ACN
SUPER 25

1. What is ISP and ICANN? Describe the role of ICANN

"An Internet Service Provider (ISP) is an organization that provides services for accessing, using, or participating in the Internet. ISPs may offer internet connectivity, domain registration, web hosting, and other related services."

"The Internet Corporation for Assigned Names and Numbers (ICANN) is a non-profit organization responsible for coordinating the global domain name system (DNS), IP address allocation, and protocol identifiers. ICANN ensures the stable and secure operation of the internet's unique identifiers."

ICANN- Internet Corporation for Assigned Names and Numbers

- Responsible for internet Domain Names and Addresses
- **Roles of ICANN:**
- Managing and coordinating DNS
- Assigning domain names and operations to root server
- Allocating IP addresses and managing global internet protocol address space
- Ensuring that domain names and IP addresses are unique and accessible globally

2. Explain in details IPv4 Header Format

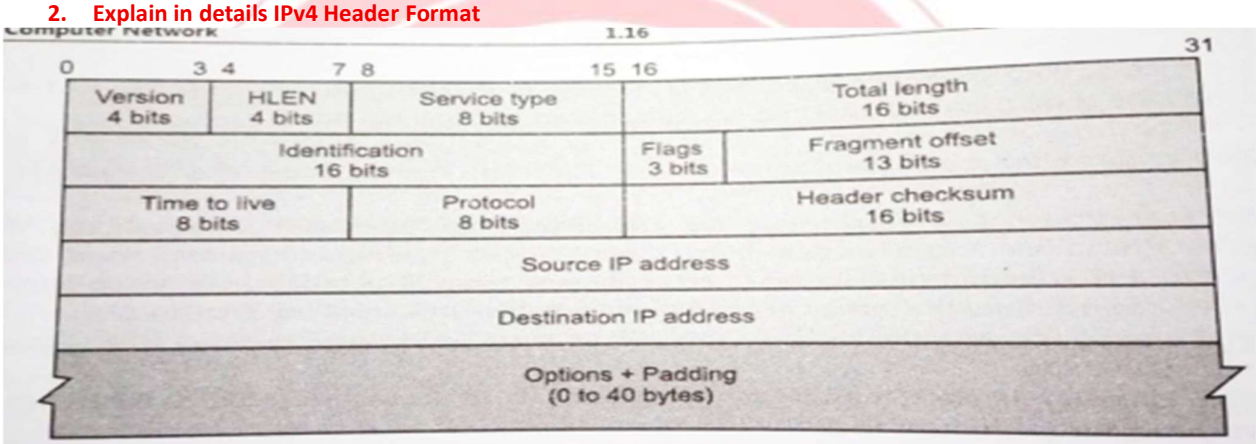
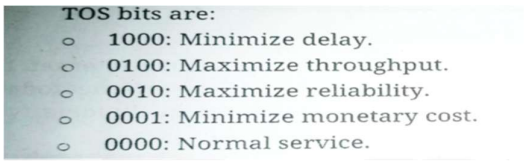


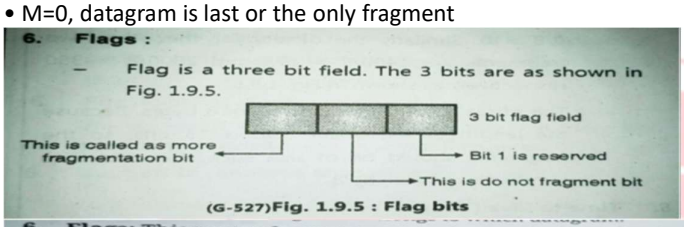
Fig. 1.13: IPv4 Header Format

- **Version(VER):**
- 4-bit field • Defines version of IP format
- Current version is 4 - hence IPv4
- Field indicates- IP software running on processing machine– that datagram belongs to IPv4
- If some other IP version - datagram is discarded
- **HLEN(Header Length):**
- 4- bit field defines datagram object in 4-byte word
- Length of header is variable • Between 20 to 60 bytes
- Default header length – 20 bytes • Value of field is 5 • 5 x 4=20
- For max size • Value of field is 15 • 15 x 4=60
- **Type of Service(TOS)**
- Provides network service parameters
- Composed of 3-bits precedence field(generally ignored)
- 4 TOS bits and unused bit- must be 0
- 4 TOS bits are:



- **Total length**
- 16-bit field • Defines total length of datagram
- Max length= 2raised to 16- 1 • Contains combined data and header length
- HLEN X 4 -> header length • Length of Data= Total Length – Header Length

• **Identification**
• Identifies datagrams source host
• when datagram is fragmented- identification field is copied to all fragments
• This number is used by destination to reassemble fragments
• **Flags**
• 3-bit field • First bit- reserved- should be 0
• Second bit- known as “Do Not Fragment” bit
• If 1 – datagram is not fragmented
• If 0 – machine should fragment datagram(if necessary)
• Third bit-(M)- “More Fragment Bit”
• M=1 ,datagram is not last fragment
• M=0, datagram is last or the only fragment

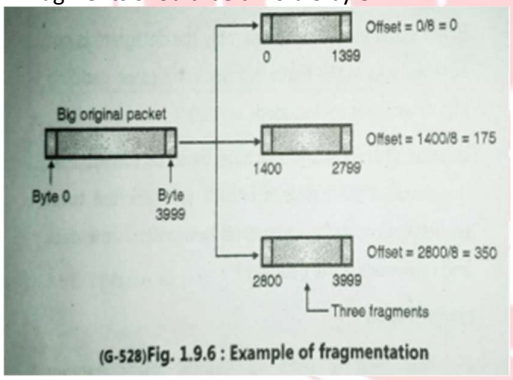


(G-527)Fig. 1.9.5 : Flag bits

6. **Flags:** This router fragment activity is controlled by following three flags:

Sr. No.	Flag	Description
1.	0	Reserved, must be zero.
2.	DF (Do not Fragment)	0 means allow fragmentation; 1 means do not allow fragmentation.
3.	MF (More Fragments)	0 means that this is the last fragment of the datagram; 1 means that additional fragments will follow.

• **Fragmentation offset**
• 13 bits field
• Used to indicate the relative position of this fragment with respect to complete datagram
• It is the offset of data in original datagram • In units of 8 bytes
• Fragments should be divisible by 8



(G-528)Fig. 1.9.6 : Example of fragmentation

• **Time to live (TTL)**
• If routing table gets corrupted ・ datagram travels between routers ・ but never reaches destination
• TTL- limits the lifetime of datagram
• It limits the journey of packet intentionally
• TTL field ia 1 – for local network packet
• When packet reaches to first router TTL is changed to 0
• **Protocol**
• 8-bit field • Defined higher level protocols which uses services of IP layer
• Data from different protocols is encapsulated into IP datagram
• Contains- name of protocol and destination IP address
• At destination this value helps in demultiplexing

Protocol	Description
0	Reserved.
1	Internet Control Message Protocol (ICMP).
2	Internet Group Management Protocol (IGMP).
3	Gateway-to-Gateway Protocol (GGP).
4	IP (IP encapsulation).
5	Stream.
6	Transmission Control Protocol (TCP).
8	Exterior Gateway Protocol (EGP).
9	Private Interior Routing Protocol.
17	User Datagram Protocol (UDP).
41	IP Version 6 (IPv6).
50	Encap Security Payload for IPv6 (ESP).
51	Authentication Header for IPv6 (AH).
89	Open Shortest Path First.

• **Header checksum**
• Covers on header only • When header field changes checksum is recomputed and verified
• It checks and monitors communication errors • It does not cover data followed by header
• **Source address**
• 32-bit field • Stores source address
• This field must remain unchanged during the time datagram travels from source host to destination host
• **Destination address**
• Defines/stores IP address of destination • 32-bit field
• This field must remain unchanged during the time datagram travels from source host to destination host
• **Options**
• Last field of header • Used for additional information
• Not required for every datagram • Used for network testing and debugging
• When used- header length is greater than 32bits

3. Explain ARP

• **ARP (Address Resolution Protocol)**
• Used to convert logical address (IP address) to physical address (MAC- media access controlled address)
• It's a network layer protocol
• This conversion is need because IP address and MAC address differ in length.
• In IPv4 : IP address- 32 bits & MAC address- 48 bits
• MAC address- data link layer (local address to router)
• Establishes and terminates connection between 2 devices

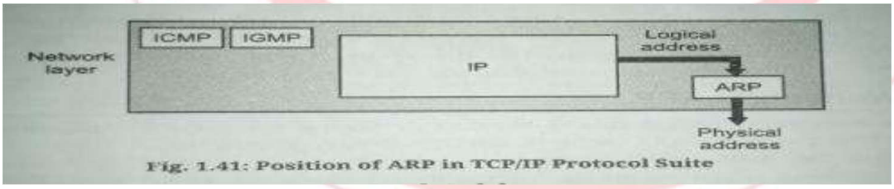


Fig. 1.41: Position of ARP in TCP/IP Protocol Suite

• Mapping of IP address(Logical) to MAC address (physical)
• 2 types:
• **Static mapping**
• Mapping table created and stored at each machine
• Table helps associating both addresses
• Mapping becomes difficult if MAC address changes
• Changed MAC address must be updated periodically
• **Dynamic mapping**
• A protocol is used for finding other addresses
• When one type of address is known
• 2 protocols
• ARP- maps IP to MAC
• RARP(Reserved Address Resolution Protocol)- maps MAC to IP

Working of ARP

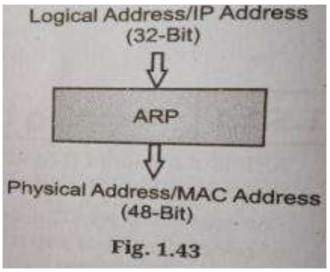


Fig. 1.43

- **Working of RARP**

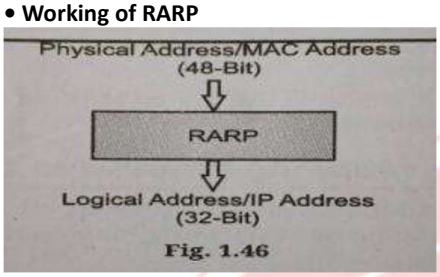
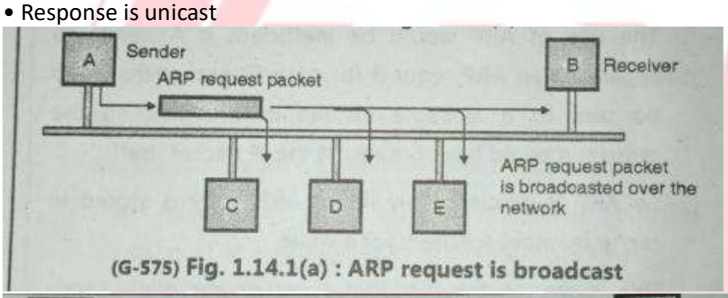
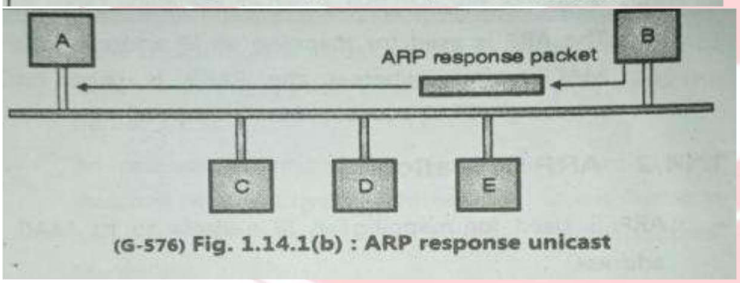


Fig. 1.46

- **ARP Operation:**
- Finds MAC address
- ARP request packet- contains IP and MAC address of A & IP address of B
- Every host receives packet but only B responds
- Response packet- contains IP and MAC address of B
- Response is unicast



(G-575) Fig. 1.14.1(a) : ARP request is broadcast



(G-576) Fig. 1.14.1(b) : ARP response unicast

- **ARP Packet Format:**

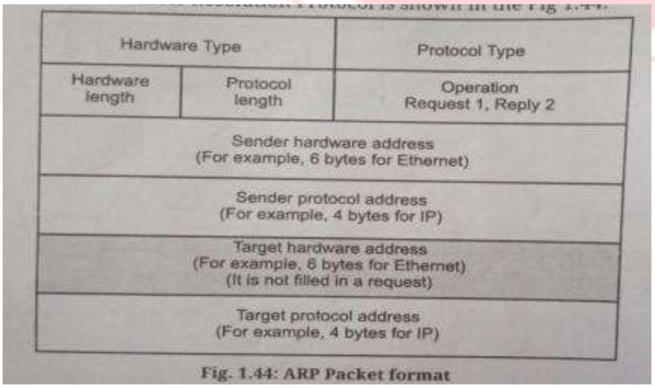


Fig. 1.44: ARP Packet format

- **Hardware Type(HTYPE):**
- 16-bits field • Defines type of network on which ARP is running

