DCN ASSIGNMENTS

DCN ASSIGNMENT - 4

Q.1 Explain subnetting with an appropriate example. (M 1)

—> Subnetting is the process of dividing a large network into smaller, more manageable sub-networks (subnets). It helps in reducing network traffic, improving performance, and enhancing security.

Key Concepts in Subnetting:

1. **IP Address:** An IP address is divided into two parts:
   - **Network part**: Identifies the network.
   - **Host part**: Identifies the specific device within the network.
2. **Subnet Mask:** A subnet mask is used to determine which portion of the IP address belongs to the network and which portion is for hosts.

Steps for Subnetting:

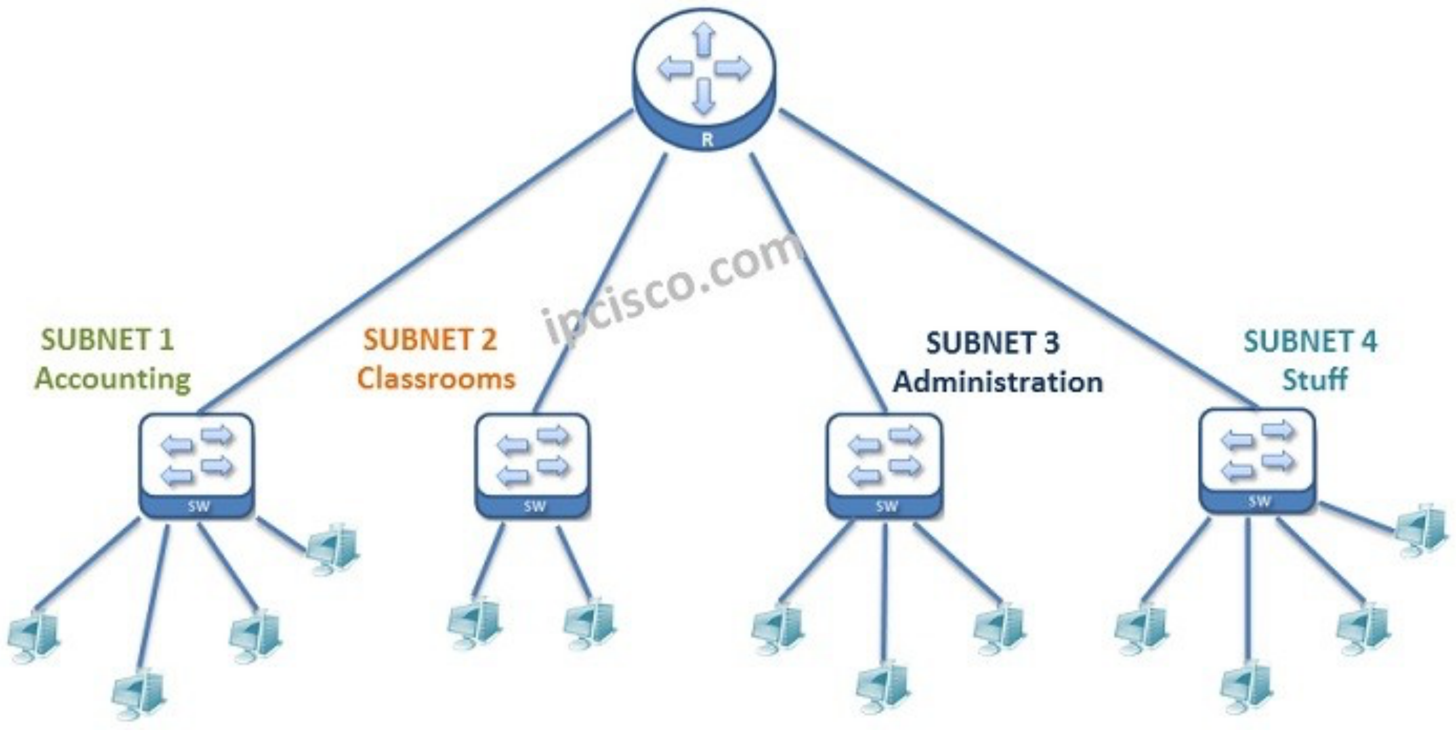
1. **Determine the network requirement**: How many subnets and hosts are needed.
2. **Choose an appropriate subnet mask**: Modify the subnet mask to divide the network into the required number of subnets.

Example:

Let's say we have an IP address: 192.168.10.0/24. This address belongs to Class C, so the default subnet mask is 255.255.255.0. This means the network can accommodate 256 IP addresses (2^8 = 256).

If we need to divide this network into 4 smaller subnets, we can borrow 2 bits from the host portion of the IP address. This changes the subnet mask to /26 (255.255.255.192).

- Total subnets: 2^2 = **4 subnets**
- Each subnet will have **64 IP addresses** (2^6 = 64).



Q.2 Explain CIDR with an appropriate example.

—> Classless Inter-Domain Routing (CIDR) is a method of allocating IP addresses and routing that replaces the old class-based system (Class A, B, C). CIDR allows for more efficient and flexible allocation of IP addresses by introducing the concept of prefix length, which defines how many bits are used for the network portion of the address.

Instead of using fixed-size subnet masks (e.g., /8, /16, or /24), CIDR uses variable-length subnet masks (VLSM), which provides more flexibility in network design.

Key Concepts of CIDR:

1. **CIDR Notation:**
   - An IP address in CIDR notation is written as:
     <IP Address>/<Prefix Length>
   - The prefix length indicates how many bits are used for the network portion. For example, 192.168.10.0/24 means the first 24 bits of the IP address represent the network.
2. **Efficient IP Address Allocation:**
   - CIDR allows networks to be divided into smaller or larger groups based on need, rather than using fixed classes (A, B, C). This helps in conserving IP addresses.
3. **Aggregation of Routes (Supernetting):**
   - CIDR also enables the aggregation of multiple networks into a single routing entry, reducing the size of routing tables.

Example:

An IP address with CIDR notation: 192.168.10.0/22

- The **/22** means the first 22 bits represent the network portion, and the remaining **10 bits** are for hosts.
- This allows for **2^(32-22) = 1024 IP addresses**, including network and broadcast addresses.

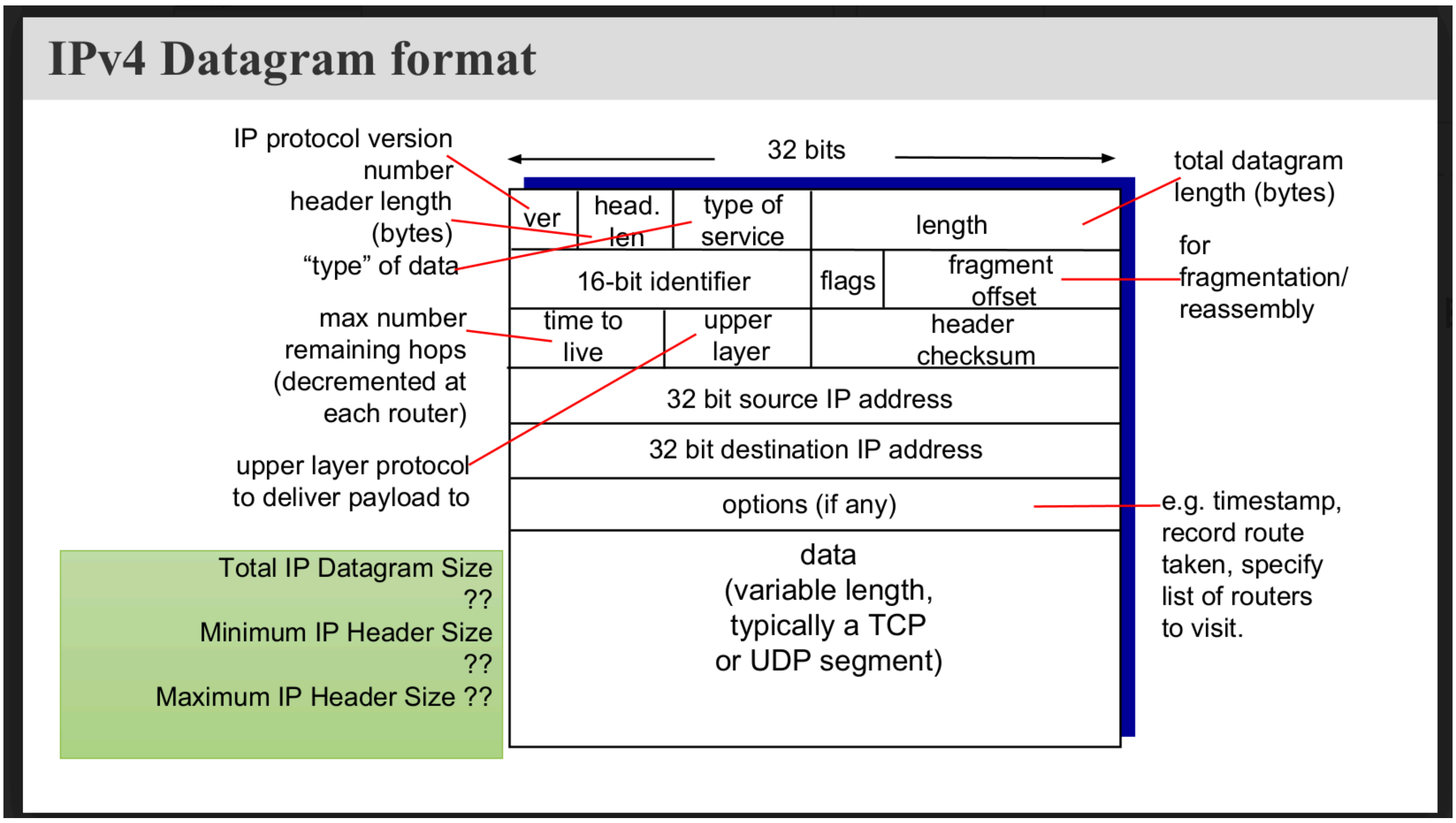
The IP range for 192.168.10.0/22:

- **Network Address:** 192.168.10.0
- **Broadcast Address:** 192.168.13.255
- **Usable IPs:** 192.168.10.1 to 192.168.13.254

Q.3 Explain IPV4 datagram header format.

—> An IPv4 datagram (packet) is the fundamental unit of data transmitted across the Internet. The IPv4 header contains essential information required for routing and delivery of the datagram from the source to the destination.

The IPv4 header is composed of a fixed part of 20 bytes (160 bits) and an optional part. It contains various fields that describe the packet, its contents, and how it should be processed.


IPv4 Datagram format
32 bits
IP protocol version number
header length (bytes)
"type" of data
max number remaining hops (decremented at each router)
upper layer protocol to deliver payload to
ver
head. len
type of service
length
16-bit identifier
flags
fragment offset
time to live
upper layer
header checksum
32 bit source IP address
32 bit destination IP address
options (if any)
data (variable length, typically a TCP or UDP segment)
total datagram length (bytes)
for fragmentation/ reassembly
e.g. timestamp, record route taken, specify list of routers to visit.
Total IP Datagram Size ??
Minimum IP Header Size ??
Maximum IP Header Size ??

**(The diagram is only till the options field and data is not included in this question)

IPv4 Header Fields

Field	Size (bits)	Description
Version	4	IP version (IPv4 = 4).
IHL	4	Header length in 32-bit words (min 5 = 20 bytes).
Type of Service	8	Specifies priority and quality of service.
Total Length	16	Total size of the packet (header + data).
Identification	16	Identifies fragments of the original packet.
Flags	3	Control fragmentation (e.g., "Don't Fragment" flag).
Fragment Offset	13	Position of a fragment in the original datagram.

TTL	8	Limits the lifespan of the packet (in hops).
Protocol	8	Identifies the higher-level protocol (TCP = 6, UDP = 17).
Header Checksum	16	Error-checking for header integrity.
Source IP	32	Sender's IP address.
Destination IP	32	Receiver's IP address.
Options (optional)	Variable	Additional optional information (e.g., security).

Q.4 Describe NAT with an appropriate example.

—> Network Address Translation (NAT) is a technique used to map private IP addresses within a local network to a public IP address before data is sent to the internet. NAT helps conserve public IP addresses and adds an extra layer of security to internal networks.

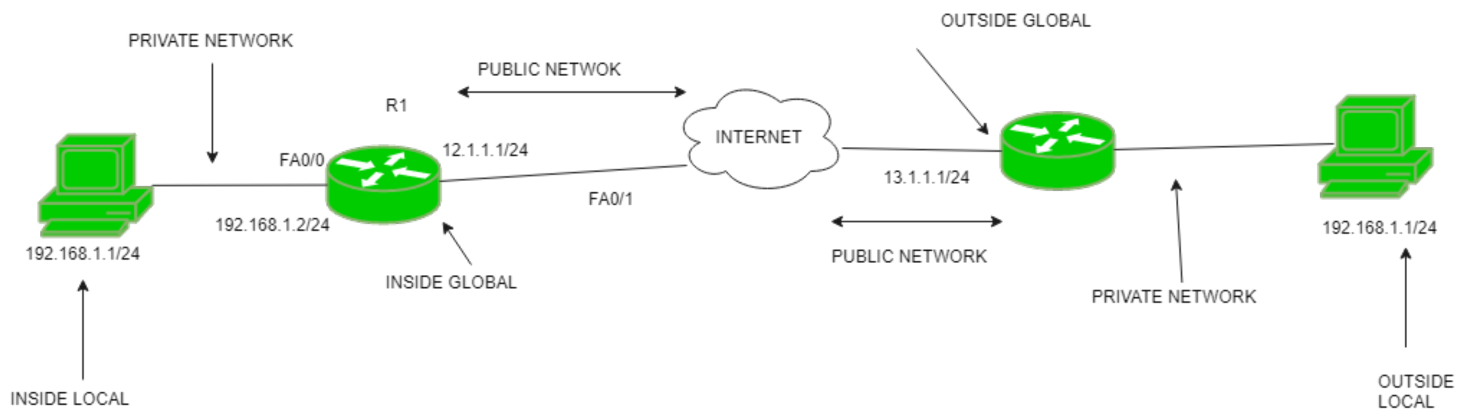
Types of NAT:

1. **Static NAT**: One-to-one mapping between a private and a public IP address. Example: 192.168.1.2 → 203.0.113.2.
2. **Dynamic NAT**: Maps a private IP address to any available public IP from a pool. Example: 192.168.1.3 → one of many public IP addresses (203.0.113.3, 203.0.113.4, etc.).
3. **PAT (Port Address Translation)**: Many private IPs can be mapped to a single public IP, using different port numbers for each session. Example: 192.168.1.4:5678 → 203.0.113.5:5678.

