MODULE 4: NETWORK LAYER

Q.1) IPv4 HEADER FORMAT

IP stands for **Internet Protocol** and **v4** stands for **Version Four** (IPv4).

IPv4 is a connectionless protocol used for packet-switched networks. **Internet Protocol Version Four** (IPv4) is the fourth revision of the Internet Protocol and a widely used protocol in data communication over different kinds of networks.

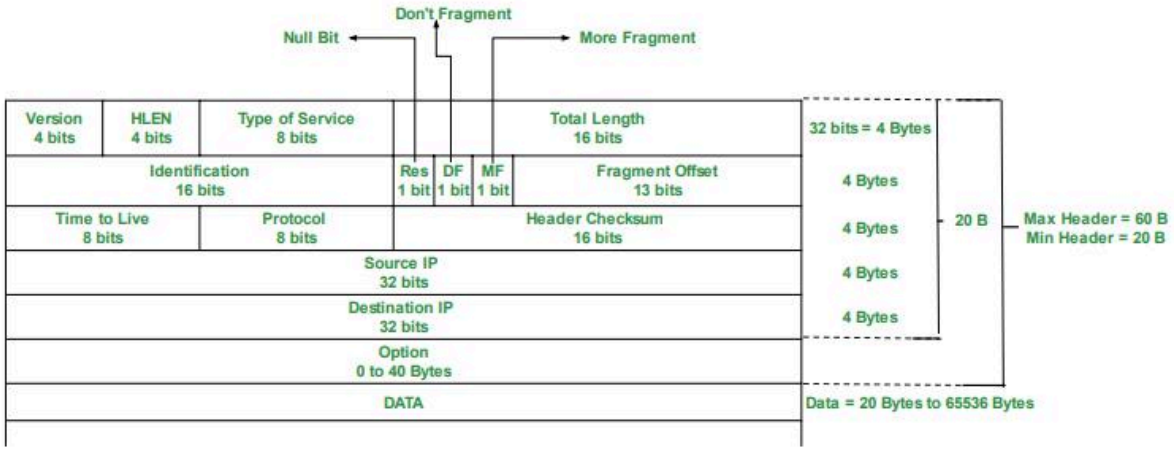
IPv4 is a connectionless protocol used in packet-switched layer networks, such as Ethernet.

It provides a logical connection between network devices by providing identification for each device.

There are many ways to configure IPv4 with all kinds of devices – including manual and automatic configurations – depending on the network type.

IPv4 uses 32-bit addresses for Ethernet communication in five classes: A, B, C, D and E. Classes A, B, and C have a different bit length for addressing the network host.

Class D addresses are reserved for multicasting, while class E addresses are reserved for military purposes. IPv4 addresses are written in the dot-decimal notation, which comprises four octets of the address expressed individually in decimal and separated by periods, for instance, 192.168.1.5.



IPv4 Datagram Header

- **VERSION:** Version of the IP protocol (4 bits), which is 4 for IPv4
- **HLEN:** IP header length (4 bits), which is the number of 32 bit words in the header. The minimum value for this field is 5 and the maximum is 15.
- **Type of service:** Low Delay, High Throughput, Reliability (8 bits)
- **Total Length:** Length of header + Data (16 bits), which has a minimum value 20 bytes and the maximum is 65,535 bytes.
- **Identification:** Unique Packet Id for identifying the group of fragments of a single IP datagram (16 bits)
- **Flags:** 3 flags of 1 bit each : reserved bit (must be zero), do not fragment flag, more fragments flag (same order)
- **Fragment Offset:** Represents the number of Data Bytes ahead of the particular fragment in the particular Datagram. Specified in terms of number of 8 bytes, which has the maximum value of 65,528 bytes.
- **Time to live:** Datagram's lifetime (8 bits), It prevents the datagram to loop through the network by restricting the number of Hops taken by a Packet before delivering to the Destination.
- **Protocol:** Name of the protocol to which the data is to be passed (8 bits)
- **Header Checksum:** 16 bits header checksum for checking errors in the datagram header
- **Source IP address:** 32 bits IP address of the sender
- **Destination IP address:** 32 bits Ip adress of the receiver
- **Option:** Optional information such as source route, record route. Used by the Network administrator to check whether a path is working or not.

Q.2)Classfull and Classless IPv4.

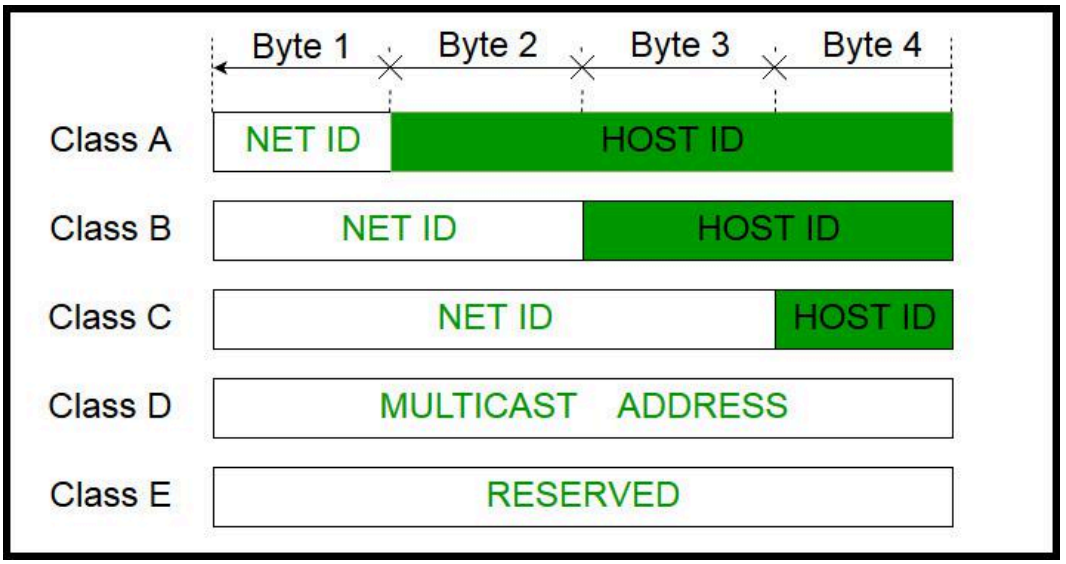
Classful and classless addressing are methods used in networking to manage IP addresses. Classful addressing divides IP addresses into fixed classes (A, B,