- ARP can be used on any physical network
- **Protocol Type(PTYPE):**
- 16- bits field • Defines protocol using ARP
- ARP can be used with any higher-level protocol like IPv4
- **Hardware Length(HLEN):**
- 8-bit field • Defines length of physical address in bytes • E.g. value 6 is for Ethernet
- **Protocol Length(PLEN):**
- 8-bits field • Defines length of logical address in bytes • Value is 4 for IPv4
- **Operation (OPER):**
- 16-bits field • Defines type of packet • Request packet or reply packet
- **Sender Hardware Address(SHA):**
- Variable length field • Defines physical address of sender
- **Sender Protocol Address(SPA):**
- Variable length field • Defines logical address of sender
- **Target Hardware Address(THA):**
- Variable length field • Defines physical address of target • Contains all 0's for ARP
- As receivers physical address is not known
- **Target Protocol Address(TPA):**
- Variable length field • Defines logical address of target

4. Explain Classful Addressing in detail

Classful Addressing – divides address space in 5 classes

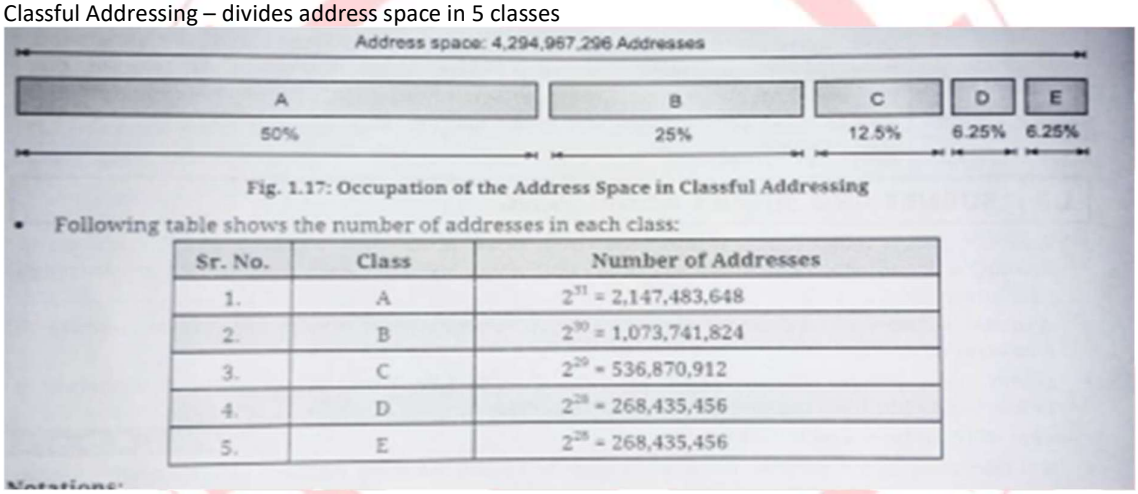


Fig. 1.17: Occupation of the Address Space in Classful Addressing

- Following table shows the number of addresses in each class:

Sr. No.	Class	Number of Addresses
1.	A	2^{31} = 2,147,483,648
2.	B	2^{30} = 1,073,741,824
3.	C	2^{29} = 536,870,912
4.	D	2^{28} = 268,435,456
5.	E	2^{28} = 268,435,456

Classful addresses

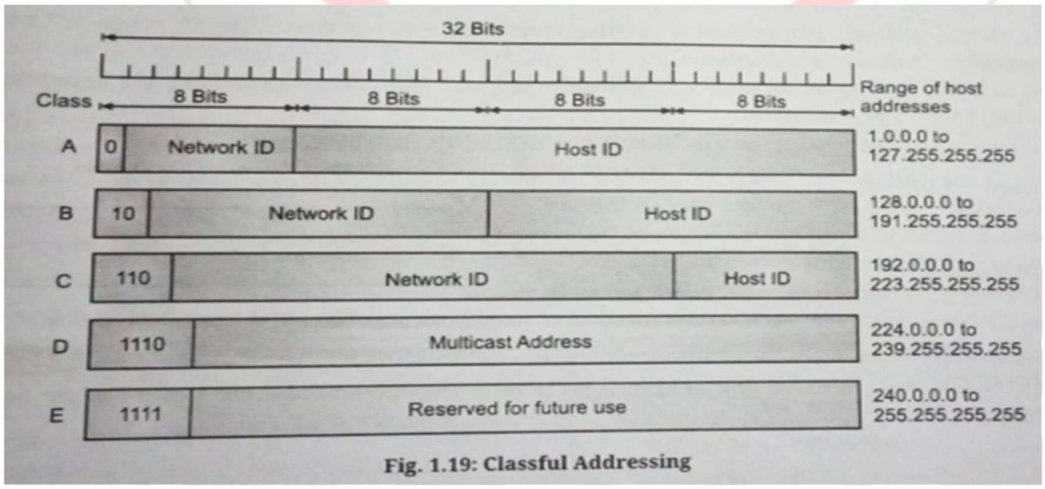


Fig. 1.19: Classful Addressing

Class A

- **Starting bits:** 0
- **IP range:** 0.0.0.0 to 127.255.255.255
- **Default subnet mask:** 255.0.0.0

- **Network/Host split:** First 8 bits for network, remaining 24 bits for host
- **Total networks:** 128 (but 0 and 127 are reserved)
- **Hosts per network:** Over 16 million
- **Use case:** Very large organizations or backbone networks

Class B

- **Starting bits:** 10
- **IP range:** 128.0.0.0 to 191.255.255.255
- **Default subnet mask:** 255.255.0.0
- **Network/Host split:** First 16 bits for network, remaining 16 bits for host
- **Total networks:** Around 16,000
- **Hosts per network:** Around 65,000
- **Use case:** Medium-sized organizations, universities

Class C

- **Starting bits:** 110
- **IP range:** 192.0.0.0 to 223.255.255.255
- **Default subnet mask:** 255.255.255.0
- **Network/Host split:** First 24 bits for network, last 8 bits for host
- **Total networks:** Over 2 million
- **Hosts per network:** 254
- **Use case:** Small businesses, LANs

Class D

- **Starting bits:** 1110
- **IP range:** 224.0.0.0 to 239.255.255.255
- **Subnet mask:** Not applicable
- **Use case:** Reserved for **multicasting** (sending data to multiple receivers)

Class E

- **Starting bits:** 1111
- **IP range:** 240.0.0.0 to 255.255.255.255
- **Subnet mask:** Not applicable
- **Use case:** Reserved for **experimental or future use**

5. Explain Super netting/ CIDR with diagram

"**Supernetting** is a technique used in IP networking to combine two or more **contiguous (adjacent)** networks with the same subnet mask into a **single larger network**, called a **supernet**. It is the **reverse of subnetting**, which divides a network into smaller parts. Supernetting helps reduce the number of entries in routing tables and improves routing efficiency."

• Process of combining multiple networks into a single larger network
• Also called as CIDR (Classless Inter-Domain Routing)
• A lot of unused address spaces in classful addressing
• Ex.
• Class A has more than 16million host addresses
• Class B has more than 65000 host addresses
• But limited number of address space has been allocated for internet use
• Class C has max 256 addresses
• Midsize organization may need more addresses
• Solution is supernetting – organization can combine several class C blocks.

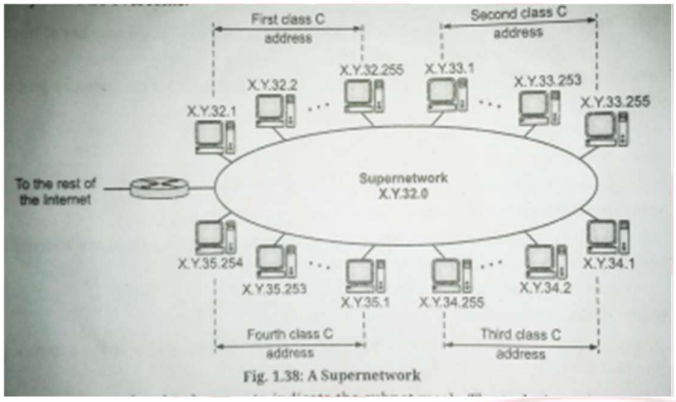


Fig. 1.38: A Supernetwork

- Supernet mask
- Reverse of subnet mask
- For supernetting we need to know first address in the block and supernet mask

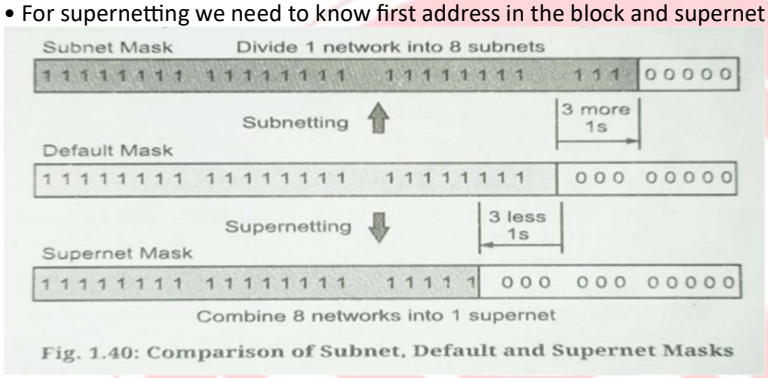


Fig. 1.40: Comparison of Subnet, Default and Supernet Masks

6. Explain Routing Architecture in details.

Routing Architecture: "Design and organization of routers which is responsible for forwarding packets in a network"

This architecture performs 2 main functions:

o Process routable protocols

o Use routing protocols to determine the best path

Router has 4 components as shown in the diagram

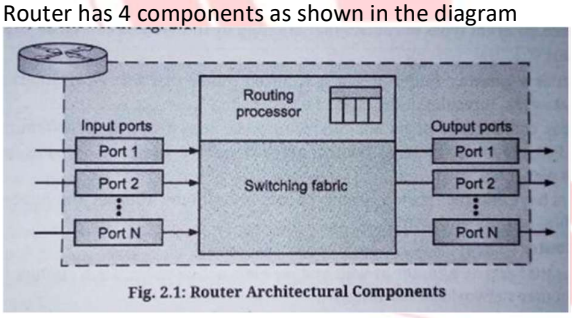
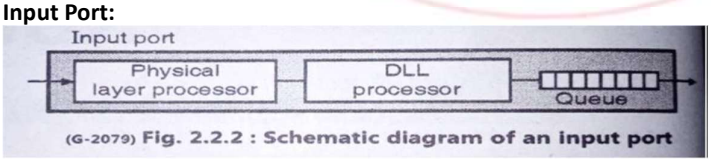


Fig. 2.1: Router Architectural Components

Input Port:



(G-2079) Fig. 2.2.2 : Schematic diagram of an input port

Performs the physical and data link layer functions of the router

Functions of input port:

o Constructs bits from received signal

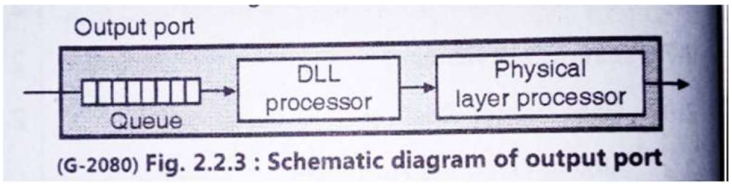
o Decapsulates packet from the frame

o Detects and corrects the errors

o Packet is ready to be forwarded

o Buffer will hold the ready to forward packet before directing to the switching fabric

Output Port:



(G-2080) Fig. 2.2.3 : Schematic diagram of output port

Performs same functions like input port but in reverse order
o The outgoing packets are buffered and queued
o Packets are encapsulated to create the frames
o Physical layer function are applied to the frame
o This will create signal that can be sent on the line

Routing Processor:

Performs function of network layer
Uses destination address to find :
o Output port number from where the packet is to be sent out
o The address of the next hop
Routing processor works by accessing routing table
Activity of routing processor is called as table lookup

Switching Fabric in a router:

Connects the router's input port to output port
It is a combination of hardware and software which moves data coming into network node out by correct port to the next node in the network
Modern days routers use different types of switching fabrics:
o Crossbar switch
o Banyan switch
o Batcher banyan switch

7. Explain different types of Routing

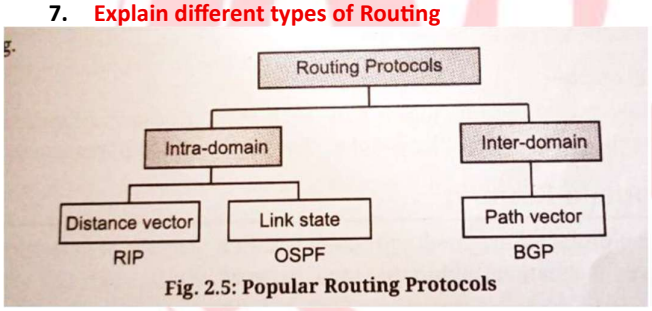


Fig. 2.5: Popular Routing Protocols

1. Static Routing

- In static routing, routes are **manually configured** by the network administrator.
- Routing paths do **not change automatically**.
- Suitable for **small and stable networks**.
- Requires **manual updates** if the network topology changes.
- Example: Setting a fixed route to a gateway in a LAN.