Example:

Let's say a local network has private IP addresses (e.g., 192.168.1.10) but wants to access the internet. The router has a public IP address (e.g., 203.0.113.5) assigned by the ISP.

1. **Before NAT**:
   - Device's private IP: **192.168.1.10**
   - Tries to connect to an external website.
2. **NAT Process**:
   - NAT converts **192.168.1.10** (private) to **203.0.113.5** (public).
   - The external website sees the request coming from **203.0.113.5**.
3. **After NAT**:
   - The response from the website is sent to **203.0.113.5**.
   - NAT translates it back to the original private IP **192.168.1.10**.



Q.5 Describe DHCP with an appropriate example. (M 2)

—> DHCP(Dynamic Host Configuration Protocol) is a network management protocol used to dynamically assign IP addresses and other network configuration parameters (e.g., subnet mask, gateway, DNS) to devices on a network, enabling them to communicate on an IP-based network.

Key Functions of DHCP:

1. **IP Address Assignment**: DHCP automatically assigns a unique IP address to each device connected to the network.
2. **Network Configuration**: Along with the IP address, DHCP provides other configuration settings, such as:
   - **Subnet Mask**: Defines the network segment.
   - **Default Gateway**: Specifies the router for outbound traffic.
   - **DNS Server**: Provides the address of the server that translates domain names to IP addresses.
3. **Address Lease Time**: DHCP assigns IP addresses for a limited time (lease), after which they must be renewed by the client.

Example of DHCP in Action:

A laptop connects to a network without a pre-configured IP address and uses DHCP to obtain one.

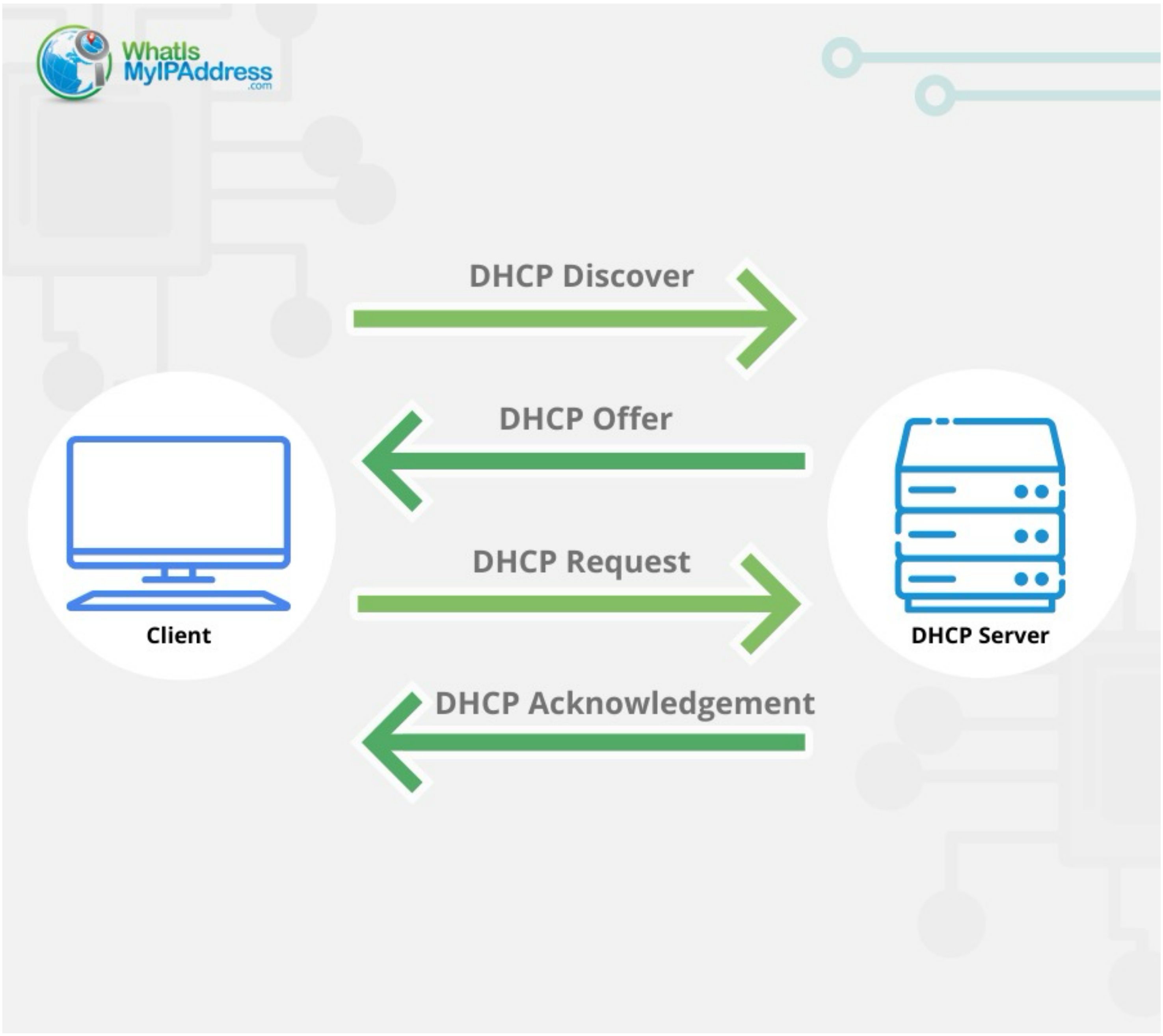
1. **DHCP Discover**:
   - The laptop broadcasts a message: *"I need an IP address!"*
2. **DHCP Offer**:
   - The router (DHCP server) responds: *"Here's IP 192.168.1.10, subnet mask 255.255.255.0, gateway 192.168.1.1, DNS 8.8.8.8."*
3. **DHCP Request**:
   - The laptop replies: *"I'd like to use IP 192.168.1.10."*
4. **DHCP ACK**:
   - The router confirms: *"You are assigned IP 192.168.1.10 for 24 hours."*

Now the laptop can communicate using IP 192.168.1.10.

Benefits of DHCP:

- **Automatic IP Assignment**: No need for manual IP configuration.
- **Efficient IP Management**: Ensures that IP addresses are not statically assigned or wasted.

- **Centralized Control**: Allows network administrators to manage IP addresses from a central server.



Q.6 Write a note on classful addressing. (M 3)

—> Classful addressing is a method of dividing the IP address space into five predefined classes (A, B, C, D, E) based on the leading bits of the address, which helps in determining the network and host portions of the address. This method was used in the early days of the Internet to define the size and structure of IP address blocks.

In classful addressing, an IP address is divided into two parts:

1. **Network Part**: Identifies the network to which the IP address belongs.
2. **Host Part**: Identifies the specific host (device) within the network.

IP Address Classes: (Write in paragraphs)

Class	Leading Bits	Address Range	Network/ Host Bits	Number of Networks	Number of Hosts per Network	Default Subnet Mask
A	0	0.0.0.0 to 127.255.255.255	8 bits / 24 bits	128	16,777,214	255.0.0.0
B	10	128.0.0.0 to 191.255.255.255	16 bits / 16 bits	16,384	65,534	255.255.0.0
C	110	192.0.0.0 to 223.255.255.255	24 bits / 8 bits	2,097,152	254	255.255.255.0
D	1110	224.0.0.0 to 239.255.255.255	Multicast	N/A	N/A	N/A
E	1111	240.0.0.0 to 255.255.255.255	Reserved for experi-mental use	N/A		

Class A : Designed for very large networks.

Class B : Suitable for medium-sized networks.

Class C : Intended for smaller networks.

Class D : Used for multicast groups rather than standard host addressing.

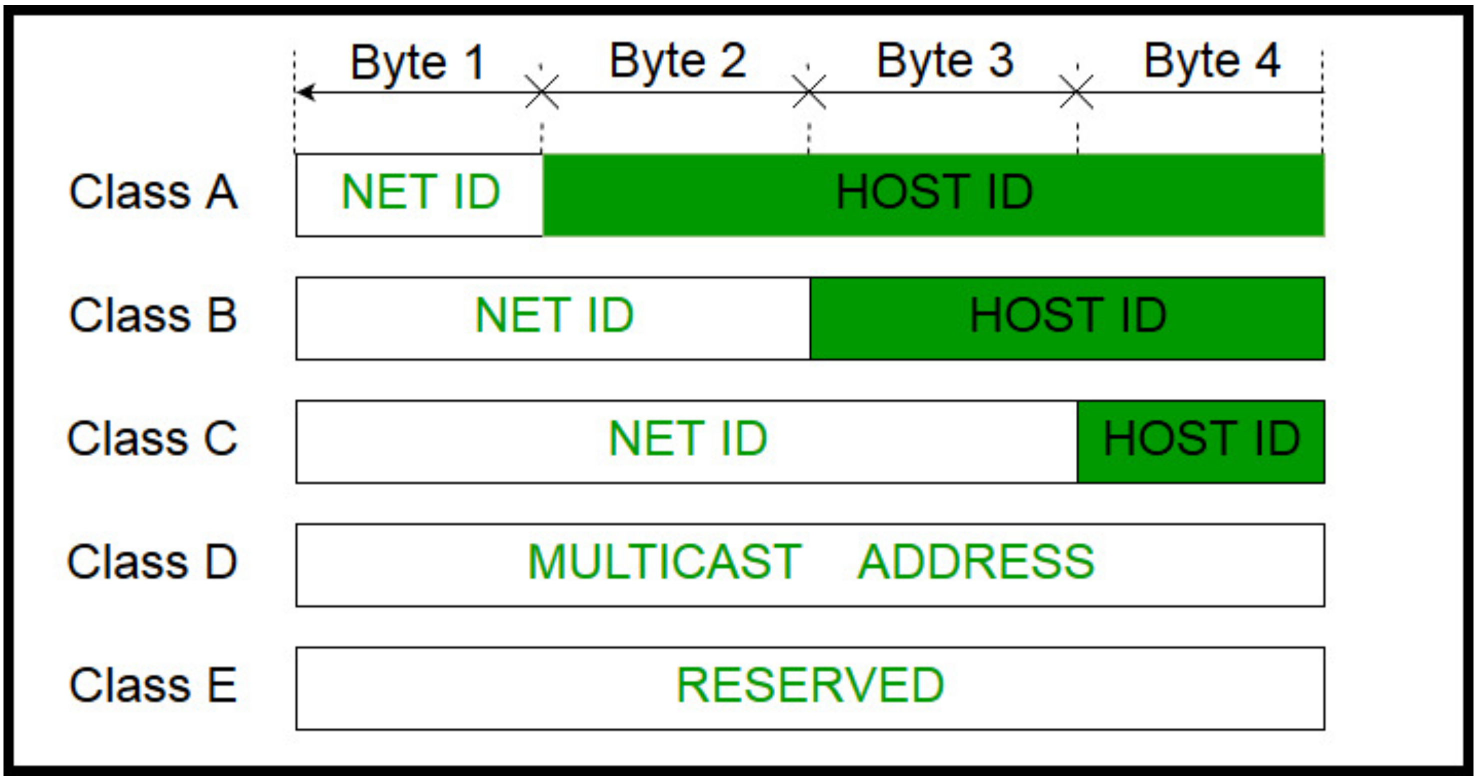
Class E : Reserved for experimental purposes and future use.

Key Features

- **Fixed Classes**: Classful addressing uses fixed boundaries for each class, making it easy to categorize networks based on size.
- **Simple Routing**: Simplified routing was possible because routers could determine the class of an address and forward packets accordingly.

Limitations

- **Lack of Flexibility**: The fixed subnet masks and class boundaries do not allow for more flexible address allocation, which can be a problem in networks with varying sizes.
- **Wasted IP Addresses**: The rigid division of network and host bits often leads to inefficient use of IP addresses. For example, a Class B network might have more host addresses than necessary, leaving many unused.



Q.7 Discuss various issues related to mobile IP.

—> Mobile IP is a protocol that allows devices (mobile nodes) to move from one network to another while maintaining a permanent IP address. This ensures continuous internet connectivity without needing to change the IP address as the device changes locations. It enables devices to continue communicating without interruption, even when they change their point of attachment to the internet (e.g., switching from a Wi-Fi network to a cellular network).

1. Security Issues

Mobile IP introduces various security challenges because it involves routing information between different

networks. Some of the key security concerns include:

- **Authentication**: Binding updates, which inform of the node's location, must be authenticated to prevent **location spoofing** by malicious actors.
- **Data Integrity**: Data transmitted between the mobile node and its home or foreign agent could be intercepted or altered. Ensuring **data integrity** is crucial to avoid tampering.
- **Tunneling Vulnerabilities**: Mobile IP uses tunneling to route packets. Attackers can exploit this to **intercept or hijack** traffic.
- **Denial of Service (DoS) Attacks**: Mobile IP is vulnerable to **DoS attacks**, where attackers overwhelm home or foreign agents with false registration or location updates.

