service that provides a mapping between the name of a host on the network and its numerical address.
- DNS is required for the functioning of the internet.
- Each node in a tree has a domain name, and a full domain name is a sequence of symbols specified by dots.
- DNS is a service that translates the domain name into IP addresses. This allows the users of networks to utilize user-friendly names when looking for other hosts instead of remembering the IP addresses.
- For example, suppose the FTP site at EduSoft had an IP address of 132.147.165.50, most people would reach this site by specifying ftp.EduSoft.com. Therefore, the domain name is more reliable than IP address.

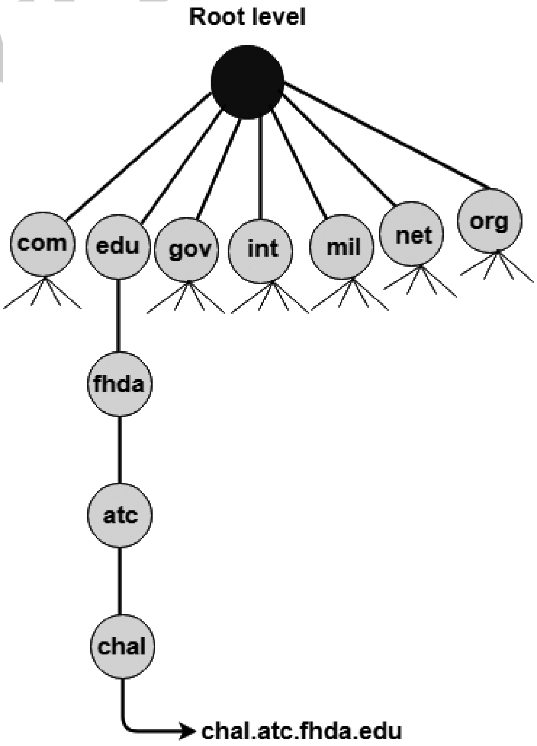
DNS is a TCP/IP protocol used on different platforms. The domain name space is divided into three different sections: generic domains, country domains, and inverse domain.



Generic Domains

- It defines the registered hosts according to their generic behavior.
- Each node in a tree defines the domain name, which is an index to the DNS database.
- It uses three-character labels, and these labels describe the organization type.

Label	Description
aero	Airlines and aerospace companies
biz	Businesses or firms
com	Commercial Organizations
coop	Cooperative business Organizations
edu	Educational institutions
gov	Government institutions
info	Information service providers
int	International Organizations
mil	Military groups
museum	Museum & other nonprofit organizations
name	Personal names
net	Network Support centers
org	Nonprofit Organizations
pro	Professional individual Organizations



Country Domain

The format of country domain is same as a generic domain, but it uses two-character country abbreviations (e.g., us for the United States) in place of three character organizational abbreviations.

Inverse Domain

The inverse domain is used for mapping an address to a name. When the server has received a request from the client, and the server contains the files of only authorized clients. To determine whether the client is on the authorized list or not, it sends a query to the DNS server and ask for mapping an address to the name.

Working of DNS

- DNS is a client/server network communication protocol. DNS clients send requests to the server while DNS servers send responses to the client.
- Client requests contain a name which is converted into an IP address known as a forward DNS lookups while requests containing an IP address which is converted into a name known as reverse DNS lookups.
- DNS implements a distributed database to store the name of all the hosts available on the internet.
- If a client like a web browser sends a request containing a hostname, then a piece of software such as DNS resolver sends a request to the DNS server to obtain the IP address of a hostname. If DNS server does not contain the IP address associated with a hostname, then it forwards the request to another DNS server. If IP address has arrived at the resolver, which in turn completes the request over the internet protocol.

5.3 Simple Mail Transfer Protocol (SMTP)

Q3. Explain briefly about SMTP protocol.

Ans : **(Imp.)**

SMTP is an application layer protocol. The client who wants to send the mail opens a TCP connection to the SMTP server and then sends the mail across the connection. The SMTP server is an always-on listening mode. As soon as it listens for a TCP connection from any client, the SMTP process initiates a connection through port 25. After successfully establishing a TCP connection the client process sends the mail instantly.

SMTP Protocol

The SMTP model is of two types:

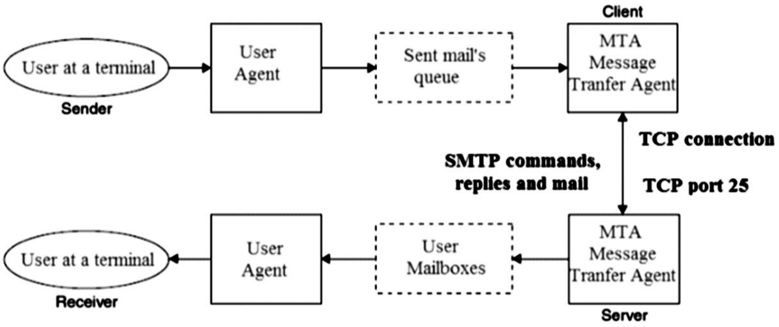
1. End-to-end method
2. Store-and- forward method

The end-to-end model is used to communicate between different organizations whereas the store and forward method is used within an organization. An SMTP client who wants to send the mail will contact the destination's host SMTP directly, in order to send the mail to the destination. The SMTP server will keep the mail to itself until it is successfully copied to the receiver's SMTP.

The client SMTP is the one that initiates the session so let us call it client- SMTP and the server SMTP is the one that responds to the session request so let us call it receiver-SMTP. The client- SMTP will start the session and the receiver-SMTP will respond to the request.

Model of SMTP system

In the SMTP model user deals with the user agent (UA), for example, Microsoft Outlook, Netscape, Mozilla, etc. In order to exchange the mail using TCP, MTA is used. The user sending the mail doesn't have to deal with MTA as it is the responsibility of the system admin to set up a local MTA. The MTA maintains a small queue of mails so that it can schedule repeat delivery of mails in case the receiver is not available. The MTA delivers the mail to the mailboxes and the information can later be downloaded by the user agents.



Both the SMTP-client and SMTP-server should have 2 components:

1. User-agent (UA)
2. Local MTA

Communication between sender and the receiver :

The sender's user agent prepares the message and sends it to the MTA. The MTA's responsibility is to transfer the mail across the network to the receiver's MTA. To send mails, a system must have a client MTA, and to receive mails, a system must have a server MTA.

SENDING EMAIL:

Mail is sent by a series of request and response messages between the client and the server. The message which is sent across consists of a header and a body. A null line is used to terminate the mail header and everything after the null line is considered as the body of the message, which is a sequence of ASCII characters. The message body contains the actual information read by the receipt.

RECEIVING EMAIL:

The user agent at the server-side checks the mailboxes at a particular time of intervals. If any information is received, it informs the user about the mail. When the user tries to read the mail it displays a list of emails with a short description of each mail in the mailbox. By selecting any of the mail users can view its contents on the terminal.

Some SMTP Commands:

- HELO – Identifies the client to the server, fully qualified domain name, only sent once per session
- MAIL – Initiate a message transfer, fully qualified domain of originator
- RCPT – Follows MAIL, identifies an addressee, typically the fully qualified name of the addressee, and for multiple addressees use one RCPT for each addressee
- DATA – send data line by line

Advantages of SMTP:

- If necessary, the users can have a dedicated server.
- It allows for bulk mailing.
- Low cost and wide coverage area.
- Offer choices for email tracking.
- reliable and prompt email delivery.

Disadvantages of SMTP:

- SMTP's common port can be blocked by several firewalls.
- SMTP security is a bigger problem.
- Its simplicity restricts how useful it can be.
- Just 7 bit ASCII characters can be used.
- If a message is longer than a certain length, SMTP servers may reject the entire message.
- Delivering your message will typically involve additional back-and-forth processing between servers, which will delay sending and raise the likelihood that it won't be sent.

5.4 File Transfer Protocol (FTP)

Q4. Explain briefly about FTP Protocol.

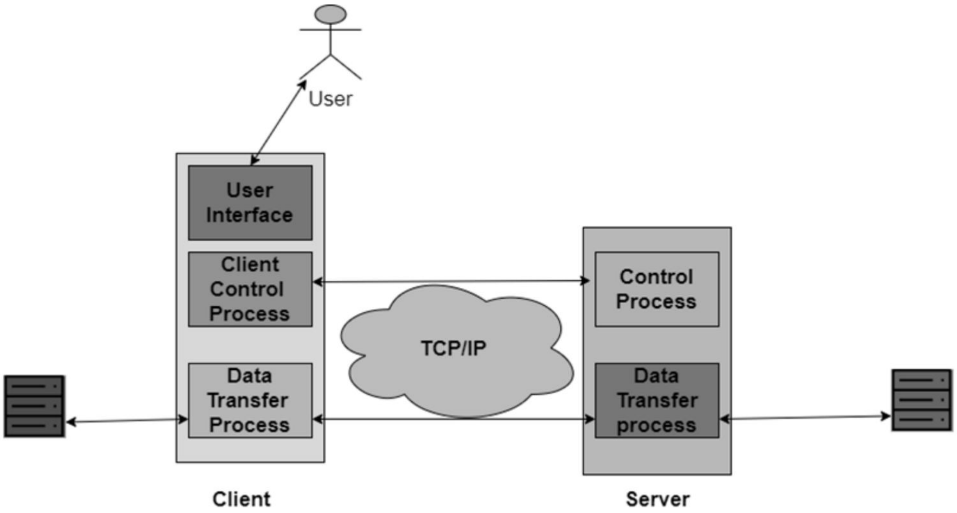
Ans : **(Imp.)**

FTP means File Transfer Protocol and it is the standard mechanism provided by the TCP/IP in order to copy a file from one host to another.

- File Transfer Protocol is a protocol present at the Application layer of the OSI Model.
- FTP is one of the easier, simpler, and secure ways to exchange files over the Internet.
- FTP is different from the other client/server applications as this protocol establishes two connections between the hosts.
  - o where one connection is used for the data transfer and is known as a data connection.
  - o while the other connection is used to control information like commands and responses and this connection is termed as control connection.
- FTP is more efficient as there is the separation of commands.
- The File Transfer Protocol makes the use of two protocols; Port 21 for the Control connection and Port 20 is used for Data connection.
- The control connection in FTP makes the use of very simple rules of communication, we just need to transfer a line of command or a line of response at a time.
- On the other hand, the data connection needs more complex rules; and the reason behind this is there are a variety of types of data that needs to be transferred.
- The transferring of files from the client computer to the server is termed as "uploading", while the transferring of data from the server to the client computer is termed as "downloading".
- The types of files transferred using the FTP are ASCII files, EBCDIC files, or image files.

Working of FTP

Given below figure shows the basic model of file Transfer Protocol, where the client comprises of three components: User Interface, Client control process, and client data transfer process. On the other hand, the server comprises of two components mainly the server control process and the server data transfer process.



1. Also, the control connection is made between the control processes while the data connection is made between the data transfer processes.

2. The control Connection remains connected during the entire interactive session of FTP while the data connection is opened and then closed for each file transferred.

3. In simple terms when a user starts the FTP connection then the control connection opens, while it is open the data connection can be opened and closed multiple times if several files need to be transferred.

Given below are three data structures supported by FTP:

1. **File Structure** In the File data structure, the file is basically a continuous stream of bytes.
2. **Record Structure** In the Record data structure, the file is simply divided into the form of records.
3. **Page Structure** In the Page data structure, the file is divided into pages where each page has a page number and a page header. These pages can be stored and accessed either randomly or sequentially.

FTP Clients

It is basically software that is designed to transfer the files back-and-forth between a computer and a server over the Internet. The FTP client needs to be installed on your computer and can only be used with the live connection to the Internet.

Some of the commonly used FTP clients are Dreamweaver, FireFTP, and Filezilla.

Q5. State the Features of FTP.

Ans :

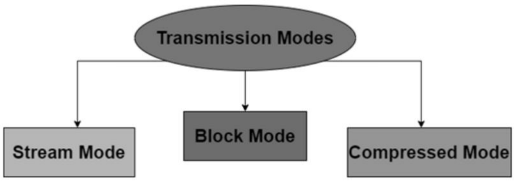
Following are the features offered by the File transfer protocol:

- FTP is mainly used to transfer one file at a time.
- Other actions performed by FTP are listing files, creating and deleting directories, deleting files, renaming files, and many more.
- FTP also hides the details of individual computer systems.
- FTP allows those files that have ownership and access restrictions.
- It is a connection-oriented protocol.
- FTP is a stateful protocol as in this the client establishes a control connection for the duration of an FTP session that typically spans multiple data transfers.

Q6, Explain the Transmission Modes of FTP.

Ans :

FTP can transfer a file across the data connection using one of the three given modes:



1. **Stream Mode**

Stream Mode is the default mode of transmission used by FTP. In this mode, the File is transmitted as a continuous stream of bytes to TCP.

If the data is simply in the form of the stream of bytes then there is no need for End-of-File, Closing of data connection by the sender is considered as EOF or end-of-file. If the data is divided into records (that is the record structure), each record has an I-byte of EOR(end-of-record).

2. **Block Mode**

Block mode is used to deliver the data from FTP to TCP in the form of blocks of data. Each block of data is preceded by 3 bytes of the header where the first byte represents the block descriptor while the second and third byte represents the size of the block.

3. **Compressed Mode**

In this mode, if the file to be transmitted is very big then the data can be compressed. This method is normally used in Run-length encoding. In the case of a text file, usually, spaces/blanks are removed. While in the case of the binary file, null characters are compressed.

Q7. State the Advantages and Disadvantages of FTP.

Ans :

Advantages of FTP.

Following are some of the benefits of using File Transfer protocol:

- Implementation of FTP is simple.
- FTP provides one of the fastest ways to transfer files from one computer to another.
- FTP is a standardized protocol and is widely used.
- File Transfer protocol is more efficient as there is no need to complete all the operations in order to get the entire file,

Disadvantages of FTP

Let us take a look at the drawbacks of FTP:

- File Transfer Protocol is not a secure way to transfer the data.
- FTP does not allow the copy from server to server and also not allows removal operations for the recursive directory.
- Scripting the jobs is hard using the FTP protocol.
- The spoofing of the server can be done in order to send data to a random unknown port on any unauthorized computer

5.5 Hyper Text Transfer Protocol (HTTP)

Q8. Explain briefly about HTTP protocol.

Ans :

Meaning

HTTP stands for Hypertext Transfer Protocol and is mainly used to access the data on the world wide web i.e (WWW). The HTTP mainly functions as the combination of FTP(File Transfer Protocol) and SMTP(Simple Mail Transfer Protocol).

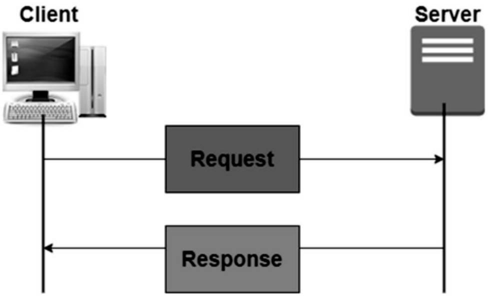
- HTTP is one of the protocols used at the Application Layer.
- The HTTP is similar to FTP because HTTP is used to transfer the files and it mainly uses the services of TCP.
- Also, HTTP is much simpler than FTP because there is only one TCP connection.
- In HTTP, there is no separate control connection, as only data is transferred between the client and the server.
- The HTTP is like SMTP because the transfer of data between the client and server simply looks like SMTP messages. But there is a difference unlike SMTP, the HTTP messages are not destined to be read by humans as they are read and interpreted by HTPP Client(that is browser) and HTTP server.
- Also, SMTP messages are stored and then forwarded while the HTTP messages are delivered immediately.
- The HTTP mainly uses the services of the TCP on the well-known port that is port 80.
- HTTP is a stateless protocol.

- In HTTP, the client initializes the transaction by sending a request message, and the server replies by sending a response.
- This protocol is used to transfer the data in the form of plain text, hypertext, audio as well as video, and so on.

Q9. Explain the mechanism of HTTP.

Ans :

The HTTP makes use of Client-server architecture. As we have already told you that the browser acts as the HTTP client and this client mainly communicates with the webserver that is hosting the website.



The Figure shows the HTTP Transaction

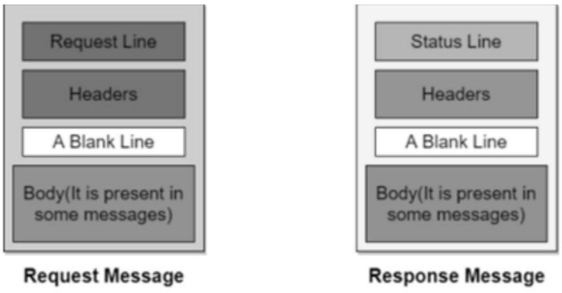
The format of the request and the response message is similar. The Request Message mainly consists of a request line, a header, and a body sometimes. A Response message consists of the status line, a header, and sometimes a body.

At the time when a client makes a request for some information (say client clicks on the hyperlink) to the webserver. The browser then sends a request message to the HTTP server for the requested objects.

After that the following things happen:

- There is a connection that becomes open between the client and the webserver through the TCP.
- After that, the HTTP sends a request to the server that mainly collects the requested data.
- The response with the objects is sent back to the client by HTTP
- At last, HTTP closes the connection.

Let us take a look at the format of the request message and response message:



Request Line and Status Line

The first line in the Request message is known as the request line, while the first line in the Response message is known as the Status line.

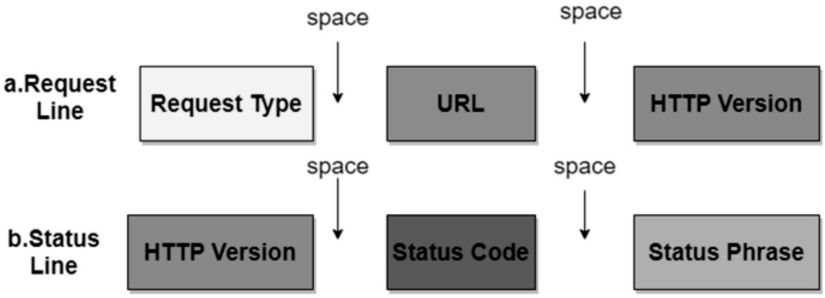


Fig. : Request Line and Status Line

Request Type

This field is used in the request line. The are several request types that are defined and these are mentioned in the table given below;

Name of Method	Actions
GET	This method is used to request a document from the server.
HEAD	This method mainly requests information about a document and not the document itself
POST	This method sends some information from the client to the server.
PUT	This method sends a document from the server to the client.
TRACE	This method echoes the incoming request.
CONNECT	This method means reserved
OPTION	In order to inquire about the available options.

URL

URL is a Uniform Resource locator and it is mainly a standard way of specifying any kind of information on the Internet.

Status Code

The status code is the field of the response message.The status code consists of three digits.

Status Phrase

This field is also used in the response message and it is used to explain the status code in the form of text.

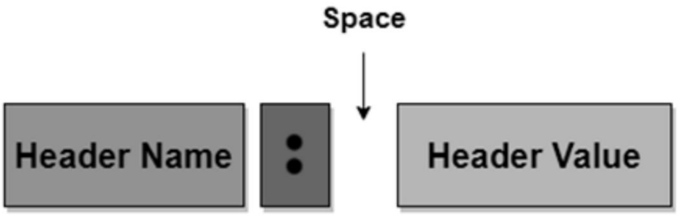
Header

The header is used to exchange the additional information between the client and the server. The header mainly consists of one or more header lines. Each header line has a header name, a colon, space, and a header value.

The header line is further categorized into four:

- **General Header** It provides general information about the message and it can be present in both request and response.

- **Request Header** It is only present in the request message and is used to specify the configuration of the client and the format of the document preferred by the client
- **Response Header** This header is only present in the response header and mainly specifies the configuration of the server and also the special information about the request.
- **Entity Header** It is used to provide information about the body of the document.



Body

It can be present in the request message or in the response message. The body part mainly contains the document to be sent or received.

Q10. State the Features of HTTP.

Ans :

The HTTP offers various features and these are as follows:

1. **HTTP is simple:** The HTTP protocol is designed to be plain and human-readable.
2. **HTTP is stateless:** Hypertext transfer protocol(HTTP) is a stateless protocol, which simply means that there is no connection among two requests that are being consecutively carried out on the same connection. Also, both the client and the server know each other only during the current requests and thus the core of the HTTP is itself a stateless one, On the other hand, the HTTP cookies provide in making use of stateful sessions.
3. **HTTP is extensible:** The HTTP can be integrated easily with the new functionality by providing a simple agreement between the client and the server.
4. **HTTP is connectionless:** As the HTTP request is initiated by the browser (HTTP client) and as per the request information by the user, after that the server processes the request of the client and then responds back to the client

Q11. Explain the Advantages and Disadvantages of HTTP.

Ans :

Advantages of HTTP

1. There is no runtime support required to run properly.
2. As it is connectionless so there is no overhead in order to create and maintain the state and information of the session.
3. HTTP is usable over the firewalls and global application is possible.
4. HTTP is platform-independent.
1. HTTP reports the errors without closing the TCP connection.
2. Offers Reduced Network congestions.

Disadvanatges of HTTP

- HTTP is not optimized for mobile.
- HTTP is too verbose.
- It can be only used for point-to-point connections.
- This protocol does not have push capabilities.
- This protocol does not offer reliable exchange without the retry logic.

HTTP Connections

HTTP connections can be further classified into two:

1. Persistent Connection

In the persistent HTTP connection, all the requests and their corresponding responses are sent over the same TCP connections. The 1.1 version of the HTTP specifies a persistent connection by default.

In this type of connection, the server leaves the connection open for more requests after sending a response. Also, the server can close the connection at the request of the client or upon reaching the time-out.

In a Persistent connection, a single TCP connection is mainly used for sending multiple objects one after the other.

Usually, the length of the data is sent along with each response. There are some cases when the server does not know the length of the data this happens when the document is created dynamically and in such cases, the server informs the client that length is not known and closes the connection after sending the data so in order let the client Inform about the end of the data.

2. Nonpersistent Connection

In the Nonpersistent HTTP connection, one TCP connection is made for each request/response; it means there is a separate for each object.

Following are the steps used;

- The client opens a TCP connection and then sends a request.
- After that, the server sends the response and then closes the connection.
- Then the client reads the data and until it encounters an end-of-file marker then it closes the connection.

This connection imposes a high overhead on the server because N different buffers are required by the server, and the start procedure is slow each time when a connection is opened.

The nonpersistent connection is supported by the HTTP 1.0 version.