2. Dynamic Routing

- In dynamic routing, routers use **routing protocols** to learn and update routes automatically.
- Adapts to **network changes** like link failures or new paths.
- Suitable for **large and complex networks**.
- Common protocols: **RIP, OSPF, EIGRP, BGP**.
- Requires more **CPU, memory, and bandwidth**.

3. Interdomain Routing

- Routing between **different autonomous systems (AS)**.
- Used in **large-scale networks** like the internet.
- Focuses on **policy-based routing** and **scalability**.
- Protocol used: **BGP (Border Gateway Protocol)**.
- Example: Routing between ISP networks.

4. Intradomain Routing

- Routing **within a single autonomous system (AS)**.

- Focuses on **efficiency and speed** inside the organization.
- Protocols used: **RIP, OSPF, EIGRP**.
- Example: Routing between departments in a university network.

8. What is Distance Vector Routing? Explain with example.

"**Distance Vector Routing** is a dynamic routing technique in which each router shares its routing table with its directly connected neighbours at regular intervals. The routing table contains information about the **distance (cost)** to reach each destination and the **next hop** router. Routers use this information to calculate the shortest path to all networks using algorithms like **Bellman-Ford**."

- The least-cost route between any two nodes →the route with minimum distance.
- each node maintains a vector (table) of minimum distances to every node.
- The table at each node also guides the packets to the desired node by showing the next stop in the route (next-hop routing).

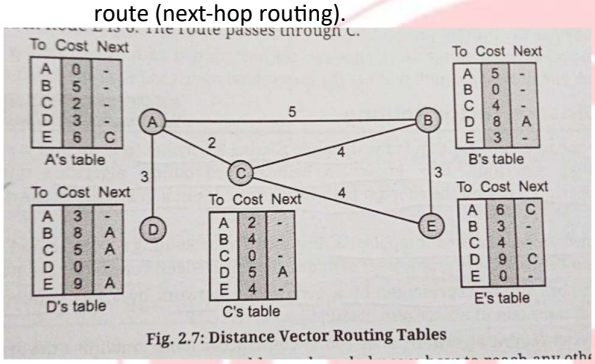


Fig. 2.7: Distance Vector Routing Tables

- The table for node A shows how we can reach any node from this node.
- For example, our least cost to reach node E is 6.
- The route passes through C.
- ***Initialization***
- The tables are stable
- each node knows how to reach any other node and the cost.
- At the beginning→Each node can know only the distance between itself and its neighbours (directly connected)
- each node can send a message to the immediate neighbours and find the distance between itself and these neighbours.
- The distance for any entry that is not a neighbour is marked as **infinite** (unreachable).

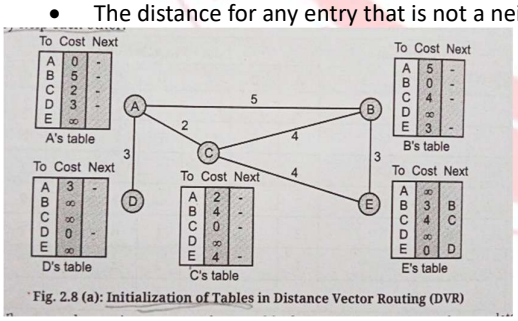


Fig. 2.8 (a): Initialization of Tables in Distance Vector Routing (DVR)

- ***Sharing***
- sharing of information between neighbours.
- node A does not know about node E, node C does.
- if C shares its routing table with A→ node A can also know how to reach node E.
- node C does not know how to reach node D, but node A does.
- A shares its routing table with node C→node C also knows how to reach node D.
- A and C – neighbours→ can **improve** their routing tables if they **help** each other.
- each node → sends its entire table to neighbour →neighbour decides what part to use and what part to discard.

- third column (next stop) →not useful to neighbour.
- this column → replaced with the **sender's** name.
- If any row can be used - the next node is the sender of the table.
- node can send - only first two columns
- each node shares its routing table with its immediate neighbours **periodically** and when there is a **change**.
- ***Updating***
- node receives →two-column table from neighbour →updates its routing table.
- Three steps:

1. receiving node adds the cost between itself and the sending node to each value in the second column. If node C claims that its distance to a destination is *x* mi and distance between A and C is *y* mi→ then distance between A and destination (via C) is *x* + *y* mi.
2. receiving node adds the name of the sending node to each row as the third column if the receiving node uses information from any row→The sending node is the next node in the route.
3. receiving node compares each row of its old table with the corresponding row of modified version of the received table.
    a. If the next-node entry is different→ chooses the row with the smaller cost.
    b. If there is a tie, the old one is kept.
    c. If the next-node entry is the same, the receiving node chooses the new row. **example**, suppose node C has previously advertised a route to node X with distance 3 → now there is no path between C and X →node C now advertises this route with a distance of infinity. Node A must not ignore this value even though its old entry is smaller. The old route does not exist any more. The new route has a distance of infinity.

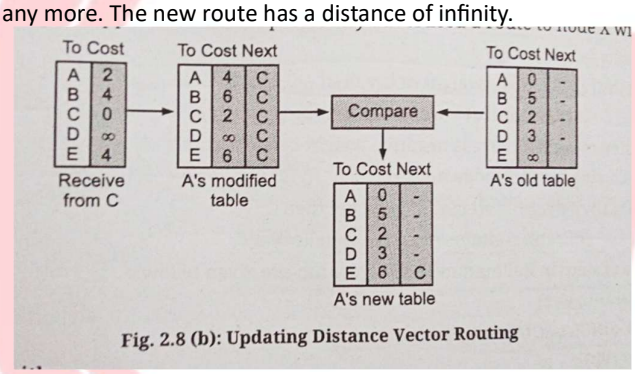


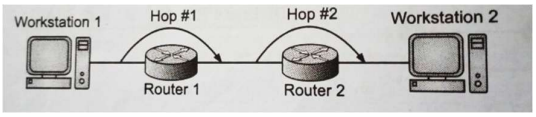
Fig. 2.8 (b): Updating Distance Vector Routing

- when we add any number to infinity, the result is still infinity.
- The modified table shows how to reach A from A via C.
- If A needs to reach itself via C, it needs to go to C and come back, a distance of 4.
- the only benefit from this updating of node A is the last entry, how to reach E. it knows that the cost is 6 via C.
- Each node can update its table by using the tables received from other nodes.
- if there is no change in the network - each node reaches a stable condition in which the contents of its table remain the same.

9. Explain working of RIPv2

- Intra domain routing protocol
- Used inside autonomous system
- One routing protocol cant handle table updating over internet
- **RIP** → is distance vector protocol→uses Bellman Ford algorithm→ for calculating routing tables
- **RIP uses port 520**
- It's a dynamic routing protocol
- Routing metric→ hop count
- **HOP COUNT:**
    - Hop→ portion of a path
    - Data packet passes through → bridges, routers and gateways → to reach the destination
    - Passing through a device → HOP occurs
    - Hop→ no. of routers occurring on the way (source to destination)

- Lowest hop count→ best route→places in routing table
- RIP→ prevents routing loops→ by limiting no. of hops
- max. Hop count→ 15 :: 16→ network unreachable
- " hop count means number of intermediate devices through which data must pass between source and destination"



- **Features of RIP:**
  - Updates of network are exchanged periodically
  - Updates are always broadcasted
  - Full routing tables are sent in updates
  - Router always trusts on routing information received from neighbours→ known as routing on rumours
- **Working Example of RIP:**

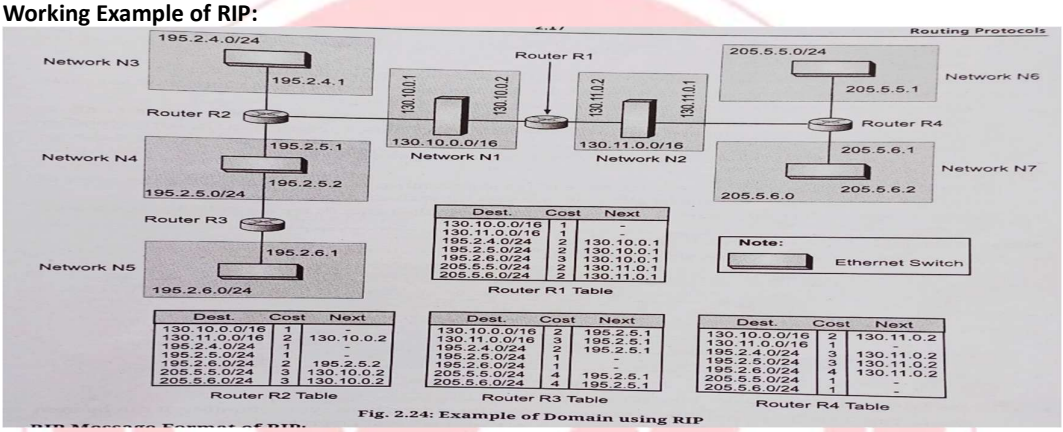
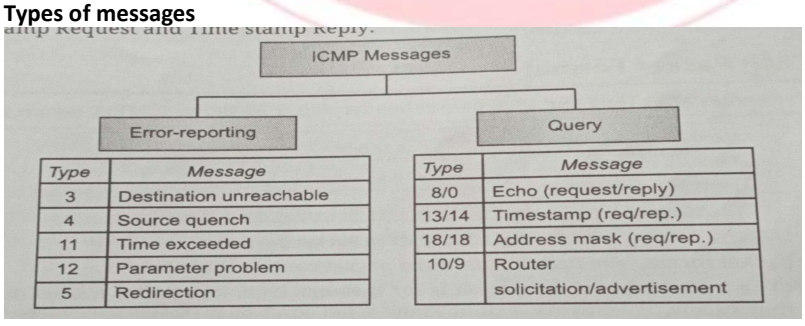


Fig. 2.24: Example of Domain using RIP

- AS with -7 networks and 4 routers
- Routing table of R1 – 7 entries – to reach each network in AS
- R1 is directly connected to 130.10.0.0 and 130.11.0.0→ no next hop for this entry
- For reaching N3, N4 and N5 → packet has to pass through R2
- So next node entry is of 130.10.0.1
- For sending packet to N6 and N7 – it has to pass through R4 – with IP 130.11.0.1

10. Explain ICMP and error reporting messages in it

- Simplest protocol in TCP/IP suite
- Used for reporting errors and management queries
- Communication is considered smooth until no one reports problem
- If middleman reports mistake→ ICMP helps to notify sender(feedback)
- Communication can be adjusted or rerouted to keep everything running smoothly
- **Types of messages**



Type	Message
3	Destination unreachable
4	Source quench
11	Time exceeded
12	Parameter problem
5	Redirection

Type	Message
8/0	Echo (request/reply)
13/14	Timestamp (req/rep.)
18/18	Address mask (req/rep.)
10/9	Router solicitation/advertisement

Types of ICMP Messages:

Category	Type	Message Name	Description
Error-Reporting Messages	3	Destination Unreachable	Indicates that a datagram could not be delivered to its destination. The Code value provides more information on the nature of the error.
	4	Source Quench	Lets a congested IP device tell a device that is sending it datagrams to slow down the rate at which it is sending them.
	11	Time Exceeded	Sent when a datagram has been discarded prior to delivery due to expiration of its Time to Live field.
	12	Parameter Problem	Indicates a miscellaneous problem (specified by the Code value) in delivering a datagram.
	5	Redirect	Allows a router to inform a host of a better route to use for sending datagrams.
	0	Echo Reply	Sent in reply to an Echo (Request) message; used for testing connectivity.
	8	Echo Request	Sent by a device to test connectivity to another device on the internetwork. The word Request sometimes appears in the message name.

	13	Timestamp Request	Sent by a device to request that another send it a timestamp value for propagation time calculation and clock synchronization. The word Request sometimes appears in the message name.
Query Message	14	Timestamp Reply	Sent in response to a Timestamp (Request) to provide time calculation and clock synchronization information.
	17	Address Mask Request	Used to request that a device send a subnet mask.
	18	Address Mask Reply	Contains a subnet mask sent in reply to an Address Mask Request.
	9	Router Advertisement	Used by routers to tell hosts of their existence and capabilities.
	10	Router Solicitation	Used by hosts to prompt any listening routers to send a Router Advertisement.