2. Triangle Routing Problem

Triangle Routing is one of the most common issues with Mobile IP, where packets from the sender are sent to the mobile node through the home agent, even when the mobile node is located in a foreign network. This results in:

- **Inefficient Routing**: The packets take an indirect route (via the home network) instead of a direct path, causing higher latency.
- **Increased Network Load**: More bandwidth is consumed because the traffic must be routed through the home agent unnecessarily.

3. Latency and Handover Delays

When a mobile node moves between networks, it must register its new location with the home agent. This process can introduce delays, especially in real-time applications such as voice or video conferencing. Some specific issues include:

- **Handover Latency**: Updating the home agent with the new location can introduce delays in communication.
- **Packet Loss During Handover**: Some packets may be lost while the mobile node updates its location, causing temporary disruption.
- **Handover Delay**: In fast-moving environments (e.g., cars, trains), frequent handovers between networks like Wi-Fi and cellular can lead to noticeable delays.

4. Scalability

As the number of mobile nodes increases, Mobile IP faces challenges related to scalability:

- **Home Agent Overload**: The home agent tracks mobile nodes' locations, and a large number of nodes can lead to processing overload.
- **Foreign Agent Load**: Foreign agents can become overwhelmed with registration requests from visiting nodes, causing congestion.
- **Route Optimization Overhead**: Implementing route optimization to resolve triangle routing adds processing overhead on routers and mobile devices, straining network resources.

5. Addressing and Location Management

Mobile IP requires the coordination of three entities: the mobile node, home agent, and foreign agent. Managing the IP addresses and maintaining correct location information of mobile nodes can present issues, such as:

- **Addressing Conflict**: The mobile node retains its home IP address, which can lead to **IP address conflicts** in the foreign network.
- **Location Update Overhead**: Frequent location updates to the home agent increase **signaling overhead** every time the node changes networks.
- **Care-of Address Assignment**: Managing the **temporary care-of address** while maintaining communication

with the home agent and original IP address can be complex.

** Summary of Issues in Mobile IP

Issue	Description
Security	Authentication, data integrity, DoS attacks, tunneling vulnerabilities.
Triangle Routing	Inefficient routing, causing higher latency and increased network load.
Latency and Handover Delays	Delays during network switches and possible packet loss.
Scalability	Overload of home and foreign agents with many mobile nodes.
Addressing and Location	Challenges with location updates, address conflicts, and care-of address management.
IPv4/IPv6 Interoperability	Difficulties ensuring smooth operation between IPv4 and IPv6 networks.
Resource Constraints	Limited battery, memory, and processing power in mobile devices.
Tunneling Overhead	Additional overhead from encapsulation and potential fragmentation issues.

Q.8 Discuss different services provided by network layer.

—> The network layer (Layer 3) of the OSI model is responsible for facilitating the transfer of data between two devices across multiple networks. It handles the routing of packets, error handling, addressing and managing network congestion, ensuring that data is correctly transmitted from the source to the destination.

1. Packetizing: The process of dividing large data streams into smaller, manageable packets at the network layer. Each packet is encapsulated with a header containing control information like source and destination IP addresses. This is necessary because the data may need to be transmitted across different networks with varying Maximum Transmission Unit (MTU) sizes.

Example: When you send a large file over the internet, it is broken into smaller packets by the network layer. Each packet will be sent separately and reassembled by the destination.

2. Routing: The process of selecting the best path for a packet from its source to its destination across multiple interconnected networks. Routers use **routing algorithms** (like hop count, bandwidth, or latency) and **routing tables** to determine the best path based on factors like distance, cost, and network topology.

Example: When you access a website, your data is sent via several routers, each determining the next best hop towards the destination.

3. Forwarding: The action taken by routers to send packets towards their destination after the best route has been determined. It involves passing the packet from one router to the next based on the destination address and forwarding table. (hop-by-hop process)

Example: After a router receives a packet, it checks the destination IP address and forwards the packet to the next router or destination based on the routing decision.

4. Error Control: A mechanism to detect and handle errors that occur during packet transmission. This service ensures that corrupted or lost packets are retransmitted or discarded, helping maintain data integrity. Error detection techniques (like checksums, CRC) are used to verify the integrity of the packet header and data.

Example: If a router detects that a packet has been corrupted (e.g., through an incorrect checksum), it will discard the packet and the sender will retransmit it.

5. Flow Control: A mechanism used to manage the rate at which data is transmitted between the sender and receiver. In the context of the network layer, flow control ensures that the receiver can handle the incoming data rate and prevents packet loss.

Example: A sender will adjust the amount of data it sends to match the receiver's processing capacity, ensuring that the receiver does not become overwhelmed.

**(Other services are Congestion Control, Quality of Service(QoS) and Security)

Q.9 Explain IPV6 addressing. (M 4)

—> IPv6 (Internet Protocol version 6) is the most recent version of the IP designed to replace IPv4. With the exhaustion of available IPv4 addresses, IPv6 provides a vastly larger address space and enhanced features.

Key Features :

1. **Address Length**: IPv6 addresses are 128 bits long, compared to 32 bits in IPv4. This allows for a vastly larger address space, providing approximately 3.4×10^{38} unique addresses.
2. **Address Format:** IPv6 addresses are written in hexadecimal notation, separated by colons. It consists of 8 groups of 4 hexadecimal digits, each group representing 16 bits.
   eg: 2001:0db8:85a3:0000:0000:8a2e:0370:7334
3. **Zero Compression**: Consecutive sections of zeros can be compressed using a double colon (::). However, this can only be used once in an address to avoid ambiguity.
   eg: 2001:0db8:85a3::8a2e:0370:7334
4. **Address Types**:

    - **Unicast**: A unique address for a single interface (e.g., a single device).
    - **Multicast**: An address for a group of interfaces, allowing data to be sent to multiple destinations.
    - **Anycast**: An address assigned to multiple interfaces, where data is sent to the nearest interface.
5. **Address Allocation:** IPv6 addresses are allocated in a hierarchical manner to reduce routing complexity. Address blocks are assigned to organisations, which can further divide them into subnets.
    - Subnetting: IPv6 allows for efficient subnetting, where a subnet mask defines the network portion of an address.
    - Common subnet sizes include /64, /48 and /32, allowing organisation's to have a large number of subnets and devices.

Structure of an IPv6 Address

An IPv6 address consists of eight groups of four hexadecimal digits, each group representing 16 bits.

xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx

Where xxxx can be any hexadecimal number (0-9, A-F).

Example: 2001:0db8:0000:0042:0000:8a2e:0370:7334

- **Global Routing Prefix**: 2001:0db8 (first 48 bits)
- **Subnet ID**: 0000:0042 (next 16 bits)
- **Interface ID**: 0000:8a2e:0370:7334 (last 64 bits)

Advantages

- **Larger Address Space**: Solves the issue of IPv4 address exhaustion.
- **Improved Security**: IPv6 was designed with security in mind, incorporating features like IPsec for encryption and authentication.
- **Better Multicast and Anycast**: More efficient data transmission methods.

Q.10 Write a short note on ICMPv4.

—> ICMPv4 (Internet Control Message Protocol version 4) is a network layer protocol, primarily used for diagnostic and error-reporting purposes in IPv4 networks. It helps manage network communication by reporting errors, sending control messages, and providing feedback about network issues like unreachable destinations, packet loss or congestion.

ICMPv4 is defined in RFC 792 and operates at the Network Layer (Layer 3) of the OSI model. It works alongside IP (Internet Protocol) to enable the reporting of network problems and assist in routing decisions.

Key Functions of ICMPv4:

1. **Error Reporting**: ICMPv4 is used to notify the sender when a problem occurs during packet delivery. For example, if a router cannot deliver a packet due to an unreachable destination, it will send an ICMP error message back to the source.
2. **Diagnostics**: ICMPv4 is used for diagnostic purposes, particularly for tools like **Ping** and **Traceroute**, which test the reachability of a host and diagnose network problems.

ICMPv4 Message Format:

An ICMP message consists of:

1. **Type**: The type of message (e.g., Echo Request, Destination Unreachable).
2. **Code**: Provides additional details about the message (e.g., the reason for an unreachable destination).
3. **Checksum**: Used for error-checking the ICMP message itself.
4. **Rest of the Header**: This can vary depending on the message type. For example, an Echo Request will contain a **sequence number** and an **identifier** to match requests and responses.

Common ICMPv4 Message Types

Type	Message	Description
0	Echo Reply	Response to an echo request (ping).
3	Destination Unreachable	Indicates a packet could not reach its destination.
5	Redirect	Informs a host to use a different gateway for a destination.
11	Time Exceeded	Indicates that a packet was discarded due to TTL expiration.
12	Parameter Problem	Indicates an issue with a header parameter in the packet.

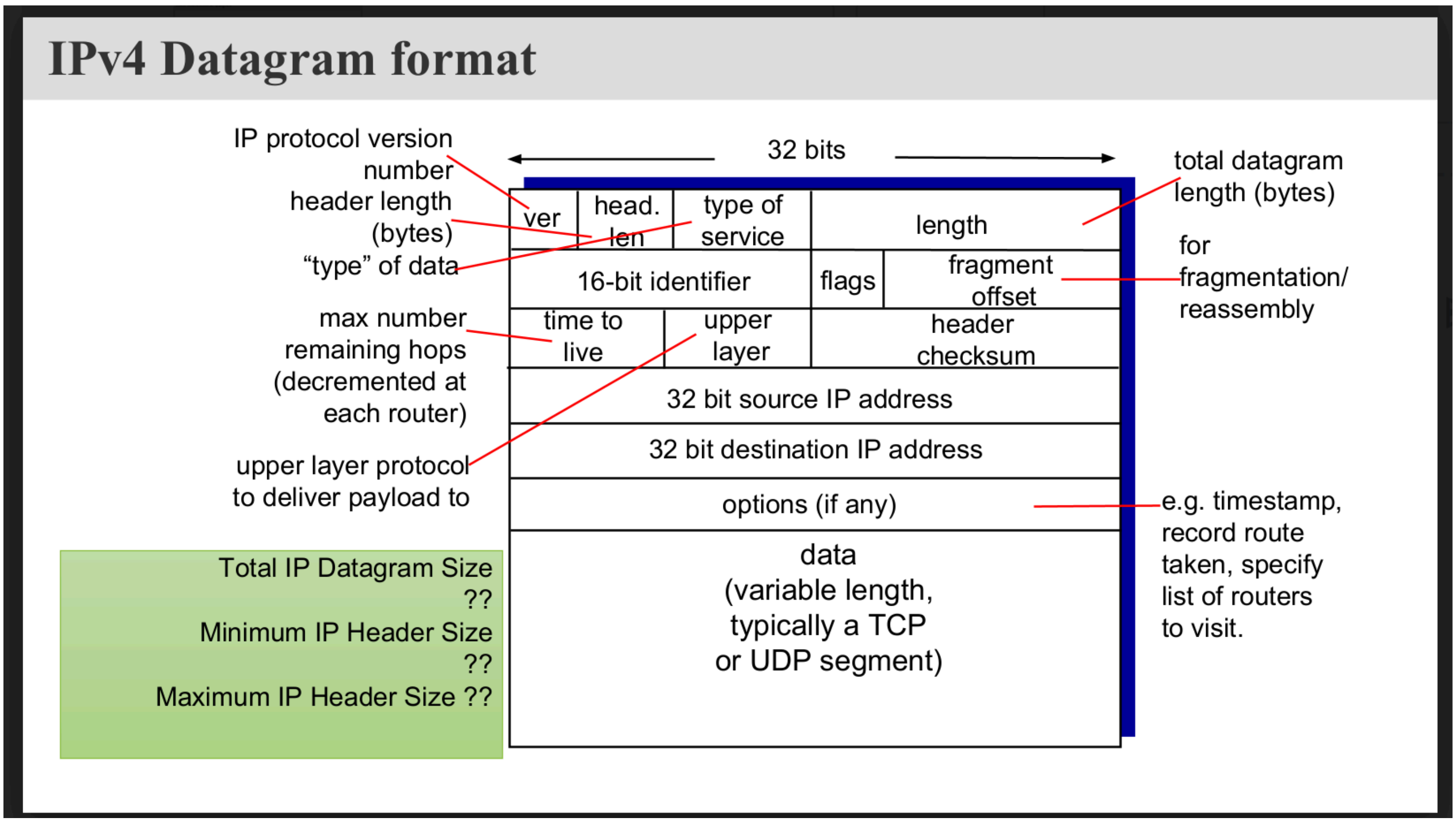
Q.11 Write a short note on datagram format of internet protocol.

—> The datagram format of the Internet Protocol (IP) defines how data is encapsulated for transmission over a network. An IP datagram consists of a header and a payload (data), allowing for the reliable transfer of packets across different networks.

The structure of the IP datagram varies slightly between IPv4 and IPv6, but the core concepts remain consistent.

The IP Datagram consists of two parts:

1. **Header**: Contains essential control and routing information.
2. **Data (Payload)**: Contains the actual data being transmitted.


IPv4 Datagram format
32 bits
ver
head. len
type of service
length
16-bit identifier
flags
fragment offset
time to live
upper layer
header checksum
32 bit source IP address
32 bit destination IP address
options (if any)
data (variable length, typically a TCP or UDP segment)
IP protocol version number
header length (bytes)
“type” of data
max number remaining hops (decremented at each router)
upper layer protocol to deliver payload to
total datagram length (bytes)
for fragmentation/ reassembly
e.g. timestamp, record route taken, specify list of routers to visit.
Total IP Datagram Size ??
Minimum IP Header Size ??
Maximum IP Header Size ??