Short Question and Answers

1. Application Layer

Ans :

Meaning

The application layer is the topmost layer of the OSI model and the TCP/IP model. In TCP/IP model, the application layer is formed by combining the top three layers, i.e., the application layer, the presentation layer, and the session layer. An application layer is an abstraction layer that specifies the shared communications protocols and interface methods used by hosts in a communications network. It is the layer closest to the end-user, implying that the application layer and the end-user can interact directly with the software application.

2. Functions of the Application Layer

Ans :

The application layer provides the following functions.

1. The Application Layer provides protocols that allow the software to communicate and receive data and finally present it to users in a meaningful way.
2. This layer allows users to log on as a remote host.
3. The Application Layer provides various facilities for users to forward multiple emails and a storage facility.
4. This layer acts as a window via which users and application processes can access network resources.
5. This layer provides services such as email, file transfer, results distribution, directory services, network resources, etc.

3. Application Layer Protocols

Ans :

1. HTTP

Hypertext transfer protocol enables us to access data via the internet. It sends data in plain text, audio, and video formats. Client and servers exchange resources over the internet using the HTTP protocol. Client devices request servers for the resources required to load a web page, and the servers respond by sending responses to the client.

2. SMTP

The SMTP (Simple Mail Transfer Protocol) is the TCP/IP protocol that handles email. The data is sent to another email address using this protocol. SMTP uses a procedure known as "store and forward" to transmit user emails on and across networks. It works with the Mail Transfer Agent to ensure that your message is sent to the correct computer and email mailbox. The port number for SMTP is 25.

3. TELNET

It's a tool that allows you to save and manage files over the internet. This protocol can be used to set up equipment for the first time, such as switches. The Telnet protoco is used by telnet command to communicate with a remote device or system. The port number for Telnet is 23.

4. FTP

The FTP (File Transfer Protocol) is a standard internet protocol for transferring data from one computer to another. FTP uses TCP to transmit data because TCP provides reliability and error-free data transmission. It facilitates file sharing via remote computer devices while ensuring dependable and efficient data delivery. For data control, FTP utilizes port 21, and for data access, it uses port 20.

5. TFTP

The TFTP (Trivial File Transfer Protocol) is a simple file transfer protocol. The TFTP uses User Datagram Protocol (UDP) to transmit data from one end to the other. The most common usage of TFTP is to read and write files/mail to and from a distant server. Compared to File Transfer Protocol, Trivial File Transfer Protocol is straightforward in design and has limited functionalities. While transmitting files, TFTP does not provide any authentication or security. As a result, it's commonly used to transfer boot files or configuration information between workstations in a local setup. Users in a computer network rarely use it due to its simplistic architecture. It is also unsafe to use the internet due to its lack of security. The TFTP port number is 69.

6. NFS

It's known as a NFS (network file system). It enables remote computers to mount file systems over a network and interact with them as if they were mounted locally. System administrators can combine resources on the network's centralized servers as a result of this. Port number for the NFS is 2049.

4. DNS.

Ans :

An application layer protocol defines how the application processes running on different systems, pass the messages to each other.

- DNS stands for Domain Name System.
- DNS is a directory service that provides a mapping between the name of a host on the network and its numerical address.
- DNS is required for the functioning of the internet.
- Each node in a tree has a domain name, and a full domain name is a sequence of symbols specified by dots.
- DNS is a service that translates the domain name into IP addresses. This allows the users of networks to utilize user-friendly names when looking for other hosts instead of remembering the IP addresses.

5. SMTP protocol.

Ans :

SMTP is an application layer protocol. The client who wants to send the mail opens a TCP connection to the SMTP server and then sends the mail across the connection. The SMTP server is an always-on listening mode. As soon as it listens for a TCP connection from any client, the SMTP process initiates a connection through port 25. After successfully establishing a TCP connection the client process sends the mail instantly.

SMTP Protocol

The SMTP model is of two types:

1. End-to-end method
2. Store-and- forward method

6. FTP Protocol.

Ans :

FTP means File Transfer Protocol and it is the standard mechanism provided by the TCP/IP in order to copy a file from one host to another.

- File Transfer Protocol is a protocol present at the Application layer of the OSI Model.
- FTP is one of the easier, simpler, and secure ways to exchange files over the Internet.
- FTP is different from the other client/server applications as this protocol establishes two connections between the hosts.
  - o where one connection is used for the data transfer and is known as a data connection.
  - o while the other connection is used to control information like commands and responses and this connection is termed as control connection.

7. Disadvantages of FTP

Ans :

Let us take a look at the drawbacks of FTP:

- File Transfer Protocol is not a secure way to transfer the data.
- FTP does not allow the copy from server to server and also not allows removal operations for the recursive directory.
- Scripting the jobs is hard using the FTP protocol.
- The spoofing of the server can be done in order to send data to a random unknown port on any unauthorized computer

8. HTTP protocol.

Ans :

Meaning

HTTP stands for Hypertext Transfer Protocol and is mainly used to access the data on the world wide web i.e (WWW). The HTTP mainly functions as the combination of FTP(File Transfer Protocol) and SMTP(Simple Mail Transfer Protocol).

- HTTP is one of the protocols used at the Application Layer.
- The HTTP is similar to FTP because HTTP is used to transfer the files and it mainly uses the services of TCP.
- Also, HTTP is much simpler than FTP because there is only one TCP connection.
- In HTTP, there is no separate control connection, as only data is transferred between the client and the server.

9. Advantages of HTTP

Ans :

1. There is no runtime support required to run properly.
2. As it is connectionless so there is no overhead in order to create and maintain the state and information of the session.
3. HTTP is usable over the firewalls and global application is possible.
4. HTTP is platform-independent.

1. HTTP reports the errors without closing the TCP connection.
2. Offers Reduced Network congestions.

10. Disadvanatges of HTTP

Ans :

- HTTP is not optimized for mobile.
- HTTP is too verbose.
- It can be only used for point-to-point connections.
- This protocol does not have push capabilities.
- This protocol does not offer reliable exchange without the retry logic.

Choose the Correct Answer

1. Which protocol is used in the application layer for email transmission? [b]

   (a) FTP (b) SMTP

   (c) POP (d) DNS

2. Which protocol is used for file transfer in the application layer? [a]

   (a) FTP (b) SMTP

   (c) HTTP (d) DNS

3. Which application layer protocol is used for remote terminal access? [a]

   (a) SSH (b) SMTP

   (c) HTTP (d) SNMP

4. Which protocol is used for resolving domain names to IP addresses in the application layer? [a]

   (a) DNS (b) SMTP

   (c) FTP (d) HTTP

5. Which application layer protocol is used for web page retrieval? [c]

   (a) SMTP (b) FTP

   (c) HTTP (d) SSH

6. Which protocol is used for real-time data transfer in the application layer? [d]

   (a) FTP (b) SMTP

   (c) HTTP (d) RTP

7. Which status code in HTTP indicates that the requested resource is not available on the server? [c]

   (a) 200 OK (b) 201 Created

   (c) 404 Not Found (d) 500 Internal Server Error

8. Which port is used by DNS for communication between client and server? [b]

   (a) Port 21 (b) Port 53

   (c) Port 80 (d) Port 443

9. Which port does FTP use for communication? [a]

   (a) Port 21 (b) Port 23

   (c) Port 25 (d) Port 53

10. Which method in HTTP is used to retrieve data from a server? [d]

    (a) POST (b) PUT

    (c) DELETE (d) GET

Fill in the blanks

1. The packet of information at the application layer is called ______
2. Application layer offers ______ service.
3. ______ protocol is used for remote network monitoring and management in the application layer.
4. A program or service that converts domain names to IP addresses is called ______
5. ______ is the default mode of FTP data transfer
6. ______ command is used to download a file from the server to the client
7. HTTP stands for______
8. The ______ header in HTTP is used to specify the type of data being sent in the request or response?
9. ______ is a network protocol used for sending email messages between servers.
10. When an email message is sent using SMTP, it is broken down into ______ and then transmitted to the recipient's email server.

Answers

1. Message
2. End to end
3. SNMP
4. DNS Resolver
5. Binary mode
6. Get
7. Hypertext Transfer Protocol
8. Content-Type
9. SMTP
10. Packets

Practical Programs

Networking Concepts Demonstration

1. Demonstrations of IP Address and Ports in Computer System.

Ans :

Public IPv4 addresses

There are two main IPv4 address spaces—the public address space and the private address space. The primary difference between both address spaces is that the public IPv4 addresses are routable on the internet, which means that any device that requires communication to other devices on the internet will need to be assigned a public IPv4 address on its interface, which is connected to the internet.

The public address space is divided into five classes:

Class A	0.0.0.0 – 126.255.255.255
Class B	128.0.0.0 – 191.255.255.255
Class C	192.0.0.0 – 223.255.255.255
Class D	224.0.0.0 – 239.255.255.255
Class E	240.0.0.0 – 255.255.255.255

Private IPv4 Addresses

This means that if a device is directly connected to the internet with a private IPv4 address, there will be no network connectivity to devices on the internet. Most ISPs usually have a filter to prevent any private addresses (RFC 1918) from entering their network.

The private address space is divided into three classes:

Class A—10.0.0.0/8 network block	10.0.0.0 – 010.255.255.255
Class B—172.16.0.0/12 network block	172.16.0.0 – 172.31.255.255
Class C—192.168.0.0/16 network block	192.168.0.0 – 192.168.255.255

Subnetting in IPv4

What is subnetting and why do we need to subnet a network?

First, subnetting is the process of breaking down a single IP address block into smaller subnetworks (subnets). Second, the reason we need to subnet is to efficiently distribute IP addresses with the result of less wastage. This brings us to other questions, such as why do we need to break down a single IP address block, and why is least wastage so important? Could we simply assign a Class A, B, or C address block to a network of any size? To answer these questions, we will go more in depth with this topic by using practical examples and scenarios.

Let's assume that you are a network administrator at a local company and one day the IT manager assigns a new task to you. The task is to redesign the IP scheme of the company. He has also told you to use an address class that is suitable for the company's size and to ensure that there is minimal wastage of IP addresses.

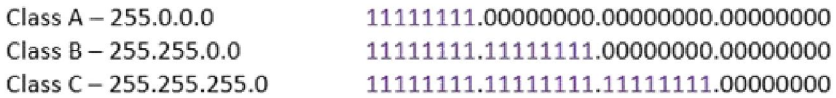
The first thing you decided to do was draw a high-level network diagram indicating each branch, which shows the number of hosts per branch office and the Wide Area Network (WAN) links between each branch router:

Step 1 – determining an appropriate class of address and why

The subnet mask can tell us a lot about a network, such as the following:

- The network and host portion of an IP address
- The number of hosts within a network

If we use a network block from either of the address classes, we will get the following available hosts:



Class A – 255.0.0.0	11111111.00000000.00000000.00000000
Class B – 255.255.0.0	11111111.11111111.00000000.00000000
Class C – 255.255.255.0	11111111.11111111.11111111.00000000

As you may remember, the network portion of an address is represented by 1s in the subnet mask, while the 0s represent the host portion. We can use the following formula to calculate the total number of IP addresses within a subnet by the known the amount of host bits in the subnet mask.

Using the formula $2^{H,}$ where H represents the host bit, we get the following results:

- Class A = 2^{24} = 16,777,216 total IPs
- Class B = 2^{16} = 65,536 total IPs
- Class C = 2^{8} = 256 total IPs

In IPv4, there are two IPs that cannot be assigned to any devices. These are the Network ID and the Broadcast IP address. Therefore, you need to subtract two addresses from the total IP formula.

Using the formula 2^{H}-2 to calculate usable IPs, we get the following:

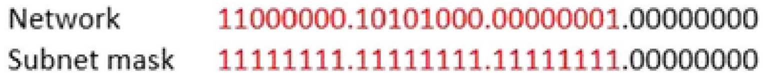
- Class A = 2^{24} – 2 = 16,777,214 total IPs
- Class B = 2^{16} – 2 = 65,534 total IPs
- Class C = 2^{8} – 2 = 254 total IPs

Looking back at Network diagram, we can identify the following seven networks:

- **Branch A LAN**: 25 hosts
- **Branch B LAN**: 15 hosts
- **Branch C LAN**: 28 hosts
- **Branch D LAN**: 26 hosts
- **WAN R1-R2**: 2 IPs are needed
- **WAN R2-R3**: 2 IPs are needed
- **WAN R3-R4**: 2 IPs are needed

Determining the appropriate address class depends on the largest network and the number of networks needed. Currently, the largest network is Branch C, which has 28 host devices that needs an IP address. We can use the smallest available class, which is any Class C address because it will be able to support the largest network we have. However, to do this, we need to choose a Class C address block. Let's use the 192.168.1.0/24 block.

Remember, the subnet mask is used to identify the network portion of the address. This also means that we are unable to modify the network portion of the IP address when we are subnetting, but we can modify the host portion:



Network	11000000.10101000.00000001.00000000
Subnet mask	11111111.11111111.11111111.00000000

The first 24-bits represent the network portion and the remaining 8-bits represent the host portion. Using the formula 2H – 2 to calculate the number of usable host IPs, we get the following:

$2^H - 2$

$2^8 - 2 = 256 - 2 = 254$ usable IP addresses

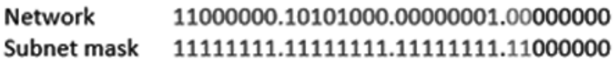
Assigning this single network block to either of the seven networks, there will be a lot of IP addresses being wasted. Therefore, we need to apply our subnetting techniques to this Class C address block.

Step 2 – Creating subnets (subnetworks)

To create more subnets or subnetworks, we need to borrow bits on the host portion of the network. The formula 2N is used to calculate the number of subnets, where N is the number of bits borrowed on the host portion. Once these bits are borrowed, they will become part of the network portion and a new subnet mask will be presented.

So far, we have a Network ID of `192.168.1.0/24`. We need to get seven subnets, and each subnet should be able to fit our largest network (which is Branch C—28 hosts).

Let's create our subnets. Remember that we need to borrow bits on the host portion, starting where the 1s end in the subnet mask. Let's borrow two host bits and apply them to our formula to determine whether we are able to get the seven subnets:



```
Network        11000000.10101000.00000001.00000000
Subnet mask    11111111.11111111.11111111.11000000
```

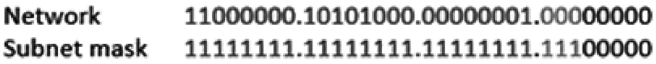
When bits are borrowed on the host portion, the bits are changed to 1s in the subnet mask. This produces a new subnet mask for all of the subnets that have been created.

Let's use our formula for calculating the number of networks:

Number of Networks = 2^N

$2^2 = 2 \times 2 = 4$ networks

As we can see, two host bits are not enough as we need at least seven networks. Let's borrow one more host bit:



```
Network        11000000.10101000.00000001.00000000
Subnet mask    11111111.11111111.11111111.11100000
```

Once again, let's use our formula for calculating the number of networks:

Number of Networks = 2^N

$2^3 = 2 \times 2 \times 2 = 8$ networks

Using 3 host bits, we are able to get a total of 8 subnets. In this situation, we have one additional network, and this additional network can be placed aside for future use if there's an additional branch in the future.

Since we borrowed 3 bits, we have 5 host bits remaining. Let's use our formula for calculating usable IP addresses:

Usable IP addresses = $2^H - 2$

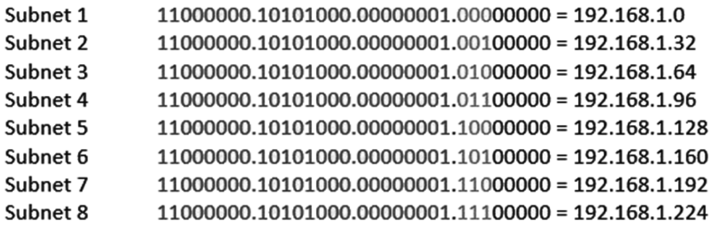
$2^5 - 2 = 32 - 2 = 30$ usable IPs

This means that each of the 8 subnets will have a total of 32 IP addresses, with 30 usable IP addresses inclusive. Now we have a perfect match. Let's work out our 8 new subnets.

The guidelines we must follow at this point are as follows:

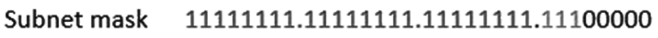
- We cannot modify the network portion of the address (red)
- We cannot modify the host portion of the address (black)
- We can only modify the bits that we borrowed (green)

Starting with the Network ID, we get the following eight subnets:



Subnet 1	11000000.10101000.00000001.00000000 = 192.168.1.0
Subnet 2	11000000.10101000.00000001.00100000 = 192.168.1.32
Subnet 3	11000000.10101000.00000001.01000000 = 192.168.1.64
Subnet 4	11000000.10101000.00000001.01100000 = 192.168.1.96
Subnet 5	11000000.10101000.00000001.10000000 = 192.168.1.128
Subnet 6	11000000.10101000.00000001.10100000 = 192.168.1.160
Subnet 7	11000000.10101000.00000001.11000000 = 192.168.1.192
Subnet 8	11000000.10101000.00000001.11100000 = 192.168.1.224

We can't forget about the subnet mask:



Subnet mask 11111111.1111111.11111111.11100000

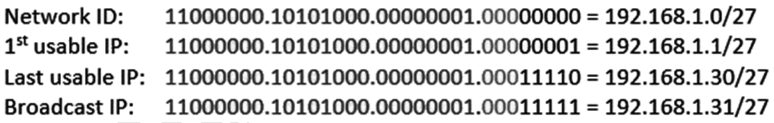
As we can see, there are twenty-seven 1s in the subnet mask, which gives us 255.255.255.224 or /27 as the new subnet mask for all eight subnets we've just created.

Take a look at each of the subnets. They all have a fixed increment of 32. A quick method to calculate the incremental size is to use the formula 2^x. This assists in working out the decimal notation of each subnet much easier than calculating the binary. The last network in any subnet always ends with the customized ending of the new subnet mask. From our example, the new subnet mask 255.255.255.224 ends with 224, and the last subnet also ends with the same value, 192.168.1.224.

Step 3 – Assigning each network an appropriate subnet and calculating the ranges

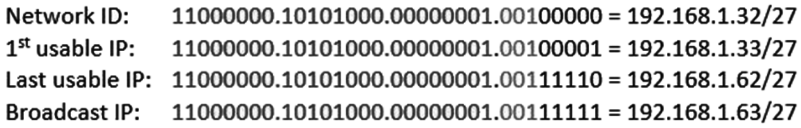
To determine the first usable IP address within a subnet, the first bit from the right must be 1. To determine the last usable IP address within a subnet all of the host bits except the first bit from the right should all be 1s. The broadcast IP of any subnet is when all of the host bits are 1s.

Let's take a look at the first subnet. We will assign subnet 1 to the Branch A LAN:



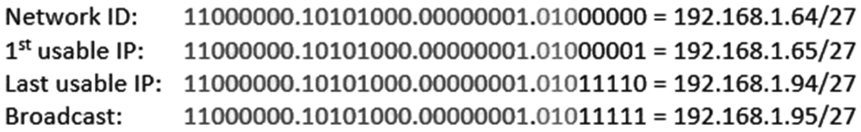
Network ID:	11000000.10101000.00000001.00000000 = 192.168.1.0/27
1st usable IP:	11000000.10101000.00000001.00000001 = 192.168.1.1/27
Last usable IP:	11000000.10101000.00000001.00011110 = 192.168.1.30/27
Broadcast IP:	11000000.10101000.00000001.00011111 = 192.168.1.31/27

The second subnet will be allocated to the Branch B LAN:



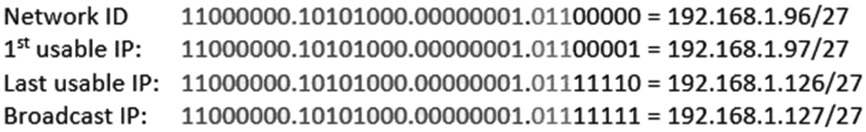
Network ID:	11000000.10101000.00000001.00100000 = 192.168.1.32/27
1st usable IP:	11000000.10101000.00000001.00100001 = 192.168.1.33/27
Last usable IP:	11000000.10101000.00000001.00111110 = 192.168.1.62/27
Broadcast IP:	11000000.10101000.00000001.00111111 = 192.168.1.63/27

The third subnet will be allocated to the Branch C LAN:



Network ID:	11000000.10101000.00000001.01000000 = 192.168.1.64/27
1st usable IP:	11000000.10101000.00000001.01000001 = 192.168.1.65/27
Last usable IP:	11000000.10101000.00000001.01011110 = 192.168.1.94/27
Broadcast:	11000000.10101000.00000001.01011111 = 192.168.1.95/27

The fourth subnet will be allocated to Branch D LAN:

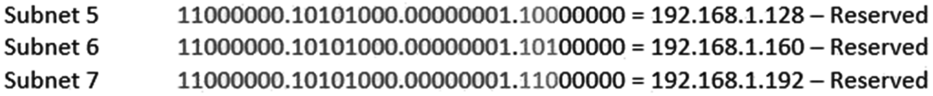


Network ID	11000000.10101000.00000001.01100000 = 192.168.1.96/27
1st usable IP:	11000000.10101000.00000001.01100001 = 192.168.1.97/27
Last usable IP:	11000000.10101000.00000001.01111110 = 192.168.1.126/27
Broadcast IP:	11000000.10101000.00000001.01111111 = 192.168.1.127/27

At this point, we have successfully allocated subnets 1 to 4 to each of the branch's LANs. During our initial calculation for determining the size of each subnet, we saw that each of the eight subnets are equal, and that we have 32 total IPs with 30 usable IP addresses. Currently, we have subnets 5 to 8 for allocation, but if we allocate subnet 5, 6 and 7 to the WAN links between the branches R1-R2, R2-R3 and R3-R4, we would be wasting 28 IP addresses since each WAN link (point-to-point) only requires 2 IP addresses.

What if we can take one of our existing subnets and create even more but smaller networks to fit each WAN (point-to-point) link? We can do this with a process known as Variable Length Subnet Masking (VLSM). By using this process, we are subnetting a subnet.

For now, we will place aside subnets 5, 6, and 7 as a future reservation for any future branches:



Subnet 5	11000000.10101000.00000001.10000000 = 192.168.1.128 – Reserved
Subnet 6	11000000.10101000.00000001.10100000 = 192.168.1.160 – Reserved
Subnet 7	11000000.10101000.00000001.11000000 = 192.168.1.192 – Reserved

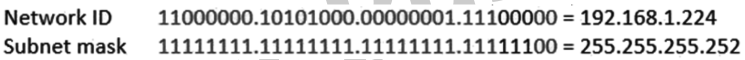
Step 4 – VLSM and Subnetting a subnet

For the WAN links, we need at least three subnets. Each must have a minimum of two usable IP addresses. To get started, let's use the following formula to determine the number of host bits that are needed so that we have at least two usable IP addresses: 2H – 2, where H is the number of host bits.