- **Message packet format**

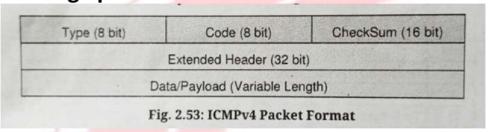


Fig. 2.53: ICMPv4 Packet Format

- **First 32 bits contain 3 fields**
- **Type:**
- 8-bit
  - Type 0 – Echo Reply
  - Type 3 – Destination Unreachable
  - Type 5 – Redirect Message
  - Type 8 – Echo Request
  - Type 11 – Time Exceeded
  - Type 12 – Parameter Problem
- **Code:**
- 8-bits
- Carries additional information about error message and type
- **Checksum:**
- Last 16 bits
- Used to check number of bits of complete message
- Enables ICMP tool to ensure that complete data is received
- **Extended Header:**
- 32 bits
- Points out problem in IP message
- Byte's location is identified by pointers which causes the problem message
- Receiving device looks here for pointing to a problem
- **Data or Payload:**
- Variable length field
- **Error reporting messages**

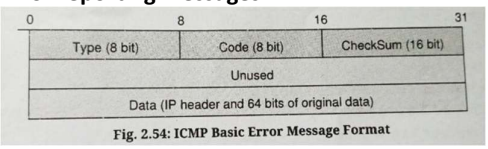


Fig. 2.54: ICMP Basic Error Message Format

- **Type:**
- Identifies type of the message
- **Code:**
- Describes purpose of the message
- **Checksum:**
- Used to validate ICMP message

- **Unused:**
- Reserved for future use
- Set to 0
- **Data:**
- Includes IP header of datagram
- Also contains first 8 bytes of data
- Used by sender to get more details about error that has occurred

11. What is UDP? Enlist/Explain the services provided by UDP?

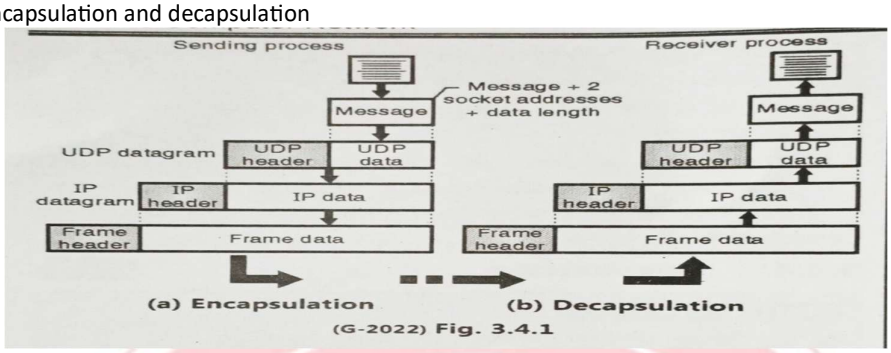
UDP- User Datagram Protocol

- It is a transport layer protocol
- Belongs to TCP/IP protocol suite
- Serves as intermediary between application programs and network operations
- Designed by David P. Reed in 1980
- Defined in RFC 768
- UDP is connectionless protocol
- UDP is suitable for streaming applications
- **UDP Services:**
  - Process to process communication
  - Connectionless services
  - Flow control
  - Error control
  - Checksum
  - Congestion control
  - Encapsulation and decapsulation
  - Queuing
  - Multiplexing and demultiplexing
- Process to process communication:
  - UDP provides process to process co by using sockets → combination of IP address and port numbers.
  - (List of Ports is given in further question)
- Connectionless service
  - UDP is connectionless, hence unreliable.
  - Each user datagram sent by UDP is an independent datagram
  - Different user datagrams sent by UDP have no relationship between them →even though they are origination from same process and sent to same destination
  - User datagram does not have any number
  - Ther is no connection establishment and no connection termination
  - Each datagram is free to travel in path
  - Processes sending short messages can successfully use UDP
  - Messages less than 65507 bytes can use UDP
- Flow and error control
  - It does not provide flow control hence receiver can overflow with incoming messages.
  - It also does not support error control mechanism except for the checksum
  - No acknowledge is sent from destination to the sender so sender does not know whether message has reached or lost or duplicated
  - If receiver detects any error using checksum → the datagram is discarded.

- Checksum
  - Checksum is provided for data integrity
  - It is different from Checksum calculation IP
  - Checksum is calculated by considering following 3 sections
    1. Pseudo header
    2. UDP header
    3. Data coming from application layer
  - Checksum is optional in UDP

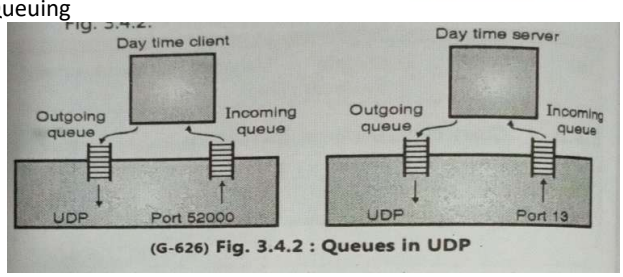
- Sender can make a decision of not calculating the checksum
- If so → field is filled with zeros before sending packet

- Congestion control
  - UDP does not provide Congestion control
  - It assumes the packets being small will not crest any congestion
  - The assumption may not be true → in case of real time transfer of audio and video
- Encapsulation and decapsulation



(G-2022) Fig. 3.4.1

  - UDP encapsulates and decapsulates messages→ to exchange message between two communication processes
  - Encapsulation:
    - Message produced by a process is to be sent by UDP
    - Process passes→ the message and 2 socket addresses with the length of data to UDP
    - UDP receives the data and adds the UDP header to it
    - "This is called UDP datagram which is passes to IP with the socker address"
    - IP adds its own header to UDP datagram
    - It enters value 17 into the protocol field→ indicating UDP is being used
    - IP datagram is then passed to data link layer
    - DLL adds its own header and trailer to create frame and sends it to physical later
    - Finally physical layer converts these bits into electrical/optical signals and sends to the destination
  - Decapsulation:
    - Encoded message arrives at destination physical layer
    - Optical/electrical signal is decoded into bits and passes them to DLL
    - DLL checks the data using header and trailer
    - Header and trailer are discarded if no errors
    - Datagram is passed to IP
    - IP carries out checking of errors
    - If no errors datagram is passed to UDP after dropping IP header
    - Datagram from IP to UDP also contains sender and receiver IP address
    - Entire datagram is checked using checksum
    - If no errors UDP header is dropped
    - Application data + senders' sockets are handed over to the process
    - Process uses socket address to respond that message is received
- Queuing



(G-626) Fig. 3.4.2 : Queues in UDP

  - The process starts at the **client** site by requesting a port number from OS
  - Every process gets one port number→ hence can create one outgoing and incoming queue
  - Queues function only when process is running

- They are destroyed as soon as process is terminated
- **Server** creates incoming and outgoing queues using its well-known ports
- Queues exist as long as server is running

- Multiplexing and demultiplexing

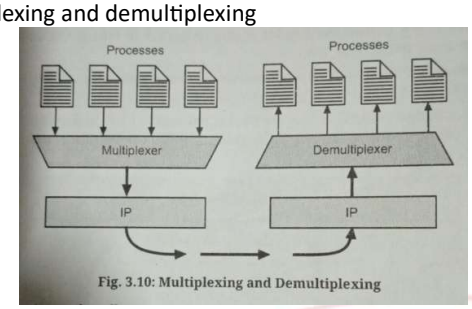


Fig. 3.10: Multiplexing and Demultiplexing

- Multiplexing: at the sender's side -several processes need to send user datagram
- There is only one UDP
- Many-to-one relationship
- Hence requires multiplexing
- Demultiplexing: at receiver's side there is only one UDP
- Many processes can receive user datagram
- One-to-many relationship
- Hence requires demultiplexing

12. What is Transmission Control Protocol (TCP)? Explain TCP services (ANY 2) in details.

- It's a reliable connection-oriented protocol
- Connection is established between sender and receiver before data is transmitted
- It divides the data received from upper layer into segments and tags the sequence number to each segment→which are used at receiving end for reordering the data
- TCP relies on application layer and network layer
- It is a transport layer protocol
- Following are the services provided by TCP
  a. Stream delivery service
  b. Sending and receiving buffers
  c. Bytes and segments
  d. Full duplex service
  e. Connection oriented service
  f. Reliable service
  g. Process to process communication

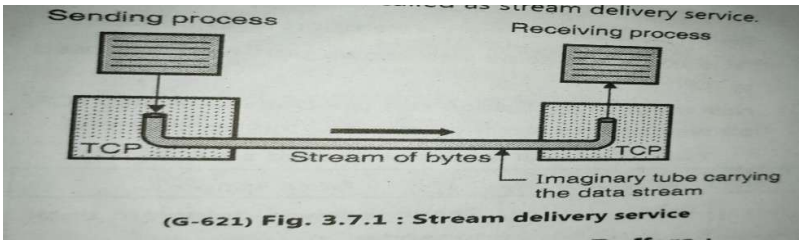
1. Process to Process Communication

- TCP uses port numbers- similar to UDP

Table 3.7.1 : Well known ports used by TCP

Port	Protocol	Description
7	Echo	Sends received datagram back to sender
9	Discard	Discards any received packet
11	Users	Active users
13	Daytime	Sends the date and the time
17	Quote	Sends a quote of the day
19	Chargen	Sends a string character
20	FTP, Data	File Transfer protocol for data
21	FTP, Control	File Transfer protocol for control
23	TELNET	Terminal network
25	SMTP	Simple Mail Transfer Protocol
53	DNS	Domain Name server
67	BOOTP	Bootstrap Protocol
79	Finger	Finger
80	HTTP	Hypertext Transfer Protocol
111	RPC	Remote Procedure Call

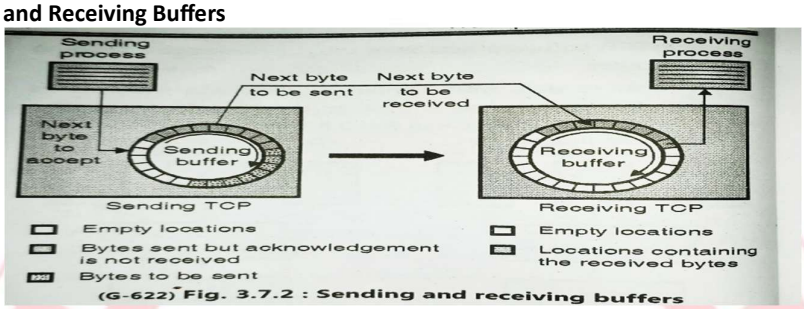
2. Stream Oriented/ Stream Delivery Service



(G-621) Fig. 3.7.1 : Stream delivery service

- TCP is a stream-oriented protocol
- A process delivers/receives the data in the form of stream of bytes.
- Sending and Receiving processes seams to be connected by imaginary tubes
- This is called as stream delivery services.

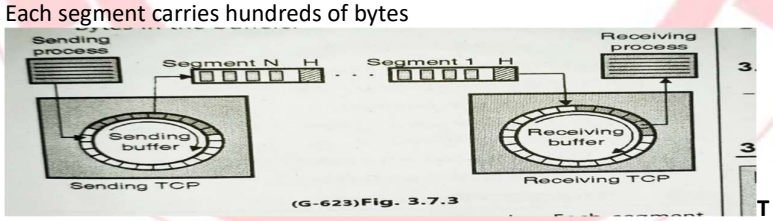
3. **Sending and Receiving Buffers**



(G-622) Fig. 3.7.2 : Sending and receiving buffers

- Sending and Receiving process may not produce or receive data at the same speed hence TCP needs buffer at both ends
- Two types of buffers used → sending buffer, & receiving buffer
- Figure shows direction of movement of data
- Sending buffer has three types of locations
- Receiving buffer has two types of locations

4. **Bytes and Segments**
   - Only buffering is not enough → we need one more step before sending data
   - TCP groups a number of bytes to form a packet called a segment
   - Header is added to each segment
   - Segment is inserted into IP datagram and transmitted.
   - Segments may be received out of order or lost or corrupted
   - All segments are not of same size
   - Each segment carries hundreds of bytes



(G-623)Fig. 3.7.3

5. **Full Duplex Service**
   - TCP offers Full Duplex Service → data can travel in both directions simultaneously.
   - Each TCP has sending and receiving buffer hence full duplex service possible.
6. **Connection Oriented Service**

- TCP is a connection-oriented protocol
- When process 1 wants to communicate with process 2
- Sequence of operations is as follows:
  1. TCP of process 1 informs TCP of process 2 and create a connection between them
  2. Both TCP exchange data in both the directions
  3. After completing the exchange both the buffers are empty – TCP destroys buffer to terminate the connection.
- The connection is virtual
- Datagram is encapsulated
- If segment get lost or corrupted – have to be resend

- Each segment may take different path to reach destination

7. **Reliable Service**
   - TCP is a reliable protocol
   - It uses checksum for error detection
   - Attempts to recover lost or corrupted packets by retransmission, acknowledge policy and timers
   - To ensure reliability it uses- byte number, sequence number, acknowledge number
   - It uses congestion control mechanism
8. **Multiplexing and Demultiplexing Service**
   - TCP does multiplexing and demultiplexing at sender and receiver ends respectively as a number of logical connections can be established between port numbers over a physical connection

13. Draw and explain TCP Segment structure / TCP header format

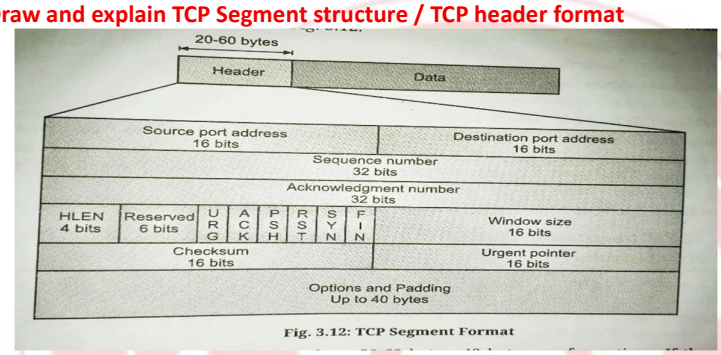


Fig. 3.12: TCP Segment Format

1. **Source port address**
   - 16-bit field
   - Identifies the application that is sending the data segment
   - 3 Ranges of port
   - 0 to 1023 → well known ports
   - 1024 to 49151 → registered ports
   - 49152 to 65535 → private ports
   - Ports are used by TCP as an interface to application layer
2. **Destination port address**
   - 16 bit field
   - Identifies destination→ receiving hosts
   - Destination port uses the same port number
   - Used to reassemble the message at the receiving end
3. **Sequence Number**
   - 32 bit field
   - Holds sequence number
     i.e. byte number of first byte sent in that segment
4. **Acknowledgement number**
   - 32 bit field
   - Identified next data byte that sender expects from receiver
   - It is an acknowledgment for previous byte being received successfully.
5. **Header length (HLEN)**
   - 4 bit fiend
   - Specifies length of TCP header in 32 bit words(4 byte words)
   - Minimum length - If header is of 20 bytes - field will hold 5 (5 X 4 = 20 bytes)
   - Maximum length - If header is of 60 bytes - field will hold 15 (15 X 4 = 60 bytes)
   - This field value is always between 5 to 15
6. **Reserved**
   - Reserved for future use
7. **Control Flags**
   6 Flags that control connection establishment, termination flow control, mode of transfer etc.