IP Datagram Format:

Field	Size	Description
Version	4 bits	Indicates the IP version (IPv4 = 4).
IHL (Internet Header Length)	4 bits	Length of the header in 32-bit words (minimum value is 5 for 20 bytes).
Type of Service (TOS)	8 bits	Specifies the quality of service (e.g., priority, delay, throughput).
Total Length	16 bits	Total length of the datagram, including header and data (maximum of 65,535 bytes).
Identification	16 bits	A unique identifier for fragments of a datagram

		(used in fragmentation).
Flags	3 bits	Control flags: 1) DF (Don't Fragment), 2) MF (More Fragments).
Fragment Offset	13 bits	Indicates the position of a fragment within the original datagram (if fragmented).
Time to Live (TTL)	8 bits	Limits the lifespan of the datagram; each router decrements the TTL by 1.
Protocol	8 bits	Specifies the upper-layer protocol (e.g., 1 for ICMP, 6 for TCP, 17 for UDP).
Header Checksum	16 bits	Error-checking value for the header (to verify header integrity).
Source IP Address	32 bits	The IP address of the sender (source).
Destination IP Address	32 bits	The IP address of the recipient (destination).
Options (Optional)	Variable	Optional fields for additional routing or security (if IHL > 5).
Data (Payload)	Variable	The actual data being transmitted (e.g., from upper-layer protocols like TCP/UDP).

Example:
Consider a simple IP datagram that is being sent from 192.168.1.1 (source) to 8.8.8.8 (destination). The IP header will include:

- **Version**: 4 (IPv4)
- **IHL**: 5 (standard 20-byte header)
- **Total Length**: 1500 bytes (including header + data)
- **Source Address**: 192.168.1.1
- **Destination Address**: 8.8.8.8
- **Protocol**: 6 (TCP)
- **TTL**: 64 (for 64 hops in the network)
- **Data (Payload)**: The actual data being sent, such as a TCP segment.

Q.12 Explain Unicast and Multicast routing protocols .

—> Routing protocols are essential for directing packets across networks. They can be classified based on the type of communication they support.

1. Unicast Routing Protocols

Unicast routing protocols are used for one-to-one communication, where a packet is sent from one sender to one specific receiver. In this type of routing, a single unique address identifies the destination, and the network forwards the packet to that address.

Characteristics of Unicast Routing Protocols

- **One-to-One Communication**: Each packet is directed to a single destination.
- **Direct Path**: The routing algorithm determines a direct path from the source to the destination.
- **Widely Used**: Most common type of routing protocol in the Internet (e.g., web browsing, file transfers).

Examples of Unicast Routing Protocols:

- **RIP (Routing Information Protocol)**: A distance-vector protocol that uses hop count as a metric to determine the best path.
- **OSPF (Open Shortest Path First)**: A link-state routing protocol that uses the Dijkstra algorithm to compute the shortest path.
- **BGP (Border Gateway Protocol)**: A path-vector protocol used for routing between different Autonomous Systems (AS) on the Internet. It is used for large-scale routing in inter-domain communication.

Example Scenario:

When you access a website (say, www.example.com), your computer (source) sends a request to the website's server (destination) using unicast routing. The data is delivered from your device directly to the server via a set of routers that determine the optimal path.

2. Multicast Routing Protocols

Multicast routing protocols are used for one-to-many communication, where a packet is sent from one sender to multiple specific receivers. In this routing type, a multicast address identifies a group of receivers, and packets are forwarded to that group.

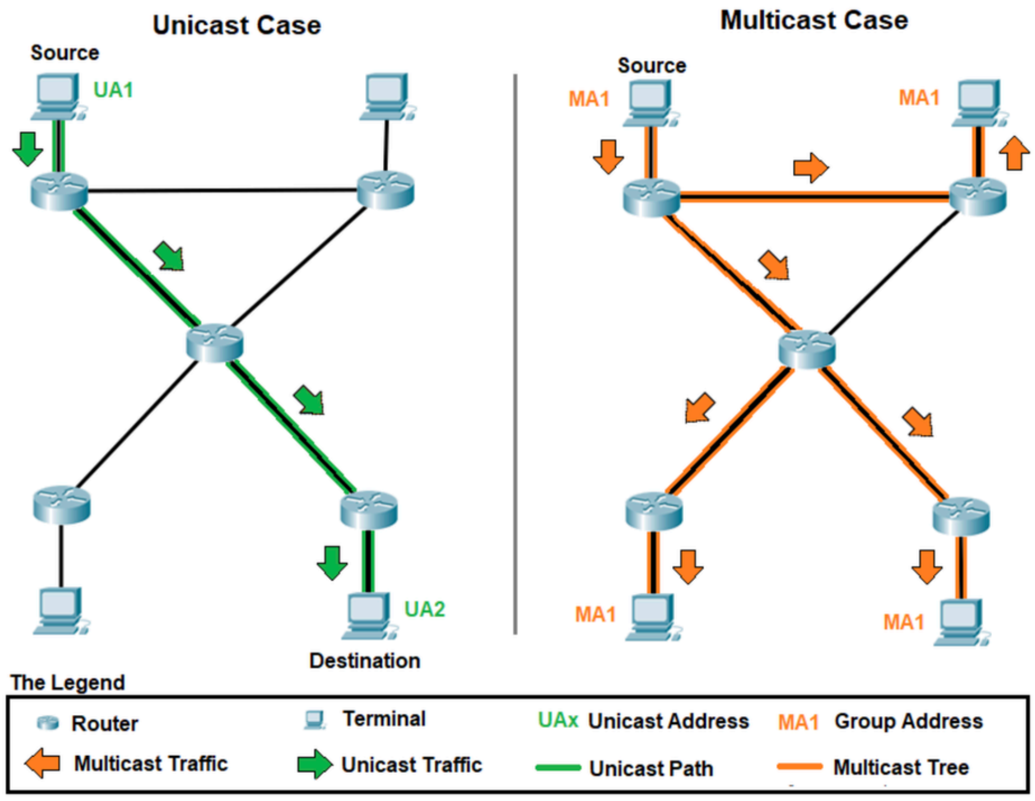
Characteristics of Multicast Routing Protocols

- **One-to-Many Communication**: A single packet can be delivered to multiple destinations simultaneously.
- **Group Addressing**: Uses special multicast addresses (e.g., in IPv4, addresses in the range 224.0.0.0 to 239.255.255.255).
- **Efficiency**: Reduces network traffic since only one copy of the packet is sent through the network, regardless of the number of receivers.

Examples of Multicast Routing Protocols

- **PIM (Protocol Independent Multicast)**: A widely used multicast routing protocol that operates independently of the underlying unicast routing protocols. It supports both Sparse Mode (PIM-SM) for less dense receiver environments and Dense Mode (PIM-DM) for dense environments.
- **IGMP (Internet Group Management Protocol)**: A network layer protocol used by hosts and adjacent routers to establish multicast group memberships. Facilitates communication between multicast routers and hosts.
- **MSDP (Multicast Source Discovery Protocol)**: Allows for the sharing of multicast source information between different PIM-SM domains. Useful for inter-domain multicast routing.

Example Scenario:
Consider a live streaming application where multiple users are watching a video. Instead of sending a separate copy of the video stream to each user (unicast), the server sends a single copy of the stream to all users in the multicast group. Routers will forward this multicast stream to all users who are part of the group.



DCN ASSIGNMENT - 5

Q.1 Explain leaky bucket and token bucket algorithm. (M 5)

—> The Leaky Bucket and Token Bucket algorithms are both techniques used for controlling data flow in networking, particularly for traffic shaping and rate limiting.

1. Leaky Bucket Algorithm

The Leaky Bucket algorithm is used to control the amount of data that can be sent over a network. It allows data to flow out at a constant rate, even if it arrives at irregular intervals.

How It Works (Bucket Capacity, Data Inflow, Data Outflow)

- Imagine a bucket with a hole at the bottom:
  - **Bucket Capacity**: Represents the maximum amount of data that can be held (buffer).
  - **Leak Rate**: Data leaks out at a constant rate.
- Incoming data is added to the bucket. If the bucket overflows (exceeds its capacity), the excess data is discarded.

Characteristics:

- Provides a steady output rate.
- Helps prevent network congestion.
- Can discard excess traffic if the bucket overflows.

2. Token Bucket Algorithm

The Token Bucket algorithm allows for bursts of traffic while also enforcing a specified average rate of data transmission. It uses tokens to control the sending rate.

How It Works (Bucket Capacity, Token Generation, Data Transmission, Burst Transmission)

- Imagine a bucket that collects tokens at a constant rate:
  - **Bucket Capacity**: Represents the maximum number of tokens that can be held.
  - **Token Generation Rate**: Tokens are generated at a fixed rate and added to the bucket.
- Each token allows sending a certain amount of data (e.g., one token = one packet).
- If there are no tokens available, the sender must wait until tokens are generated.
- The algorithm allows for burst transmission if tokens are available, up to the bucket's capacity.

Characteristics:

- Allows bursts of data transmission up to the bucket capacity.
- Ensures that average data transmission adheres to a specified rate.
- Efficiently utilizes available bandwidth.

Q.2 Differentiate between UDP and TCP. (M 6)

—> UDP (User Datagram Protocol) and TCP (Transmission Control Protocol) are two core protocols used for data transmission over the internet.

Feature	UDP (User Datagram Protocol)	TCP (Transmission Control Protocol)

Connection	Connectionless	Connection-oriented
Reliability	Unreliable (no guarantee of delivery)	Reliable (guarantees delivery)
Error Checking	Optional (checksums for errors)	Automatic error checking and correction
Data Order	No guaranteed order	Guarantees ordered delivery
Flow Control	No flow control	Uses flow control mechanisms (e.g., sliding window)
Congestion Control	No congestion control	Implements congestion control (e.g., slow start)
Overhead	Lower overhead (minimal header)	Higher overhead (larger header due to control info)
Use Cases	Suitable for applications needing speed (e.g., video streaming, gaming)	Used for applications requiring reliability (e.g., web browsing, file transfers)
Header Size	8 bytes	20 bytes (minimum)
Transmission Type	Datagram (message-oriented)	Stream (byte-oriented)
Speed	Faster due to lower overhead	Slower due to reliability features

Q.3 Explain connection oriented and connection-less services.

—> Connection-oriented and Connection-less services define the way data is transmitted between two devices over a network. The distinction is based on whether a dedicated communication path is established before data transmission and whether there is any guarantee of delivery.

1. Connection-Oriented Service

A connection-oriented service is a communication method where a dedicated communication path is

established between the sender and receiver before any data is exchanged. This service ensures that the connection is set up, maintained, and then terminated once the data transfer is complete. It provides a reliable, ordered, and error-checked transmission of data.

Key Characteristics:

- **Connection Setup**: A connection must be established between the sender and receiver before data transfer begins (e.g., through a handshake process).
- **Reliability**: It ensures **error recovery**, **flow control**, and **sequencing** of data, making it reliable.
- **Acknowledgment**: The receiver acknowledges receipt of data, and lost or corrupted packets are retransmitted.
- **Guaranteed Delivery**: Data is guaranteed to be delivered to the destination in the same order it was sent.
- **Resource Reservation**: Resources (such as bandwidth) may be reserved for the duration of the connection.

Protocols: TCP (Transmission Control Protocol) commonly used and ATM (Asynchronous Transfer Mode) used in high-speed networks.

Applications: Web browsing (HTTP/HTTPS), file transfers (FTP) and email (SMTP) rely on connection-oriented services.

Example:
When you download a file from the internet, the **TCP** protocol ensures that the file is transferred reliably, without loss, and in the correct order.

2. Connection-Less Service
A connection-less service is a communication method where no dedicated path is established between the sender and receiver before data is transmitted. Each data packet is sent independently, without checking if the other packets reach the destination. There is no need for connection setup, and the network does not guarantee delivery, order, or error correction.

Key Characteristics:

- **No Connection Setup**: No need to establish a connection or reserve resources before transmitting data.
- **Unreliable**: Data may arrive out of order, or not arrive at all. The network does not guarantee that packets will be delivered.
- **No Acknowledgment**: There is no acknowledgment from the receiver, and no automatic retransmission of lost packets.
- **Faster**: Because there's no connection setup or error checking, it is generally faster and requires less overhead.

Protocols: UDP (User Datagram Protocol) commonly used and IP (Internet Protocol) used in delivering packets across networks.
Applications: Streaming media (video/audio), online gaming and voice over IP (VoIP) utilize connectionless services for faster transmission.

Example:

In online gaming, **UDP** is often used because it requires low latency and can tolerate some packet loss, which does not affect the overall experience as much.

Q.4 Explain TCP connection establishment.
—> Same answer as Question 7, The Three-way Handshake for TCP connection establishment.

Q.5 Explain about congestion control.
—> Congestion control refers to the mechanisms and techniques used to prevent or reduce network congestion, ensuring that network resources (such as bandwidth, routers, and switches) are used efficiently without overwhelming the system. Congestion occurs when the demand for network resources exceeds the available capacity, leading to packet loss, delays, and reduced throughput.

In the context of Transmission Control Protocol (TCP), congestion control is crucial to maintain network performance and stability. The goal is to avoid overloading the network and to ensure that data is transmitted at an optimal rate.

Key Concepts in Congestion Control

- **Congestion**: This happens when too many packets are sent into the network, causing routers or switches to become overloaded. This results in packet delays, retransmissions, and a significant reduction in network performance.
- **Congestion Window (cwnd)**: A mechanism in TCP that controls the amount of data that can be sent over the network before receiving an acknowledgment from the receiver. The congestion window size adapts dynamically based on network conditions.
- **Available Bandwidth**: The capacity of the network to handle data transmission. When congestion occurs, the available bandwidth reduces, leading to packet loss.