We are going to use one bit, $2^1 - 2 = 2 - 2 = 0$ usable IP addresses. Let's add an extra host bit in our formula, that is, $2^2 - 2 = 4 - 2 = 2$ usable IP addresses. At this point, we have a perfect match, and we know that only two host bits are needed to give us our WAN (point-to-point) links.

We are going to use the following guidelines:

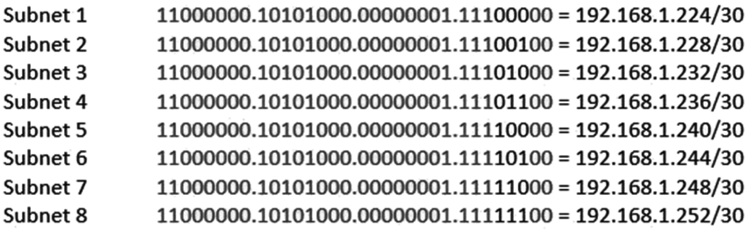
- We cannot modify the network portion of the address (red)
- Since we know that the two host bits are needed to represent two usable IP addresses, we can lock it into place (purple)
- The bit between the network portion (red) and the locked-in host bits (purple) will be the new network bits (black)



Network ID	11000000.10101000.00000001.11100000 = 192.168.1.224
Subnet mask	11111111.11111111.11111111.11111100 = 255.255.255.252

- To calculate the number of networks, we can use $2^N = 2^3 = 8$ networks. Even though we got a lot more networks than we actually needed, the remainder of the networks can be set aside for future use.
- To calculate the total IPs and increment, we can use $2^H = 2^2 = 4$ total IP addresses (inclusive of the Network ID and Broadcast IP addresses).
- To calculate the number of usable IP addresses, we can use $2^H - 2 = 2^2 - 2 = 2$ usable IP addresses per network.

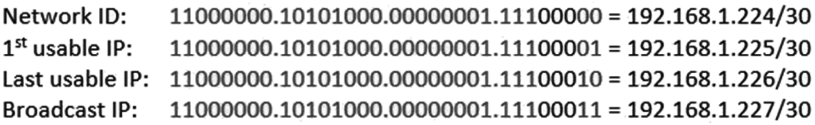
Let's work out our eight new subnets for any existing and future WAN (point-to-point) links:



Subnet 1	11000000.10101000.00000001.11100000 = 192.168.1.224/30
Subnet 2	11000000.10101000.00000001.11100100 = 192.168.1.228/30
Subnet 3	11000000.10101000.00000001.11101000 = 192.168.1.232/30
Subnet 4	11000000.10101000.00000001.11101100 = 192.168.1.236/30
Subnet 5	11000000.10101000.00000001.11110000 = 192.168.1.240/30
Subnet 6	11000000.10101000.00000001.11110100 = 192.168.1.244/30
Subnet 7	11000000.10101000.00000001.11111000 = 192.168.1.248/30
Subnet 8	11000000.10101000.00000001.11111100 = 192.168.1.252/30

Now that we have eight new subnets, let's allocate them accordingly.

The first subnet will be allocated to WAN 1, R1-R2:



Network ID:	11000000.10101000.00000001.11100000 = 192.168.1.224/30
1st usable IP:	11000000.10101000.00000001.11100001 = 192.168.1.225/30
Last usable IP:	11000000.10101000.00000001.11100010 = 192.168.1.226/30
Broadcast IP:	11000000.10101000.00000001.11100011 = 192.168.1.227/30

The second subnet will be allocated to WAN 2, R2-R3:

Network ID:	11000000.10101000.00000001.11100100 = 192.168.1.228/30
1st usable IP:	11000000.10101000.00000001.11100101 = 192.168.1.229/30
Last usable IP:	11000000.10101000.00000001.11100110 = 192.168.1.230/30
Broadcast IP:	11000000.10101000.00000001.11100111 = 192.168.1.231/30

The third subnet will be allocated to WAN 3, R3-R4:

Network ID:	11000000.10101000.00000001.11101000 = 192.168.1.232/30
1st usable IP:	11000000.10101000.00000001.11101001 = 192.168.1.233/30
Last usable IP:	11000000.10101000.00000001.11101010 = 192.168.1.234/30
Broadcast IP:	11000000.10101000.00000001.11101011 = 192.168.1.235/30

Now that we have allocated the first three subnets to each of the WAN links, the following remaining subnets can be set aside for any future branches which may need another WAN link. These will be assigned for future reservation:

Subnet 4	11000000.10101000.00000001.11101100 = 192.168.1.236/30
Subnet 5	11000000.10101000.00000001.11110000 = 192.168.1.240/30
Subnet 6	11000000.10101000.00000001.11110100 = 192.168.1.244/30
Subnet 7	11000000.10101000.00000001.11111000 = 192.168.1.248/30
Subnet 8	11000000.10101000.00000001.11111100 = 192.168.1.252/30

2. Explanation of Settings un Network Connections

Ans :

The process of setting it up in 7 simple steps.

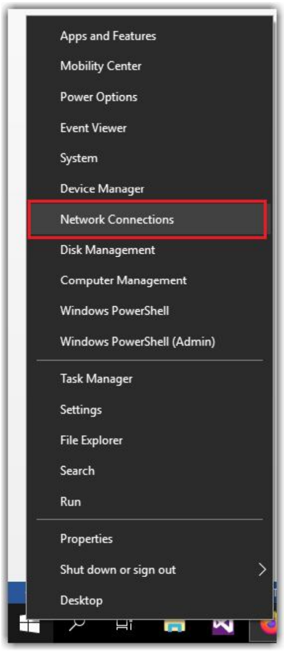
Step 1

Connect the device to your PC and start configuring network

The process of setting up network connection depends on the operation system you are using. In this example, Windows 10 is installed on PC.

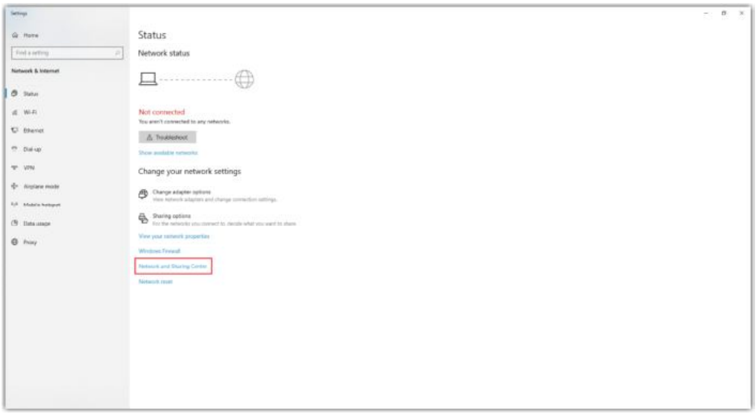
Step 2

To set up your network connection right click on Start and select Network connections.



Step 3

Press Network and Sharing Center.



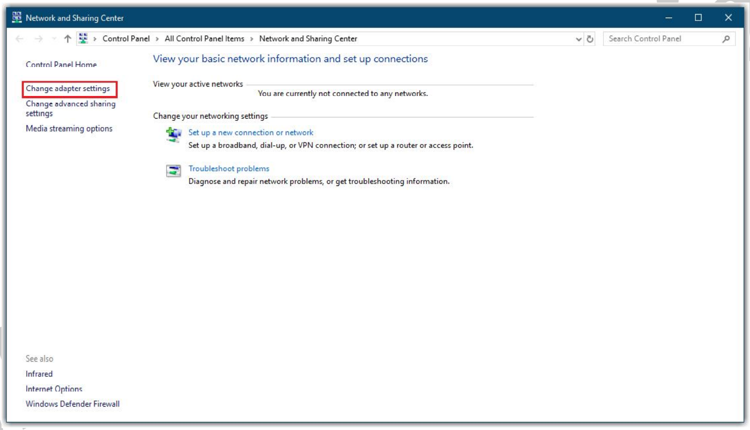
You aren't connected to any networks.

View network adapters and change connection settings.

For the networks you connect to, decide what you want to share.

Step 4

Open Change adapter settings window.



Network and Sharing Center
Control Panel › All Control Panel Items › Network and Sharing Center
Search Control Panel
Control Panel Home
Change adapter settings
Change advanced sharing settings
Media streaming options
See also
Infrared
Internet Options
Windows Defender Firewall
Set up a new connection or network
Troubleshoot problems

View your basic network information and set up connections

View your active networks

You are currently not connected to any networks.

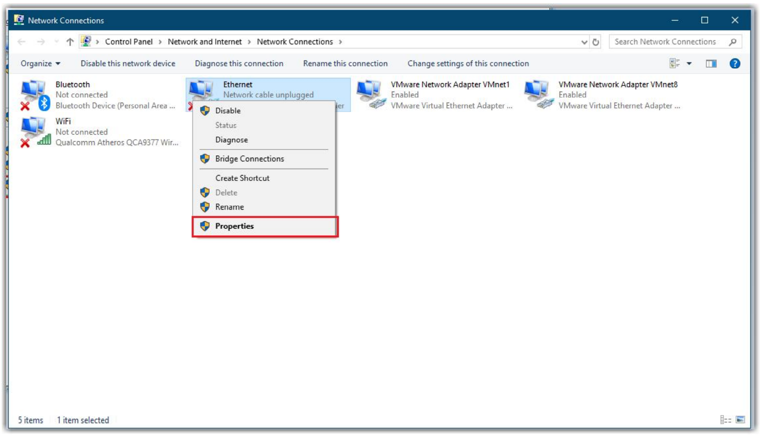
Change your networking settings

Set up a broadband, dial-up, or VPN connection; or set up a router or access point.

Diagnose and repair network problems, or get troubleshooting information.

Step 5

In this Change adapter settings window, you will see all network adapters available on your PC. Select one of them, click on it with the right mouse button and choose "Properties" from the context menu to open a Settings window.



Bluetooth
Not connected
Bluetooth Device (Personal Area ...

WiFi
Not connected
Qualcomm Atheros QCA9377 Wir...

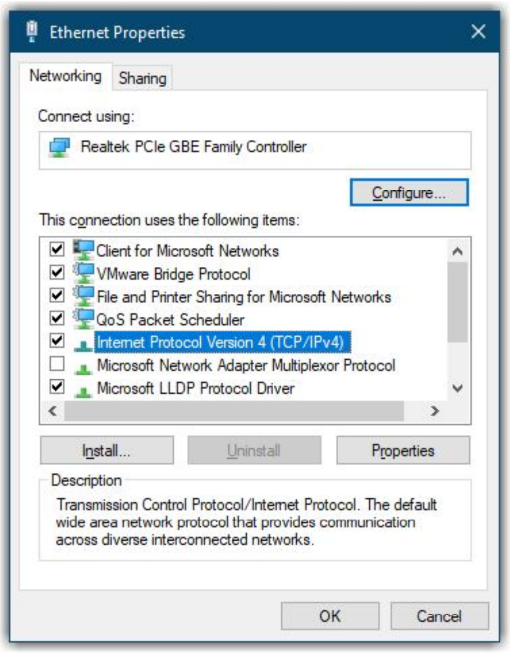
Ethernet
Network cable unplugged

VMware Network Adapter VMnet1
Enabled
VMware Virtual Ethernet Adapter ...

VMware Network Adapter VMnet8
Enabled
VMware Virtual Ethernet Adapter ...

Step 6

In the Settings windows, choose the item "Internet protocol Version 4 (TCP/IPv4)" (check the box opposite to the item, if it is not there), then press "Properties".



Ethernet Properties

Networking | Sharing

Connect using:

Realtek PCIe GBE Family Controller

This connection uses the following items:

- ☑ Client for Microsoft Networks
- ☑ VMware Bridge Protocol
- ☑ File and Printer Sharing for Microsoft Networks
- ☑ QoS Packet Scheduler
- ☑ Internet Protocol Version 4 (TCP/IPv4)
- □ Microsoft Network Adapter Multiplexor Protocol
- ☑ Microsoft LLDP Protocol Driver

Description

Transmission Control Protocol/Internet Protocol. The default wide area network protocol that provides communication across diverse interconnected networks.

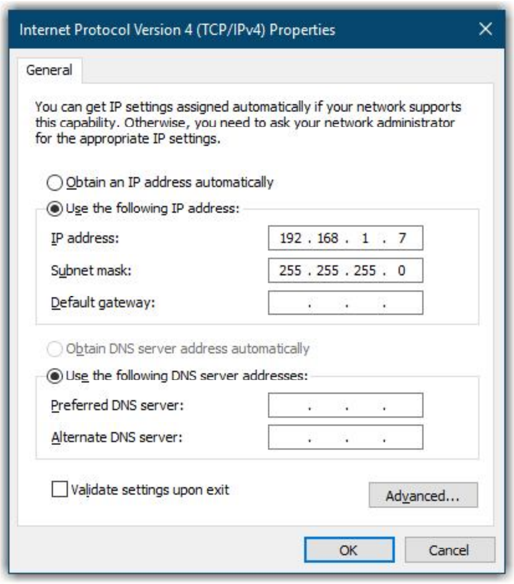
Step 7

Select the "Use the following IP-address:" option. Then enter the IP-address and the subnet mask. If the PC is part of a delivery package, it has the following network card settings:

- IP-address 192.168.1.7;
- Subnet mask 255.255.255.0.

The IP-address is written based on the following principles:

- the first 3 groups of digits should have same value as on any device (if there are devices with IP-addresses 192.168.1.9 and 192.168.1.10, then the numbers 192, 168 and 1 must be entered in the first 3 groups);
- the last group of numbers should make the IP-address of the PC stand out among the other devices in the set; for example, if there are devices with IP-addresses 192.168.1.9 and 192.168.1.10, then any number from 1 to 254 may be entered except 9, 10 and 127 (these numbers are reserved by the operation system)



Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically

● Use the following IP address:

IP address: 192 . 168 . 1 . 7

Subnet mask: 255 . 255 . 255 . 0

Default gateway: . . .

○ Obtain DNS server address automatically

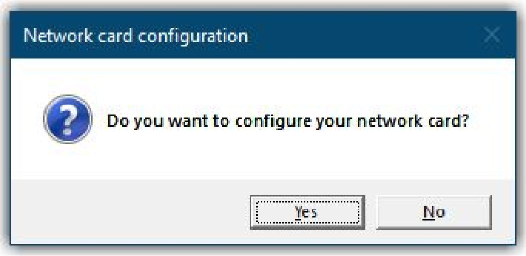
● Use the following DNS server addresses:

Preferred DNS server: . . .

Alternate DNS server: . . .

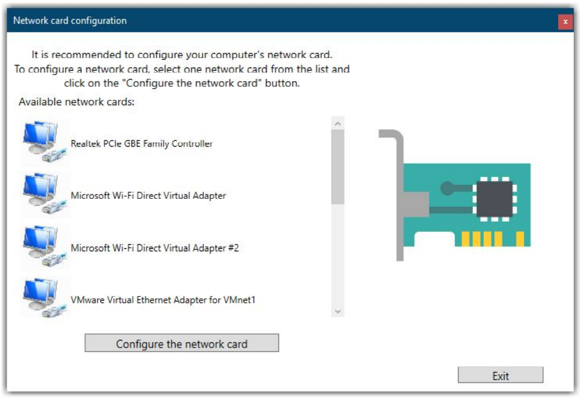
□ Validate settings upon exit

We also have a network manager in our RULA Software installer. To use it, accept network card configuration during setup. Select your network card in the new window and press the "Configure the network card" button.



Network card configuration

Do you want to configure your network card?

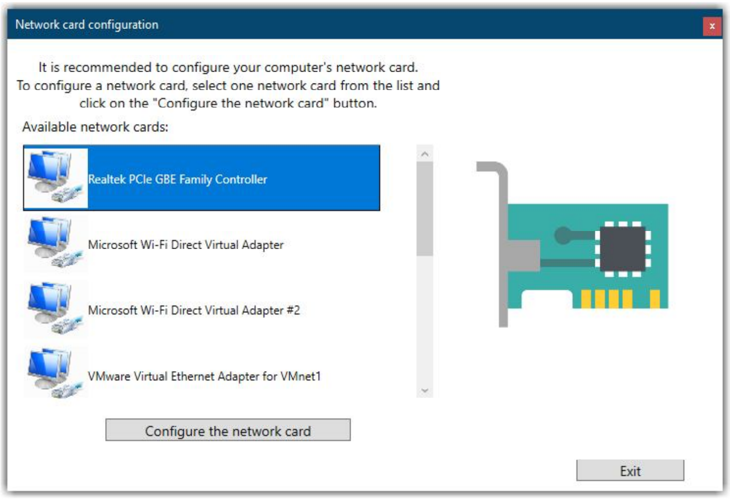


Network card configuration

It is recommended to configure your computer's network card.
To configure a network card, select one network card from the list and click on the "Configure the network card" button.

Available network cards:

- Realtek PCIe GBE Family Controller
- Microsoft Wi-Fi Direct Virtual Adapter
- Microsoft Wi-Fi Direct Virtual Adapter #2
- VMware Virtual Ethernet Adapter for VMnet1

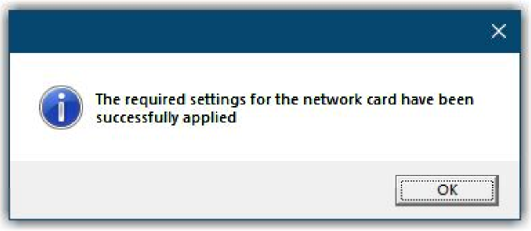


Network card configuration

It is recommended to configure your computer's network card.
To configure a network card, select one network card from the list and click on the "Configure the network card" button.

Available network cards:

- Realtek PCIe GBE Family Controller
- Microsoft Wi-Fi Direct Virtual Adapter
- Microsoft Wi-Fi Direct Virtual Adapter #2
- VMware Virtual Ethernet Adapter for VMnet1



The required settings for the network card have been successfully applied

3. Testing of Networking Connectivity using Ping, Tracepath

Ans :

How does ping work?

Ping uses ICMP (Internet Control Message Protocol) Echo messages to see if a remote host is active or inactive, how long a round trip message takes to reach the target host and return, and any packet loss.

It sends a request and waits for a reply (which it receives if the destination responds back within the timeout period).

It's basically a quick, easy way to verify that you can reach a destination on the internet. If you can, great! If not, you can use traceroute to investigate what's happening at every step between your device and the destination.

Example ping command and results:

```
hostname ~ % ping -c 5 www.google.com
PING www.google.com (216.58.212.228): 56 data bytes
```

The ping command, set to send 5 packets to google.com.

```
64 bytes from 216.58.212.228: icmp_seq=0 ttl=113 time=42.262 ms
64 bytes from 216.58.212.228: icmp_seq=1 ttl=113 time=34.796 ms
64 bytes from 216.58.212.228: icmp_seq=2 ttl=113 time=35.805 ms
64 bytes from 216.58.212.228: icmp_seq=3 ttl=113 time=45.299 ms
64 bytes from 216.58.212.228: icmp_seq=4 ttl=113 time=150.292 ms
```

This shows the results from each individual ping, with their round trip time in milliseconds.

```
— www.google.com ping statistics —
```

5 packets transmitted, 5 packets received, 0.0% packet loss

round-trip min/avg/max/stddev = 34.796/61.691/150.292/44.474 ms

The stats from the entire test - the minimum time it took to reach the destination, the average, the maximum, and the standard deviation.

How does traceroute work?

By default, traceroute sends three packets of data to test each 'hop' (when a packet is passed between routers it is called a 'hop').

It will first send 3 packets to an unreachable port on the target host, each with a Time-To-Live (TTL) value of 1. This means that as soon as it hits the first router in the path (within your network), it will timeout. The first router will respond with an ICMP Time Exceeded Message (TEM), as the datagram has expired.

Then another 3 datagrams are sent, with the TTL set to 2, causing the second router (your ISP) in the path to respond with an ICMP TEM.

This continues until the datagrams eventually have a TTL long enough to reach the destination. When it does, as the messages are being sent to an invalid port, an ICMP port unreachable message is returned, signaling that the traceroute is finished.

In this case, an error message is actually expected behavior, not a sign that something has gone wrong.

The most important part of a traceroute is usually the round trip times. Ideally you're looking for consistent times over the course of the trace.

If you see times suddenly increase (elevated latency) on a specific hop, and continue to increase as the trace approaches the target, this may indicate a problem starting with the sudden increase.

However, if there is elevated latency in the middle, but it remains consistent toward the end, or if the elevated latency decreases toward the end, that doesn't necessarily indicate a problem.

If you see high latency at the beginning of the trace, it may indicate a problem with your local network. You should work with your local admin (or yourself, if you are your own local admin) to fix it. By default, Windows uses ICMP to transmit the data while Linux uses UDP.

Example traceroute command and result:

```
hostname ~ % traceroute www.google.com
```

```
traceroute to www.google.com (216.58.212.228), 64 hops max, 52 byte packets
```

The command to traceroute to google.

```
1. homerouter.cpe (192.168.8.1)   10.129 ms   1.528 ms   1.373 ms
```

The first hop is within a local network. Here, we have the hop number (1), the domain name/IP address (in this case a home router), then RTT1, RTT2, and RTT3 (Round Trip Time - the time it takes for a packet to get to the hop and back to the computer, in milliseconds). This is the latency of the hop.

There are three numbers because, by default, the command sends three data packets. In general, times over 150ms are unusual for a trip within the continental US, though signals crossing an ocean may exceed this time.

```
2   * * *
```

Hop 2: There are two possibilities for stars like this - either ICMP/UDP were not configured on the receiving device and it did not respond, or the packets were dropped due to a network issue (such as a firewall or packet timeouts).

In this case, as it's very close to the beginning of the trace, it's likely that this is due to the device not being configured to send responses to a traceroute.

```
3   192.168.213.21 (192.168.213.21)   26.641 ms   31.671 ms   26.824 ms
4   192.168.213.22 (192.168.213.22)   20.294 ms   22.496 ms   19.922 ms
5   * * *
6   * * *
```

These stars, further along in the trace, are more likely to be due to a target's firewall blocking requests (though HTTP requests should still be able to be processed in most cases), a possible connection problem, or a return path issue (that is, the signal is reaching the router, but isn't getting a response).

The trace will then continue until it reaches the target.

4. Checking Network Statistics with Netstat.

Ans :

Steps on How to Check Network Statistics using Netstat in Windows

Netstat or Network Statistics is a network utility that allows you to check incoming and outgoing network connections across various protocols, routing table, owning processes, statistics, etc.