There are 6,1-bits
- **URG (Urgent Pointer)**
- Urgent pointer is valid
- **ACK(Acknowledgement)**
- Acknowledge number is valid
- **PSH (Push function)**
- Request for push (ex. Break request which can jump ahead the queued data
- **RST (Re set the connection)**
- Reset the connection
- **SYN1(Synchronise)**
- Synchronise sequence number
- **FIN (NO more data from sender)**
- Terminate the connection

8. **Window size**
   - 16-bit field
   - Tells the sender how much data the receiver is willing to accept
   - Maximum window size 65535

9. **Checksum**
- 16 bit value
- Sender calculates the value based on the content of TCP header and data field
- Receiver generates the same computation
- If values match- receiver is confident that segment is arrived intact

10. **Urgent pointer**
    - It is 16 bit field
    - It is used to point to a data that should be processed urgently
    - It tells receiver when the last byte of urgent data in the segment ends

11. **Options**
    - Provides additional functionality
    - Variable length field
    - Can not be larger than 40 bytes
    - Most common option is MSS – maximum segment size

12. **Padding**
    - As options may vary in size – it is necessary to pad TCP header with 0's to maintain standard size

13. **Data**
    - Variable length field 'Carries application data from sender to receiver
    - TCP header + data field – TCP segment

14. **Explain TCP connection establishment using Three-way handshaking mechanism**
- 3 phases in connection-oriented TCP transmission are:
  - **Connection establishment**

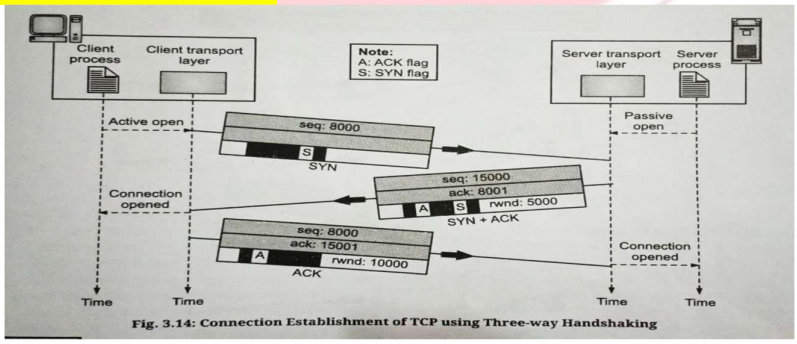


Fig. 3.14: Connection Establishment of TCP using Three-way Handshaking

- To establish a connection TCP uses 3-way handshaking
- Consider an application program known as client wants to make connection with another application program known as server using TCP.
- Process starts with server

- Server tells TCP that it is ready to accept connection.
- This request called as passive open
- Although TCP server is ready to accept connection from any machine in the world → it can not make the connection itself.
- Client program initiates a request for an active open
- Client that wishes to connect to an open server tells its TCP to connect to a particular server.
- TCP can now start the 3-way handshaking process.
- To establish a connection 3-way handshake occurs

1. **SYN segment**
   - Only SYN flag is set
   - SYN Segment is for synchronization of sequence number
   - Client chooses random number as first sequence number and sends it to server
   - Sequence number is called as ISN-Initial Sequence Number
   - SYN is a controlled segment – carries no data
   - It consumes one sequence number
   - When data transfer starts ISN is incremented by one
   - It contains no real data but imaginary bytes
2. **SYN +ACK segment**
   - It sets 2 flag bits – SYN and ACK
   - This segment cannot carry data
   - But it consumes one sequence number
   - Segment has dual purpose
   - 1st – SYN segment for communication in other direction→ server uses this segment to initialise sequence number
   - Server also acknowledge receipt of SYN from the client by setting ACK flag
   - It also needs to define receive window size
3. **ACK segment**
   - ACK segment acknowledges receipt of second segment with ACK flag and acknowledges field
   - It carries no data
   - Consumes no sequence number

➢ **Data Transfer**

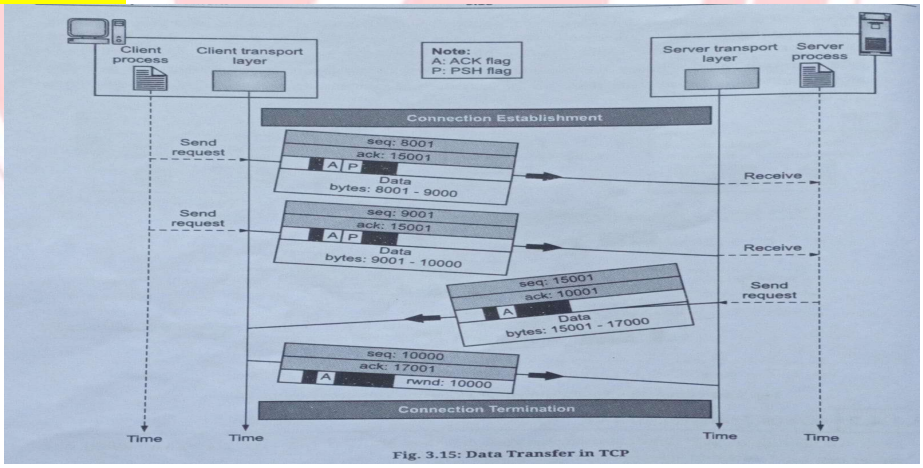


Fig. 3.15: Data Transfer in TCP

- After connection is established, bidirectional data transfer can take place.
- The client and server can both send data and acknowledgments.
- the client sends 2000 bytes of data in two segments.
- The server then sends 2000 bytes in one segment.
- The client sends one more segment.
- The first three segments carry both data and acknowledgment,
- But the last segment carries only an acknowledgment because there are no more data to be sent.
- The data segments sent by the client have the PSH (push) flag set
- so server TCP knows to deliver data to the server process as soon as they are received.
- The segment from the server→ does not set the push flag.

- Most TCP implementations have the option to set or not set this flag.
  - **Connection termination**

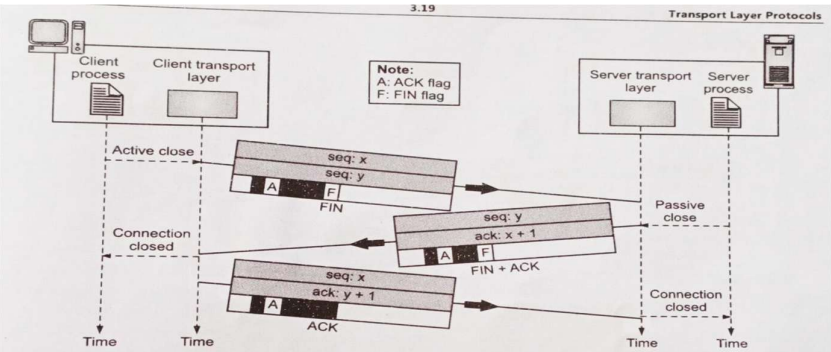


Fig. 3.16: three-way Handshaking for Connection Termination for TCP

- Any of the two parties involved in exchanging data (client or server) can close the connection, although it is usually initiated by the client.
- Most implementations today allow two options for connection termination→ three-way handshaking and four-way handshaking with a half-close option.
  - **3 -way handshaking**
- **FIN segment:**
- The client TCP, after receiving a close command from the
- client process, sends the first segment
- A FIN segment in which the FIN flag is set.
- It can include the last chunk of data sent by the client, or it can be just a control segment
- if it is only a control segment or number ifit does not carry data - it consumes only one sequence number.
- **FIN + ACK segment:**
- Server TCP- after receiving the FIN segment- informs its process of the situation and sends the second segment→ a FIN +ACK segment, to confirm the receipt of the FIN segment from the client
- At the same time to announce the closing of the connection in the other direction.
- This segment can also contain the last chunk of data from the server.
- The FIN +ACK segment consumes one sequence
- number if it does not carry data.
- **ACK segment:**
- client TCP sends the last segment→an ACK segment→ to confirm the receipt of the FIN segment from the TCP server.
- This segment contains the acknowledgment number
- which is 1 plus the sequence number received in the FIN segment from the server.
- This segment cannot carry data and consumes no sequence numbers.
  - **4-way handshaking in brief.**
- Four-way handshaking in TCP uses half-close option
- one end can stop sending data while still receiving data.
- Either end can issue a half-close
- It is normally initiated by the client.
- It can occur when the server needs all the data before processing can begin.
- A good example is sorting→When the client sends data to the server to be sorted→ server needs to receive all the data before sorting can start
- after sending
- all the data, can close the connection in the client-to-server direction.
- However, server-to-client direction must remain open to receive the sorted data.
- server, after receiving the data- needs time for sorting
- Its outbound direction must remain open.

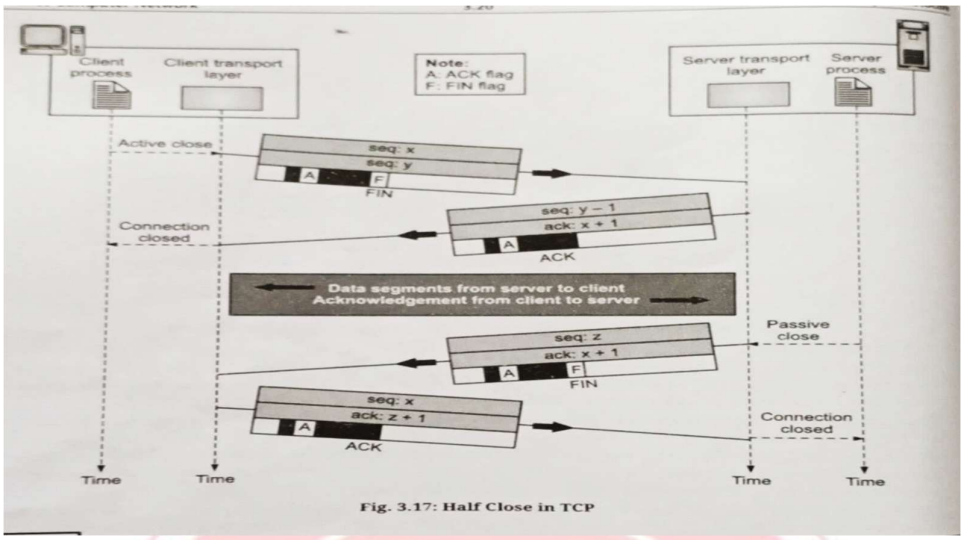


Fig. 3.17: Half Close in TCP

- Figure shows an example of a half-close.
- client half-closes the connection by sending a FIN segment.
- server accepts the half-close by sending ACK segment.
- The data transfer from the client to the server stops.
- Server can still send data.
- When the server has sent all the processed data, it sends a FIN segment→acknowledged by an ACK from the client.
- After half-closing of the connection→data can travel from the server to the client
- Acknowledgments can travel from the client to the server.
- Client cannot send any more data to the server.
- second segment (ACK) consumes no sequence number

15. What is SCTP and enlist SCTP services. Explain features of SCTP

Stream Control Transmission Protocol (SCTP)

- Transport layer protocol
- Combined features of UDP and TCP
- Can detect lost data
- Robust and flexible communication
- Can handle multimedia and stream traffic
- Lies between application layer and network layer

- services

1. **Process to process communication**
   - Done via following ports
   - Table lists some extra port numbers used by SCTP:

Sr. No.	Protocol	Port Number	Description
1.	IUA	9990	ISDN over IP.
2.	M2UA	2904	SS7 telephony signaling.
3.	M3UA	2905	SS7 telephony signaling.
4.	H.248	2945	Media gateway control.
5.	H.323	1718, 1719, 1720, 11720	IP telephony.
6.	SIP	5060	IP telephony.