Importance

- **Network Efficiency**: Ensures that the network can handle traffic without bottlenecks, maintaining performance and reliability.
- **Fairness**: Aims to allocate network resources fairly among all users, preventing any single user from monopolizing bandwidth.
- **Quality of Service (QoS)**: Maintains the quality of network services, which is crucial for real-time applications like VoIP and video streaming.

TCP Congestion Control Algorithm (Extra Information)

TCP uses feedback-based algorithms to adjust data transmission based on network conditions:

1. **Slow Start**:
   - Starts with a small congestion window (cwnd), typically 1 or 2 MSS.
   - Doubles cwnd for each ACK received, leading to exponential growth.
   - Example: If cwnd = 1 MSS, it increases to 2 MSS, then 4 MSS, etc.
2. **Congestion Avoidance**:

    - Once cwnd reaches a threshold (ssthresh), growth becomes linear (increases by 1 MSS per RTT).
    - Prevents congestion by cautiously increasing transmission.
3. **Fast Retransmit & Fast Recovery**:
    - On detecting packet loss via duplicate ACKs, the lost packet is retransmitted.
    - Cwnd is reduced to half instead of resetting to 1 MSS, allowing continued transmission without slow start.
4. **Timeout & Retransmission**:
    - If no ACK is received in time, TCP enters slow start again and sets ssthresh to half of cwnd, reducing transmission rate to avoid congestion.

While TCP's congestion control is the most commonly known, other network protocols also implement various methods for managing congestion:
1. Random Early Detection (RED)
2. Explicit Congestion Notification (ECN)

Q.6 Write a note on transport layer.
—> The Transport Layer is a crucial layer in the OSI (Open Systems Interconnection) model and the TCP/IP model, responsible for providing reliable and efficient data transfer between devices across a network. It acts as an intermediary between the application layer (where end-user applications operate) and the network layer (which manages data routing and addressing).

Key Functions of the Transport Layer

1. **End-to-End Communication**:
    - The transport layer facilitates communication between processes or applications running on different devices. It provides logical communication between application processes, abstracting the details of the underlying network.
2. **Segmentation and Reassembly**:
    - Data from the application layer is divided into smaller chunks called **segments** (in TCP) or **datagrams** (in UDP). These segments are sent across the network and reassembled at the destination.
3. **Multiplexing**:
    - The transport layer allows multiple applications on the same device to communicate simultaneously. It achieves this by using port numbers, which identify different applications or services running on a device.

Protocols in the Transport Layer

1. **Transmission Control Protocol (TCP)**: A **connection-oriented** protocol that guarantees reliable data delivery, correct ordering and error recovery. It uses a three-way handshake for connection establishment and provides flow control, congestion control and error correction mechanisms.
    - **Applications**: Web browsing (HTTP), email (SMTP), file transfer (FTP).
2. **User Datagram Protocol (UDP)**: A **connectionless** protocol that provides no guarantees for delivery, ordering, or error correction. It is faster and has lower overhead than TCP, making it suitable for applications where speed is more important than reliability (e.g., streaming, VoIP, gaming).
    - **Applications**: Video streaming, DNS, VoIP.

Services Provided by the Transport Layer

- **Connection-Oriented Service** (TCP): Establishes, maintains, and terminates connections between devices. Ensures data is transmitted reliably.
- **Connection-Less Service** (UDP): Sends data without establishing a connection, useful for applications requiring low latency and can tolerate some data loss.
- **Flow Control**: Prevents data loss by regulating the rate of data transfer between sender and receiver.
- **Error Detection and Correction**: Ensures data integrity by detecting and correcting errors.
- **Multiplexing**: Allows multiple communication sessions on the same device through port numbers.

Q.7 Explain three-way handshaking approach for TCP connection establishment and termination. (M 7)

—> The three-way handshake is a process used in the TCP (Transmission Control Protocol) to establish a reliable connection between a client and a server. It ensures that both parties are synchronized and ready to communicate before data transmission begins. The same process can be adapted for connection termination, though it is commonly described as a four-way handshake.

TCP Connection Establishment (Three-Way Handshake) :

Step 1: SYN (Synchronize)

- The client sends a SYN packet to the server to initiate a connection. This packet includes a sequence number that indicates the starting point for data transmission. The purpose of this packet is to request a connection to the server.

  Client → Server: SYN, Seq = x

Step 2: SYN-ACK (Synchronize-Acknowledgment)

- The server responds with a SYN-ACK packet. This packet acknowledges the client's SYN request (by including an acknowledgment number) and also sends its own SYN packet (with its own sequence number) to the client.

  Server → Client: SYN-ACK, Seq = y, Ack = x + 1

Step 3: ACK (Acknowledgment)

- The client sends an ACK packet back to the server, acknowledging the receipt of the server's SYN-ACK packet. At this point, the connection is established, and both parties can start transmitting data.

  Client → Server: ACK, Seq = x + 1, Ack = y + 1

(After these three steps, a TCP connection is successfully established, and both the client and server can begin to transmit data.)

TCP Connection Termination (Four-Way Handshake) :

Step 1: FIN (Finish)

- The client (or server) that wants to terminate the connection sends a **FIN** packet to the other party, indicating it has finished sending data.

  Client → Server: FIN, Seq = x + 1

Step 2: ACK (Acknowledgment)

- The other party acknowledges the receipt of the FIN packet by sending an **ACK** packet back. This indicates that it has received the FIN request.
  Server → Client: ACK, Seq = y + 1, Ack = x + 2

Step 3: FIN (Finish)

- The other party (which received the FIN) may still have data to send. Once it is finished sending its data, it sends its own **FIN** packet.
  Server → Client: FIN, Seq = y + 2

Step 4: ACK (Acknowledgment)

- Finally, the original sender acknowledges the receipt of the FIN packet with an **ACK** packet.
  Client → Server: ACK, Seq = x + 2, Ack = y + 3

(After these four steps, the connection is gracefully terminated, ensuring that both parties have completed their data exchange.)

Q.8 Explain working of piggybacking protocol.

—> The **piggybacking protocol** is an optimization technique, particularly in protocols like TCP (Transmission Control Protocol) for reliable data transmission. The key idea behind piggybacking is to combine data and acknowledgment (ACK) messages into a single frame, reducing the number of packets sent over the network and thus improving efficiency. This approach is primarily used in **bidirectional communication** where both parties (sender and receiver) can transmit data.

How Piggybacking Works

In a traditional protocols like TCP, when a sender transmits data to a receiver, it may require an acknowledgment from the receiver to confirm successful receipt. This often results in two separate messages: one for the data and another for the acknowledgment. This can be inefficient, especially in a bidirectional communication scenario.

With piggybacking, the sender can "piggyback" the acknowledgment of received data onto the outgoing data packet when it has data to send.

Step 1: Data Transmission

1. **Sender Sends Data**: The sender transmits a data packet to the receiver, which includes the data segment.
   Sender → Receiver: DATA

Step 2: Receiver Acknowledges

2. **Receiver Receives Data**: Upon receiving the data, the receiver sends an acknowledgment back to the sender. Without piggybacking, this would be a separate ACK message.
   Receiver → Sender: ACK

Step 3: Piggybacking Acknowledgment

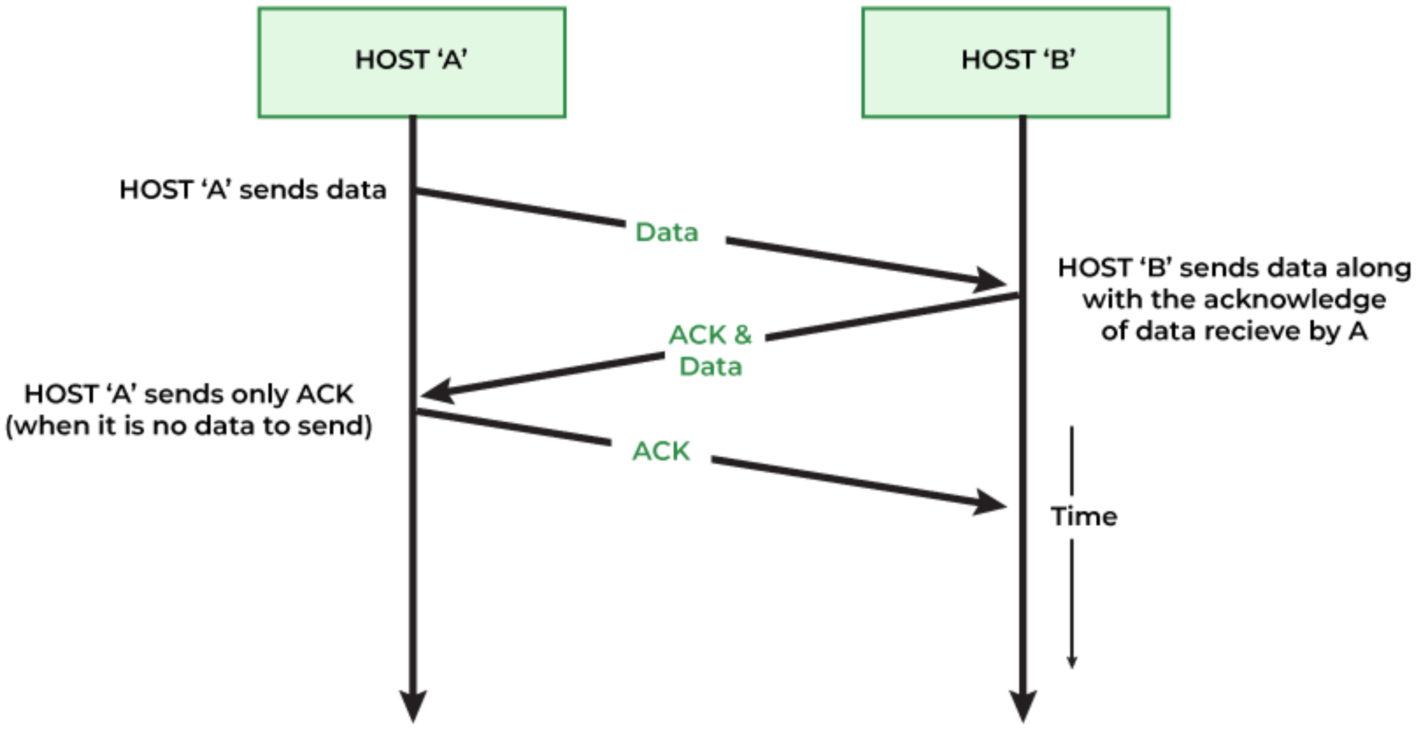
3. **Receiver Has Data to Send**: If the receiver has data of its own to send back to the sender, instead of sending a separate ACK, it combines the ACK for the received data with the new data packet. The acknowledgment will typically indicate the next expected sequence number.
   Receiver → Sender: DATA + ACK

4. **Sender Receives Combined Message**: The sender receives the combined message and processes both the acknowledgment and the new data simultaneously.

Example Scenario

Let's consider a simple example of how piggybacking works in a TCP-like protocol:

1. **Sender A sends data**: A1 to Receiver B.
2. **Receiver B receives A1** and wants to send its own data B1 back to Sender A.
3. Instead of sending an ACK for A1 separately, Receiver B combines the ACK for A1 (indicating the next expected sequence number) with its own data B1.
4. **Receiver B sends**: ACK for A1 + B1 to Sender A.


HOST 'A'
HOST 'B'
HOST 'A' sends data
Data
HOST 'B' sends data along with the acknowledge of data recieve by A
ACK & Data
HOST 'A' sends only ACK (when it is no data to send)
ACK
Time

Q.9 Explain Go-Back-N protocol. (M 8_1)

—> Go-Back-N is an Automatic Repeat reQuest (ARQ) protocol used for reliable data transfer. It is a sliding window protocol that allows a sender to send multiple frames before needing an acknowledgment for the first one, but the receiver can only acknowledge frames in order.

Key Concepts of Go-Back-N Protocol

1. **Sliding Window**

    - It defines the range of frames that can be sent but not yet acknowledged.
    - The sender maintains a window of size *N,* allowing it to send up to *N* frames before requiring an acknowledgment (ACK) from the receiver.
2. **Sequence Numbers**
    - Each frame sent is assigned a unique sequence number, which helps both sender and receiver keep track of which frames have been sent and acknowledged.
    - Sequence numbers are typically modulo 2^m, where *m* is the number of bits used for sequence numbering.
3. **Acknowledgments**
    - The receiver sends an acknowledgment for the last correctly received frame, known as a cumulative acknowledgment.
    - If any frame is lost or corrupted, the receiver discards that frame and any subsequent frames, causing the sender to "go back" and retransmit from the lost frame onward.

How It Works

1. **Sending Frames**:
    - The sender transmits multiple frames in sequence, up to the maximum window size *N*.
2. **Receiving Frames**:
    - The receiver checks the frames. If frames are received in order, it sends an acknowledgment for the highest numbered frame.
    - If a frame is missing, the receiver discards it and all subsequent frames, sending an acknowledgment for the last correctly received frame.
3. **Retransmission**:
    - When the sender receives a negative acknowledgment (or no acknowledgment within a timeout period), it retransmits the lost or erroneous frame along with all subsequent frames.