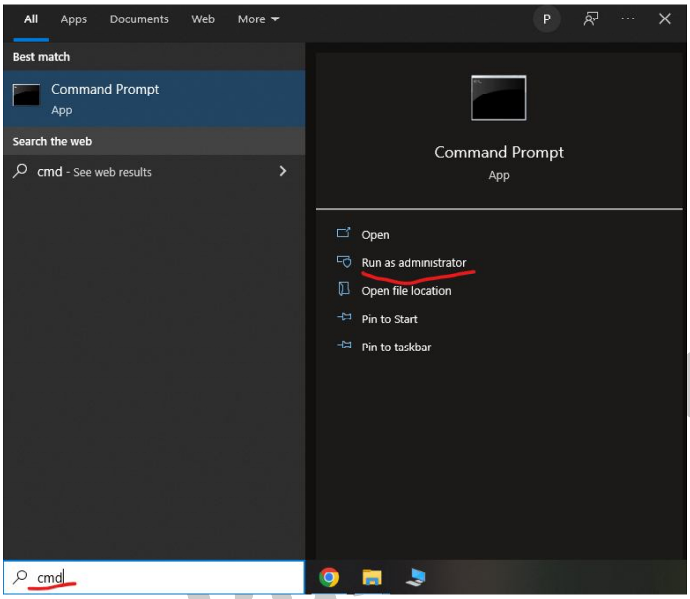
We are going to learn how to leverage netstat from our command prompt in Windows 10.

Before we learn how to check which port number is free we will learn more about some 'netstat' commands.

It is advisable to be in an administrative mode because generally many port numbers are kept hidden from the guest users.

Step 1: Open cmd (command prompt) in your system as an administrator.

If you don't know how to open it as an administrator here's how to do that:



As shown above, you can simply type 'cmd' in the search bar at the left bottom of your screen. Then click on Run as administrator.

Note: If UAC (User Account Control) asks you for confirmation, just click on Yes.

Using Netstat to Check Network Statistics in Windows

Post this point remember to use the above-mentioned Step.

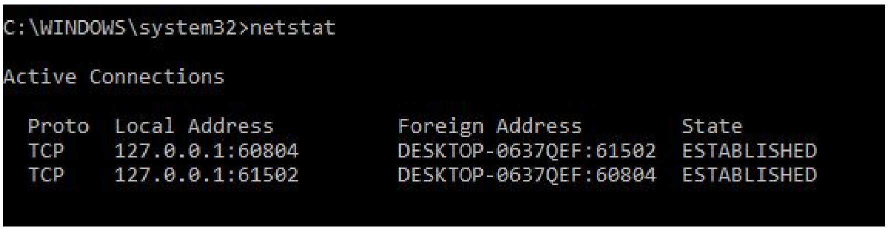
How to Check Active Connections

Perform Step 1 (going forward assume Step 1 has to be performed) and then in the opened command prompt window, type the following:

netstat

and press enter.

Doing so will list down the active connections in a tabular fashion with the first column specifying the Protocol. The second one tells the local address, the third foreign, and the last one tells you their current State.



```
C:\WINDOWS\system32>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    127.0.0.1:60804        DESKTOP-0637QEF:61502  ESTABLISHED
  TCP    127.0.0.1:61502        DESKTOP-0637QEF:60804  ESTABLISHED
```

In the example above, you can see in the local address column, that the first part 127.0.0.1 is your localhost or loopback address. The second bit i.e. 60804 is the port number of your local device which has established a connection.

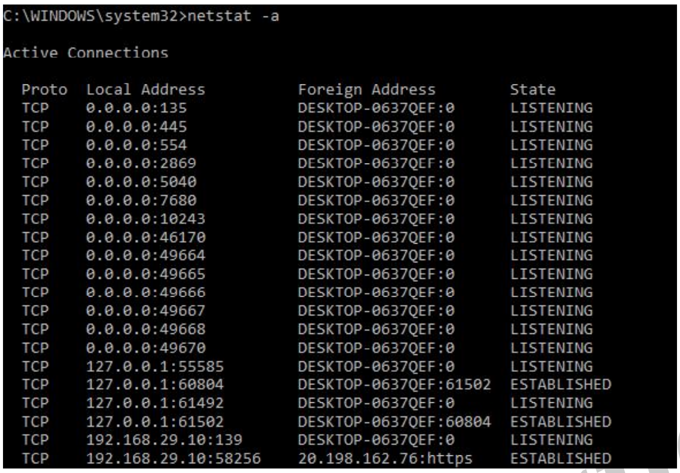
Note: You can press Ctrl + C, if you do not want the search to continue at any point.

How to Check all Connections and Listening Ports

If you want to check all the connections of all the protocols you can use the following command:

netstat -a

where a means all sockets of all protocols like TCP, UDP etc.



```
C:\WINDOWS\system32>netstat -a

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:445            DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:554            DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:2869           DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:5040           DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:7680           DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:10243          DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:46170          DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:49664          DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:49665          DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:49666          DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:49667          DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:49668          DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:49670          DESKTOP-0637QEF:0      LISTENING
  TCP    127.0.0.1:55585        DESKTOP-0637QEF:0      LISTENING
  TCP    127.0.0.1:60804        DESKTOP-0637QEF:61502  ESTABLISHED
  TCP    127.0.0.1:61492        DESKTOP-0637QEF:0      LISTENING
  TCP    127.0.0.1:61502        DESKTOP-0637QEF:60804  ESTABLISHED
  TCP    192.168.29.10:139      DESKTOP-0637QEF:0      LISTENING
  TCP    192.168.29.10:58256    20.198.162.76:https    ESTABLISHED
```

Understanding Port Messages

All the 'Listening' ports are available or the 'Open' ports that you can use. All the ports that say 'Established' are taken.

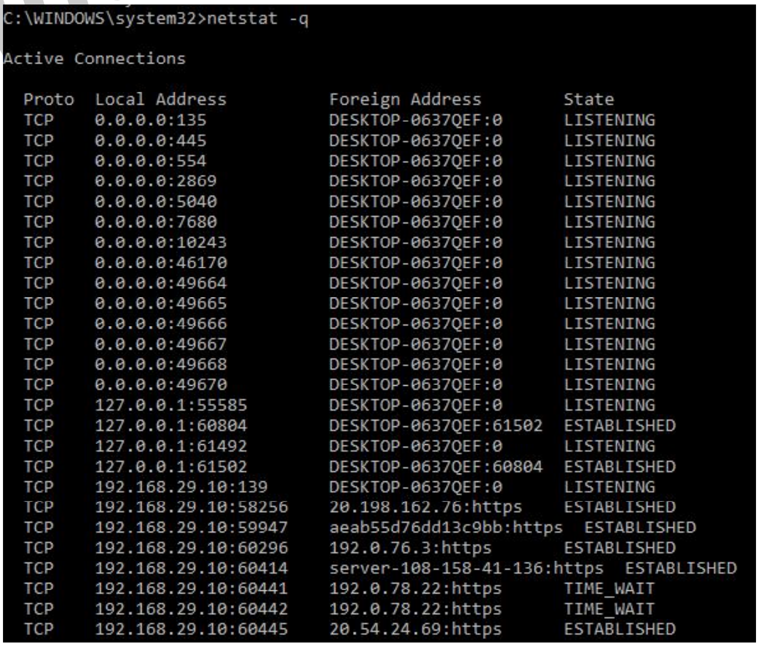
In other words, LISTENING means that a classic open port is listening for inbound connections.

How to Check Bound Non-Listening Ports

If you also want to check bound nonlistening ports, you could alternatively use:

netstat -q

and press Enter.

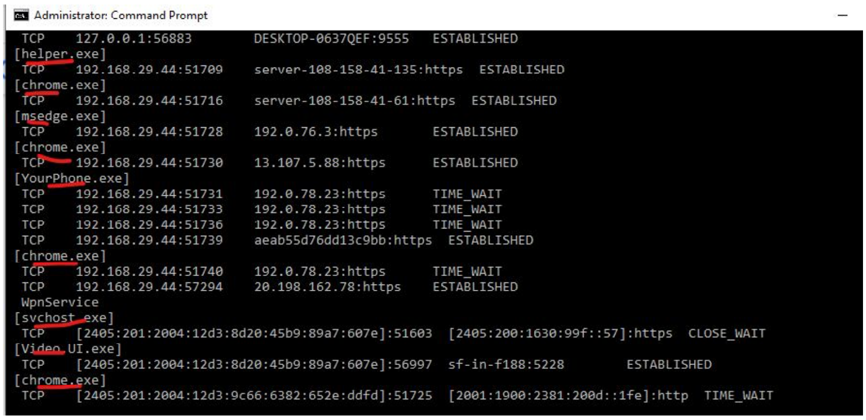


```
C:\WINDOWS\system32>netstat -q

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:445            DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:554            DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:2869           DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:5040           DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:7680           DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:10243          DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:46170          DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:49664          DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:49665          DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:49666          DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:49667          DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:49668          DESKTOP-0637QEF:0      LISTENING
  TCP    0.0.0.0:49670          DESKTOP-0637QEF:0      LISTENING
  TCP    127.0.0.1:55585        DESKTOP-0637QEF:0      LISTENING
  TCP    127.0.0.1:60804        DESKTOP-0637QEF:61502  ESTABLISHED
  TCP    127.0.0.1:61492        DESKTOP-0637QEF:0      LISTENING
  TCP    127.0.0.1:61502        DESKTOP-0637QEF:60804  ESTABLISHED
  TCP    192.168.29.10:139      DESKTOP-0637QEF:0      LISTENING
  TCP    192.168.29.10:58256    20.198.162.76:https    ESTABLISHED
  TCP    192.168.29.10:59947    aeab55d76dd13c9bb:https  ESTABLISHED
  TCP    192.168.29.10:60296    192.0.76.3:https       ESTABLISHED
  TCP    192.168.29.10:60414    server-108-158-41-136:https  ESTABLISHED
  TCP    192.168.29.10:60441    192.0.78.22:https      TIME_WAIT
  TCP    192.168.29.10:60442    192.0.78.22:https      TIME_WAIT
  TCP    192.168.29.10:60445    20.54.24.69:https      ESTABLISHED
```

How to Check Executable files involved in Each Connection

The command to check the executable files in each connection is:

netstat -b



```
 TCP    127.0.0.1:56883        DESKTOP-0637QEF:9555   ESTABLISHED
[helper.exe]
 TCP    192.168.29.44:51709    server-108-158-41-135:https  ESTABLISHED
[chrome.exe]
 TCP    192.168.29.44:51716    server-108-158-41-61:https  ESTABLISHED
[msedge.exe]
 TCP    192.168.29.44:51728    192.0.76.3:https       ESTABLISHED
[chrome.exe]
 TCP    192.168.29.44:51730    13.107.5.88:https      ESTABLISHED
[YourPhone.exe]
 TCP    192.168.29.44:51731    192.0.78.23:https      TIME_WAIT
 TCP    192.168.29.44:51733    192.0.78.23:https      TIME_WAIT
 TCP    192.168.29.44:51736    192.0.78.23:https      TIME_WAIT
 TCP    192.168.29.44:51739    aeab55d76dd13c9bb:https  ESTABLISHED
[chrome.exe]
 TCP    192.168.29.44:51740    192.0.78.23:https      TIME_WAIT
 TCP    192.168.29.44:57294    20.198.162.78:https    ESTABLISHED
 WpnService
[svchost.exe]
 TCP    [2405:201:2004:12d3:8d20:45b9:89a7:607e]:51603  [2405:200:1630:99f::57]:https  CLOSE_WAIT
[Video.UI.exe]
 TCP    [2405:201:2004:12d3:8d20:45b9:89a7:607e]:56997  sf-in-f188:5228        ESTABLISHED
[chrome.exe]
 TCP    [2405:201:2004:12d3:9c66:6382:652e:ddfd]:51725  [2001:1900:2381:200d::1fe]:http  TIME_WAIT
```

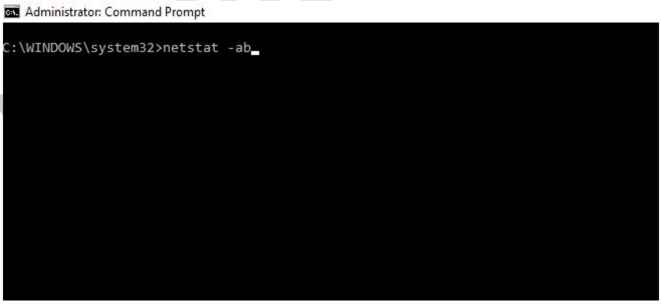
As you can see in the image they are the executable files that have either established connection or are in waiting mode.

How to Display all Connections and Listening Ports and Executable Files

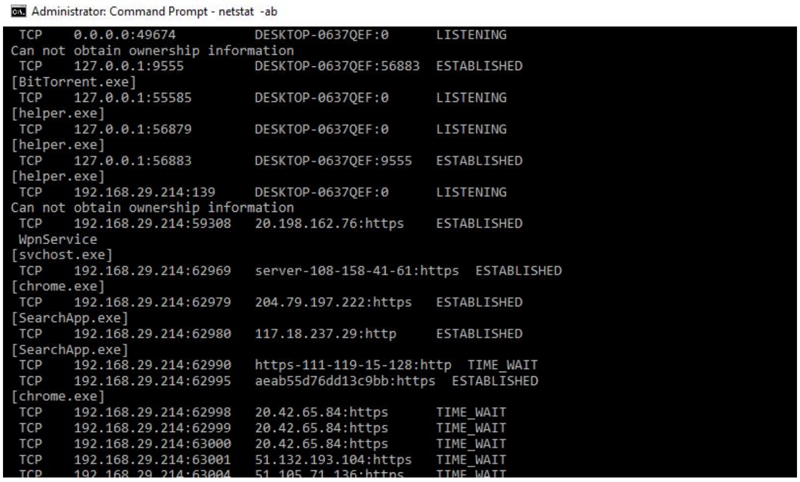
The above commands can be joined together as well, hence if you want to check all the connections and listening ports along with the executable file information, you can just type:

netstat -ab

and press enter where -a means all sockets of all protocols like TCP, UDP etc. and -b is used to display executable files.



```
C:\WINDOWS\system32>netstat -ab
```



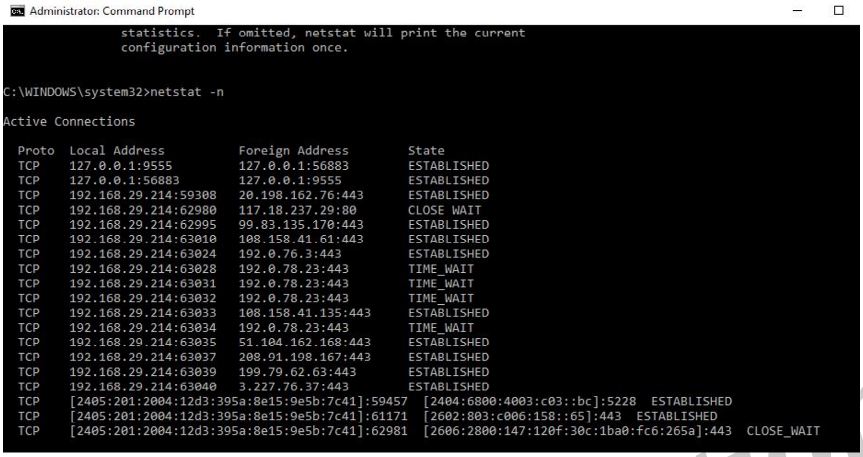
```
 TCP    0.0.0.0:49674          DESKTOP-0637QEF:0      LISTENING
Can not obtain ownership information
 TCP    127.0.0.1:9555         DESKTOP-0637QEF:56883  ESTABLISHED
[BitTorrent.exe]
 TCP    127.0.0.1:55585        DESKTOP-0637QEF:0      LISTENING
[helper.exe]
 TCP    127.0.0.1:56879        DESKTOP-0637QEF:0      LISTENING
[helper.exe]
 TCP    127.0.0.1:56883        DESKTOP-0637QEF:9555   ESTABLISHED
[helper.exe]
 TCP    192.168.29.214:139     DESKTOP-0637QEF:0      LISTENING
Can not obtain ownership information
 TCP    192.168.29.214:59308   20.198.162.76:https    ESTABLISHED
 WpnService
[svchost.exe]
 TCP    192.168.29.214:62969   server-108-158-41-61:https  ESTABLISHED
[chrome.exe]
 TCP    192.168.29.214:62979   204.79.197.222:https   ESTABLISHED
[SearchApp.exe]
 TCP    192.168.29.214:62980   117.18.237.29:http     ESTABLISHED
[SearchApp.exe]
 TCP    192.168.29.214:62990   https-111-119-15-128:http  TIME_WAIT
 TCP    192.168.29.214:62995   aeab55d76dd13c9bb:https  ESTABLISHED
[chrome.exe]
 TCP    192.168.29.214:62998   20.42.65.84:https      TIME_WAIT
 TCP    192.168.29.214:62999   20.42.65.84:https      TIME_WAIT
 TCP    192.168.29.214:63000   20.42.65.84:https      TIME_WAIT
 TCP    192.168.29.214:63001   51.132.193.104:https   TIME_WAIT
 TCP    192.168.29.214:63004   51.105.71.136:https    TIME_WAIT
```

How to Display Addresses and Port Numbers in Numerical Form

If you want to display the addresses in numerical form type the following:

netstat -n

As you can see in the image below, even the foreign address has been converted into number format.



```
               statistics.  If omitted, netstat will print the current
               configuration information once.

C:\WINDOWS\system32>netstat -n

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    127.0.0.1:9555         127.0.0.1:56883        ESTABLISHED
  TCP    127.0.0.1:56883        127.0.0.1:9555         ESTABLISHED
  TCP    192.168.29.214:59308   20.198.162.76:443      ESTABLISHED
  TCP    192.168.29.214:62980   117.18.237.29:80       CLOSE_WAIT
  TCP    192.168.29.214:62995   99.83.135.170:443      ESTABLISHED
  TCP    192.168.29.214:63010   108.158.41.61:443      ESTABLISHED
  TCP    192.168.29.214:63024   192.0.76.3:443         ESTABLISHED
  TCP    192.168.29.214:63028   192.0.78.23:443        TIME_WAIT
  TCP    192.168.29.214:63031   192.0.78.23:443        TIME_WAIT
  TCP    192.168.29.214:63032   192.0.78.23:443        TIME_WAIT
  TCP    192.168.29.214:63033   108.158.41.135:443     ESTABLISHED
  TCP    192.168.29.214:63034   192.0.78.23:443        TIME_WAIT
  TCP    192.168.29.214:63035   51.104.162.168:443     ESTABLISHED
  TCP    192.168.29.214:63037   208.91.198.167:443     ESTABLISHED
  TCP    192.168.29.214:63039   199.79.62.63:443       ESTABLISHED
  TCP    192.168.29.214:63040   3.227.76.37:443        ESTABLISHED
  TCP    [2405:201:2004:12d3:395a:8e15:9e5b:7c41]:59457  [2404:6800:4003:c03::bc]:5228  ESTABLISHED
  TCP    [2405:201:2004:12d3:395a:8e15:9e5b:7c41]:61171  [2602:803:c006:158::65]:443  ESTABLISHED
  TCP    [2405:201:2004:12d3:395a:8e15:9e5b:7c41]:62981  [2606:2800:147:120f:30c:1ba0:fc6:265a]:443  CLOSE_WAIT
```

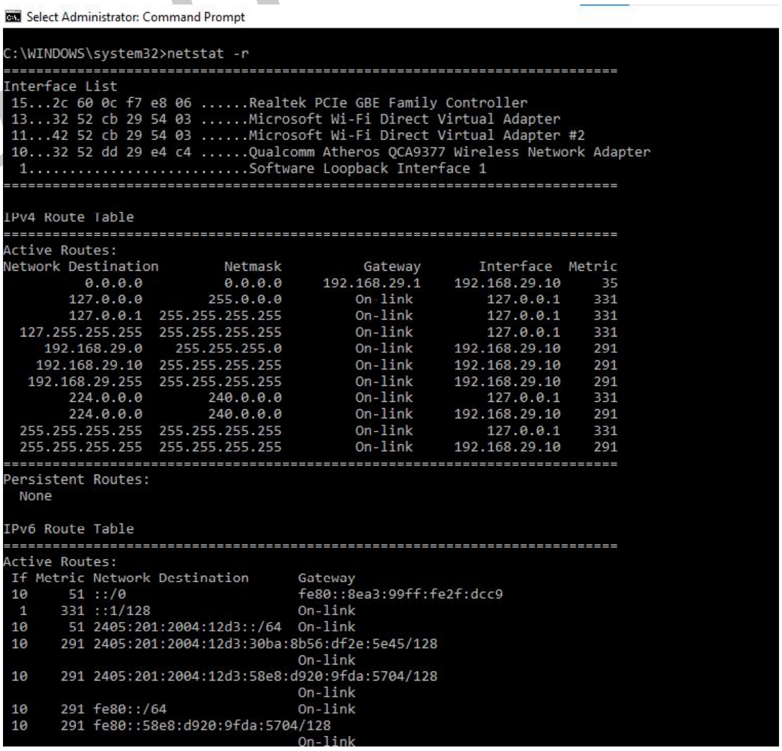
How to Display Routing Table

In order to display the routing table just use the following command:

netstat -r

and press Enter.