C, D, E), each with predefined ranges. In contrast, classless addressing, also known as CIDR (Classless Inter-Domain Routing), offers more flexibility by allowing addresses to be subdivided into smaller blocks called subnets. This flexibility helps optimize address allocation and supports the growth of the internet by efficiently managing IP address resources.

What is Classful Addressing?

Classful Addressing was introduced in 1981, with classful routing, IPv4 addresses were divided into 5 classes(A to E), each with a predetermined range. The class of an IP address determines the network portion and the host portion based on its class-specific subnet mask. Classful addressing was inflexible and led to inefficiencies in address allocation, which prompted the development of classless addressing (CIDR) for more efficient use of IP address space.

Classes A-C: unicast addresses
Class D: multicast addresses
Class E: reserved for future use



Class A

In a class A address, the first bit of the first octet is always ‘0’. Thus, class A addresses range from 0.0.0.0 to 127.255.255.255(as 01111111 in binary converts to 127 in decimal).

The first 8 bits or the first octet denote the network portion and the rest 24 bits or the 3 octets belong to the host portion. Its Subnet mask is 255.0.0.0.

Example: 10.1.1.1

The actual range of class A addresses is: 1.0.0.0 to 126.255.255.255

Class B

In a class B address, the first octet would always start with ’10’. Thus, class B addresses range from 128.0.0.0 to 191.255.255.255. The first 16 bits or the first two octets denote the network portion and the remaining 16 bits or two octets belong to the host portion. Its Subnet mask is 255.255.0.0.

Example: 172.16.1.1

Class C

In a class C address, the first octet would always start with ‘110’. Thus, class C addresses range from 192.0.0.0 to 223.255.255.255. The first 24 bits or the first three octets denote the network portion and the rest 8 bits or the remaining one octet belong to the host portion. Its Subnet mask is 255.255.255.0.

Example: 192.168.1.1

Class D

Class D is used for **multicast addressing** and in a class D address the first octet would always start with ‘1110’. Thus, class D addresses range from 224.0.0.0 to 239.255.255.255. Its Subnet mask is not defined.

Example: 239.2.2.2

Class D addresses are used by routing protocols like OSPF, RIP, etc.

Class E

Class E addresses are reserved for research purposes and future use. The first octet in a class E address starts with ‘1111’. Thus, class E addresses range from 240.0.0.0 to 255.255.255.255. Its Subnet mask is not defined.

Classless IPv4 Addressing (CIDR)

Overview:

- Classless Inter-Domain Routing (CIDR) was introduced to overcome the limitations of classful addressing.
- CIDR allows for variable-length subnet masking (VLSM), enabling more efficient use of IP addresses.

Characteristics:

- **Variable Length**: Subnet masks can be of any length, allowing for more precise control over the number of available addresses in a network.
- **CIDR Notation**: Addresses are often represented using CIDR notation, which includes the base IP address followed by a slash and the number of bits in the subnet mask (e.g., `192.168.1.0/24`).
- **More Efficient**: Reduces wastage by allowing organizations to obtain IP addresses tailored to their specific needs, thus accommodating the growth of the internet.
- **Hierarchical Routing**: CIDR improves routing efficiency by aggregating routes, which reduces the size of routing tables.

Example of CIDR:

- A network with a base address of `192.168.0.0/22` can accommodate up to 1,022 usable hosts. This flexibility allows an organization to optimize its IP address allocation based on actual needs.

Q.3) EXPLAIN DIJKSTRA"S AGORITHM AS SHORTEST PATH ROUTING WITH EXAMPLE

Dijkstra's algorithm is a popular algorithm used to find the shortest path from a starting node (source) to all other nodes in a weighted graph. This algorithm is particularly useful in various applications, such as network routing, where finding the most efficient path is essential.

How Dijkstra's Algorithm Works

1. **Initialization**:
    - Set the initial node's distance to zero and all other nodes' distances to infinity.
    - Create a priority queue (or min-heap) to store nodes based on their distances.
2. **Process Nodes**:
    - While there are unvisited nodes:
        1. Extract the node with the smallest distance from the priority queue (let's call this node `current`).
        2. For each neighboring node of `current`, calculate the tentative distance from the source node to this neighbor through `current`.
        3. If the calculated distance is less than the previously known distance, update the shortest distance and set the predecessor of the neighbor to `current`.
        4. Add the neighbor to the priority queue if it is not already present.
3. **Repeat** until all nodes have been visited or the shortest path to the desired node is found.
4. **Construct the Path**: Once the algorithm completes, the shortest paths to all nodes from the source can be retrieved.