Example :

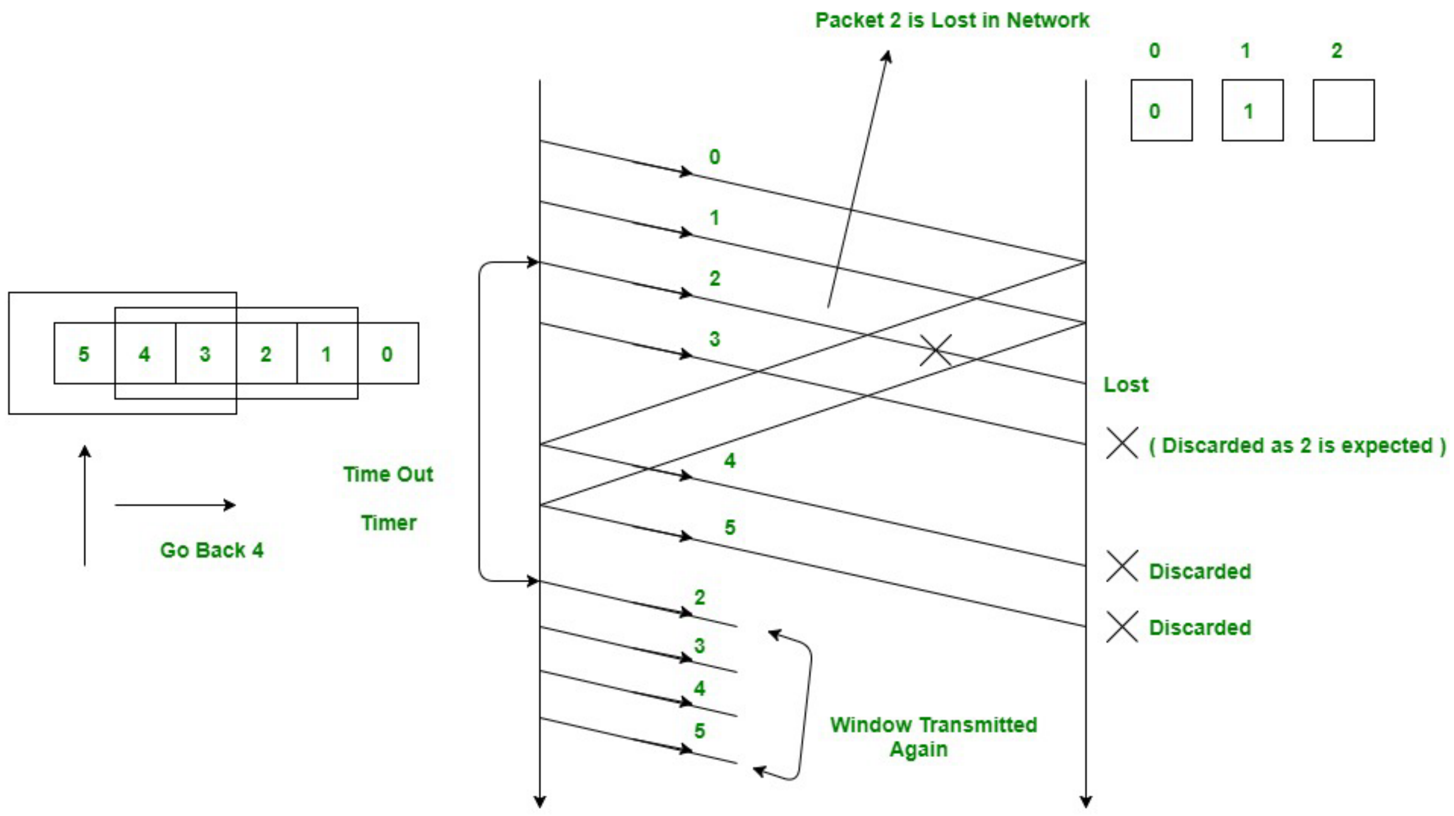
1. **Window Size**: Assume a window size of 4 frames.
2. **Sent Frames**: The sender sends frames 0, 1, 2, 3.
3. **Acknowledgment**:
    - The receiver successfully receives frames 0, 1, 2, but frame 3 is lost.
    - The receiver acknowledges frame 2.
4. **Retransmission**:
    - The sender, upon noticing the lack of acknowledgment for frame 3, retransmits frames 3 and 4, even though frame 4 may have been sent successfully.

Advantages

- **Simplicity**: The protocol is relatively easy to implement.
- **Efficiency for Low Error Rates**: It can be efficient when errors are infrequent, as only the lost frames need to be retransmitted.

Disadvantages

- **Inefficiency with High Error Rates**: If many frames are lost, it can lead to a large number of retransmissions, causing bandwidth inefficiency.
- **Increased Latency**: Retransmitting multiple frames can lead to increased latency.



Q.10 Explain stop-and-wait protocol.

—> The Stop-and-Wait Protocol is a simple and fundamental protocol used for reliable data transfer between two devices (sender and receiver). In this protocol, the sender sends a single data packet and waits for an acknowledgment (ACK) from the receiver before sending the next packet. This process ensures reliable delivery but can result in lower efficiency, especially over long distances with high latency.

Key Characteristics:

1. **Simplicity**: The protocol is easy to implement due to its straightforward structure and minimal requirements.
2. **One Frame at a Time**: The sender can only send one frame and must wait for an acknowledgment before sending the next one.
3. **Reliability**: The protocol ensures that each frame is received correctly before proceeding, allowing for error detection and retransmission of lost or corrupted frames.

How Stop-and-Wait Protocol Works

1. Frame Transmission

- The sender transmits a single data frame to the receiver.

  Sender → Receiver: Frame 0

2. Waiting for Acknowledgment

- After sending the frame, the sender enters a waiting state and does not send any further frames until it receives an acknowledgment from the receiver.

3. Receiving the Frame

- The receiver checks the integrity of the received frame (e.g., using checksums).
- If the frame is received correctly, the receiver sends an acknowledgment back to the sender.

  Receiver → Sender: ACK for Frame 0

4. Sending the Next Frame

- Upon receiving the acknowledgment, the sender sends the next frame and repeats the process.

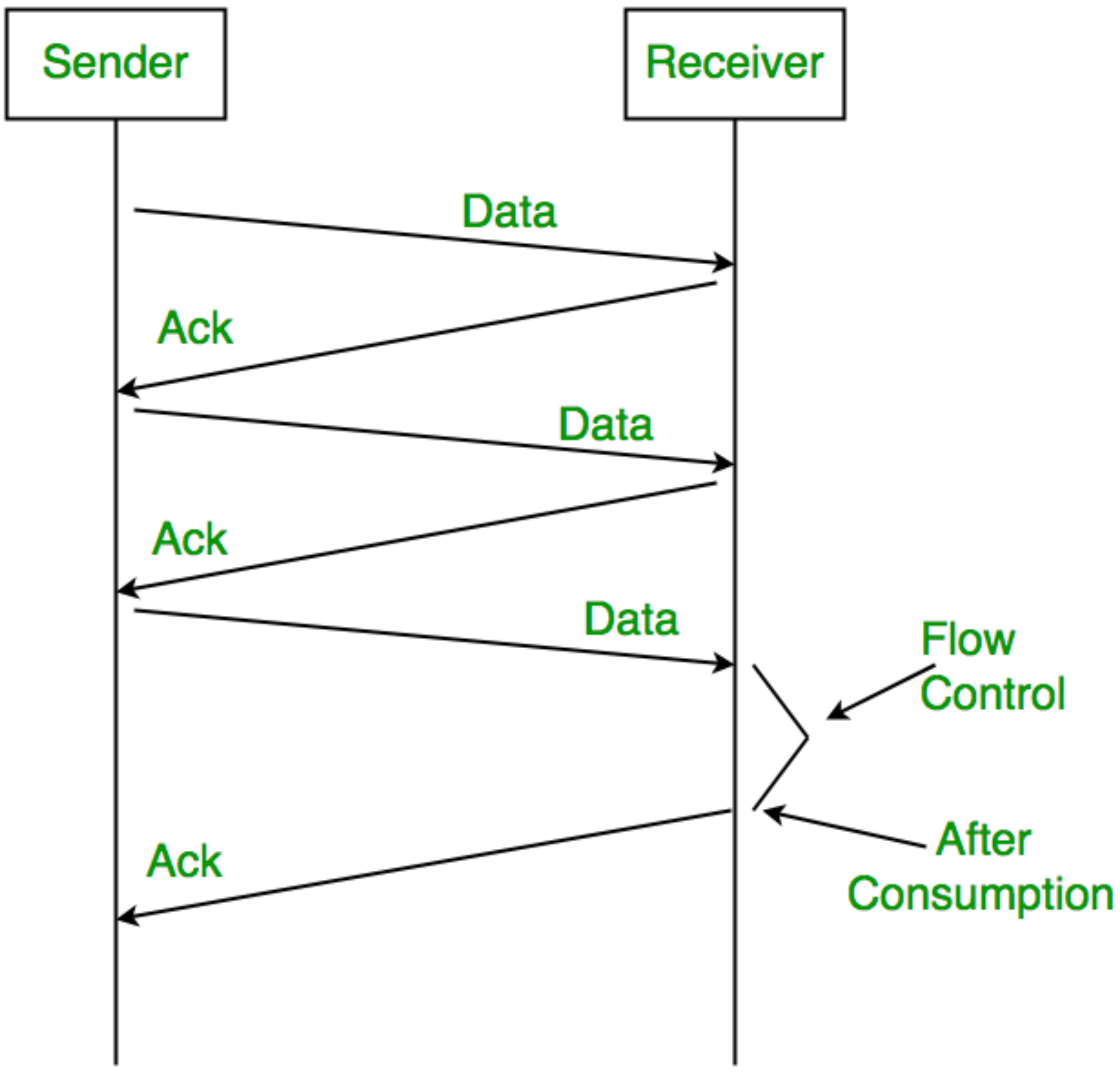
  Sender → Receiver: Frame 1

Example Scenario:

1. **Sender sends Frame 0** and waits for an acknowledgment.
2. **Receiver receives Frame 0** and sends an ACK.
3. **Sender receives ACK for Frame 0** and sends Frame 1.
4. **Receiver receives Frame 1** and sends an ACK for Frame 1.

Handling Errors

- If the sender does not receive an acknowledgment within a specified timeout period (due to frame loss, corruption, etc.), it retransmits the same frame.
- The receiver may also send a negative acknowledgment (NAK) if it detects a corrupted frame, prompting the sender to retransmit.



1.

Q.11 Discuss working of Selective-Repeat protocol. (M 8_2)

—> The **Selective Repeat (SR)** protocol is an advanced automatic repeat request (ARQ) protocol used for reliable data transfer. It is part of the family of sliding window protocols and improves upon the Go-Back-N protocol by allowing the receiver to selectively acknowledge individual frames instead of requiring the sender to retransmit all frames after a lost or corrupted frame. This can lead to more efficient use of network resources and better throughput.

Key Features of Selective Repeat Protocol

1. **Sliding Window**: Similar to Go-Back-N, Selective Repeat uses a sliding window mechanism. However, it allows both the sender and receiver to manage frames more efficiently by permitting selective

acknowledgment of frames.
2. **Frame Buffering**: The receiver can buffer out-of-order frames until the missing frame is received. This is crucial for maintaining the correct sequence of data without the need to discard frames.
3. **Individual Acknowledgment**: Each frame sent can be acknowledged independently, allowing for greater efficiency, especially in networks with higher frame loss rates.

How It Works

1. Frame Transmission

- The sender can transmit multiple frames up to a specified window size *N*.

2. Receiving Frames

- The receiver checks each received frame for errors (e.g., using checksums).
- If a frame is received correctly, it sends an acknowledgment (ACK) for that specific frame.

3. Buffering Out-of-Order Frames

- If the receiver gets an out-of-order frame (e.g., frame 2 is received after frame 4), it buffers the out-of-order frame until the missing frame is received.
- The receiver can acknowledge the frames it has received correctly, even if they are not in order.

Example :