Apart from giving an Interface List, you will get a glimpse of how your IP connections are being routed.



```
C:\WINDOWS\system32>netstat -r
===========================================================================
Interface List
 15...2c 60 0c f7 e8 06 ......Realtek PCIe GBE Family Controller
 13...32 52 cb 29 54 03 ......Microsoft Wi-Fi Direct Virtual Adapter
 11...42 52 cb 29 54 03 ......Microsoft Wi-Fi Direct Virtual Adapter #2
 10...32 52 dd 29 e4 c4 ......Qualcomm Atheros QCA9377 Wireless Network Adapter
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0     192.168.29.1    192.168.29.10     35
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
     192.168.29.0    255.255.255.0         On-link     192.168.29.10    291
    192.168.29.10  255.255.255.255         On-link     192.168.29.10    291
   192.168.29.255  255.255.255.255         On-link     192.168.29.10    291
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link     192.168.29.10    291
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    331
  255.255.255.255  255.255.255.255         On-link     192.168.29.10    291
===========================================================================
Persistent Routes:
  None

IPv6 Route Table
===========================================================================
Active Routes:
 If Metric Network Destination      Gateway
 10     51 ::/0                     fe80::8ea3:99ff:fe2f:dcc9
  1    331 ::1/128                  On-link
 10     51 2405:201:2004:12d3::/64  On-link
 10    291 2405:201:2004:12d3:30ba:8b56:df2e:5e45/128
                                    On-link
 10    291 2405:201:2004:12d3:58e8:d920:9fda:5704/128
                                    On-link
 10    291 fe80::/64                On-link
 10    291 fe80::58e8:d920:9fda:5704/128
                                    On-link
```

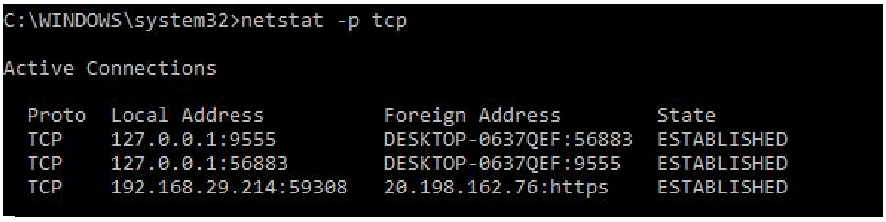
How to Check Connections for a specific Protocol

You can use the following command for that:

netstat -p proto

where replace the word proto with the protocol you want to check connections for and press enter.

Like this:

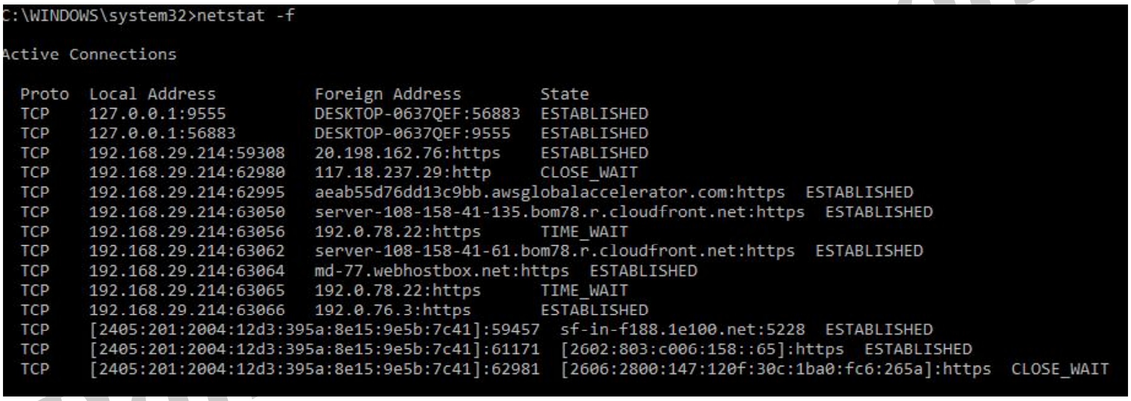


```
C:\WINDOWS\system32>netstat -p tcp

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    127.0.0.1:9555         DESKTOP-0637QEF:56883  ESTABLISHED
  TCP    127.0.0.1:56883        DESKTOP-0637QEF:9555   ESTABLISHED
  TCP    192.168.29.214:59308   20.198.162.76:https    ESTABLISHED
```

How to Get FQDNs for Foreign Addresses

If you want to get FQDN i.e. Fully Qualified Domain Names for foreign addresses, the command to use is:

netstat -f



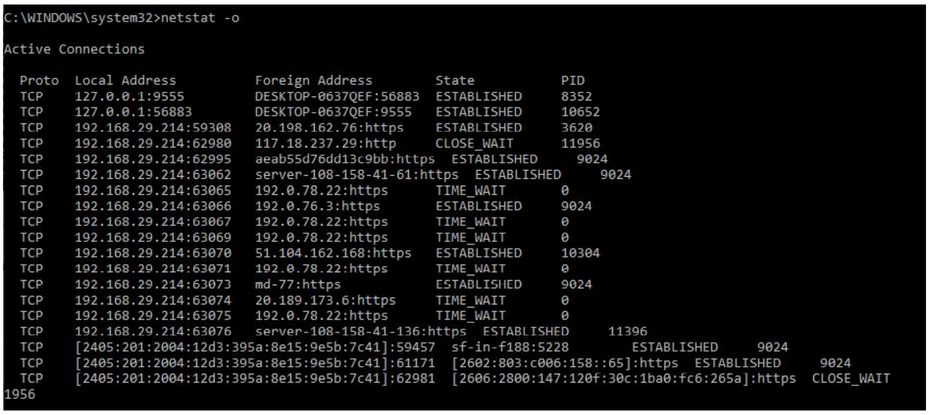
```
C:\WINDOWS\system32>netstat -f

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    127.0.0.1:9555         DESKTOP-0637QEF:56883  ESTABLISHED
  TCP    127.0.0.1:56883        DESKTOP-0637QEF:9555   ESTABLISHED
  TCP    192.168.29.214:59308   20.198.162.76:https    ESTABLISHED
  TCP    192.168.29.214:62980   117.18.237.29:http     CLOSE_WAIT
  TCP    192.168.29.214:62995   aeab55d76dd13c9bb.awsglobalaccelerator.com:https  ESTABLISHED
  TCP    192.168.29.214:63050   server-108-158-41-135.bom78.r.cloudfront.net:https  ESTABLISHED
  TCP    192.168.29.214:63056   192.0.78.22:https      TIME_WAIT
  TCP    192.168.29.214:63062   server-108-158-41-61.bom78.r.cloudfront.net:https  ESTABLISHED
  TCP    192.168.29.214:63064   md-77.webhostbox.net:https  ESTABLISHED
  TCP    192.168.29.214:63065   192.0.78.22:https      TIME_WAIT
  TCP    192.168.29.214:63066   192.0.76.3:https       ESTABLISHED
  TCP    [2405:201:2004:12d3:395a:8e15:9e5b:7c41]:59457  sf-in-f188.1e100.net:5228  ESTABLISHED
  TCP    [2405:201:2004:12d3:395a:8e15:9e5b:7c41]:61171  [2602:803:c006:158::65]:https  ESTABLISHED
  TCP    [2405:201:2004:12d3:395a:8e15:9e5b:7c41]:62981  [2606:2800:147:120f:30c:1ba0:fc6:265a]:https  CLOSE_WAIT
```

How to Check Owning Process Id

Adding just -o to netstat command will give you the owning process Id as well.

So punch in the following command and press enter:

netstat -o



```
C:\WINDOWS\system32>netstat -o

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    127.0.0.1:9555         DESKTOP-0637QEF:56883  ESTABLISHED     8352
  TCP    127.0.0.1:56883        DESKTOP-0637QEF:9555   ESTABLISHED     10652
  TCP    192.168.29.214:59308   20.198.162.76:https    ESTABLISHED     3620
  TCP    192.168.29.214:62980   117.18.237.29:http     CLOSE_WAIT      11956
  TCP    192.168.29.214:62995   aeab55d76dd13c9bb:https  ESTABLISHED     9024
  TCP    192.168.29.214:63062   server-108-158-41-61:https  ESTABLISHED     9024
  TCP    192.168.29.214:63065   192.0.78.22:https      TIME_WAIT       0
  TCP    192.168.29.214:63066   192.0.76.3:https       ESTABLISHED     9024
  TCP    192.168.29.214:63067   192.0.78.22:https      TIME_WAIT       0
  TCP    192.168.29.214:63069   192.0.78.22:https      TIME_WAIT       0
  TCP    192.168.29.214:63070   51.104.162.168:https   ESTABLISHED     10304
  TCP    192.168.29.214:63071   192.0.78.22:https      TIME_WAIT       0
  TCP    192.168.29.214:63073   md-77:https            ESTABLISHED     9024
  TCP    192.168.29.214:63074   20.189.173.6:https     TIME_WAIT       0
  TCP    192.168.29.214:63075   192.0.78.22:https      TIME_WAIT       0
  TCP    192.168.29.214:63076   server-108-158-41-136:https  ESTABLISHED     11396
  TCP    [2405:201:2004:12d3:395a:8e15:9e5b:7c41]:59457  sf-in-f188:5228        ESTABLISHED     9024
  TCP    [2405:201:2004:12d3:395a:8e15:9e5b:7c41]:61171  [2602:803:c006:158::65]:https  ESTABLISHED     9024
  TCP    [2405:201:2004:12d3:395a:8e15:9e5b:7c41]:62981  [2606:2800:147:120f:30c:1ba0:fc6:265a]:https  CLOSE_WAIT
1956
```

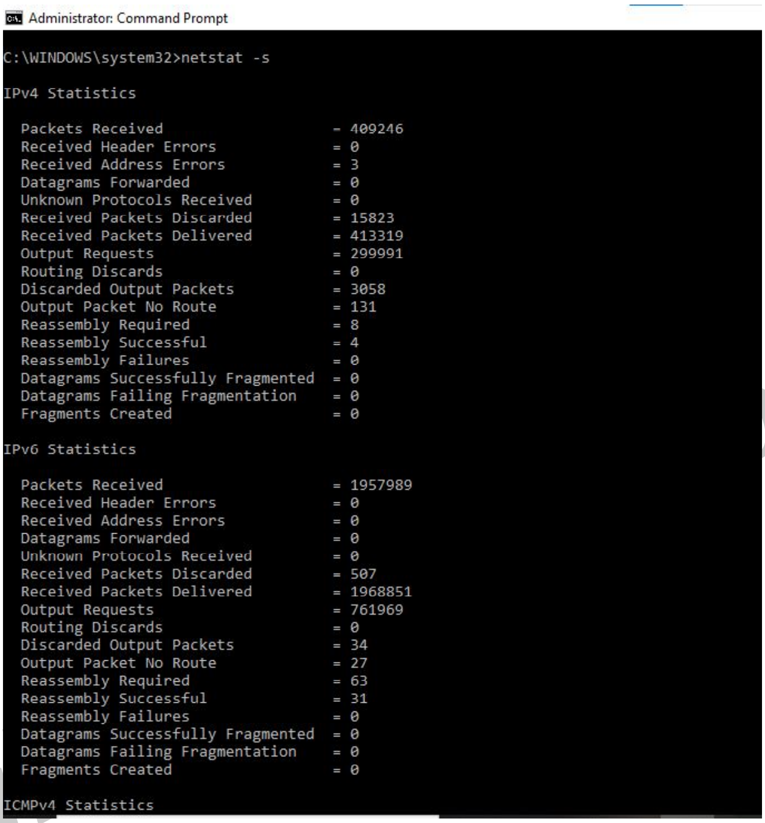
How to Check Statistics for your Network using Netstat

Netstat gives you the ability to check per-protocol statistics which gives you a glimpse of how many packets were received, errors, discarded or delivered status etc.

You can obtain that using the following command:

netstat -s

Press Enter after you have typed the above.



```
Administrator: Command Prompt

C:\WINDOWS\system32>netstat -s

IPv4 Statistics

  Packets Received                   = 409246
  Received Header Errors             = 0
  Received Address Errors            = 3
  Datagrams Forwarded                = 0
  Unknown Protocols Received         = 0
  Received Packets Discarded         = 15823
  Received Packets Delivered         = 413319
  Output Requests                    = 299991
  Routing Discards                   = 0
  Discarded Output Packets           = 3058
  Output Packet No Route             = 131
  Reassembly Required                = 8
  Reassembly Successful              = 4
  Reassembly Failures                = 0
  Datagrams Successfully Fragmented  = 0
  Datagrams Failing Fragmentation    = 0
  Fragments Created                  = 0

IPv6 Statistics

  Packets Received                   = 1957989
  Received Header Errors             = 0
  Received Address Errors            = 0
  Datagrams Forwarded                = 0
  Unknown Protocols Received         = 0
  Received Packets Discarded         = 507
  Received Packets Delivered         = 1968851
  Output Requests                    = 761969
  Routing Discards                   = 0
  Discarded Output Packets           = 34
  Output Packet No Route             = 27
  Reassembly Required                = 63
  Reassembly Successful              = 31
  Reassembly Failures                = 0
  Datagrams Successfully Fragmented  = 0
  Datagrams Failing Fragmentation    = 0
  Fragments Created                  = 0

ICMPv4 Statistics
```

By default, you get the statistics for the following protocols:

IP, IPv6, ICMP, ICMPv6, TCP, TCPv6, UDP, and UDPv6

How to Check Statistics for a Specific Protocol

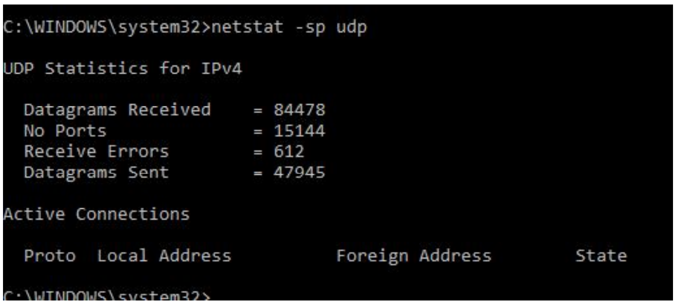
If you want to check statistics based on a specific protocol, you can type the following:

netstat -sp proto

where replace the word proto with the protocol you want the statistics for and then press enter.

for e.g.:

netstat -sp udp



```
C:\WINDOWS\system32>netstat -sp udp

UDP Statistics for IPv4

  Datagrams Received    = 84478
  No Ports              = 15144
  Receive Errors        = 612
  Datagrams Sent        = 47945

Active Connections

  Proto  Local Address          Foreign Address        State

C:\WINDOWS\system32>
```

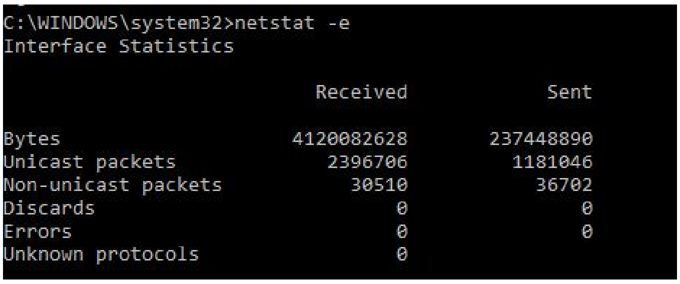
One thing to note here is that what it also does is show you the active connections along with the statistical information.

How to Check Ethernet Statistics

In a similar fashion if you want to grab ethernet interface statistics, you can type the following:

netstat -e

and press Enter.



```
C:\WINDOWS\system32>netstat -e
Interface Statistics

                           Received            Sent

Bytes                    4120082628       237448890
Unicast packets             2396706         1181046
Non-unicast packets           30510           36702
Discards                          0               0
Errors                            0               0
Unknown protocols                 0
```

You can combine other flags for various results too.

For e.g.

netstat -es

or

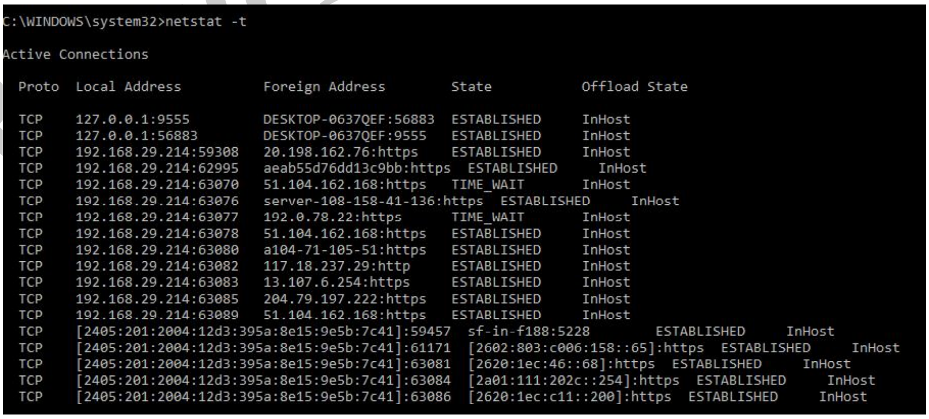
netstat -se

will give you results for both ethernet and protocol statistics.

How to Check Offload State of Connections

The command to use in order to check the offload state of current connections is:

netstat -t

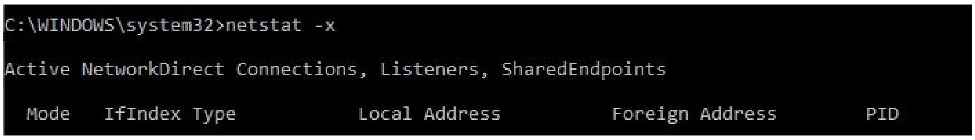


```
C:\WINDOWS\system32>netstat -t

Active Connections

  Proto  Local Address          Foreign Address        State           Offload State

  TCP    127.0.0.1:9555         DESKTOP-0637QEF:56883  ESTABLISHED     InHost
  TCP    127.0.0.1:56883        DESKTOP-0637QEF:9555   ESTABLISHED     InHost
  TCP    192.168.29.214:59308   20.198.162.76:https    ESTABLISHED     InHost
  TCP    192.168.29.214:62995   aeab55d76dd13c9bb:https  ESTABLISHED     InHost
  TCP    192.168.29.214:63070   51.104.162.168:https   TIME_WAIT       InHost
  TCP    192.168.29.214:63076   server-108-158-41-136:https  ESTABLISHED     InHost
  TCP    192.168.29.214:63077   192.0.78.22:https      TIME_WAIT       InHost
  TCP    192.168.29.214:63078   51.104.162.168:https   ESTABLISHED     InHost
  TCP    192.168.29.214:63080   a104-71-105-51:https   ESTABLISHED     InHost
  TCP    192.168.29.214:63082   117.18.237.29:http     ESTABLISHED     InHost
  TCP    192.168.29.214:63083   13.107.6.254:https     ESTABLISHED     InHost
  TCP    192.168.29.214:63085   204.79.197.222:https   ESTABLISHED     InHost
  TCP    192.168.29.214:63089   51.104.162.168:https   ESTABLISHED     InHost
  TCP    [2405:201:2004:12d3:395a:8e15:9e5b:7c41]:59457  sf-in-f188:5228        ESTABLISHED     InHost
  TCP    [2405:201:2004:12d3:395a:8e15:9e5b:7c41]:61171  [2602:803:c006:158::65]:https  ESTABLISHED     InHost
  TCP    [2405:201:2004:12d3:395a:8e15:9e5b:7c41]:63081  [2620:1ec:46::68]:https  ESTABLISHED     InHost
  TCP    [2405:201:2004:12d3:395a:8e15:9e5b:7c41]:63084  [2a01:111:202c::254]:https  ESTABLISHED     InHost
  TCP    [2405:201:2004:12d3:395a:8e15:9e5b:7c41]:63086  [2620:1ec:c11::200]:https  ESTABLISHED     InHost
```

How to Display NetworkDirect connections, listeners, and shared Endpoints

The command that would help you grab Active, NetworkDirect Connections, Listeners and Shared Endpoints is:

netstat -x



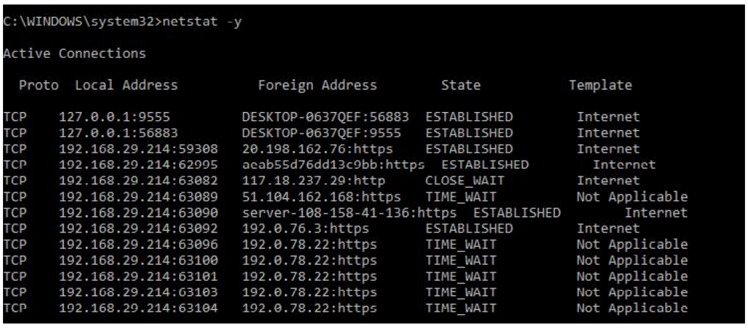
```
C:\WINDOWS\system32>netstat -x

Active NetworkDirect Connections, Listeners, SharedEndpoints

  Mode   IfIndex Type           Local Address          Foreign Address         PID
```

How to Display TCP Connection Template for Connections

The command that displays the TCP connection template is:

netstat -y

However, remember this cannot be clubbed with other commands.



```
C:\WINDOWS\system32>netstat -y

Active Connections

  Proto  Local Address          Foreign Address        State           Template

TCP    127.0.0.1:9555         DESKTOP-0637QEF:56883  ESTABLISHED       Internet
TCP    127.0.0.1:56883        DESKTOP-0637QEF:9555   ESTABLISHED       Internet
TCP    192.168.29.214:59308   20.198.162.76:https    ESTABLISHED       Internet
TCP    192.168.29.214:62995   aeab55d76dd13c9bb:https  ESTABLISHED       Internet
TCP    192.168.29.214:63082   117.18.237.29:http     CLOSE_WAIT        Internet
TCP    192.168.29.214:63089   51.104.162.168:https   TIME_WAIT         Not Applicable
TCP    192.168.29.214:63090   server-108-158-41-136:https  ESTABLISHED       Internet
TCP    192.168.29.214:63092   192.0.76.3:https       ESTABLISHED       Internet
TCP    192.168.29.214:63096   192.0.78.22:https      TIME_WAIT         Not Applicable
TCP    192.168.29.214:63100   192.0.78.22:https      TIME_WAIT         Not Applicable
TCP    192.168.29.214:63101   192.0.78.22:https      TIME_WAIT         Not Applicable
TCP    192.168.29.214:63103   192.0.78.22:https      TIME_WAIT         Not Applicable
TCP    192.168.29.214:63104   192.0.78.22:https      TIME_WAIT         Not Applicable
```

5. Demonstration of Static and Dynamic IP Address Settings.

Ans :

Static Configuration

NetworkManager primarily handles network configuration. NetworkManager can be used in a GUI, TUI, or CLI environment.

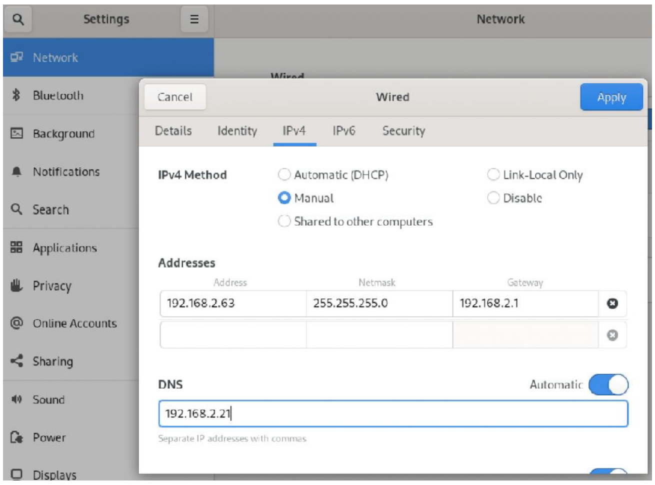
The nmcli process to set a static IP configuration is to create a connection profile and then set the values desired. Red Hat has documentation here.