Example

Consider the following weighted graph:

Graph Representation:

- Nodes: A, B, C, D, E
- Edges with weights:
  - A → B (1)
  - A → C (4)
  - B → C (3)
  - B → D (2)
  - C → E (1)
  - D → E (6)

Step-by-Step Execution of Dijkstra's Algorithm from Node A:

1. **Initialization**:
   - Distances:
     - A: 0 (source)
     - B: ∞
     - C: ∞
     - D: ∞
     - E: ∞
   - Priority Queue: `[(0, A)]` (Distance, Node)
2. **First Iteration**:
   - Current node: A (Distance 0)
   - Update neighbors:
     - B: 0 + 1 = 1 (Update)
     - C: 0 + 4 = 4 (Update)
   - Distances:
     - A: 0
     - B: 1
     - C: 4

    - D: ∞
    - E: ∞
  - Priority Queue: `[(1, B), (4, C)]`
3. **Second Iteration**:
  - Current node: B (Distance 1)
  - Update neighbors:
    - C: min(4, 1 + 3) = 4 (No Update)
    - D: 1 + 2 = 3 (Update)
  - Distances:
    - A: 0
    - B: 1
    - C: 4
    - D: 3
    - E: ∞
  - Priority Queue: `[(3, D), (4, C)]`
4. **Third Iteration**:
  - Current node: D (Distance 3)
  - Update neighbors:
    - E: 3 + 6 = 9 (Update)
  - Distances:
    - A: 0
    - B: 1
    - C: 4
    - D: 3
    - E: 9
  - Priority Queue: `[(4, C), (9, E)]`
5. **Fourth Iteration**:

- Current node: C (Distance 4)
- Update neighbors:
  - E: min(9, 4 + 1) = 5 (Update)
- Distances:
  - A: 0
  - B: 1
  - C: 4
  - D: 3
  - E: 5
- Priority Queue: `[(5, E)]`

6. **Fifth Iteration**:
   - Current node: E (Distance 5)
   - No unvisited neighbors.
   - Priority Queue: `[]` (empty)

Final Distances from Node A:

- A: 0
- B: 1
- C: 4
- D: 3
- E: 5 ii

Shortest Paths:

- **A to B**: A → B (Distance 1)
- **A to C**: A → B → C (Distance 4)
- **A to D**: A → B → D (Distance 3)
- **A to E**: A → B → D → E or A → B → C → E (Distance 5)

Conclusion

Dijkstra's algorithm efficiently finds the shortest paths from a source node to all other nodes in a weighted graph. It is widely used in routing protocols and other applications where optimal pathfinding is required. The algorithm's efficiency is significantly enhanced when implemented with a priority queue, making it suitable for large-scale networks.

Q.4) EXPLAIN THE CONTENTS AND THE RQUIREMENTS OF LINK STATE PACKET. WHAT ARE THE STEPS INVOLVED IN LINK STATE ROUTING

Link State Packet (LSP)

A Link State Packet (LSP) is a fundamental component of Link State Routing protocols. It contains essential information about a router's state and the state of its links, enabling other routers in the network to construct a complete view of the network topology. This information is critical for efficient routing decisions.

Contents of a Link State Packet

The exact structure of a Link State Packet can vary depending on the specific Link State Routing protocol (e.g., OSPF, IS-IS), but generally, an LSP includes the following components:

1. **Router ID**:
    - A unique identifier for the router originating the LSP. This is typically an IP address.
2. **Link State Information**:
    - **Link Identifiers**: Information about each link (interface) that the router has, which may include link addresses and types (e.g., point-to-point, broadcast).
    - **Link Costs**: The cost or metric associated with each link, indicating the resource utilization or distance to reach the neighbor.
3. **Sequence Number**:

- A number that allows routers to determine the freshness of the LSP. It helps prevent outdated information from being used.

4. **Timestamp**:
    - A timestamp indicating when the LSP was generated. This helps in managing the staleness of the information.
5. **Acknowledgment**:
    - A field that may be used to confirm receipt of the LSP by neighboring routers.
6. **Other Information**:
    - Depending on the protocol, additional fields may be present, such as flags indicating the status of the router (up, down, etc.) or information on router capabilities.

Requirements for Link State Packet

To ensure effective communication and network operation using Link State Routing, several requirements must be met:

1. **Unique Identifiers**: Each router must have a unique identifier to avoid confusion in the network.
2. **Reliable Transmission**: LSPs must be transmitted reliably, ensuring that all routers receive the latest state information.
3. **Timely Updates**: Routers need to be updated promptly about changes in link states to maintain an accurate view of the network.
4. **Efficient Processing**: Routers must have sufficient processing power and memory to handle incoming LSPs, store the state information, and compute the shortest paths.
5. **Routing Algorithms**: Implementation of a routing algorithm (like Dijkstra's) to calculate the best paths based on the information contained in LSPs.

Steps Involved in Link State Routing

The process of Link State Routing can be broken down into several key steps:

1. **Initialization**:
    - Each router initializes its data structures, including its own link state information and its view of the network topology.

2. **Link State Advertisement (LSA) Generation**:
    - Routers create and populate Link State Packets (LSPs) with information about their links and states. This includes the router ID, link states, costs, and other relevant data.
3. **LSP Dissemination**:
    - The LSPs are flooded throughout the network. Each router sends its LSP to all its neighbors, ensuring that all routers receive the same information about the network topology.
4. **LSP Reception and Processing**:
    - Upon receiving LSPs, routers process the information, updating their link state databases to reflect the current state of the network. This step includes checking the sequence numbers to ensure they are processing the latest information.
5. **Network Topology Construction**:
    - Each router constructs a complete view of the network topology using the collected LSPs. This is essentially a graph representing routers as nodes and links as edges with associated costs.
6. **Shortest Path Calculation**:
    - Using an algorithm (usually Dijkstra's algorithm), routers calculate the shortest paths to all other routers in the network based on the updated topology information.
7. **Routing Table Update**:
    - Routers update their routing tables with the newly calculated shortest paths, enabling them to forward packets efficiently.
8. **Continuous Monitoring**:
    - Routers continuously monitor their links. If a link state changes (e.g., a link goes down), the router generates a new LSP and floods it through the network, repeating the process to ensure all routers have updated information.

Conclusion

Link State Routing, with its use of Link State Packets, provides a robust mechanism for managing routing in complex networks. By disseminating

information about link states and utilizing algorithms to compute shortest paths, Link State Routing ensures efficient and accurate routing. The structured process of generating, distributing, and processing LSPs enables routers to adapt to changes in the network quickly, maintaining optimal routing performance.