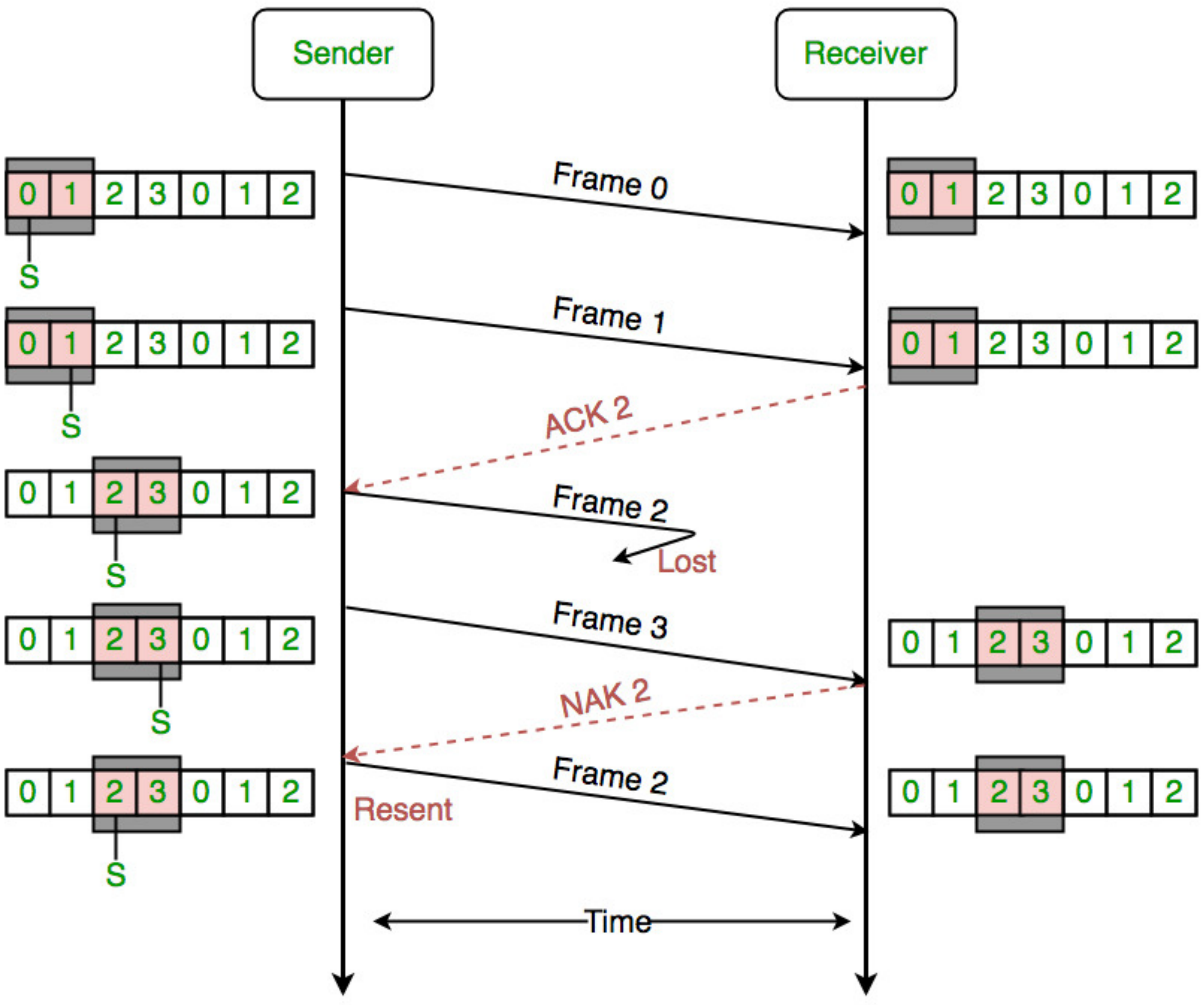
1. **Window Size**: Assume a window size of 4 frames.
2. **Sent Frames**: The sender sends frames 0, 1, 2, 3.
3. **Receiving Frames**:
    - The receiver successfully receives frames 0, 1, and 3, but frame 2 is lost.
    - The receiver acknowledges frames 0 and 1, and does not send an acknowledgment for frame 2.
4. **Retransmission**:
    - The sender, after a timeout for frame 2, retransmits only frame 2. Frames 0, 1, and 3 remain unaffected.

Advantages

- **Efficiency**: Only the lost or erroneous frames are retransmitted, leading to more efficient use of bandwidth compared to Go-Back-N.
- **Reduced Latency**: By not requiring the retransmission of all subsequent frames, the protocol can reduce latency in data transmission.

Disadvantages

- **Complexity**: The protocol is more complex to implement compared to Go-Back-N due to the need for managing individual acknowledgments and out-of-order frames.
- **Buffering**: The receiver must have sufficient buffering capacity to store out-of-order frames until they can be processed.



Q.12 Explain SCTP packet format.

—> SCTP (Stream Control Transmission Protocol) is a message-oriented, reliable transport protocol used to transfer data over IP networks. It is designed to provide features like multi-homing, multi-streaming and out-of-order message delivery, which makes it suitable for applications such as telephony and streaming media.

The SCTP packet format consists of a series of chunks, which contain control and data information. Each SCTP packet begins with a header followed by one or more chunks.

SCTP Packet Structure

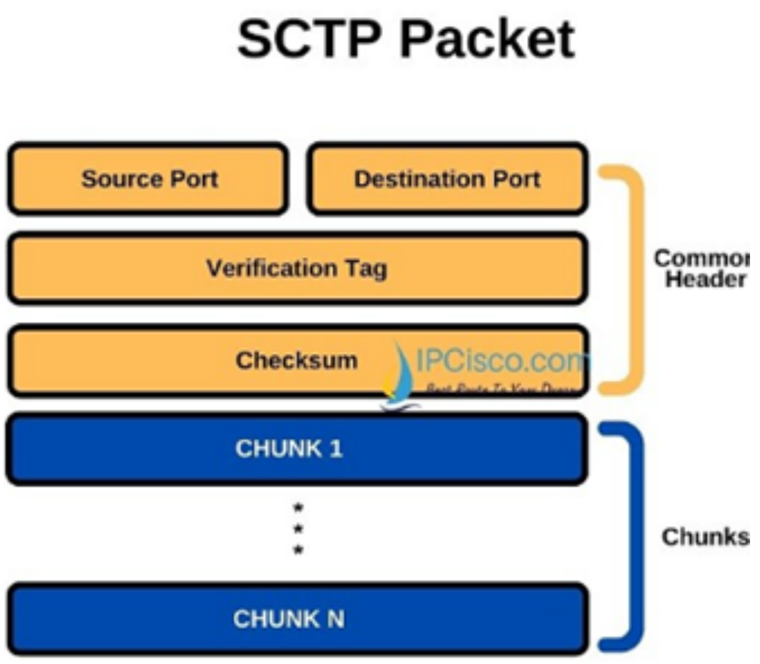
1. **SCTP Header**: The SCTP header is fixed in size and appears at the beginning of every SCTP packet. It contains essential information required for packet processing.
    - **Length (4 bytes)**: The total length of the SCTP packet, including the header and all chunks.
    - **Source Port (2 bytes):** The port of the sender's endpoint.
    - **Destination Port (2 bytes):** The port of the receiver's endpoint.

    - **Verification Tag (4 bytes)**: Used for security and to ensure the packet is valid for the association. It helps prevent the use of packets from other associations.
    - **Checksum (4 bytes)**: A 32-bit CRC checksum used for error-checking the entire packet.
2. **Chunks**: The SCTP packet is composed of multiple chunks, each of which can have different types and formats. Chunks can be of various types, such as data chunks, control chunks, and so on. Each chunk also has a header and a type field that defines its purpose.

SCTP Chunk Structure

Each chunk has a specific format consisting of the following fields:

1. **Chunk Type (1 byte)**: Indicates the type of chunk (e.g., DATA, INIT, HEARTBEAT, etc.).
2. **Chunk Flags (1 byte)**: Contains flags relevant to the specific chunk type (e.g., whether the chunk is the last chunk in a message).
3. **Chunk Length (2 bytes)**: The total length of the chunk, including the chunk header and data.
4. **Chunk Value (Variable length)**: The actual data or control information contained in the chunk. The structure of this part varies depending on the chunk type.



Padding (if needed, to ensure the total packet length is a multiple of 4 bytes).

DCN ASSIGNMENT - 6

Q.1 Explain HTTP and HTTPS in detail. (M 9)

—> HTTP(Hypertext Transfer Protocol) is a protocol used for transmitting hypertext via the web. It allows clients (like browsers) to communicate with servers. It is a request-response protocol, where a client (typically a web browser) sends requests to a server, which then responds with the requested resources (like HTML documents).

Key Features:

1. **Stateless Protocol:**
    - Each request from the client to the server is treated as an independent transaction. The server does not store any information about the client's previous requests.
2. **Methods:**
    - Common HTTP methods include:
        - **GET:** Retrieve data from a server.
        - **POST:** Send data to the server (e.g., form submissions).
        - **PUT:** Update existing data on the server.
        - **DELETE:** Remove data from the server.
3. **Port:**
    - By default, HTTP uses port 80.
4. **Request/Response Structure:**
    - An HTTP request consists of:
        - Request Line (method, URL, HTTP version)
        - Headers
        - Body (optional)
    - An HTTP response consists of:
        - Status Line (HTTP version, status code, status message)
        - Headers
        - Body (content)
5. **Status Codes:**
    - HTTP responses include status codes indicating the result of the request:
        - **200 OK:** Successful request.
        - **404 Not Found:** Requested resource not found.
        - **500 Internal Server Error:** Server encountered an error.

—> HTTPS(Hypertext Transfer Protocol Secure) is the secure version of HTTP, combining HTTP with Transport Layer Security (TLS) to provide encryption and secure communication over a computer network. It is used to ensure the confidentiality, integrity, and authenticity of the data transmitted between the client and server.

Key Features:

1. **Encryption:**
    - HTTPS encrypts the data exchanged between the client and server, protecting it from eavesdropping and man-in-the-middle attacks.
2. **Authentication:**
    - HTTPS uses digital certificates to authenticate the server. This ensures that clients are communicating with the intended server, not an impostor.
3. **Data Integrity:**
    - HTTPS verifies that the data sent and received has not been altered during transmission.
4. **Port:**
    - HTTPS typically uses port 443.
5. **SEO Benefits:**
    - Search engines like Google give preference to HTTPS sites, improving their ranking in search results.

Q.2 Describe application layer and its protocol.

—> The Application Layer is the topmost layer in the OSI (Open Systems Interconnection) model and is responsible for providing network services directly to the end-users or applications. It acts as an interface between the user's applications and the underlying network infrastructure. Also provides functions like file transfers, email, and web browsing.

Key Functions of the Application Layer:

- **User Interface**: Facilitates interactions between users and applications.
- **Data Formatting**: Prepares data for the application in a usable format.
- **Session Management**: Manages sessions between applications.
- **Error Handling**: Provides error detection and correction mechanisms.

Common Protocols in the Ap×lication Layer

1. **HTTP (Hypertext Transfer Protocol):**
   - Used for transferring web pages and web resources over the internet. It operates over TCP (Transmission Control Protocol) and provides request-response communication between clients (browsers) and servers.
2. **HTTPS (HTTP Secure):**
   - A secure version of HTTP that uses SSL/TLS to encrypt data transferred between the client and server, ensuring secure transactions.
3. **FTP (File Transfer Protocol):**
   - Used for transferring files b×tween a client and server. It supports both Uploading and downloading files and can operate in active or passive modes.
4. **SMTP (Simple Mail Transfer Protocol):**
   - Used for sending emails between servers. It operates over TCP and typically uses port 25.
5. **DNS (Domain Name System):**
   - Translates human-readable domain names (like www.example.com) into IP addresses that computers can understand. It operates over UDP (User Datagram Protocol) for queries and responses.

**** Others are POP3 (Post Office Protocol version 3), IMAP (Internet Message Access Protocol), Telnet and SSH (Secure Shell)**

Q.3 Explain with diagram email format. (M 10)

—> An email consists of several key components, each playing a vital role in the structure and delivery of the message. The basic format of an email can be divided into two main parts :

1. Email Header

The email header contains essential metadata about the email i.e., the sender, recipient, subject, and date.

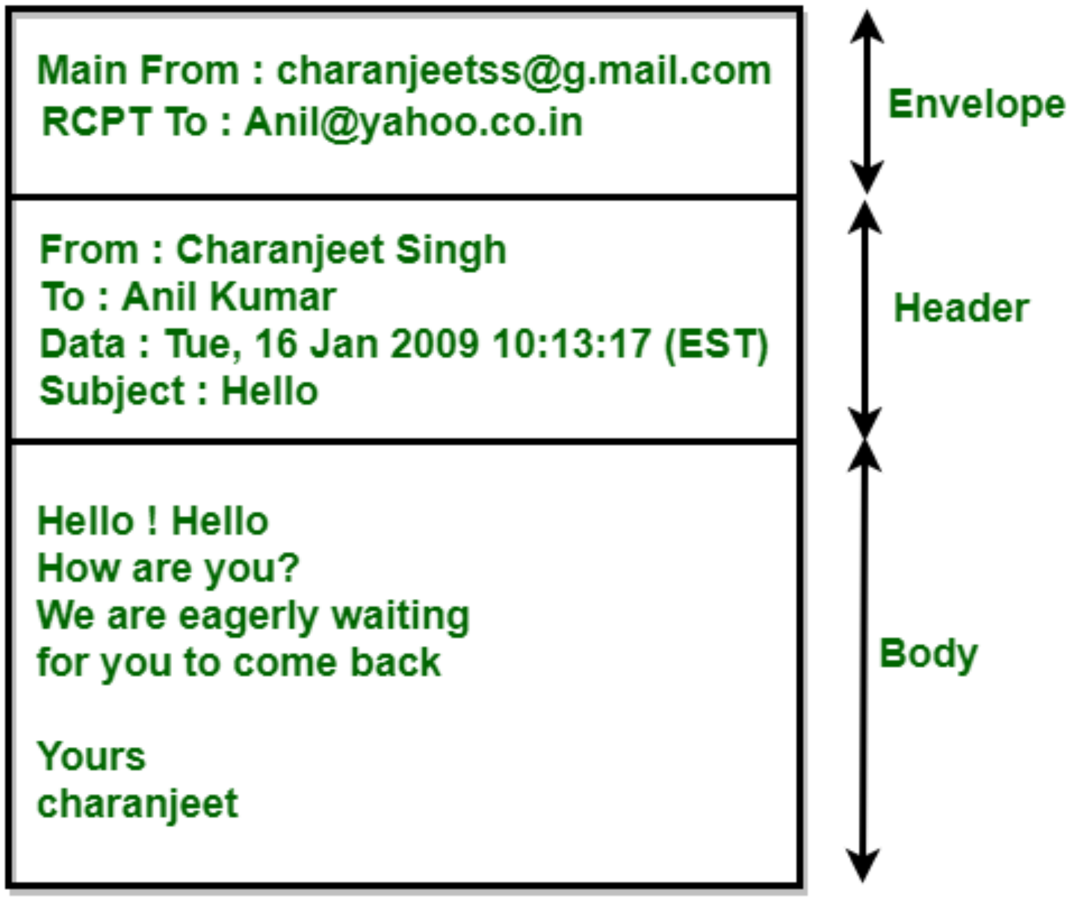
- **From**: Email address of the sender.
- **To**: Email address of the recipient.
- **CC**: (Carbon Copy) Other recipients who receive a copy.
- **BCC**: (Blind Carbon Copy) Recipients who receive a copy without others knowing.
- **Subject**: A brief description of the email content.

- **Date**: The date and time when the email was sent.