Here is an example of creating a network connection named home-network with an IP address of 192.168.2.200/24, a default gateway of 192.168.2.1, and a name server of 8.8.8.8:

```
# nmcli connection add con-name home-network ifname enp7s0 type ethernet
# nmcli connection modify home-network ipv4.gateway 192.168.2.1
# nmcli connection modify home-network ipv4.addresses 192.168.2.200/24
# nmcli connection modify home-network ipv4.dns 8.8.8.8
# nmcli connection up home-network
```

The GUI configuration can be accomplished by selecting the Manual button and then filling in the blanks with the appropriate information.

Image



IPv4 Method ○ Automatic (DHCP) ● Manual ○ Shared to other computers ○ Link-Local Only ○ Disable

Addresses

Address	Netmask	Gateway
192.168.2.63	255.255.255.0	192.168.2.1

DNS Automatic

192.168.2.21

Separate IP addresses with commas

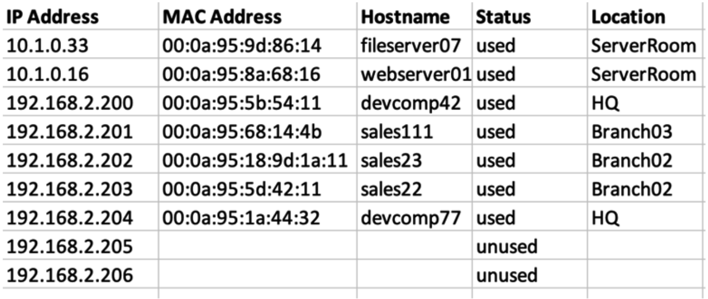
Figure 1: Static IP configuration in the NetworkManager GUI.

Recall that you can make no typographical errors when configuring IP addresses, and duplicate addresses will cause network connection problems.

Tracking IPs

It is essential to track your statically assigned IP addresses. Depending on the size of your environment, this might be so simple as a text document or a spreadsheet, all the way up to specialized software that integrates with directory services and DHCP. I find it's best to at least track IP address (and subnet mask), MAC address (not essential), hostname, role on the network (justifies why the devices have a static IP), and any additional notes.

Image



IP Address	MAC Address	Hostname	Status	Location
10.1.0.33	00:0a:95:9d:86:14	fileserver07	used	ServerRoom
10.1.0.16	00:0a:95:8a:68:16	webserver01	used	ServerRoom
192.168.2.200	00:0a:95:5b:54:11	devcomp42	used	HQ
192.168.2.201	00:0a:95:68:14:4b	sales111	used	Branch03
192.168.2.202	00:0a:95:18:9d:1a:11	sales23	used	Branch02
192.168.2.203	00:0a:95:5d:42:11	sales22	used	Branch02
192.168.2.204	00:0a:95:1a:44:32	devcomp77	used	HQ
192.168.2.205			unused	
192.168.2.206			unused	

Figure 2: Use a spreadsheet to track static IP address configurations.

Dynamic Configurations

The devices that require a static IP configuration are a relatively small percentage of your network. Most network devices are end-user systems such as workstations, laptops, phones, tablets, and other transient devices. In addition, these systems do not usually host network services that need to be discoverable by other computers.

IP address configurations are unforgiving when it comes to duplicates and typos. In addition, static IP address settings are fairly time-consuming. Finally, IP address settings tend to be temporary, especially with the advent of portable devices like laptops, phones, and tablets. To save time and reduce the chances of a mistake, dynamic IP address allocation is preferable for these kinds of nodes.

Linux systems are configured as DHCP clients by using NetworkManager.

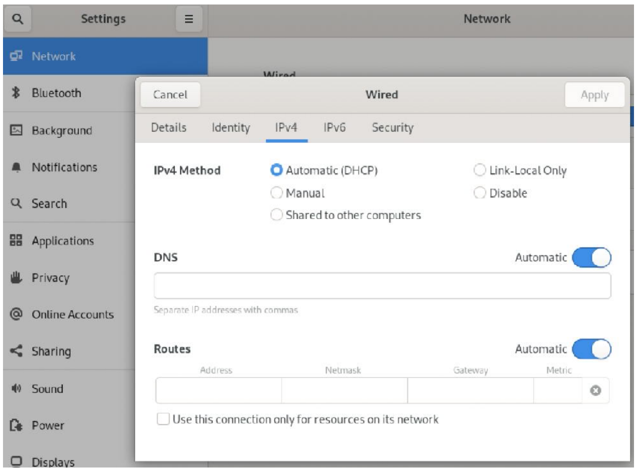
Here is an example of adding a network connection profile configured to lease an IP address from DHCP:

```
# nmcli connection add con-name home-network ifname enp7s0 type ethernet
```

By not specifying an address NetworkManager assumes the DHCP client role.

Here is a screenshot of a dynamic IP address configuration from the GUI:

Image



IPv4 Method ● Automatic (DHCP) ○ Link-Local Only ○ Manual ○ Disable ○ Shared to other computers

DNS Automatic ●

Separate IP addresses with commas

Routes Automatic ●

Address	Netmask	Gateway	Metric

□ Use this connection only for resources on its network

Figure 3: The NetworkManager GUI with a dynamic address configuration.

The dhclient command

The dhclient command is also used to manage dynamic IP address configurations. However, in RHEL 8, network configurations, including DHCP, are handled by NetworkManager. Older RHEL versions rely on dhclient, as do some other distributions.

```
# dhclient
```

The ip route command displays lease information.

6. Understanding Ethernet Cabling and Switched Networks.

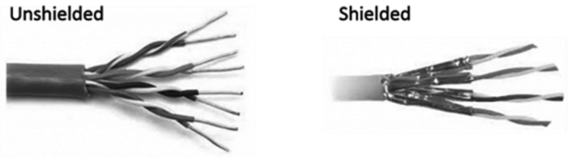
Ans :

Twisted Pair

Twisted Pair wiring is a type of cable which uses eight individual wires in a bundle. The eight individual wires are paired in sets of two, and each pair is twisted around each other. This creates four pairs of wire, each of which serve as a channel through which data can be transmitted.

The pairing of the wires is very important, and we will look at why later on in this article, but the short version is it helps negate and minimize the effects of Crosstalk and Electromagnetic Interference (EMI).

There are two prominent types of Twisted Pair wiring, a Shielded variant and an Unshielded variant:



Notice in both cases, the pairs of wire create four distinct channels or lanes through which data will be sent.

Unshielded Twisted Pair (UTP)

This is the more commonly deployed variation. There is no additional shielding against electromagnetic noise, but none the less, UTP can carry a signal reliably due to innate features of twisted pair wiring. We will explore these in more depth later on in this article.

UTP is less expensive, more (physically) resilient, and more flexible. These attributes typically make UTP the preferred choice.

Shielded Twisted Pair (STP)

STP has additional shielding around each pair of wires and then one more shield around all four pairs. This helps contain and isolate the electromagnetic noise that occurs when signals travel through a wire.

That said, if any part of the shielding is damaged, or if the wires aren't perfectly grounded on either side of the connection, the shielding can act as an antenna and introduce additional electromagnetic noise from stray radio waves and Wi-Fi signals in the air.

Ethernet

As was said before, Ethernet is a family of specifications that governs a few different things. One of those things are all the different wiring specifications: 10BASE-T, 100BASE-TX, 1000BASE-T, and so on.

100 BASE-T

The number at the beginning simply refers to the speed of the wire in Millions of bits per second, or more often referred to as Megabits per second (Mbps). A wire rated at 100 Mbps can theoretically transmit 100,000,000 bits per second, which equates to roughly 12.5 MegaBytes per second (MBps). Notice the capital B vs the lower case b to refer to Bytes vs bits.

100 BASE-T

The term base is short for baseband signaling. Its counterpart is broadband signaling. When these terms originated, the difference between them was baseband signaling sends digital signals across the medium, whereas broadband sends analog signals across the medium.

100 BASE-T

The "–T" stands for Twisted Pair. This is in contrast to other wiring standards like -2 and -5 which indicate Coaxial wiring with maximum ranges of 200~ and 500 meters, or -SR and -LR which are Short Range and Long Range Fiber Optic wiring standards.

100 BASE-T4

100BASE-T4 uses all four pairs in the bundle (all eight wires). One pair is used solely for Transmitting signals (TX). One pair is used solely for Receiving signals (RX). The remaining two pairs can be used for either RX or TX, and it's up to both sides of the wire to negotiate which of the remaining pairs are used for what.

T4 is one of the earlier specifications for Twisted Pair, and doesn't see much modern use due to unnecessary complexity in the design for very little gain over the 100BASE-TX iteration described next.

100BASE-TX

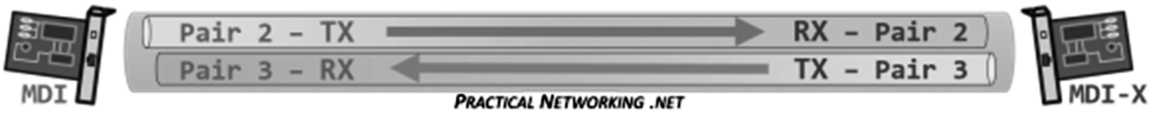
100BASE-TX uses only two pairs, one dedicated to TX, and the other dedicated to RX. The other two pairs on the wire are unused. You could very well construct a 100BASE-TX wire which only had 4 of the 8 wires in the correct pin-positions (1,2,3,6), but often the other four wires are still included mostly as place holders for the remaining pin-positions, as well as for future compatibility.

Why Crossover

The 100BASE-TX and 10BASE-T specifications both call for 8 wires in a twisted pair cable to be grouped into four pairs.

Of the four pairs, only two will actually be used: pair 2 and pair 3. Each individual wire in the pair is a simplex medium, which means the signal can only ever cross any one wire in one direction.

In order to attain full-duplex communication, some wires are permanently set aside for communication in one direction, and the other wires are permanently set aside for communication in the opposite direction.



The configuration of the Network Interface Card (NIC) will determine which pair is used to transmit and which pair is used to receive.

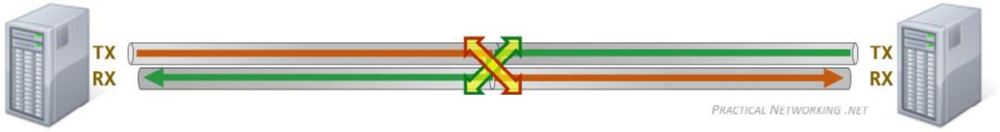
A NIC that transmits (TX) signals over pair 2 (pin 1&2) and receives (RX) signals over pair 3 (pin 3&6) is called a Media Dependent Interface (MDI) NIC. While a NIC that does the opposite (TX on pair 3, and RX on pair 2) is called a Media Dependent Interface Crossover (MDI-X).

PC to PC

A PC uses an MDI NIC, which means PCs always transmit on pair 2, and receive on pair 3. But if two PCs connected directly to each other are both trying to transmit over pair 2, it would lead to a collision of their signals. And worse, neither PC would receive anything on pair 3.

As a result, the pin-pairs need to be crossed on the wire, so that what is sent from one PC on pair 2, arrives on the other PC on pair 3, and vice versa.

Here is a simplified illustration (the colors below are irrelevant, they simply indicate two different paths, for two different directions of the communication):



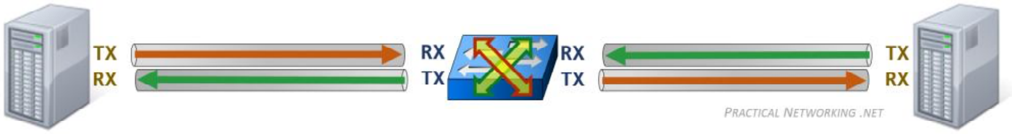
Notice both PCs can transmit signals through a dedicate channel, and due to the cross of the pairs in the wire (represented by the giant X), both PCs can receive what the other transmitted from a dedicated channel.

Hence, a connection from a PC directly to another PC requires a crossover cable.

PC to Switch to PC

A switch is a device that is meant to facilitate communication between two PCs on the same network. To that end, a switch NIC uses the MDI-X specification, which means a switch always transmits on pair 3, and receives on pair 2 (the exact inverse of an MDI NIC on a PC).

This causes the switch to have a built-in crossover function. The wire doesn't need to cross the pairs, because the switch will take care of it:



As you can see, a PC connected to a switch can simply use a straight-through cable, and let the Switch deal with crossing the pairs. The end to end path remains consistent: every device is transmitting on its TX ports, and receiving on its RX ports.

Routers and Hubs

But what of routers and hubs? What type of NIC do they use?

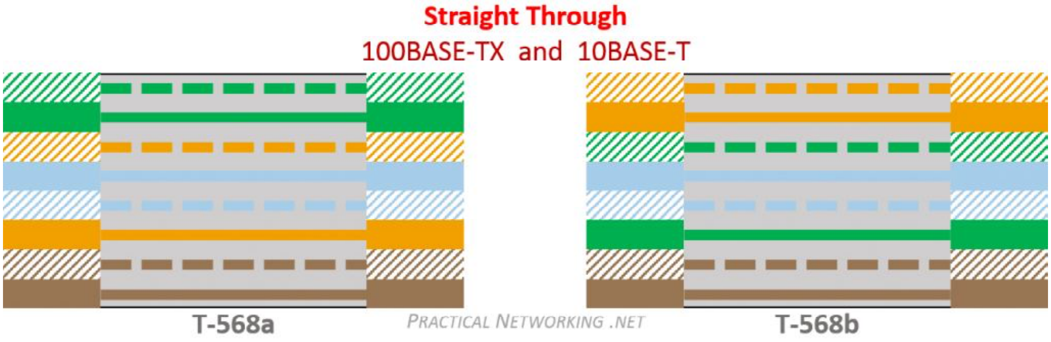
It turns out, a Router, like a PC, uses the MDI specification – TX on pair 2, and RX on pair 3. As such, you can replace any picture of a PC in any of the illustrations above with a Router, and can easily determine which connections would require a straight-through cable and which would require a crossover cable.

Furthermore, a Hub's ports use the MDI-X specification – TX on pair 3, and RX on pair 2. You can replace any picture of a switch above with a Hub and can also easily determine what cables are required.

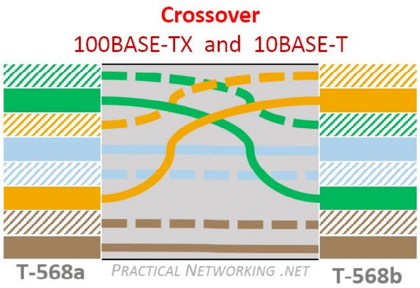
Ethernet Cable Wiring Diagram

Recall that there are two standards for the colors in the RJ45 specification: T568a and T568b. The standard being utilized on either side of a Twisted Pair wire is what determines whether the cable is straight-through or crossover.

To make a Straight-through cable, simply order the wires on both sides of the cable to one specification (either both T568a or both T568b):



To make a Crossover cable, simply use one standard on one side, and the other standard on the opposite side:



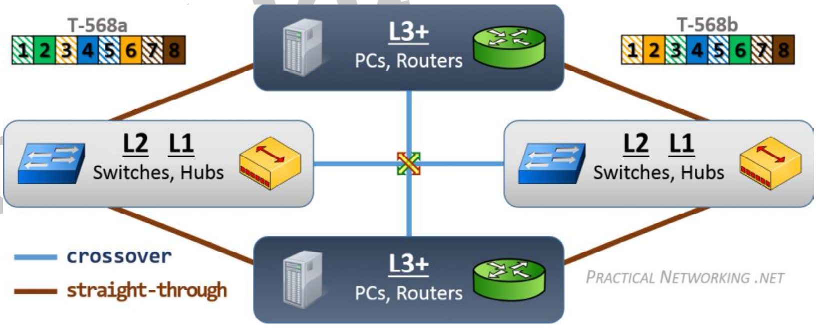
Note that wire pair 1 and pair 4 are not used (the blue and brown wires). You could, theoretically not include the wires in the cable at all, but this would make keeping the remaining wires in the proper order rather difficult.

Moreover, since they are not used, they do not need to be crossed in a crossover cable. However, the Gigabit specification does require using all 8 wires, and often all pairs are crossed for consistency. We will discuss Gigabit Ethernet later in this article.

And lastly, remember that the signal doesn't really care what color the wire is. As long as the correct pins are connected to each other, communication will work. You could use all green wires, and as long as Pins 1&2 are connected to Pins 3&6 on the other side (and vice versa), you would have a fully functioning cross-over wire. But just because it works, doesn't mean it is a good idea – such a cable would be a nightmare to maintain.

Easy Memorization Chart

We can aggregate everything we learned above regarding crossover wires and straight-through wires into a simple chart:



A benefit to how the graphic above is displayed is that it makes it very easy to sketch out. Simply draw L2/ L1 on the left and right, and L3+ on top and bottom and connect everything to each other. The lines that cross each other require a crossover cable when connecting devices that operate at those layers of the OSI model. The lines that connect straight to each other require a straight-through cable.

In summary:

An L1 or L2 device connected to another L1 or L2 device requires a crossover cable.

An L1 or L2 device connected to a L3+ device requires a straight-through cable.

An L3+ device connected to another L3+ device requires a crossover cable.

Or even simpler:

Like devices require a crossover cable.

Unlike devices require a straight-through cable.

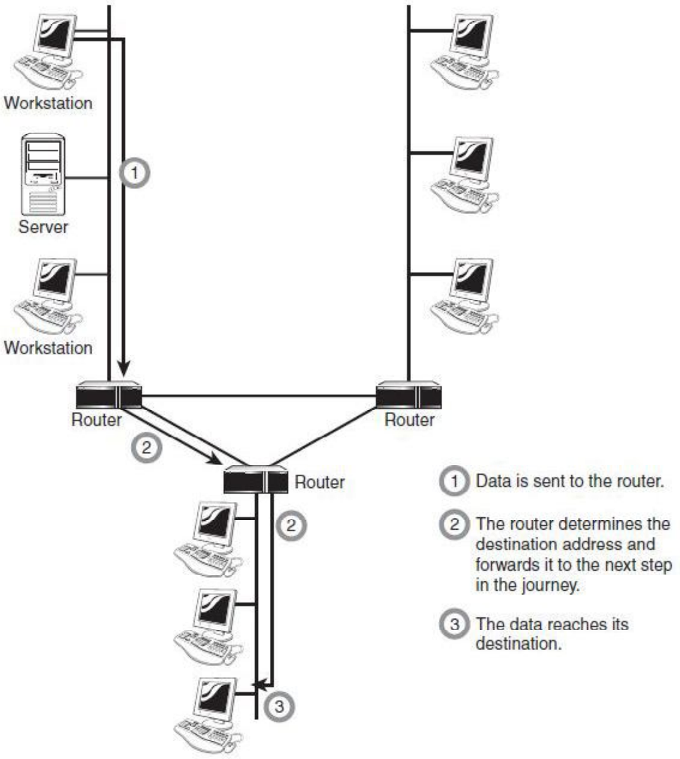
7. Comprehension of Routers and Firewalls.

Ans :

Routers

Routers are an increasingly common sight in any network environment, from a small home office that uses one to connect to an Internet service provider (ISP) to a corporate IT environment where racks of routers manage data communication with disparate remote sites. Routers make internetworking possible, and in view of this, they warrant detailed attention.

Routers are network devices that literally route data around the network. By examining data as it arrives, the router can determine the destination address for the data; then, by using tables of defined routes, the router determines the best way for the data to continue its journey. Unlike bridges and switches, which use the hardware-configured MAC address to determine the destination of the data, routers use the software-configured network address to make decisions. This approach makes routers more functional than bridges or switches, and it also makes them more complex because they have to work harder to determine the information.


Workstation
Server
Workstation
1
Router
Router
2
Router
2
3
1 Data is sent to the router.
2 The router determines the destination address and forwards it to the next step in the journey.
3 The data reaches its destination.

The basic requirement for a router is that it must have at least two network interfaces. If they are LAN interfaces, the router can manage and route the information between two LAN segments. More commonly, a router is used to provide connectivity across wide area network (WAN) links.

A router derives can route data it receives from one network onto another. When a router receives a packet of data, it reads the header of the packet to determine the destination address. Once it has determined the address, it looks in its routing table to determine whether it knows how to reach the destination and, if it does, it forwards the packet to the next hop on the route. The next hop might be the final destination, or it might be another router.

A routing tables play a very important role in the routing process. They are the means by which the router makes its decisions. For this reason, a routing table needs to be two things. It must be up-to-date, and it must be complete.

There are two ways that the router can get the information for the routing table

1. Static routing
2. Dynamic routing

Static Routing

In environments that use static routing, routes and route information are entered into the routing tables manually. Not only can this be a time-consuming task, but also errors are more common. Additionally, when there is a change in the layout, or topology, of the network, statically configured routers must be manually updated with the changes. Again, this is a time consuming and potentially error-laden task. For these reasons, static routing is suited to only the smallest environments with perhaps just one or two routers. A far more practical solution, particularly in larger environments, is to use dynamic routing.

Dynamic Routing

In a dynamic routing environment, routers use special routing protocols to communicate. The purpose of these protocols is simple; they enable routers to pass on information about themselves to other routers so that other routers can build routing tables. There are two types of routing protocols used—the older distance vector protocols and the newer link state protocols.

Firewalls

A firewall is a networking device, either hardware or software based, that controls access to your organization's network. This controlled access is designed to protect data and resources from an outside threat. To do this, firewalls are typically placed at entry/exit points of a network—for example, placing a firewall between an internal network and the Internet. Once there, it can control access in and out of that point.

Although firewalls typically protect internal networks from public networks, they are also used to control access between specific network segments within a network—for example, placing a firewall between the Accounts and the Sales departments.

As mentioned, firewalls can be implemented through software or through a dedicated hardware device. Organizations implement software firewalls through network operating systems (NOS) such as Linux/UNIX, Windows servers, and Mac OS servers. The firewall is configured on the server to allow or permit certain types of network traffic. In small offices and for regular home use, a firewall is commonly installed on the local system and configured to control traffic. Many third-party firewalls are available.

Hardware firewalls are used in networks of all sizes today. Hardware firewalls are often dedicated network devices that can be implemented with very little configuration and protect all systems behind the firewall from outside sources. Hardware firewalls are readily available and often combined with other devices today. For example, many broadband routers and wireless access points have firewall functionality built in. In such case, the router or WAP might have a number of ports available to plug systems in to.

8. Significance of DNS.

What is Domain Name System (DNS) Server?