Q.5) NOTE ON ARP & RARP

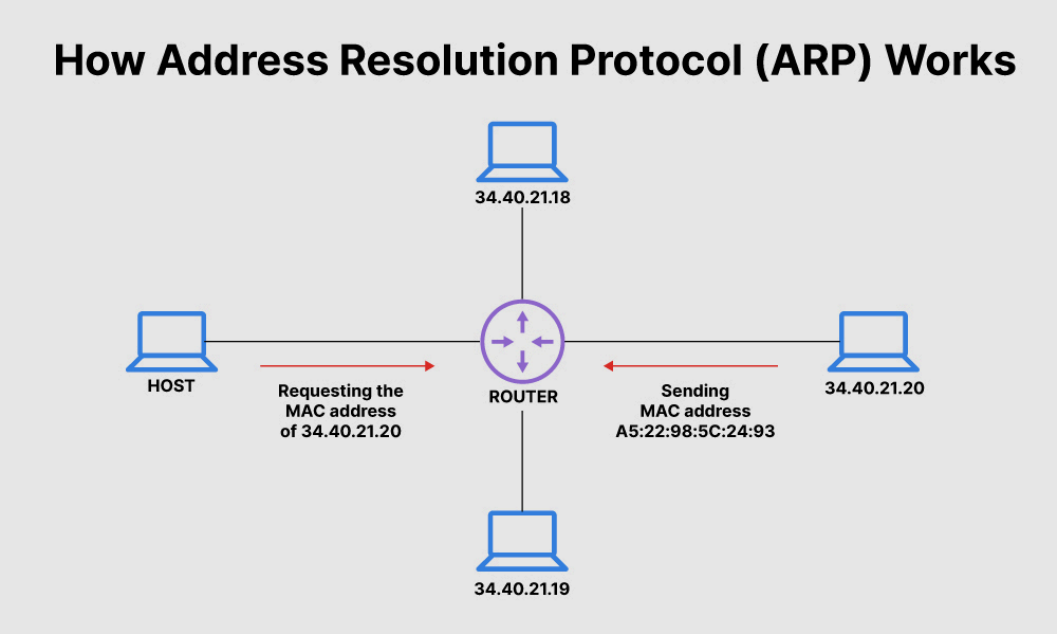
Address Resolution Protocol (ARP)

ARP is a network protocol used for mapping an Internet Protocol (IP) address to a physical machine address (MAC address) that is recognized in the local area network (LAN).

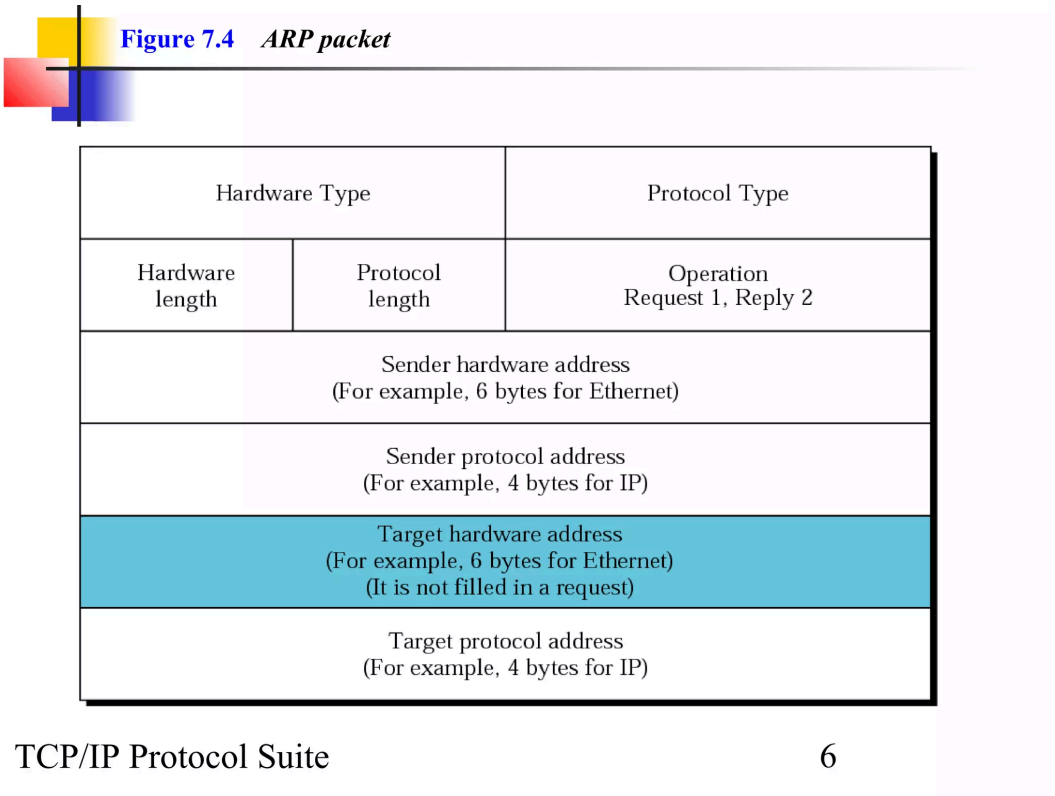
It operates at the link layer (Layer 2) of the OSI model and plays a crucial role in network communication, especially in IPv4 networks.

Key Functions of ARP

1. **Mapping IP Addresses to MAC Addresses**:
    - ARP enables devices to discover the MAC address associated with a given IP address on the same local network. This mapping is essential because while devices communicate using IP addresses, the underlying network uses MAC addresses to transmit frames.
2. **ARP Cache**:
    - Devices maintain an ARP cache, a table that stores IP-to-MAC address mappings for a limited time. This helps to reduce the number of ARP requests by storing previously resolved addresses.
3. **Broadcast Mechanism**:
    - When a device wants to communicate with another device using its IP address but does not know its MAC address, it sends out an ARP request in a broadcast manner to all devices on the local network. The device with the corresponding IP address replies with an ARP response containing its MAC address.