2. **Multi stream facility**
   - Multi stream service provided to each connection is called association. If one stream is blocked other stream can deliver the data

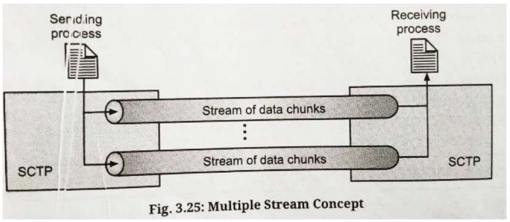


Fig. 3.25: Multiple Stream Concept

3. **Full duplex communication**
   - The data can flow in both direction at the same time
4. **Connection oriented services**
   - If user 1 wants to send and receive message from user 2:
   1. 2 SCTPs establish the connection with each other
   2. Once connection is established data gets exchanged in both directions.
   3. Finally, the association is terminated.
5. **Reliability**
   - SCTP uses acknowledgement mechanism to check the arrival of data
6. **Multihoming**
   - Sender and receiver both are multihomed → connected to more than 1 physical address with multiple IP addresses

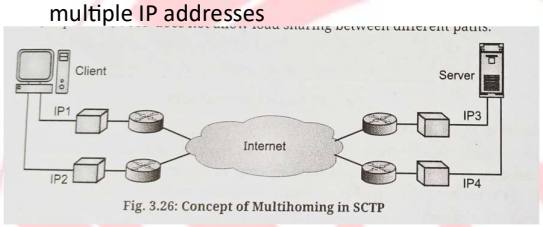


Fig. 3.26: Concept of Multihoming in SCTP

Only one pair of IP can be chosen for normal communication → alternative is used if main choice fails

features of SCTP

- **Transmission Sequence Number (TSN)**
  - Unit of SCTP is Data Chunk. Data transfer is controlled by numbering data chunks. TSN is used to assign numbers to data chunks
- **Stream Identifier (SI)**
  - 16-bit number starts with zero needed to identify streams.
    Each data chunks need to carry SI in the header so that it is properly placed in its stream on arrival.
- **Packets**
  - Data is carried out in the form of data chunks and control information is carried out in control chunks. Both are packed together in the packet.
- **Multihoming**
  - It allows both ends to define multiple IP addresses for communication. Only one is primary and rest are alternative addresses
- **Flow Control**
  - Like TCP, SCTP executes flow control to prevent overwhelming the receiver.
- **Error Control**
  - Like TCP, SCTP executes error control to support reliability. TSN numbers and acknowledgment numbers are used for error control.
- **Congestion Control**
  - Like TCP, SCTP executes congestion control to decide how many data blocks can be inserted into the network.

16. **What is DNS? How to map domain name with IP address**
    - DNS ensures the internet is not only user-friendly but also works smoothly, loading whatever content we ask for quickly and efficiently.
    - It allows the user to access remote system by entering human readable device hostnames instead of IP address. It translates domain name into IP addresses so browser can load internet resources.
    - It translates human readable domain names into the numerical identifiers associated with networking equipment, enabling devices to be located and connected worldwide.

- Analogous to a network "phone book," DNS is how a browser can translate a domain name (e.g., "facebook.com") to the actual IP address of the server, which stores the information requested by the browser.
- TCP/IP uses DNS client and DNS server to map a name to an address
- It can also perform reverse mapping

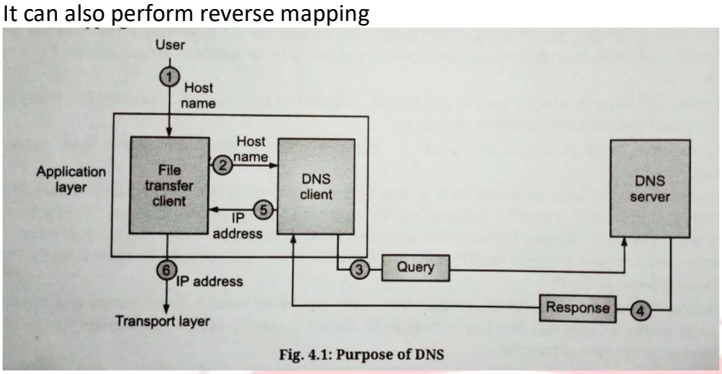


Fig. 4.1: Purpose of DNS

- When user wants to use file transfer client to access corresponding file transfer server running on remote host
- If user knows only file transfer server name.
- But TCP/IP suite needs the IP address of file transfer server → to make connection
- Six steps to map host name to an IP address
- **Step 1**: The user passes host name to file transfer client
- **Step 2** : file transfer client passes the host name to DNS client
- **Step 3**: each computer after being booted knows the address of one DNS server. The DNS client send s a messages to DNS server with a query that gives file transfer server name using known IP address of DNS server
- **Step 4**: DNS server responds with IP address of the desired file transfer server
- **Step 5**: The client passes IP address to file transfer client
- **Step 6**: File transfer client now uses received IP address to access file transfer server
- DNS is a network protocol used to translate host names to IP addresses

17. what is a name server? What are its different types?

- To distribute information among many computers DNS uses DNS name servers
- Domain name system is maintained by distributed database system
- Nodes in this system are the name servers
- Each domain has at least one DNS server → that publishes information about domain and the name servers belonging to it
- Root is standalone → can create many domains (subtrees) as first level nodes
- Created domain are very large so they are divided into subdomains
- Each and every server is responsible for either large or small domain

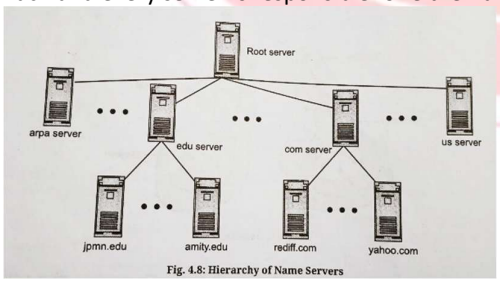


Fig. 4.8: Hierarchy of Name Servers

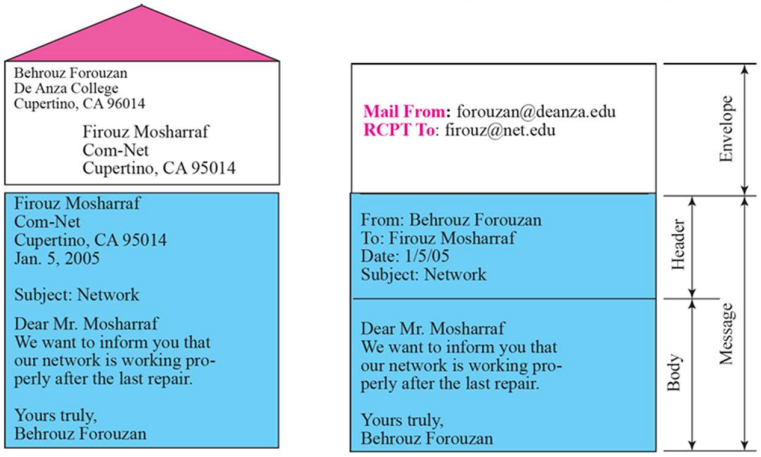
Types of name servers

1. **Root server**
   - Usually does not store any information
   - it assigns authority to other servers
   - It keeps reference of other servers
   - There are several root servers → covering whole domain name space
   - Root servers are distributed all around the world

2. **Primary server**
    - It stores a file about the zone for which it is an authority
    - It is responsible for- creating, maintaining and updating the zone file
    - It stores zone file on a local disk
3. **Secondary server**
    - It's a server that transfers complete information about a zone from another server →and stores the file on local disc
    - It neither creates nor updated the zone file
    - If updating is required it must be done by primary server
    - Primary server sends updated version to secondary server

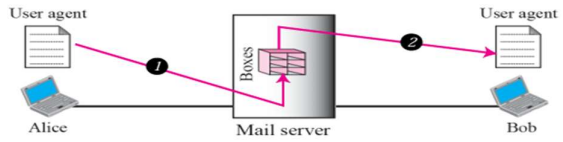
18. Draw format of E-mail. Explain 4 scenarios of email- architecture

Format of email (Email envelope)



Email architecture and services – 4 scenarios

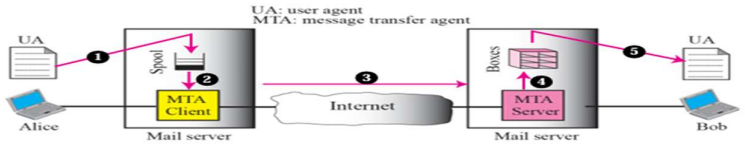
1. **First scenario**



Scenario 1: Both Users on the Same System

- **Setup:** Sender and recipient use the **same computer or local system.**
- **Flow:**
    - The sender composes the email using a User Agent (UA).
    - The message is **stored directly in the recipient's mailbox** on the same system.
- No need for a mail server or internet connection—this is a local delivery.

2. **Second scenario**



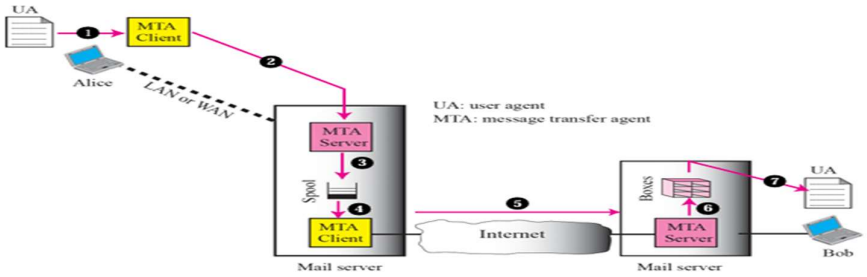
Scenario 2: Users on Different Systems

- **Setup:** Sender and recipient are on **separate computers**.
- **Flow:**
    - The sender's UA sends the message to their Mail Transfer Agent (MTA).
    - The MTA uses **SMTP** to send the message to the recipient's MTA.

- The recipient retrieves the message using **POP3** or **IMAP** via their UA.

- **Internet connection is required for communication between MTAs.**

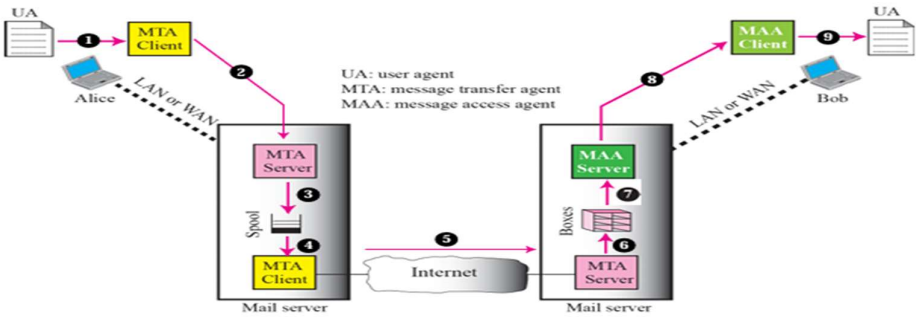
3. **Third scenario**



Scenario 3: LAN/WAN Connection

- **Setup:** Sender and recipient are on **different systems connected via a Local Area Network (LAN) or Wide Area Network (WAN).**
- **Flow:**
  - Similar to Scenario 2, but the **MTAs may be within the same organization or network.**
  - Faster delivery due to internal routing.
  - Often uses **internal mail servers** for routing and storage.

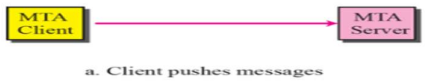
4. **Fourth scenario**



Scenario 4: Cloud-Based or Webmail Services

- **Setup:** Users use services like Gmail, Outlook, or Yahoo Mail.
- **Flow:**
  - The sender accesses the email service via a **web interface or app**.
  - The cloud-based mail server handles **SMTP**, **POP3**, and **IMAP** protocols.
  - The recipient accesses their mailbox through the **same or a different service**.
- Highly scalable, supports attachments, filtering, and spam protection.

Push versus pull:-



a. Client pushes messages

b. Client pulls messages

19. Describe SMTP components and working of message transfer agent

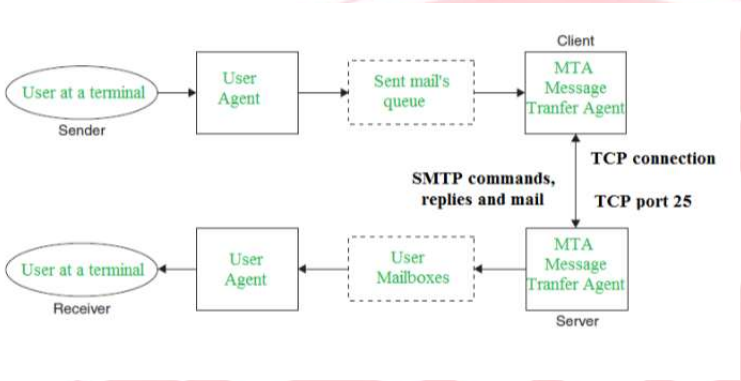
SMTP (Simple Mail Transfer Protocol)

- SMTP is an **Internet standard** for transmitting email across IP networks.
- It is a **connection-oriented**, **text-based** protocol.
- Communication occurs via commands and data streams over a TCP connection.
- SMTP is used for **sending** and **receiving** emails over TCP/IP architecture.