2. Email Body

The email body contains the main content of the message. It can include text, images, links, attachments, etc. The body can be formatted in plain text or HTML, allowing for rich text formatting.

- **Greeting**: A salutation to the recipient (e.g., "Dear John,").
- **Message**: The actual content or information being communicated.
- **Closing**: A sign-off (e.g., "Best regards,").



Q.4 Write echo client programming using UDP in c. (SKIP)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>

#define SERVER_PORT 8080
#define SERVER_IP "127.0.0.1"
#define BUFFER_SIZE 1024
```

```
int main() {
    int sockfd;
    struct sockaddr_in server_addr;
    char buffer[BUFFER_SIZE];
    socklen_t addr_len = sizeof(server_addr);

    // Create a UDP socket
    if ((sockfd = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    // Set server address
    memset(&server_addr, 0, sizeof(server_addr));
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(SERVER_PORT);

    // Convert IPv4 and IPv6 addresses from text to binary form
    if (inet_pton(AF_INET, SERVER_IP, &server_addr.sin_addr) <= 0) {
        perror("Invalid address or address not supported");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

    // Input message from user
    printf("Enter message: ");
    fgets(buffer, BUFFER_SIZE, stdin);

    // Send message to the server
    ssize_t sent_bytes = sendto(sockfd, buffer, strlen(buffer), 0,
                                (const struct sockaddr
*)&server_addr,
                                addr_len);

    if (sent_bytes < 0) {
        perror("Message send failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

    // Wait for the server's response (echo)
    ssize_t received_bytes = recvfrom(sockfd, buffer, BUFFER_SIZE, 0,
                                      (struct sockaddr
*)&server_addr,
                                      &addr_len);
    if (received_bytes < 0) {
        perror("Message receive failed");
        close(sockfd);
        exit(EXIT_FAILURE);
    }

    // Null-terminate the received message and print it
    buffer[received_bytes] = '\0';
    printf("Received echo: %s\n", buffer);

    // Close the socket
    close(sockfd);

    return 0;
}
```

Q.5 Write a TCP program in JAVA. (SKIP)

TCP Server in Java

```
import java.io.*;
import java.net.*;

public class TcpServer {
    public static void main(String[] args) {
        int port = 8080; // Server port
        try (ServerSocket serverSocket = new ServerSocket(port)) {
            System.out.println("Server is listening on port " + port);

            // Continuously listen for incoming connections
            while (true) {
                Socket socket = serverSocket×accept();
                System.out.println("New client connected");

                // Handle client in a separate thread
                new ClientHandler(socket).start();
            }
        } catch (IOException ex) {
            System.out.println("Server exception: " +
ex.getMessage());
            ex.printStackTrace();
        }
    }
}
```

```
class ClientHandler extends Thread {
    private Socket socket;

    public ClientHandler(Socket socket) {
        this×socket = socket;
    }

    public void run() {
        try (BufferedReader input = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
             PrintWriter output = new
PrintWriter(socket.getOutputStream(), true)) {

            String message;
            while ((message = input.readLine()) != null) {
                System.out.println("Received: " + message);
                output.println("Echo: " + message); // Echo the
message back to the client
            }
        } catch (IOException ex) {
            System.out.println("ClientHandler exception: " +
ex.getMessage());
        } finally {
            try {
                socket.close();
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

TCP Client in Java

```
import java.io.*;
import java.net.*;

public class TcpClient {
    public static void main(String[] args) {
        String serverAddress = "127.0.0.1"; // Server IP address
        int port = 8080; // Server port

        try (Socket socket = new Socket(serverAddress, port);
             PrintWriter output = new
PrintWriter(socket.getOutputStream(), true);
```

```
             BufferedReader input = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
             BufferedReader consoleInput = new BufferedReader(new
InputStreamReader(System.in))) {

            String userInput;
            System.out.println("Enter messages (type 'exit' to
quit):");
            while ((userInput = consoleInput.readLine()) != null) {
                output.println(userInput); // Send message to the
server
                String response = input.readLine(); // Read echo
response
                System.out.println("Server response: " + response);

                if ("exit".equalsIgnoreCase(userInput)) {
                    break; // Exit loop if user types 'exit'
                }
            }
        } catch (IOException ex) {
            System.out.println("Client exception: " +
ex.getMessage());
            ex.printStackTrace();
        }
    }
}
```

Q.6 Explain FTP. (M 11)

—> File Transfer Protocol (FTP) is a standard network protocol used to transfer files between a client and a server over a TCP/IP network. It allows users to upload, download, delete, rename, move, and copy files on a remote server.

Key Features of FTP

1. **Client-Server Architecture:**
    - FTP operates on a client-server model where the client requests files and the server provides them.
2. **Protocols**: FTP typically uses two separate channels:
    - **Command Channel**: Uses port 21 for sending commands.
    - **Data Channel**: Uses port 20 for transferring files. (Based on active or passive modes)
3. **Modes**:
    - **Active Mode**: The server connects back to the client for data transfer.
    - **Passive Mode**: The client connects to the server for both commands and data transfer, making it more firewall-friendly.
4. **Authentication**: FTP can require a username and password for access, though anonymous FTP allows users to access files without authentication.
5. **Transfer Types**:

- **ASCII Mode**: For text files; converts line endings as needed.
- **Binary Mode**: For non-text files (e.g., images, executables); transfers files without conversion.

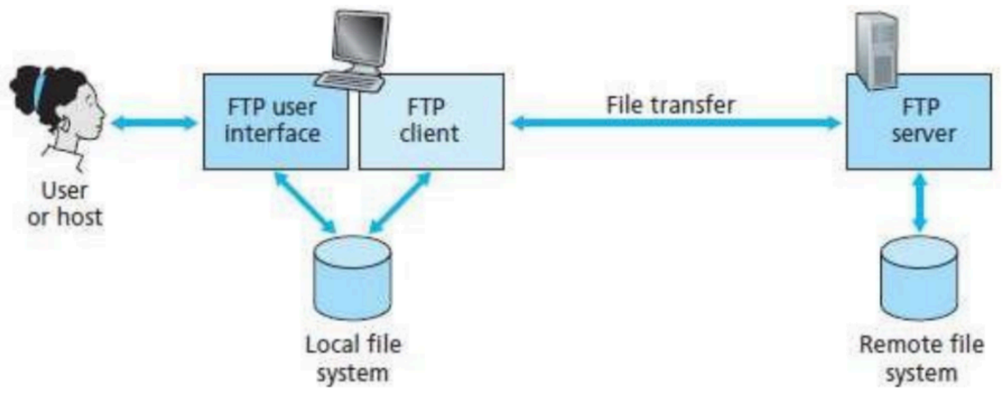


Fig. 9 FTP moves files between local and remote file systems

Advantages of FTP:

1. **Efficient File Transfer:** FTP can handle large files and multiple files at once, making it suitable for transferring large datasets.
2. **Interoperability:** Supported on most operating systems and platforms.

Disadvantages of FTP:

1. **Security Risks:** Sensitive data, including usernames and passwords, are transmitted in plaintext, making it vulnerable to interception.
2. **Complex Configuration:** Setting up FTP servers and managing user permissions can be complex.

Q.7 Explain Telnet. (M 112)

—> Telnet is a network protocol used to provide a command-line interface for communication with a remote device or server. It allows users to connect to remote systems and execute commands as if they were logged into the local machine.

Key Features of Telnet

1. **Remote Access:** Telnet allows users to remotely access and manage devices, such as servers, routers, and switches, over a network.
2. **Protocol**: Operates over TCP, typically using port 23 for communication.
3. **No Encryption**: Data sent over Telnet, including usernames and passwords, is transmitted in plain text, making it susceptible to eavesdropping.
4. **Session Control**: Users can control sessions, including the ability to log in, execute commands, and log out.
5. Bidirectional Communication: Telnet supports bidirectional communication, allowing both the client and server to send and receive data.

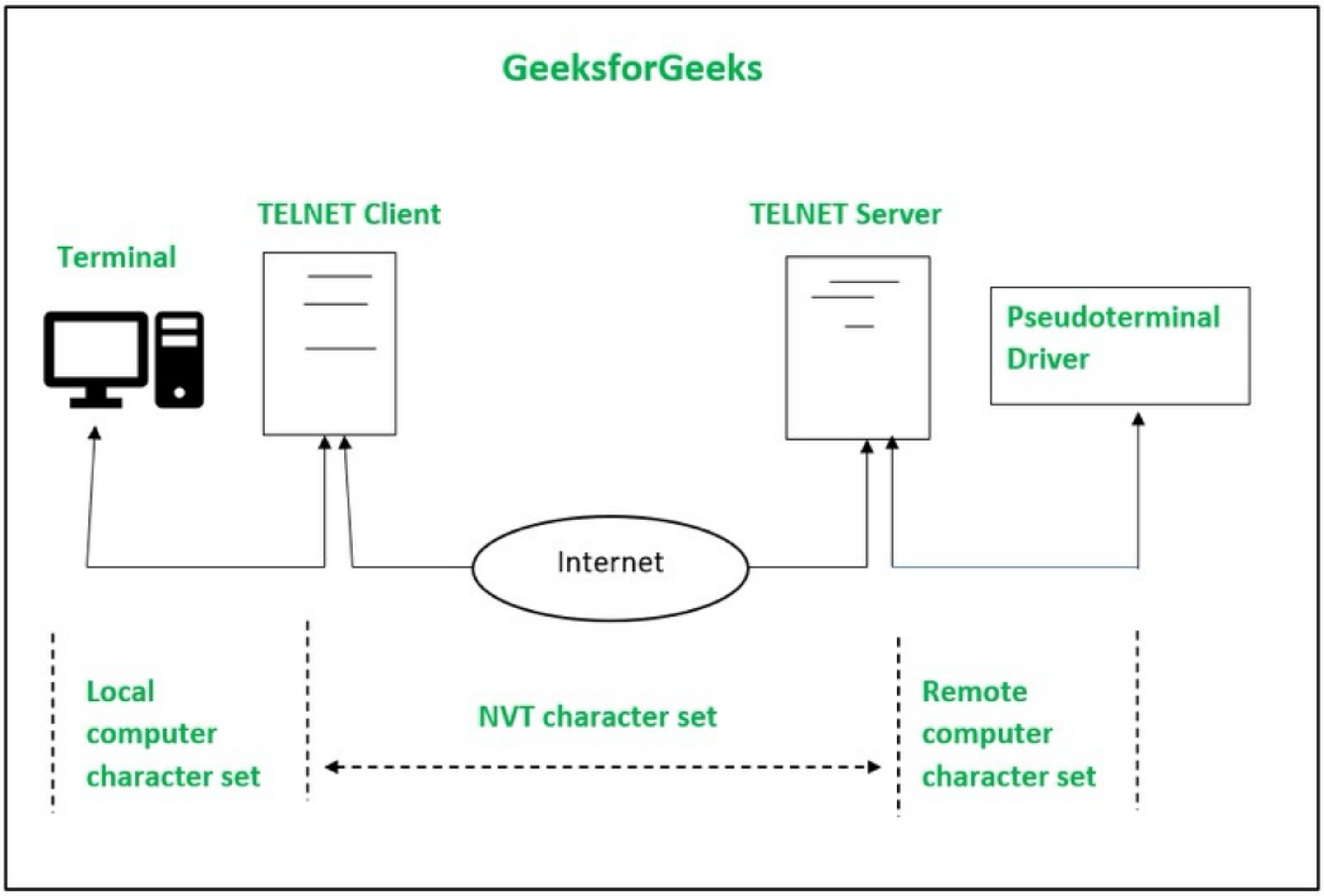
Advantages of Telnet

1. **Simplicity:** Telnet is straightforward to use for command-line operations and is widely supported.
2. **Cross-Platform:** It can be used on various operating systems, allowing for compatibility across different

systems.

Disadvantages of Telnet

1. **Security Risks:** The lack of encryption means that data is susceptible to interception. Sensitive information, like passwords, can be easily captured.
2. **Limited Functionality:** Compared to more modern protocols like SSH (Secure Shell), Telnet lacks many features related to security and encryption.



Q.8 Explain URL, URI, URN?

—> URLs, URIs, and URNs are all terms used in the context of web resources, but they have distinct meanings and purposes.

1. URL (Uniform Resource Locator)

A URL is a specific type of URI that provides a means of locating a resource by describing its primary access mechanism (network location or name). In simple terms, URLs specify where a resource is located and how to retrieve it.

Components:

- **Protocol/Scheme:** Specifies how to access the resource (e.g., http, https, ftp).
- **Host:** The domain name or IP address of the server.
- **Port (optional):** The port number on which the server is listening (e.g., :80).
- **Path:** The specific resource location on the server.
- **Query (optional):** Additional parameters for the request.
- **Fragment (optional):** A specific part of the resource.

Example:
https://www.example.com/index.html
Extra - (https://www.example.com:443/path/to/resource?query=1#fragment)

2. URI (Uniform Resource Identifier)
A URI is a broader term that encompasses both URLs and URNs. It is a string of characters that identifies a resource on the internet either by location (URL) or by name (URN) and extensible way to describe resources.

Components:
- **Scheme:** Specifies the protocol (e.g., http, https, ftp).
- **Authority:** Contains information such as the domain name or IP address.
- **Path:** The specific location of the resource on the server.
- **Query:** Optional parameters for the request.
- **Fragment:** A reference to a specific section within the resource.

Example:
http://www.example.com/index.html?user=123#section1
Extra - (mailto:info@example.com)

3. URN (Uniform Resource Name)
A URN is another type of URI that serves as a unique identifier for a resource without implying its location or how to access it. URNs are intended to serve as persistent, location-independent resource identifiers.

Example:
urn:isbn:0451450523