ARP Packet Structure



An ARP packet typically contains the following fields:

1. **Hardware Type**: Specifies the type of hardware used (Ethernet, Token Ring, etc.).
2. **Protocol Type**: Indicates the type of protocol (IPv4, IPv6, etc.).

3. **Hardware Address Length**: The length of the MAC address in bytes (usually 6 bytes for Ethernet).
4. **Protocol Address Length**: The length of the IP address in bytes (4 bytes for IPv4).
5. **Operation**: Indicates whether the packet is an ARP request (1) or an ARP response (2).
6. **Sender Hardware Address**: The MAC address of the sender.
7. **Sender Protocol Address**: The IP address of the sender.
8. **Target Hardware Address**: The MAC address of the intended recipient (set to all zeros in requests).
9. **Target Protocol Address**: The IP address of the intended recipient.

Example of ARP Process

1. **Host A** wants to send data to **Host B** (IP: 192.168.1.2) but does not know its MAC address.
2. Host A broadcasts an ARP request: "Who has IP address 192.168.1.2? Tell 192.168.1.1."
3. Host B receives the request and sends an ARP response: "I have IP address 192.168.1.2, and my MAC address is AA:BB:CC:DD:EE:FF."
4. Host A receives the response and stores the mapping in its ARP cache for future use.

Reverse Address Resolution Protocol (RARP)

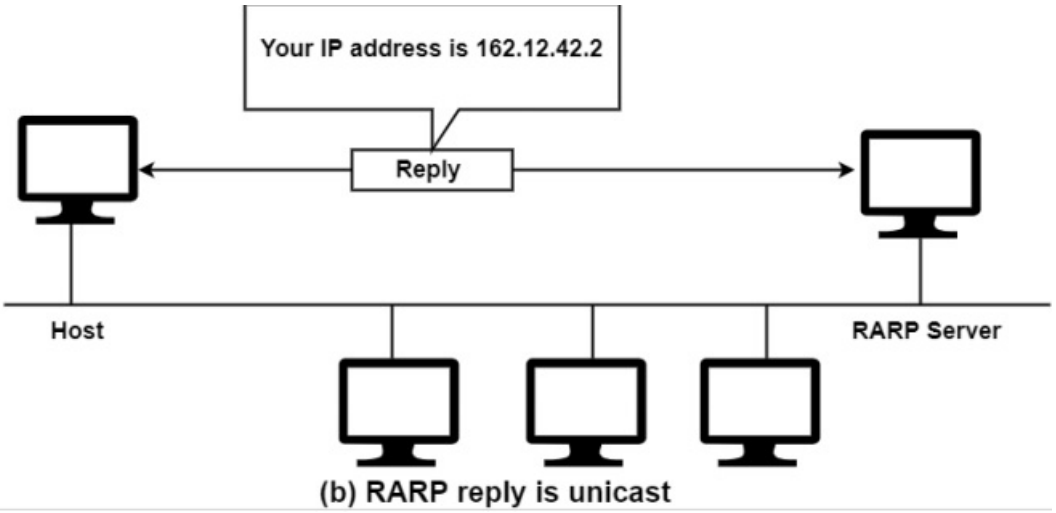
RARP is a network protocol used to map a physical machine address (MAC address) to an Internet Protocol (IP) address. It is essentially the reverse of ARP and operates at the link layer as well. However, RARP is less commonly used today due to the advent of more advanced protocols like DHCP (Dynamic Host Configuration Protocol).

Key Functions of RARP

1. **Obtaining IP Addresses**:
    - RARP enables devices (often diskless workstations) to obtain an IP address from a network server when they only know their MAC

address. This is crucial for devices that do not have a local storage medium to store their IP address configuration.

2. **RARP Server**:
   - RARP requests are sent to a RARP server, which responds with the corresponding IP address associated with the requesting device's MAC address.



(b) RARP reply is unicast

RARP Packet Structure

Figure 7.11 *RARP packet*

Hardware type		Protocol type
Hardware length	Protocol length	Operation Request 3, Reply 4
Sender hardware address (For example, 6 bytes for Ethernet)		
Sender protocol address (For example, 4 bytes for IP) (It is not filled for request)		
Target hardware address (For example, 6 bytes for Ethernet) (It is not filled for request)		
Target protocol address (For example, 4 bytes for IP) (It is not filled for request)		

TCP/IP Protocol Suite 27

Similar to ARP, RARP packets include fields such as:

1. **Hardware Type**: Specifies the type of hardware (Ethernet, etc.).
2. **Protocol Type**: Indicates the type of protocol (IPv4).
3. **Hardware Address Length**: Length of the MAC address.
4. **Protocol Address Length**: Length of the IP address.
5. **Operation**: Indicates whether the packet is a RARP request or response.
6. **Sender Hardware Address**: The MAC address of the device requesting the IP.
7. **Sender Protocol Address**: Set to all zeros in RARP requests.
8. **Target Hardware Address**: The MAC address of the requesting device.
9. **Target Protocol Address**: The IP address of the requesting device (set to all zeros in requests).

Example of RARP Process

1. **Diskless Host** with MAC address AA:BB:CC:DD:EE:FF needs to obtain an IP address.
2. It broadcasts a RARP request: "Who has MAC address AA:BB:CC:DD:EE:FF? Tell me my IP."
3. A RARP server on the network receives the request and replies with a RARP response containing the IP address assigned to that MAC address.
4. The diskless host receives the response and configures its network interface with the provided IP address.