- **Email systems rely on Message Transfer Agents (MTAs) to implement SMTP.**
- MTAs handle **sending**, **storing**, and **receiving** emails on mail servers.
- SMTP defines the formal protocol for client-server communication between MTAs.
- It ensures **reliable** email **transmission** across the Internet.

Model of SMTP

- The user interacts with a User Agent (UA) like **Microsoft Outlook, Netscape, or Mozilla to send emails.**
- The Message Transfer Agent (MTA) is responsible for exchanging mail over a TCP connection, typically on **TCP port 25**.
- Users do not directly deal with the MTA; it's managed by the system administrator.
- The MTA maintains a mail queue to retry delivery if the recipient is temporarily unavailable.
- Once delivered, the MTA stores the email in the recipient's mailbox.
- The recipient accesses the email using their User Agent (UA) at their terminal.



Components of SMTP

1. Mail User Agent (MUA)

- Interface used by the user to compose, send, and read emails.
- Examples: Gmail, Outlook, Thunderbird.

2. Mail Submission Agent (MSA)

- Accepts email from the MUA.
- Validates and forwards it to the MTA for delivery.

3. Mail Transfer Agent (MTA)

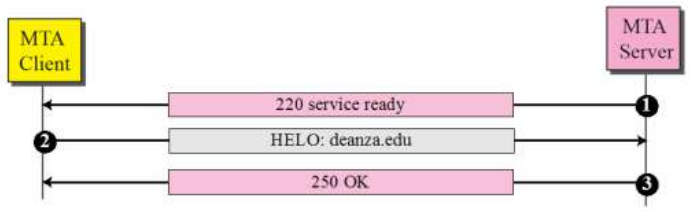
- Transfers email between servers using SMTP.
- Handles routing, queuing, and retries.

4. Message Delivery Agent (MDA)

- Delivers the email to the recipient's mailbox.
- Works with POP3 or IMAP for message retrieval.

Working or operation of SMTP

Step 1: connection establishment



Server Greeting

- The MTA Server sends a message: 220 service ready This indicates that the server is ready to begin an SMTP session.

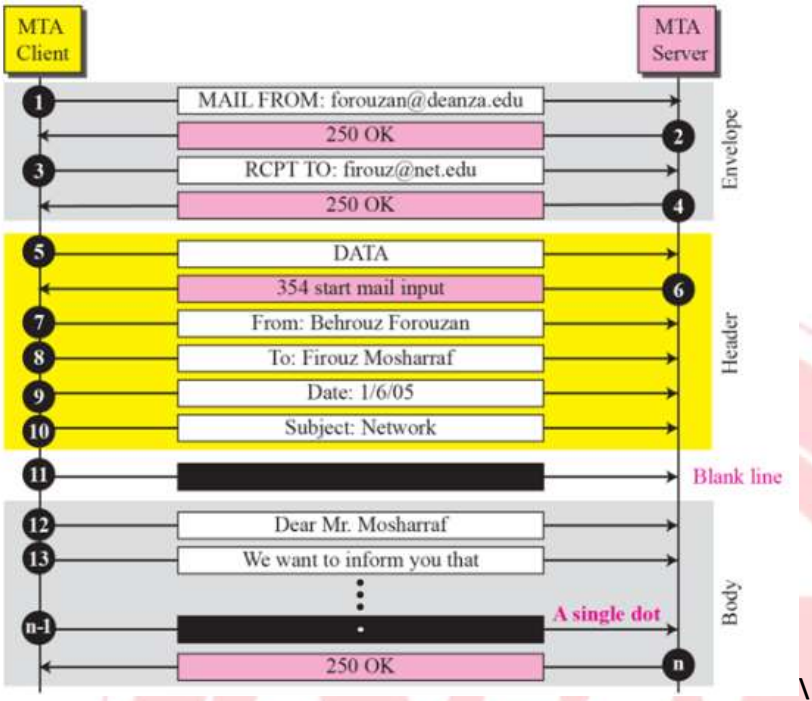
Client Introduction

- The MTA Client responds with: HELO: deanza.edu This command identifies the client to the server using its domain name (deanza.edu).

Server Acknowledgment

- The MTA Server replies: 250 OK This confirms that the server has accepted the client's identity and is ready to proceed with further commands (like MAIL FROM, RCPT TO, etc.).

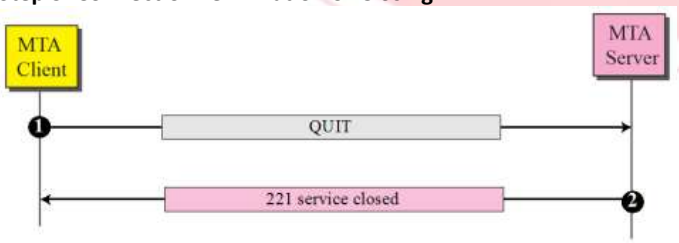
Step 2: Message Transfer



\

1. **Envelope Section**
   - **MAIL FROM:** specifies the sender's email address.
   - **RCPT TO:** specifies the recipient's email address.
   - Server responds with 250 OK to confirm each command.
2. **Header Section**
   - Begins with the DATA command.
   - Server replies with 354 Start mail input.
   - Includes fields like From, To, Date, and Subject.
3. **Blank Line**
   - Separates the header from the body of the email.
4. **Body Section**
   - Contains the actual message content.
   - Ends with a single dot (.) on a new line to signal completion.
   - Server replies with 250 OK to confirm successful delivery.

Step 3: Connection Termination or Closing



SMTP Connection Termination

1. **QUIT Command**
   - The MTA Client sends the QUIT command to the MTA Server.
   - This signals that the client has finished sending emails and wants to close the session.
2. **Server Response**
   - The MTA Server replies with 221 service closed.
   - This confirms that the server is ending the connection and releasing resources.

20. What is PGP? Explain how it uses key rings?

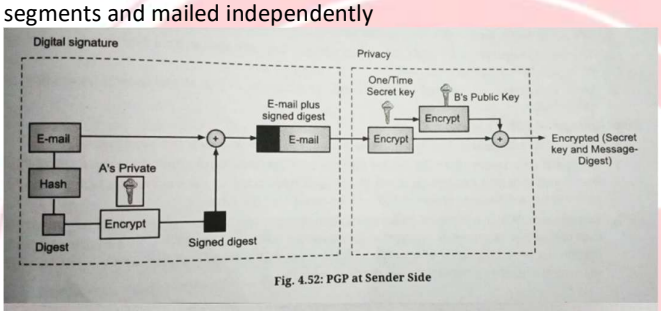
- **Pretty Good Privacy (PGP)**
- Invented by Phil Zimmermann
- PGP provides email with → privacy, integrity, authentication, non-repudiation
- It is used to create secure email message
- It uses secret key encryption
- It is open source
- Provides compression by ZIP algorithm
- **Services of PGP**
- **Authentication→** using SHA-1 algorithm
- **Confidentiality→** using 3-DES
- **Confidentiality and Authentication→** for same message
- **Compression→** before encryption to reduce size
- **E-mail Compatibility**
- **Segmentation→** max length of e-mail is 50000 octets→ if more broken down into smaller segments and mailed independently
- 

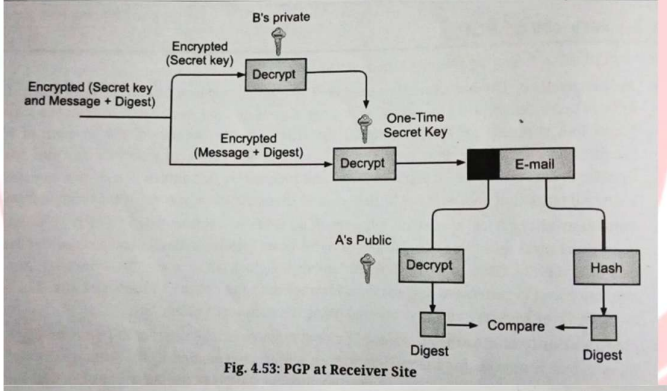
Fig. 4.52: PGP at Sender Side
- 

Fig. 4.53: PGP at Receiver Site
- Following are the steps taken by PGP to create secure e-mail at the sender site:
  - The e-mail message is hashed by using a hashing function to create a digest.
  - The digest is then encrypted to form a signed digest by using the sender's private key, and then signed digest is added to the original email message.
  - The original message and signed digest are encrypted by using a one-time secret key created by the sender.
  - The secret key is encrypted by using a receiver's public key.
  - Both the encrypted secret key and the encrypted combination of message and digest are sent together.
- Following are the steps taken to show how PGP uses hashing and a combination of three keys to generate the original message:
  - The receiver receives the combination of encrypted secret key and message digest is received.
  - The encrypted secret key is decrypted by using the receiver's private key to get the one-time secret key.
  - The secret key is then used to decrypt the combination of message and digest.
  - The digest is decrypted by using the sender's public key, and the original message is hashed by using a hash function to create a digest.
  - Both the digests are compared if both of them are equal means that all the aspects of security are preserved.
- **PGP Key Rings**

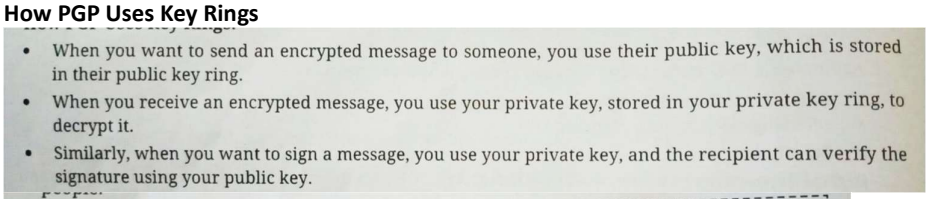
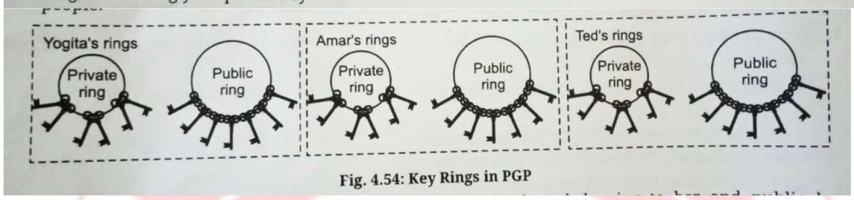
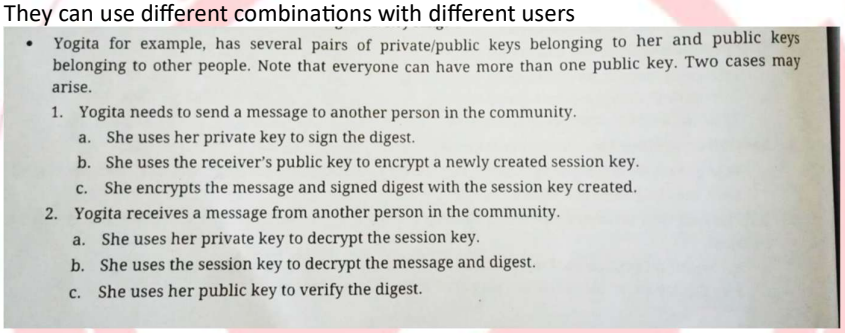
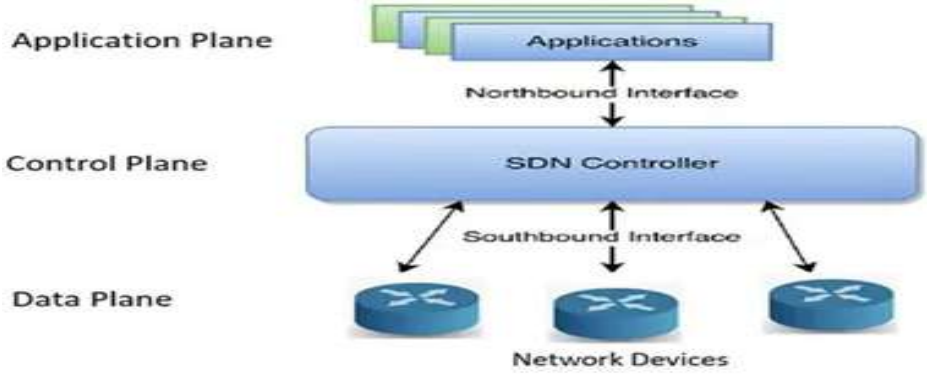
- **Key rings:** database that store cryptographic keys used for→ encryption, decryption, authentication
- It is a file that stores public or private keys
- Mainly 2 types of key rings:
- **Public Key Ring:** stores public keys of other users and its own as well
- **Private Key Ring:** stores users private key→ used for decryption and creating digital signature
- **How PGP Uses Key Rings**
- 
  - When you want to send an encrypted message to someone, you use their public key, which is stored in their public key ring.
  - When you receive an encrypted message, you use your private key, stored in your private key ring, to decrypt it.
  - Similarly, when you want to sign a message, you use your private key, and the recipient can verify the signature using your public key.
- 

  Fig. 4.54: Key Rings in PGP
- Each user has two sets of key rings: public and private
- They can use different combinations with different users
- 
  - Yogita for example, has several pairs of private/public keys belonging to her and public keys belonging to other people. Note that everyone can have more than one public key. Two cases may arise.
    1. Yogita needs to send a message to another person in the community.
       a. She uses her private key to sign the digest.
       b. She uses the receiver's public key to encrypt a newly created session key.
       c. She encrypts the message and signed digest with the session key created.
    2. Yogita receives a message from another person in the community.
       a. She uses her private key to decrypt the session key.
       b. She uses the session key to decrypt the message and digest.
       c. She uses her public key to verify the digest.