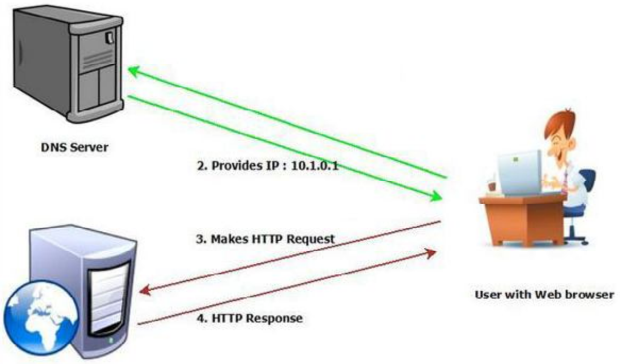
Ans :

According to the senior officer of the ministry of electronics and IT (MeitY), DNS is a system that translates domain names to Internet Protocol or IP addresses that allows browsers to load websites sought. No doubt, it is an important tool that requires to be fool-proof and has a major role in browsing the Internet.

DNS is a database that stores all the names of domain and their corresponding IP numbers for a particular top-level domain (TLD) like .com or .net. It will identify and locate the computer systems and resources on the internet.

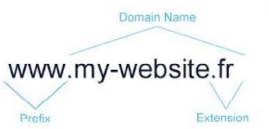
According to IT ministry official, the main aim of bringing our own public DNS is to ensure availability, particularly for smaller Interest Service Providers (ISPs) who don't have credible DNS. Bigger ones usually have their own DNS.

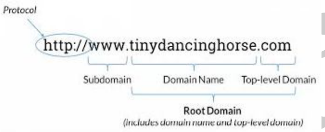
How DNS server work?



The directory of DNS that matches name to the numbers is not located all in one place in some dark corner of the internet. Just like the internet, the directory is distributed around the world which is stored on domain name servers that all communicate with each other on a very regular basis to provide updates and redundancies

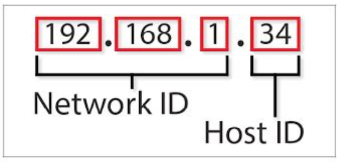
Do you know what is domain name and IP address?

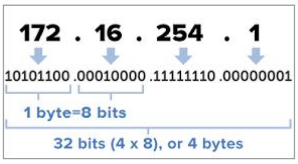



Protocol
http://www.tinydancinghorse.com
Subdomain
Domain Name
Top-level Domain
Root Domain
(includes domain name and top-level domain)

Let us take a domain name, www.fb.com. The naming convention begins from right to left and vice-versa for IP address. In the domain name for face book, first the DNS will check for com which stands for the commercial domain. Further, Face book is a sub-domain to com and subsequently, www is a sub domain to face book. The dot (.) separates the domains from their sub domains. Let us tell you that the full domain name can only consist of 253 characters.

What is Black Box? How does it Work?





If you want to find out the domain name registered against an IP address, then he will request the DNS server with the IP Address of the website. Suppose the IP address sent is 31.13.79.246 then DNS will first check 31, then 13 then 79 and then finally 246 and in this way will tell you that the IP address belongs to www.fb.com. There are different tree structures in the database like here 31 belongs to the top position of the tree and is primary domain in the hierarchy, address 13, 79, 246 are consecutive sub-domains. The 246 number is the server machine hosting the website www.fb.com.

Significance of DNS server

It will enhance security to discourage cyber attacks and a quicker site loading time. If a user inadvertently accesses a malicious or phishing site, the new public system would immediately open up a page or popup to alert the user of such potential threat so that the suspicious resource could be avoided, the official who is aware of the initiative. The security features are created by the National Informatics Centre (NIC) and has a capability to host as many as 5 million users that can be scaled up further if needed.

No doubt, DNS is an important tool that requires to be fool-proof and has a major role in browsing the internet. The new DNS will be placed across the country to minimise outage and would be available round the clock. Users simply can use by typing the IP number in to the internet browser.

NETWORK PROGRAMMING

1. Implement IPC Using A) Pipes B) FIFO

Ans :

```
// C program to demonstrate use of fork() and pipe()
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

int main()
{
        // We use two pipes
        // First pipe to send input string from parent
        // Second pipe to send concatenated string from child

        int fd1[2]; // Used to store two ends of first pipe
        int fd2[2]; // Used to store two ends of second pipe

        char fixed_str[] = “forgeeks.org”;
        char input_str[100];
        pid_t p;

        if (pipe(fd1) == -1) {
                fprintf(stderr, “Pipe Failed”);
                return 1;
        }
        if (pipe(fd2) == -1) {
                fprintf(stderr, “Pipe Failed”);
                return 1;
        }

        scanf(“%s”, input_str);
        p = fork();

        if (p < 0) {
                fprintf(stderr, “fork Failed”);
                return 1;
        }
```

```
// Parent process
else if (p > 0) {
        char concat_str[100];

        close(fd1[0]); // Close reading end of first pipe

        // Write input string and close writing end of first
        // pipe.
        write(fd1[1], input_str, strlen(input_str) + 1);
        close(fd1[1]);

        // Wait for child to send a string
        wait(NULL);

        close(fd2[1]); // Close writing end of second pipe

        // Read string from child, print it and close
        // reading end.
        read(fd2[0], concat_str, 100);
        printf(“Concatenated string %s\n”, concat_str);
        close(fd2[0]);
}

// child process
else {
        close(fd1[1]); // Close writing end of first pipe

        // Read a string using first pipe
        char concat_str[100];
        read(fd1[0], concat_str, 100);

        // Concatenate a fixed string with it
        int k = strlen(concat_str);
        int i;
        for (i = 0; i < strlen(fixed_str); i++)
                concat_str[k++] = fixed_str[i];

        concat_str[k] = ‘\0’; // string ends with ‘\0’

        // Close both reading ends
        close(fd1[0]);
        close(fd2[0]);
```

```
            // Write concatenated string and close writing end
            write(fd2[1], concat_str, strlen(concat_str) + 1);
            close(fd2[1]);

            exit(0);
      }
}
```

Output:

```
Concatenated string www.geeksforgeeks.org
```

```
// C program to implement one side of FIFO
// This side writes first, then reads
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    int fd;

   // FIFO file path
   char * myfifo = "/tmp/myfifo";

   // Creating the named file(FIFO)
   // mkfifo(<pathname>, <permission>)
   mkfifo(myfifo, 0666);

   char arr1[80], arr2[80];
   while (1)
    {
      // Open FIFO for write only
      fd = open(myfifo, O_WRONLY);

      // Take an input arr2ing from user.
      // 80 is maximum length
      fgets(arr2, 80, stdin);

      // Write the input arr2ing on FIFO
       // and close it
      write(fd, arr2, strlen(arr2)+1);
       close(fd);
```

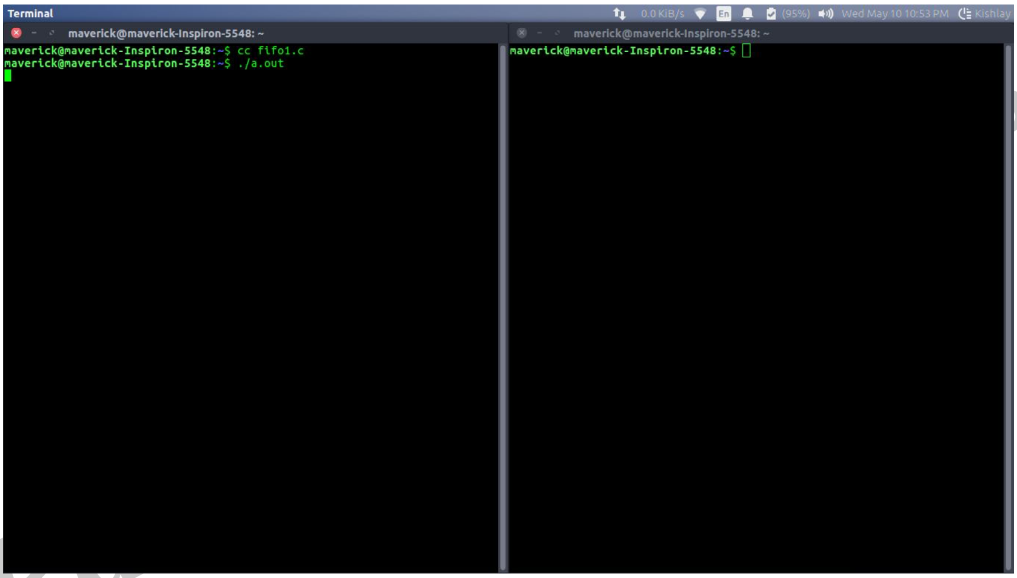
```
        // Open FIFO for Read only
        fd = open(myfifo, O_RDONLY);

        // Read from FIFO
        read(fd, arr1, sizeof(arr1));

        // Print the read message
        printf(“User2: %s\n”, arr1);
         close(fd);
     }
    return 0;
}
```

Program 2(Reads First)

```
// C program to implement one side of FIFO
// This side reads first, then reads
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    int fd1;

    // FIFO file path
    char * myfifo = “/tmp/myfifo”;

    // Creating the named file(FIFO)
    // mkfifo(<pathname>,<permission>)
    mkfifo(myfifo, 0666);

    char str1[80], str2[80];
    while (1)
     {
        // First open in read only and read
        fd1 = open(myfifo,O_RDONLY);
         read(fd1, str1, 80);

        // Print the read string and close
        printf(“User1: %s\n”, str1);
         close(fd1);
        // Now open in write mode and write
        // string taken from user.
        fd1 = open(myfifo,O_WRONLY);
        fgets(str2, 80, stdin);
        write(fd1, str2, strlen(str2)+1);
        close(fd1);
    }
    return 0;
}
```

Output:

Run the two programs simultaneously on two terminals.


Terminal
0.0 KiB/s
En
(95%)
Wed May 10 10:53 PM
Kishlay
maverick@maverick-Inspiron-5548: ~
maverick@maverick-Inspiron-5548: ~

```
maverick@maverick-Inspiron-5548:~$ cc fifo1.c
maverick@maverick-Inspiron-5548:~$ ./a.out
```

```
maverick@maverick-Inspiron-5548:~$
```

2. Implement File Transfer Using Message Queue Form of IPC

Ans :

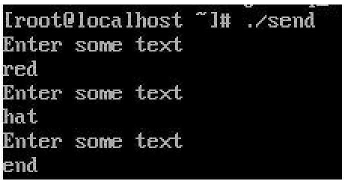
Let's write a program for IPC using Message Queues to send data to a message queue.

```
#include<stdlib.h>
 #include<stdio.h>
 #include<string.h>
 #include<unistd.h>
 #include<sys/types.h>
 #include<sys/ipc.h>
 #include<sys/msg.h>
 #define MAX_TEXT 512     //maximum length of the message that can be sent allowed
 struct my_msg{
          long int msg_type;
          char some_text[MAX_TEXT];
 };
```

```
int main()
{
        int running=1;
        int msgid;
        struct my_msg some_data;
        char buffer[50]; //array to store user input
        msgid=msgget((key_t)14534,0666|IPC_CREAT);
        if (msgid == -1) // -1 means the message queue is not created
        {
                printf("Error in creating queue\n");
                exit(0);
        }

        while(running)
        {
                printf("Enter some text:\n");
                fgets(buffer,50,stdin);
                some_data.msg_type=1;
                strcpy(some_data.some_text,buffer);
                 if(msgsnd(msgid,(void *)&some_data, MAX_TEXT,0)==-1) // msgsnd returns -
1 if the message is not sent
                {
                        printf("Msg not sent\n");
                }
                if(strncmp(buffer,"end",3)==0)
                {
                        running=0;
                }
        }
}
```

Output

The above program gives this output.



```
[root@localhost ~]# ./send
Enter some text
red
Enter some text
hat
Enter some text
end
```

3. Design TCP Iterative Client and Server Application to Reverse the Given Input Sentence.

Ans :

Server Program

```
#include<string.h>
#include<stdio.h>
```

```
#include<stdlib.h>
#include<unistd.h>
#include<sys/socket.h>
#include<sys/types.h>
#define MAXLINE 20
#define SERV_PORT 5777
main(int argc,char *argv)
{
int i,j;
ssize_t n;
char line[MAXLINE];
char revline[MAXLINE];
int listenfd,connfd,clilen;
struct sockaddr_in servaddr,cliaddr;
listenfd=socket(AF_INET,SOCK_STREAM,0);
bzero(&servaddr,sizeof(servaddr));
servaddr.sin_family=AF_INET; servaddr.sin_port=htons(SERV_PORT);
bind(listenfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
listen(listenfd,1);
for( ; ; )
{
clilen=sizeof(cliaddr);
connfd=accept(listenfd,(struct sockaddr*)&cliaddr,&clilen);
printf(“connect to client”);
while(1)
{
if((n=read(connfd,line,MAXLINE))==0)
break;
line[n-1]=’\0’;
j=0;
for(i=n-2;i>=0;i—)
revline[j++]=line[i];
revline[j]=’\0’;
write(connfd,revline,n);
}
}
}
```

Client Program

```
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/socket.h>
```

```
#include<netinet/in.h>
#include<sys/types.h>
#define MAXLINE 20
#define SERV_PORT 5777
main(int argc,char *argv)
 {
char sendline[MAXLINE],revline[MAXLINE];
 int sockfd; struct sockaddr_in servaddr;
sockfd=socket(AF_INET,SOCK_STREAM,0);
bzero(&servaddr,sizeof(servaddr));
 servaddr.sin_family=AF_INET;
servaddr.sin_port=ntohs(SERV_PORT);
connect(sockfd,(struct sockaddr*)&servaddr,sizeof(servaddr));
printf("\n enter the data to be send");
while(fgets(sendline,MAXLINE,stdin)!=NULL)
{ write(sockfd,sendline,strlen(sendline));
printf("\n line send");
 read(sockfd,revline,MAXLINE); printf("\n reverse of the given sentence is : %s",revline); printf("\n"); }
exit(0); }
Output
enter the data to be send hellow
 line send reverse of the given sentence is : wolleh
```

4. Design TCP Concurrent Client and Server Application to Reverse the Given Input Sentence.

Ans :

Client.c

```
// C client code to send string to reverse
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>

#define PORT 8090

// Driver code
int main()
{
        struct sockaddr_in address;
        int sock = 0, valread;
        struct sockaddr_in serv_addr;
        char str[100];
```

```
printf("\nInput the string:");
scanf("%[^\n]s", str);

char buffer[1024] = { 0 };

// Creating socket file descriptor
if ((sock = socket(AF_INET,
                          SOCK_STREAM, 0))
     < 0) {
     printf("\n Socket creation error \n");
     return -1;
}

memset(&serv_addr, '0', sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(PORT);

// Convert IPv4 and IPv6 addresses from
// text to binary form 127.0.0.1 is local
// host IP address, this address should be
// your system local host IP address
if (inet_pton(AF_INET, "127.0.0.1",
                    &serv_addr.sin_addr)
     <= 0) {
     printf("\nAddress not supported \n");
     return -1;
}

// connect the socket
if (connect(sock, (struct sockaddr*)&serv_addr,
                 sizeof(serv_addr))
     < 0) {
     printf("\nConnection Failed \n");
     return -1;
}

int l = strlen(str);

// send string to server side
send(sock, str, sizeof(str), 0);

// read string sent by server
valread = read(sock, str, l);
```

```
		printf("%s\n", str);

		return 0;
	}
```

Server.c

```
	// Server C code to reverse a
	// string by sent from client
	#include <netinet/in.h>
	#include <stdio.h>
	#include <stdlib.h>
	#include <string.h>
	#include <sys/socket.h>
	#include <unistd.h>

	#define PORT 8090

	// Driver code
	int main()
	{
		int server_fd, new_socket, valread;
		struct sockaddr_in address;
		char str[100];
		int addrlen = sizeof(address);
		char buffer[1024] = { 0 };
		char* hello = "Hello from server";

		// Creating socket file descriptor
		if ((server_fd = socket(AF_INET,
						SOCK_STREAM, 0)) = = 0) {
			perror("socket failed");
			exit(EXIT_FAILURE);
		}

		address.sin_family = AF_INET;
		address.sin_addr.s_addr = INADDR_ANY;
		address.sin_port = htons(PORT);

		// Forcefully attaching socket to
		// the port 8090
		if (bind(server_fd, (struct sockaddr*)&address,
						sizeof(address)) < 0) {
            perror(“bind failed”);
            exit(EXIT_FAILURE);
        }

        // puts the server socket in passive mode
        if (listen(server_fd, 3) < 0) {
            perror(“listen”);
            exit(EXIT_FAILURE);
        }
        if ((new_socket = accept(server_fd,
                          (struct sockaddr*)&address,
                          (socklen_t*)&addrlen)) < 0) {
            perror(“accept”);
            exit(EXIT_FAILURE);
        }

        // read string send by client
        valread = read(new_socket, str,
                           sizeof(str));
        int i, j, temp;
        int l = strlen(str);

        printf(“\nString sent by client:%s\n”, str);

        // loop to reverse the string
        for (i = 0, j = l - 1; i < j; i++, j—) {
            temp = str[i];
            str[i] = str[j];
            str[j] = temp;
        }

        // send reversed string to client
        // by send system call
        send(new_socket, str, sizeof(str), 0);
        printf(“\nModified string sent to client\n”);

        return 0;
    }
```

Output

Input : welcome

Output :emoclew

5. Design TCP Client and Server Application to Transfer File.

Ans :

Client

The client performs the following functions.

1. Start the program
2. Declare the variables and structures required.
3. A socket is created and the connect function is executed.
4. The file is opened.
5. The data from the file is read and sent to the server.
6. The socket is closed.
7. The program is stopped.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <arpa/inet.h>
#define SIZE 1024

voidsend_file(FILE*fp, intsockfd){
  intn;
  chardata[SIZE] = {0};

  while(fgets(data, SIZE, fp) != NULL) {
    if(send(sockfd, data, sizeof(data), 0) == -1) {
      perror(“[-]Error in sending file.”);
      exit(1);
    }
    bzero(data, SIZE);
  }
}

intmain(){
  char*ip = “127.0.0.1”;
  intport = 8080;
  inte;

  intsockfd;
  structsockaddr_in server_addr;
  FILE*fp;
  char*filename = “send.txt”;
```

```
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if(sockfd < 0) {
        perror(“[-]Error in socket”);
        exit(1);
    }
    printf(“[+]Server socket created successfully.\n”);

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = port;
    server_addr.sin_addr.s_addr = inet_addr(ip);

    e = connect(sockfd, (structsockaddr*)&server_addr, sizeof(server_addr));
    if(e == -1) {
        perror(“[-]Error in socket”);
        exit(1);
    }
   printf(“[+]Connected to Server.\n”);

    fp = fopen(filename, “r”);
    if(fp == NULL) {
        perror(“[-]Error in reading file.”);
        exit(1);
    }

    send_file(fp, sockfd);
    printf(“[+]File data sent successfully.\n”);

    printf(“[+]Closing the connection.\n”);
    close(sockfd);

    return0;
}
```

Server

The server performs the following functions.

1. Start the program.
2. Declare the variables and structures required.
3. The socket is created using the socket function.
4. The socket is binded to the specific port.
5. Start listening for the connections.
6. Accept the connection from the client.
7. Create a new file.
8. Receives the data from the client.
9. Write the data into the file.
10. The program is stopped.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <arpa/inet.h>
#define SIZE 1024

voidwrite_file(intsockfd){
    intn;
    FILE*fp;
    char*filename = "recv.txt";
    charbuffer[SIZE];

    fp = fopen(filename, "w");
    while(1) {
        n = recv(sockfd, buffer, SIZE, 0);
        if(n <= 0){
            break;
            return;
        }
        fprintf(fp, "%s", buffer);
        bzero(buffer, SIZE);
    }
    return;
}

intmain(){
    char*ip = "127.0.0.1";
    intport = 8080;
    inte;

    intsockfd, new_sock;
    structsockaddr_in server_addr, new_addr;
    socklen_t addr_size;
    charbuffer[SIZE];

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if(sockfd < 0) {
        perror("[-]Error in socket");
        exit(1);
    }
    printf("[+]Server socket created successfully.\n");

    server_addr.sin_family = AF_INET;
    server_addr.sin_port = port;
    server_addr.sin_addr.s_addr = inet_addr(ip);

    e = bind(sockfd, (structsockaddr*)&server_addr, sizeof(server_addr));
    if(e < 0) {
        perror(“[-]Error in bind”);
        exit(1);
    }
    printf(“[+]Binding successfull.\n”);

    if(listen(sockfd, 10) == 0){
  printf(“[+]Listening....\n”);
  }else{
  perror(“[-]Error in listening”);
        exit(1);
  }

    addr_size = sizeof(new_addr);
    new_sock = accept(sockfd, (structsockaddr*)&new_addr, &addr_size);
    write_file(new_sock);
    printf(“[+]Data written in the file successfully.\n”);

    return0;
}
```

6. Design UDP Client and Server Application to Reverse the Given Input Sentence.

Ans :

```
server
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>

#define S_PORT 43454
#define C_PORT 43455
#define ERROR -1
#define IP_STR “127.0.0.1”

void strrev(char *str, int len) {
    int i, j;
    char temp;
    for (i = 0, j = len -1; i < j; + +i, —j) {
        temp = str[i];
        str[i] = str[j];
        str[j] = temp;
    }
}

int main(int argc, char const *argv[]) {
        int sfd, len;
        char *str_buf;
        struct sockaddr_in servaddr, clientaddr;
        sfd = socket(AF_INET, SOCK_DGRAM,IPPROTO_UDP);
        if (sfd = = ERROR) {
                perror("Could not open a socket");
                return 1;
        }
        memset((char *) &servaddr, 0, sizeof(servaddr));
        servaddr.sin_family=AF_INET;
        servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
        servaddr.sin_port=htons(S_PORT);

        memset((char *) &clientaddr, 0, sizeof(clientaddr));
        clientaddr.sin_family=AF_INET;
        clientaddr.sin_addr.s_addr=inet_addr(IP_STR);
        clientaddr.sin_port=htons(C_PORT);

        if((bind(sfd,(struct sockaddr *)&servaddr,sizeof(servaddr)))!=0) {
                perror("Could not bind socket");
                return 2;
        }

        printf("Server is running on %s:%d\n", IP_STR, S_PORT);
        while(1) {
                recvfrom(sfd, &len, sizeof(len), 0, (struct sockaddr *)&clientaddr, (socklen_t *)&clientaddr);
                str_buf = (char *) malloc(len*sizeof(char));
                recvfrom(sfd, str_buf, len, 0, (struct sockaddr *)&clientaddr, (socklen_t *)&clientaddr);
                printf("Client at %s:%d said: %s\t", inet_ntoa(clientaddr.sin_addr), ntohs(clientaddr.sin_port),
                str_buf);
                strrev(str_buf,len);
                sendto(sfd, &len, sizeof(len), 0, (struct sockaddr *)&clientaddr, sizeof(clientaddr));
                sendto(sfd, str_buf, len, 0, (struct sockaddr *)&clientaddr, sizeof(clientaddr));
                printf("The reverse is: %s\n", str_buf);
                free(str_buf);
        }
        return 0;
}
```