Differences Between ARP and RARP

Feature	ARP (Address Resolution Protocol)	RARP (Reverse Address Resolution Protocol)
Purpose	Maps IP addresses to MAC addresses	Maps MAC addresses to IP addresses
Operation	Used by hosts to find MAC addresses of other hosts	Used by hosts to find IP addresses based on their MAC addresses
Use Case	Essential for normal IP communication in networks	Historically used by diskless devices to obtain IP addresses

Protocol Status	Widely used and essential in IPv4 networks	Less common, largely replaced by DHCP

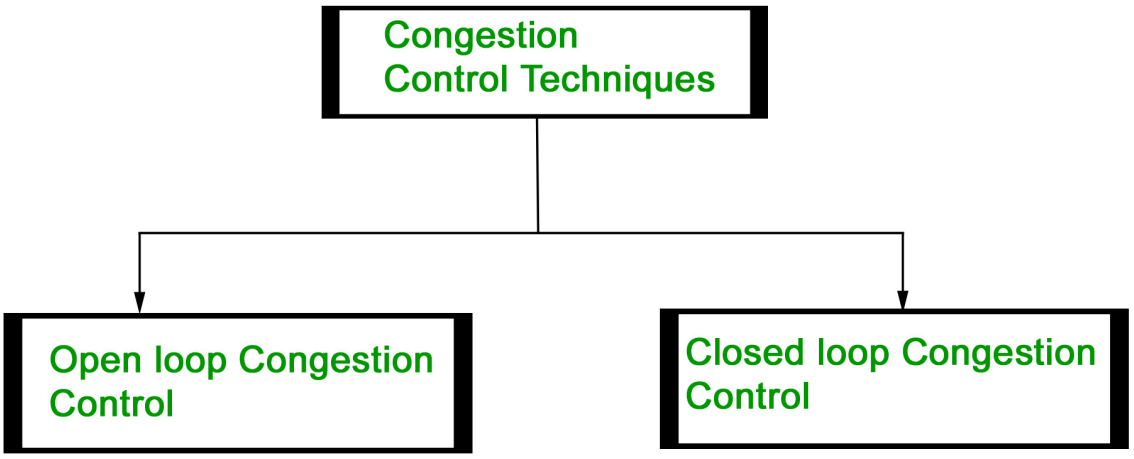
Conclusion

ARP and RARP are critical protocols in network communication, particularly in IPv4 networks. While ARP is essential for resolving IP addresses to MAC addresses and facilitating communication between devices, RARP provided a solution for devices needing to discover their IP addresses based on their MAC addresses. Today, RARP is largely superseded by DHCP, which offers more flexibility and features for dynamic IP address assignment.

Q.6) WHAT IS CONGESTION CONTROL? EXPLAIN OPEN LOOP AND CLOSE LOOP CONGESTION CONTROL

Congestion Control

Congestion Control refers to a set of mechanisms and techniques used in computer networks to manage and prevent network congestion, which occurs when the demand for network resources exceeds the capacity available. This can result in packet loss, delays, and decreased overall network performance. Effective congestion control is essential for ensuring the smooth flow of data and maintaining the quality of service (QoS) in a network.



Open Loop Congestion Control

Open loop congestion control is a proactive approach that seeks to prevent congestion before it occurs. It involves implementing certain policies and strategies to control the amount of data entering the network, without relying on feedback from the network's current state.

Open Loop Policies

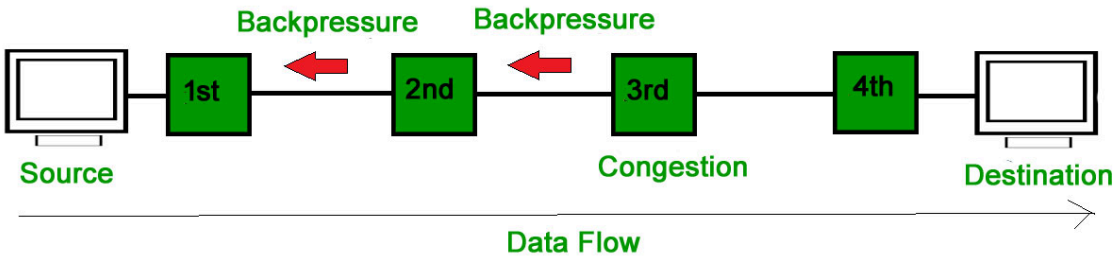
1. **Retransmission Policy**:
    - In open loop control, the strategy for retransmitting lost packets is predefined. If a packet is lost, the sender may wait for a certain timeout before retransmitting, rather than relying on acknowledgments. This policy helps limit the sending rate to avoid exacerbating congestion.
2. **Window Control**:
    - This policy involves setting a fixed maximum window size for data transmission. The sender can only send a limited number of packets before needing an acknowledgment. By maintaining a smaller window size, the sender can prevent overwhelming the network with too much data at once.
3. **Discarding Policy**:
    - When congestion occurs, routers may drop packets that arrive when the buffer is full. This policy helps manage resources by controlling the flow of packets into the network, though it may lead to increased packet loss.
4. **Acknowledgment Policy**:
    - The acknowledgment strategy may involve using cumulative acknowledgments, where the receiver acknowledges the highest sequence number received. This approach can help optimize the sender's behavior, as it allows for efficient retransmission strategies.
5. **Admission Control**:
    - Before allowing new connections or data flows into the network, the system assesses the current capacity and existing traffic. If the network is congested or nearing capacity, new sessions may be denied entry, effectively managing the load on the network.