21. Explain SDN architecture with diagram and working

Architecture

- Architecture of SDN consists of **three main layers**



1. **Infrastructure Layer (data plane)**

- Also known as: Forwarding Layer
- **Function:** Handles actual data transmission across the network.
- **Components**: Physical and virtual switches, routers, and other network devices.
- **Role:**
  - Forwards packets based on instructions from the control layer.
  - Collects network statistics and status information.
- **Key Feature:** Devices are simplified and programmable, often using protocols like OpenFlow.

2. Control Layer (Control Plane)

- Also known as: SDN Controller
- **Function:** Acts as the brain of the network.
- **Components:** Centralized software-based controller.
- **Role:**
  - Makes decisions about traffic routing and network policies.
  - Communicates with infrastructure devices to enforce rules.
  - Provides abstraction to the application layer.
  - **Key Feature:** Centralized intelligence for dynamic and flexible network control.

3. Application Layer

- **Function:** Hosts network applications and services.
- **Components:** Software applications like firewalls, load balancers, traffic analysers.
- **Role:**
  - Defines network behaviour and policies.
  - Requests services from the control layer via APIs.
- **Key Feature:** Enables innovation and customization without changing hardware

How They Work Together

- Application Layer tells the Control Layer what it wants.
- Control Layer translates those requests into instructions.
- Infrastructure Layer executes those instructions to manage traffic.

22. Explain Edge Computing and Edge Networking

Definition of Edge Computing

Edge computing is a distributed computing model that processes data closer to where it's generated—such as IoT devices or local servers—rather than relying solely on centralized cloud data centres.

Components of Edge Computing

1. Perception Layer

- Responsible for data **collection** from sensors, devices, and IoT endpoints
- Acts as the **interface** between the physical world and digital systems
- **Examples**: temperature sensors, cameras, RFID readers

2. Networking Layer

- **Transmits** data between edge devices and other layers
- **Ensures** connectivity, routing, and communication protocols
- **Technologies** include Wi-Fi, 5G, Ethernet, LPWAN

3. Edge Computing Layer

- Performs local data processing, filtering, and analytics
- **Reduces latency** by minimizing reliance on cloud data centres
- **Hosts** edge servers, gateways, and micro data centres

4. Application Processing Layer

- **Executes** business logic and application-specific tasks
- **Interfaces** with cloud services or enterprise systems
- **Supports** real-time decision-making and service delivery

Main Components of edge computing systems are:

1. Edge Devices: Sensors, cameras, IoT devices that generate and sometimes process data locally.
2. Edge Gateways: Act as intermediaries between edge devices and the cloud; handle protocol translation, preprocessing, and security.
3. Edge Servers: Perform intensive local processing and analytics; often located near data sources to reduce latency.
4. Edge Local Storage: Temporarily stores data at the edge for quick access and reduced cloud dependency.
5. Network Infrastructure: Includes routers, switches, and communication links that connect edge components and ensure data flow.

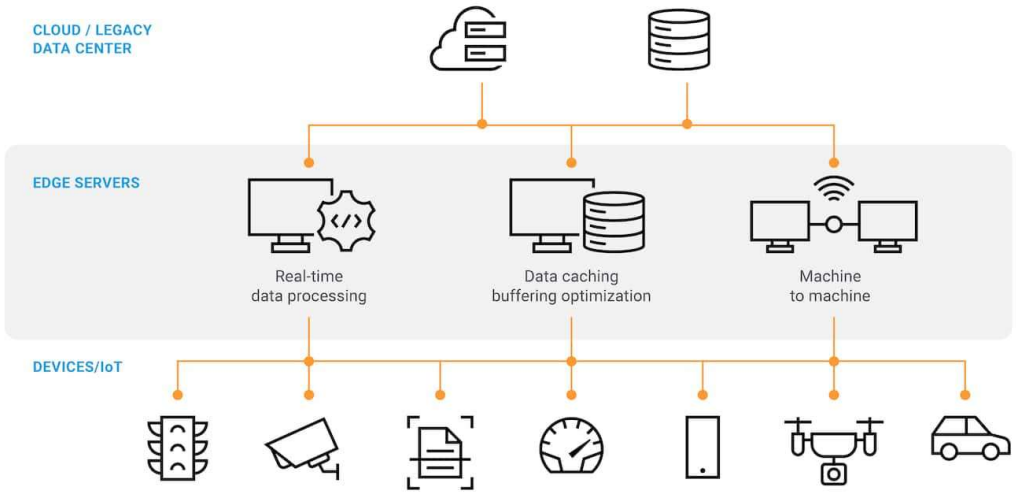
6. Edge Computing Platform/ Management Software: Manages deployment, orchestration, monitoring, and updates of edge applications and services.
7. Cloud or Centralized Data Centre: Provides long-term storage, deep analytics, and centralized control when needed.

8. Security Components: Protects data and devices through encryption, authentication, firewalls, and intrusion detection systems.

Edge Networking: Definition of Edge Networking

- Edge networking refers to the architecture that places data processing and network resources closer to the devices generating data—at the "edge" of the network—rather than relying solely on centralized cloud servers.
- Edge networking is foundational for modern innovations like smart cities, 5G, and real-time analytics.
- It is beneficial for **applications** like
  - Internet of Things (IoT)
  - Artificial Intelligence (AI)

Components of Edge Networking



1. **Edge Devices:** Sensors and IoT hardware that generate and sometimes process data locally.
2. **Edge Nodes / Edge Services**: Intermediate systems that perform computation and service delivery near the data source.
3. **Network Infrastructure:** Connectivity backbone including routers, switches, and communication protocols**.**
4. **Cloud or Central Data Centre:** Centralized systems for deep analytics, storage, and global coordination.
5. **Security Components:** Tools like firewalls, encryption, and access controls to protect edge data and devices.
6. **Software and Middleware:** Platforms that manage orchestration, data flow, and integration across edge and cloud.

23. Define VOIP and explain how it works?

Voice Over Internet Protocol (VoIP)

- Voice Over Internet Protocol (VoIP) is a technology that allows voice communication over the internet instead of traditional phone lines.

How VoIP Works

- **Definition:** VoIP (also called IP telephony) enables voice communication over IP networks like the internet.

- **Function:** It converts voice into digital packets using compression, sends them over the internet, and reassembles them into sound at the receiver's end**.**
- **Process:**

1. Your phone connects to a switch/router in your LAN.
2. When you dial a number, your IP phone signals the VoIP service provider.
3. The provider sets up the call and exchanges data packets.
4. Your IP phone converts these packets back into audible sound.

- **Requirements:** A broadband connection and a VoIP service provider are essential.
- **Difference from Analog:** Unlike traditional phone systems (DSL, cable), VoIP uses digital packet transmission.

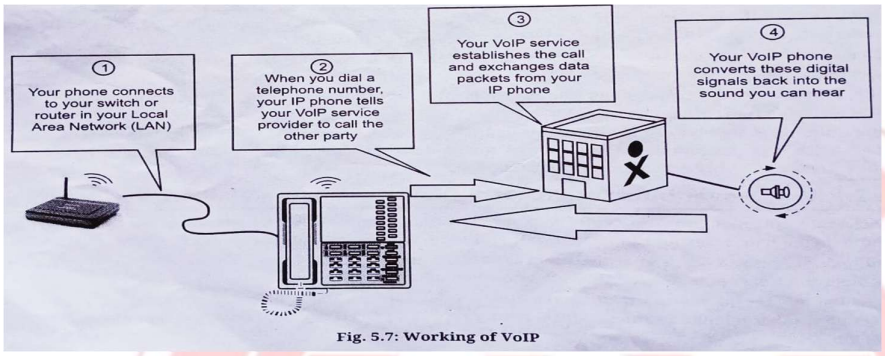


Fig. 5.7: Working of VoIP

VoIP Services:

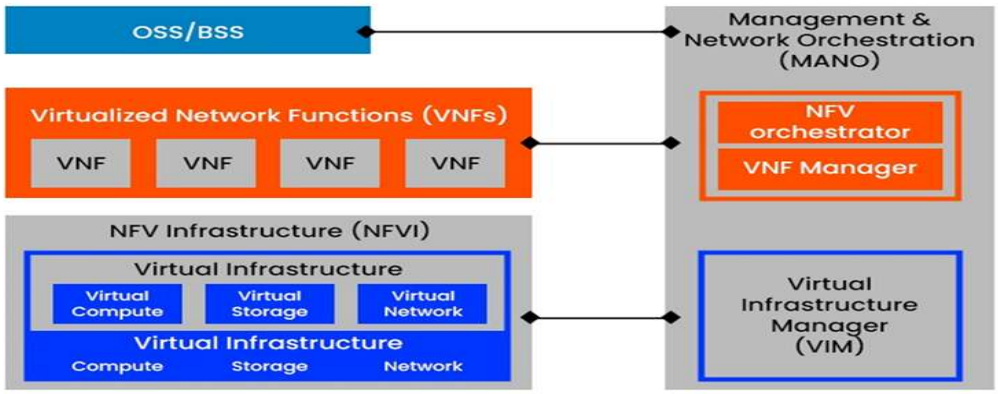
- VoIP (Voice over Internet Protocol) services allow users to **make voice and video calls** using the **internet** instead of traditional phone lines. These services convert voice into digital packets and transmit them over IP networks, offering flexibility, cost savings, and advanced features

24. Explain Working of NFV and Components of NVF Architecture

- NFV is a modern networking approach that replaces traditional hardware-based network functions (like firewalls, routers, and load balancers) with software-based solutions.
- These functions run on standard servers or virtual machines, rather than specialized hardware.
- NFV improves flexibility, scalability, and cost-efficiency by allowing service providers to deploy and manage network services faster.
- It supports technologies like cloud computing, SDN, and IoT, and is essential for 5G infrastructure.

Working of NFV

Components of NVF Architecture



1. **Virtualization Layer**

- **Abstracts** physical hardware into virtual resources

- Enables **multiple VNFs** to run on shared infrastructure
- Uses **hypervisors** (e.g., KVM, VMware) or **containers** (e.g., Docker)

2. Virtual Network Functions (VNFs)

- Software versions of traditional network functions
- **Examples**: firewall, router, load balancer, NAT
- Can be deployed, scaled, and updated independently

3. NFV Infrastructure (NFVI)

- **Physical and virtual resources**: compute, storage, and networking
- **Hosts** the VNFs and **supports** their operation
- Includes hardware + virtualization layer + resource managers

4. Management and Orchestration (MANO)

- Controls **lifecycle** and **automation** of VNFs and services
- **Subcomponents**:
  - NFV Orchestrator (NFVO) – manages network services
  - VNF Manager (VNFM) – handles VNF lifecycle
  - Virtualized Infrastructure Manager (VIM) – manages NFVI resources

25. What is Flow control and Error control in TCP?

TCP Flow Control is a protocol designed to manage the data flow between the user and the server. It ensures that there is a specific bandwidth for sending and receiving data so the data can be processed without facing any major issues. In order to achieve this, the TCP protocol uses a mechanism called the sliding window protocol

Error control in TCP is mainly done through the use of **three simple techniques** :

1. **Checksum –** Every segment contains a checksum field which is used to find corrupted segments. If the segment is corrupted, then that segment is discarded by the destination TCP and is considered lost.
2. **Acknowledgement –** TCP has another mechanism called acknowledgement to affirm that the data segments have been delivered. Control segments that contain no data but have sequence numbers will be acknowledged as well but ACK segments are not acknowledged.

Retransmission – When a segment is missing, delayed to deliver to a receiver, corrupted when it is checked by the receiver then that segment is retransmitted again. Segments are retransmitted only during two events: when the sender receives three duplicate acknowledgements (ACK) or when a retransmission timer expires.

• **Retransmission after RTO:** TCP always preserves one retransmission time-out (RTO) timer for all sent but not acknowledged segments. When the timer runs out of time, the earliest segment is retransmitted. Here no timer is set for acknowledgement. In TCP, the RTO value is dynamic in nature and it is updated using the round trip time (RTT) of segments. RTT is the time duration needed for a segment to reach the receiver and an

acknowledgement to be received by the sender.

• **Retransmission after Three duplicate ACK segments:** RTO method works well when the value of RTO is small. If it is large, more time is needed to get confirmation about whether a segment has been delivered or not. Sometimes one segment is lost and the receiver receives so many out-of-order segments that they cannot be saved. In order to solve this situation, three duplicate acknowledgement method is used and missing segment is retransmitted immediately instead of retransmitting already delivered segment. This is a fast retransmission because it makes it possible to quickly retransmit lost segments instead of waiting for timer to end.