Client

```
*/
     #include <sys/socket.h>
     #include <netdb.h>
     #include <string.h>
     #include <stdlib.h>
     #include <netinet/in.h>
     #include <arpa/inet.h>
     #include <unistd.h>
     #include <stdio.h>
     #include <string.h>

     #define S_PORT 43454
     #define C_PORT 43455
     #define ERROR -1
     #define IP_STR “127.0.0.1”

     int main(int argc, char const *argv[]) {
          int sfd, len;
          char str_buf[2048];
          struct sockaddr_in servaddr, clientaddr;
          socklen_t addrlen;
          sfd = socket(AF_INET, SOCK_DGRAM,IPPROTO_UDP);
          if (sfd = = ERROR) {
               perror(“Could not open a socket”);
               return 1;
          }
          memset((char *) &servaddr, 0, sizeof(servaddr));
          servaddr.sin_family=AF_INET;
          servaddr.sin_addr.s_addr=inet_addr(IP_STR);
          servaddr.sin_port=htons(S_PORT);

          memset((char *) &clientaddr, 0, sizeof(clientaddr));
          clientaddr.sin_family=AF_INET;
          clientaddr.sin_addr.s_addr=inet_addr(IP_STR);
          clientaddr.sin_port=htons(C_PORT);

          if((bind(sfd,(struct sockaddr *)&clientaddr,sizeof(clientaddr)))!=0) {
               perror(“Could not bind socket”);
               return 2;
          }
```

```
        printf(“Client is running on %s:%d\n”, IP_STR, C_PORT);
        printf(“Enter a string: “);
        scanf(“%[^\n]%*c”,str_buf);
        len = strlen(str_buf);
        sendto(sfd, &len, sizeof(len), 0, (struct sockaddr *)&servaddr, sizeof(servaddr));
        sendto(sfd, str_buf, len, 0, (struct sockaddr *)&servaddr, sizeof(servaddr));
        addrlen = sizeof(clientaddr);
        recvfrom(sfd, &len, sizeof(len), 0, (struct sockaddr *)&clientaddr, &addrlen);
        recvfrom(sfd, str_buf, len, 0, (struct sockaddr *)&clientaddr, &addrlen);
        printf(“Server Replied: %s\n”, str_buf);

        return 0;
}
```

7. Design UDP Client and Server Application to Reverse the Given Input Sentence.

Ans :

server

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>

#define S_PORT 43454
#define C_PORT 43455
#define ERROR -1
#define IP_STR “127.0.0.1”

void strrev(char *str, int len) {
    int i, j;
    char temp;
    for (i = 0, j = len -1; i < j; ++i, —j) {
        temp = str[i];
        str[i] = str[j];
        str[j] = temp;
    }
}

int main(int argc, char const *argv[]) {
        int sfd, len;
        char *str_buf;
        struct sockaddr_in servaddr, clientaddr;
        sfd = socket(AF_INET, SOCK_DGRAM,IPPROTO_UDP);
        if (sfd = = ERROR) {
            perror(“Could not open a socket”);
            return 1;
        }
        memset((char *) &servaddr, 0, sizeof(servaddr));
        servaddr.sin_family=AF_INET;
        servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
        servaddr.sin_port=htons(S_PORT);

        memset((char *) &clientaddr, 0, sizeof(clientaddr));
        clientaddr.sin_family=AF_INET;
        clientaddr.sin_addr.s_addr=inet_addr(IP_STR);
        clientaddr.sin_port=htons(C_PORT);

        if((bind(sfd,(struct sockaddr *)&servaddr,sizeof(servaddr)))!=0) {
            perror(“Could not bind socket”);
            return 2;
        }

        printf(“Server is running on %s:%d\n”, IP_STR, S_PORT);
        while(1) {
            recvfrom(sfd, &len, sizeof(len), 0, (struct sockaddr *)&clientaddr, (socklen_t *)&clientaddr);
            str_buf = (char *) malloc(len*sizeof(char));
            recvfrom(sfd, str_buf, len, 0, (struct sockaddr *)&clientaddr, (socklen_t *)&clientaddr);
            printf(“Client at %s:%d said: %s\t”, inet_ntoa(clientaddr.sin_addr), ntohs(clientaddr.sin_port),
            str_buf);
            strrev(str_buf,len);
            sendto(sfd, &len, sizeof(len), 0, (struct sockaddr *)&clientaddr, sizeof(clientaddr));
            sendto(sfd, str_buf, len, 0, (struct sockaddr *)&clientaddr, sizeof(clientaddr));
            printf(“The reverse is: %s\n”, str_buf);
            free(str_buf);
        }
        return 0;
    }
```

Client

```
*/
    #include <sys/socket.h>
    #include <netdb.h>
    #include <string.h>
    #include <stdlib.h>
    #include <netinet/in.h>
    #include <arpa/inet.h>
```

```
#include <unistd.h>
#include <stdio.h>
#include <string.h>

#define S_PORT 43454
#define C_PORT 43455
#define ERROR -1
#define IP_STR “127.0.0.1”

int main(int argc, char const *argv[]) {
        int sfd, len;
        char str_buf[2048];
        struct sockaddr_in servaddr, clientaddr;
        socklen_t addrlen;
        sfd = socket(AF_INET, SOCK_DGRAM,IPPROTO_UDP);
        if (sfd = = ERROR) {
                perror(“Could not open a socket”);
                return 1;
        }
        memset((char *) &servaddr, 0, sizeof(servaddr));
        servaddr.sin_family=AF_INET;
        servaddr.sin_addr.s_addr=inet_addr(IP_STR);
        servaddr.sin_port=htons(S_PORT);

        memset((char *) &clientaddr, 0, sizeof(clientaddr));
        clientaddr.sin_family=AF_INET;
        clientaddr.sin_addr.s_addr=inet_addr(IP_STR);
        clientaddr.sin_port=htons(C_PORT);

        if((bind(sfd,(struct sockaddr *)&clientaddr,sizeof(clientaddr)))!=0) {
                perror(“Could not bind socket”);
                return 2;
        }

        printf(“Client is running on %s:%d\n”, IP_STR, C_PORT);
        printf(“Enter a string: “);
        scanf(“%[ ^ \n]%*c”,str_buf);
        len = strlen(str_buf);
        sendto(sfd, &len, sizeof(len), 0, (struct sockaddr *)&servaddr, sizeof(servaddr));
        sendto(sfd, str_buf, len, 0, (struct sockaddr *)&servaddr, sizeof(servaddr));
        addrlen = sizeof(clientaddr);
        recvfrom(sfd, &len, sizeof(len), 0, (struct sockaddr *)&clientaddr, &addrlen);
        recvfrom(sfd, str_buf, len, 0, (struct sockaddr *)&clientaddr, &addrlen);
        printf(“Server Replied: %s\n”, str_buf);

        return 0;
}
```

8. Design UDP Client Server to Transfer a File

Ans :

The Server :

```
/ server code for UDP socket programming
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>

#define IP_PROTOCOL 0
#define PORT_NO 15050
#define NET_BUF_SIZE 32
#define cipherKey 'S'
#define sendrecvflag 0
#define nofile "File Not Found!"

// function to clear buffer
void clearBuf(char* b)
{
        int i;
        for (i = 0; i < NET_BUF_SIZE; i++)
                b[i] = '\0';
}

// function to encrypt
char Cipher(char ch)
{
        return ch ^ cipherKey;
}

// function sending file
int sendFile(FILE* fp, char* buf, int s)
{
        int i, len;
        if (fp == NULL) {
                strcpy(buf, nofile);
                len = strlen(nofile);
                buf[len] = EOF;
```

```
            for (i = 0; i <= len; i++)
                buf[i] = Cipher(buf[i]);
            return 1;
        }

        char ch, ch2;
        for (i = 0; i < s; i++) {
            ch = fgetc(fp);
            ch2 = Cipher(ch);
            buf[i] = ch2;
            if (ch == EOF)
                return 1;
        }
        return 0;
    }

    // driver code
    int main()
    {
        int sockfd, nBytes;
        struct sockaddr_in addr_con;
        int addrlen = sizeof(addr_con);
        addr_con.sin_family = AF_INET;
        addr_con.sin_port = htons(PORT_NO);
        addr_con.sin_addr.s_addr = INADDR_ANY;
        char net_buf[NET_BUF_SIZE];
        FILE* fp;

        // socket()
        sockfd = socket(AF_INET, SOCK_DGRAM, IP_PROTOCOL);

        if (sockfd < 0)
            printf("\nfile descriptor not received!!\n");
        else
            printf("\nfile descriptor %d received\n", sockfd);

        // bind()
        if (bind(sockfd, (struct sockaddr*)&addr_con, sizeof(addr_con)) == 0)
            printf("\nSuccessfully binded!\n");
        else
            printf("\nBinding Failed!\n");

        while (1) {
            printf("\nWaiting for file name...\n");
```

```
                // receive file name
                clearBuf(net_buf);

                nBytes = recvfrom(sockfd, net_buf,
                                  NET_BUF_SIZE, sendrecvflag,
                                  (struct sockaddr*)&addr_con, &addrlen);

                fp = fopen(net_buf, "r");
                printf("\nFile Name Received: %s\n", net_buf);
                if (fp = = NULL)
                    printf("\nFile open failed!\n");
                else
                    printf("\nFile Successfully opened!\n");

                while (1) {

                    // process
                    if (sendFile(fp, net_buf, NET_BUF_SIZE)) {
                        sendto(sockfd, net_buf, NET_BUF_SIZE,
                               sendrecvflag,
                            (struct sockaddr*)&addr_con, addrlen);
                        break;
                    }

                    // send
                    sendto(sockfd, net_buf, NET_BUF_SIZE,
                           sendrecvflag,
                        (struct sockaddr*)&addr_con, addrlen);
                    clearBuf(net_buf);
                }
                if (fp != NULL)
                    fclose(fp);
            }
            return 0;
        }
```

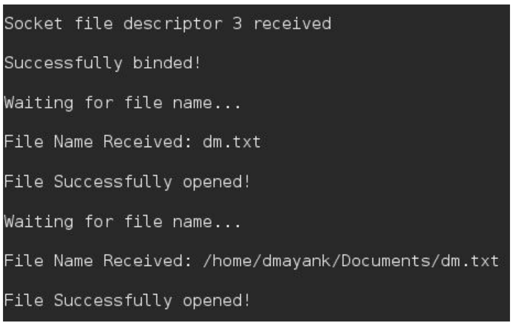
The Client:

```
    // client code for UDP socket programming
    #include <arpa/inet.h>
    #include <netinet/in.h>
    #include <stdio.h>
    #include <stdlib.h>
    #include <string.h>
    #include <sys/socket.h>
```

```
#include <sys/types.h>
#include <unistd.h>

#define IP_PROTOCOL 0
#define IP_ADDRESS "127.0.0.1" // localhost
#define PORT_NO 15050
#define NET_BUF_SIZE 32
#define cipherKey 'S'
#define sendrecvflag 0

// function to clear buffer
void clearBuf(char* b)
{
        int i;
        for (i = 0; i < NET_BUF_SIZE; i++)
                b[i] = '\0';
}

// function for decryption
char Cipher(char ch)
{
        return ch ^ cipherKey;
}

// function to receive file
int recvFile(char* buf, int s)
{
        int i;
        char ch;
        for (i = 0; i < s; i++) {
                ch = buf[i];
                ch = Cipher(ch);
                if (ch == EOF)
                        return 1;
                else
                        printf("%c", ch);
        }
        return 0;
}

// driver code
int main()
{
        int sockfd, nBytes;
        struct sockaddr_in addr_con;
        int addrlen = sizeof(addr_con);
        addr_con.sin_family = AF_INET;
        addr_con.sin_port = htons(PORT_NO);
        addr_con.sin_addr.s_addr = inet_addr(IP_ADDRESS);
        char net_buf[NET_BUF_SIZE];
        FILE* fp;

        // socket()
        sockfd = socket(AF_INET, SOCK_DGRAM,
                        IP_PROTOCOL);

        if (sockfd < 0)
                printf("\nfile descriptor not received!!\n");
        else
                printf("\nfile descriptor %d received\n", sockfd);

        while (1) {
                printf("\nPlease enter file name to receive:\n");
                scanf("%s", net_buf);
                sendto(sockfd, net_buf, NET_BUF_SIZE,
                       sendrecvflag, (struct sockaddr*)&addr_con,
                       addrlen);

                printf("\n—————Data Received—————\n");

                while (1) {
                        // receive
                        clearBuf(net_buf);
                        nBytes = recvfrom(sockfd, net_buf, NET_BUF_SIZE,
                                          sendrecvflag, (struct sockaddr*)&addr_con,
                                          &addrlen);

                        // process
                        if (recvFile(net_buf, NET_BUF_SIZE)) {
                            break;
                        }
                }
                printf("\n———————————————————\n");
        }
        return 0;
}
```

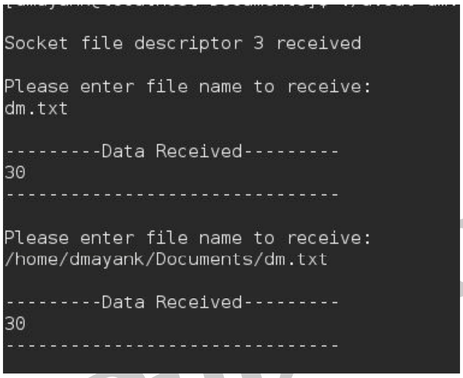
Output :

Server :



```
Socket file descriptor 3 received

Successfully binded!

Waiting for file name...

File Name Received: dm.txt

File Successfully opened!

Waiting for file name...

File Name Received: /home/dmayank/Documents/dm.txt

File Successfully opened!
```

Client :



```
Socket file descriptor 3 received

Please enter file name to receive:
dm.txt

---------Data Received---------
30
-------------------------------

Please enter file name to receive:
/home/dmayank/Documents/dm.txt

---------Data Received---------
30
-------------------------------
```

FACULTY OF INFORMATICS

BCA II-Year IV-Semester (CBCS) Examination

Model Paper - I

COMPUTER NETWORKING

Time : 3 Hours] **[Max. Marks : 70**

Note : Answer all questions from Part - A, & any five questions from Part - B Choosing one questions from each unit.

PART - A (10 × 2 = 20 Marks)

			ANSWERS
1.	(a)	Various types of network elements.	**(Unit-I, SQA-2)**
	(b)	Wireless LAN.	**(Unit-I, SQA-4)**
	(c)	IGMP protocol.	**(Unit-II, SQA-10)**
	(d)	Supernetting	**(Unit-II, SQA-6)**
	(e)	What are timers in TCP	**(Unit-III, SQA-5)**
	(f)	Limitations of UDP	**(Unit-III, SQA-9)**
	(g)	What is I/O Multiplexing	**(Unit-IV, SQA-10)**
	(h)	Unix Socket structure.	**(Unit-IV, SQA-1)**
	(i)	HTTP protocol.	**(Unit-V, SQA-8)**
	(j)	DNS.	**(Unit-V, SQA-4)**

PART - B (5 × 10 = 50 Marks)

UNIT - I

2. Explain various types of network elements. **(Unit-I, Q.No.2)**

OR

3. Compare and contrast Fast Ethernet abd Gigabit Ethernet. **(Unit-I, Q.No.8)**

UNIT - II

4. Explain about the various classes of IPv4. **(Unit-II, Q.No.3)**

OR

5. Explain IPV6 addressing with an example. **(Unit-II, Q.No.8)**

UNIT - III

6. State the Advantages and Disadvantages of TCP. **(Unit-III, Q.No.2)**

OR

7. Explain about Congestion control in TCP. **(Unit-III, Q.No.6)**

UNIT - IV

8. Explain briefly about Unix Socket structure. **(Unit-IV, Q.No.1)**

OR

9. What is I/O Multiplexing? Explain various I/O models. **(Unit-IV, Q.No.14)**

UNIT - V

10. Explain briefly about DNS. **(Unit-V, Q.No.2)**

OR

11. Explain the mechanism of HTTP. **(Unit-V, Q.No.9)**

FACULTY OF INFORMATICS

BCA II-Year IV-Semester (CBCS) Examination

Model Paper - II

COMPUTER NETWORKING

Time : 3 Hours] **[Max. Marks : 70**

Note : Answer all questions from Part - A, & any five questions from Part - B Choosing one questions from each unit.

PART - A (10 × 2 = 20 Marks)

ANSWERS

1. (a) What is Ethernet **(Unit-I, SQA-1)**

(b) Address Resolution Protocol **(Unit-I, SQA-10)**

(c) IP Version Types **(Unit-II, SQA-3)**

(d) Logical addressing **(Unit-II, SQA-1)**

(e) Transmission Control Protocol **(Unit-III, SQA-1)**

(f) Characteristics of UDP **(Unit-III, SQA-7)**

(g) Socket Address Structure **(Unit-IV, SQA-2)**

(h) Client-server Architecture **(Unit-IV, SQA-6)**

(i) Functions of the Application Layer **(Unit-V, SQA-2)**

(j) Disadvantages of FTP **(Unit-V, SQA-7)**

PART - B (5 × 10 = 50 Marks)

UNIT - I

2. Explain about various types of bridges **(Unit-I, Q.No.12)**

OR

3. Explain various types of computer networks? **(Unit-I, Q.No.3)**

UNIT - II

4. What is subnetting? Explain, How to use subnetting. **(Unit-II, Q.No.4)**

OR

5. Explain about IGMP protocol. **(Unit-II, Q.No.10)**

UNIT - III

6. What is Transmission Control Protocol (TCP)? Explain the working mechanism of TCP. **(Unit-III, Q.No.1)**

OR

7. Explain TCP sliding window protocol with neat diagram in detail. **(Unit-III, Q.No.5)**

UNIT - IV

8. Explain the concept of Socket Address Structure (SAS). **(Unit-IV, Q.No.2)**

OR

9. What is UDP? Explain the steps in doing UDP programming. **(Unit-IV, Q.No.10)**

UNIT - V

10. Explain briefly about SMTP protocol. **(Unit-V, Q.No.3)**

OR

11. Explain briefly about HTTP protocol. **(Unit-V, Q.No.8)**

FACULTY OF INFORMATICS

BCA II-Year IV-Semester (CBCS) Examination

Model Paper - III

COMPUTER NETWORKING

Time : 3 Hours] **[Max. Marks : 70**

Note : Answer all questions from Part - A, & any five questions from Part - B Choosing one questions from each unit.

PART - A (10 × 2 = 20 Marks)

			ANSWERS
1.	(a)	CSMA/CD Protocol	**(Unit-I, SQA-3)**
	(b)	Transparent Bridge	**(Unit-I, SQA-7)**
	(c)	CIDR	**(Unit-II, SQA-7)**
	(d)	IPV4 addresses	**(Unit-II, SQA-4)**
	(e)	State the Advantages and Disadvantages of TCP.	**(Unit-III, SQA-2)**
	(f)	Applications of UDP	**(Unit-III, SQA-8)**
	(g)	Advanced socket system	**(Unit-IV, SQA-3)**
	(h)	Functions of Transport Layers.	**(Unit-IV, SQA-5)**
	(i)	Advantages of HTTP	**(Unit-V, SQA-9)**
	(j)	Application Layer Protocols	**(Unit-V, SQA-3)**

PART - B (5 × 10 = 50 Marks)

UNIT - I

2. Explain about Back off algorithm. **(Unit-I, Q.No.7)**

OR

3. Explain about various types of topologies in network. **(Unit-I, Q.No.4)**

UNIT - II

4. How to Subnet a Class C Address Using the Binary Method ? Explain. **(Unit-II, Q.No.5)**

OR

5. Explain about Distance vector routing. **(Unit-II, Q.No.12)**

UNIT - III

6. Explain briefly about TCP header Format. **(Unit-III, Q.No.3)**

OR

7. Explain about UDP protocol **(Unit-III, Q.No.9)**

UNIT - IV

8. What is TCP? Explain TCP client Server model. **(Unit-IV, Q.No.6)**

OR

9. Explain briefly about Asynchronous I/O. **(Unit-IV, Q.No.15)**

UNIT - V

10. Explain briefly about FTP Protocol. **(Unit-V, Q.No. 4)**

OR

11. Explain the Advantages and Disadvantages of HTTP. **(Unit-V, Q.No.11)**

FACULTY OF INFORMATICS

BCA IV - Semester (CBCS) Examination

February - 2023

COMPUTER NETWORKS

Time : 3 Hours] [Max. Marks : 70

Note : I) Answer all questions from Part - A and answer any five questions from Part-B. Choosing one questions from each unit.

II) Missing data, if any, may be suitably assumed.

PART - A (10 × 2 = 20 Marks)

1. (a) What is wired LAN?

(b) Define bridge.

(c) Define supernetting.

(d) What is logical addressing?

(e) Define UDP.

(f) What is windowing effect?

(g) Write about synchronous I/O.

(h) Define raw sockets.

(i) What are the types of connections in FTP?

(j) Write about HTTP messages.

PART - B (5 × 10 = 50 Marks)

UNIT - I

2. (a) Explain about wireless LANIEEE802.11 architecture.

(b) Discuss in detail about CSMA/CD protocol.

OR

3. (a) Write about Fast and Gigabit Ethernet.

(b) Differentiate between ARP and RARP.

UNIT - II

4. (a) Explain about CIDR in detail.

 (b) Explain about IGMP.

OR

5. (a) Explain about Link state routing.

 (b) What are differences between OSPF and BGP?

UNIT - III

6. (a) Explain the services provided by Transport Layer.

 (b) Explain Token bucket algorithm.

OR

7. (a) Discuss in detail about the approaches for controlling congestion in a network.

 (b) Explain about UDP protocol.

UNIT - IV

8. (a) Explain about primitive system calls.

 (b) Discuss about concurrent programs.

OR

9. (a) Explain about UDP system calls.

 (b) Explain Asynchronous I/O model.

UNIT - V

10. (a) Explain about the working mechanism of SMTP.

 (b) Write about network application architecture.

OR

11. (a) Explain about platforms of DNS.

 (b) Discuss in detail about HTTP.