Closed Loop Congestion Control

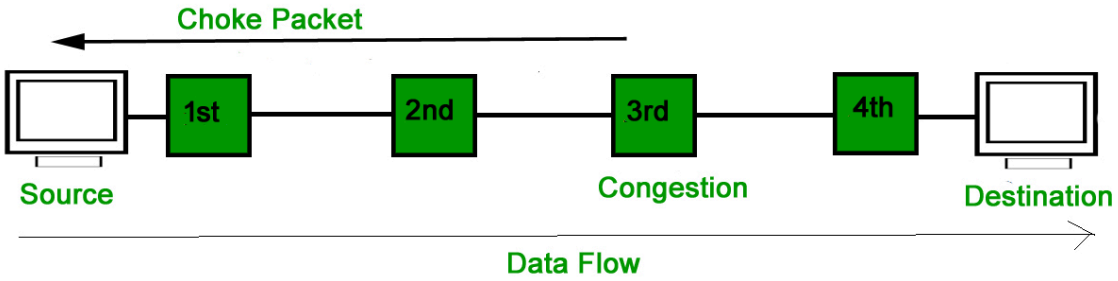
Closed loop congestion control is a reactive approach that responds to congestion after it has been detected. It involves monitoring network performance and adjusting traffic flow based on feedback mechanisms.

Closed Loop Techniques

1. **Backpressure**:
    - This technique involves sending feedback signals from routers to the sender when congestion is detected. Routers can hold packets in queues and notify the sender to reduce its sending rate. Backpressure is particularly useful in networks where the congestion is localized, allowing the sender to adjust dynamically based on router feedback.



2. **Choke Packet Technique**:
    - In this method, a router sends a "choke packet" to the sender when it detects congestion. The choke packet instructs the sender to reduce its transmission rate. This technique provides an explicit signal to the sender to slow down, helping to alleviate congestion.



3. **Implicit Signaling**:

- This technique relies on the observation of packet loss or delay as an implicit signal of congestion. For example, if a sender notices that acknowledgments are delayed or that packets are being lost, it infers that congestion is occurring and reduces its sending rate accordingly. This approach does not involve explicit feedback but relies on the sender's interpretation of network conditions.

4. **Explicit Signaling**:
    - Explicit signaling involves routers sending specific signals to the sender regarding network conditions. This can include congestion notifications or requests to reduce the data rate. Protocols like Explicit Congestion Notification (ECN) fall into this category, where routers mark packets to indicate congestion without dropping them, allowing the sender to adjust its rate accordingly.

Q.7) Distance Vector Routing (DVR) Protocol

Distance Vector Routing (DVR) Protocol is a method used by routers to find the best path for data to travel across a network.

Each router keeps a table that shows the shortest distance to every other router, based on the number of hops (or steps) needed to reach them.

Routers share this information with their neighbors, allowing them to update their tables and find the most efficient routes.

This protocol helps ensure that data moves quickly and smoothly through the network.

The protocol requires that a router inform its neighbors of topology changes periodically.

Bellman-Ford Basics

Each router maintains a Distance Vector table containing the distance between itself and All possible destination nodes. Distances, based on a chosen metric, are computed using information from the neighbors' distance vectors.

> Information kept by DV router:
> Each router has an ID

Associated with each link connected to a router,
there is a link cost (static or dynamic).
Intermediate hops.

Distance Vector Table Initialization:
Distance to itself = 0
Distance to ALL other routers = infinity number.

How Distance Vector Algorithm works?

- A **router** transmits its distance vector to each of its neighbors in a routing packet.
- Each router receives and saves the most recently received distance vector from each of its neighbors.
- A router recalculates its distance vector when:
  - It receives a distance vector from a neighbor containing different information than before.
  - It discovers that a link to a neighbor has gone down.

The DV calculation is based on minimizing the cost to each destination

Dx(y) = Estimate of least cost from x to y
C(x,v) = Node x knows cost to each neighbor v
Dx = [Dx(y): y ? N] = Node x maintains distance vector
Node x also maintains its neighbors' distance vectors
– For each neighbor v, x maintains Dv = [Dv(y): y ? N]

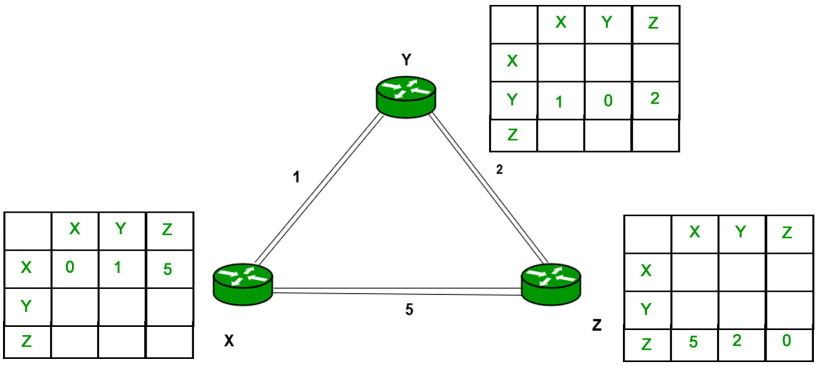
Note:

- From time-to-time, each node sends its own distance vector estimate to neighbors.
- When a node x receives new DV estimate from any neighbor v, it saves v's distance vector and it updates its own DV using B-F equation:

  Dx(y) = min { C(x,v) + Dv(y), Dx(y) } for each node y ? N

Example :

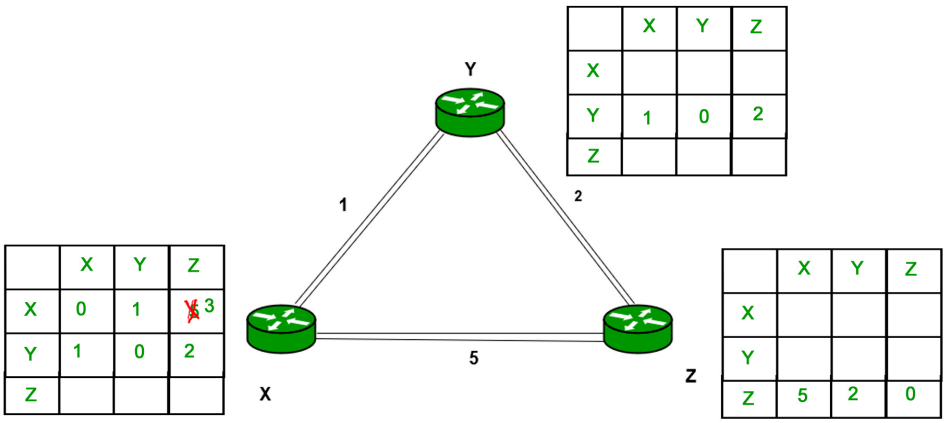
Consider 3-routers X, Y and Z as shown in figure. Each router have their routing table. Every routing table will contain distance to the destination nodes.



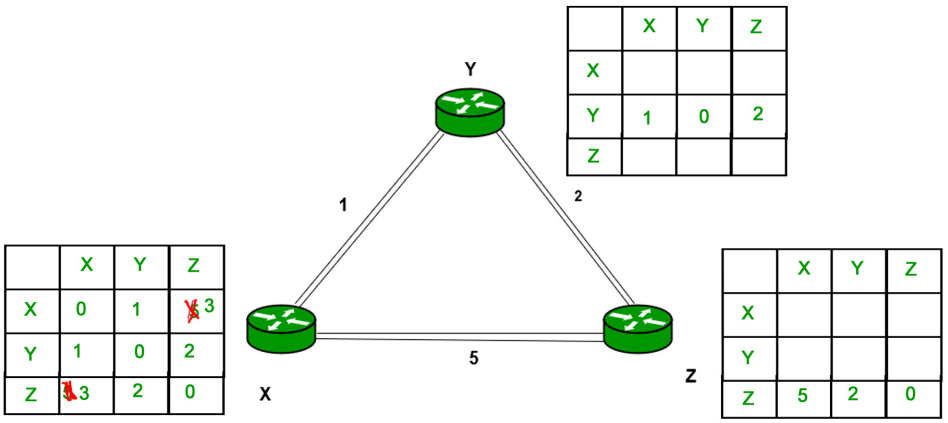
Consider router X , X will share it routing table to neighbors and neighbors will share it routing table to it to X and distance from node X to destination will be calculated using bellmen- ford equation.

$$D_x(y) = \min \{ C(x,v) + D_v(y)\} \text{ for each node } y \in N$$

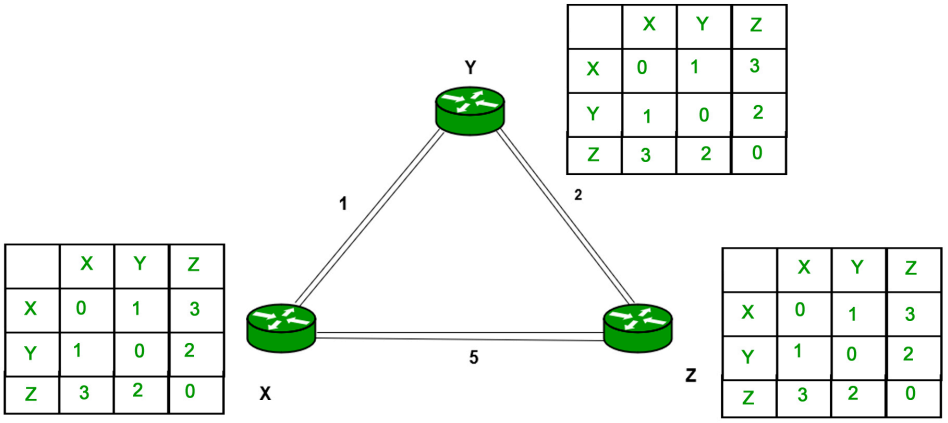
As we can see that distance will be less going from X to Z when Y is intermediate node(hop) so it will be update in routing table X.



Similarly for Z also –



Finally the routing table for all –



Applications of Distance Vector Routing Algorithm

The Distance Vector Routing Algorithm has several uses:

- **Computer Networking** : It helps route data packets in networks.
- **Telephone Systems** : It's used in some telephone switching systems.
- **Military Applications** : It has been used to route missiles.

Q.8) IPv6 header format

The IPv6 header format is designed to be more efficient and streamlined than its predecessor, IPv4. It consists of a fixed 40-byte header, which contains essential information for routing and delivery. Here's a breakdown of the IPv6 header fields:

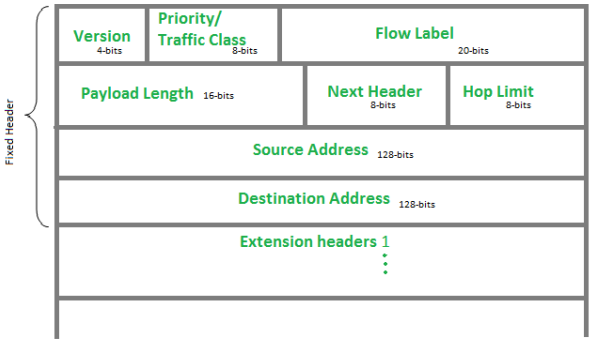
IPv6 Header Format

Field	Length (bits)	Description
Version	4	Specifies the IP version (6 for IPv6).
Traffic Class	8	Used for Quality of Service (QoS) features; includes precedence and flags.
Flow Label	20	Used for identifying a flow of packets with the same characteristics.
Payload Length	16	Indicates the length of the payload (data) following the header, in bytes.
Next Header	8	Specifies the type of the next header (protocol) following the IPv6 header (e.g., TCP, UDP, ICMPv6).
Hop Limit	8	Similar to TTL in IPv4, it specifies the maximum number of hops a packet can make before being discarded.
Source Address	128	The IPv6 address of the sender (originator) of the packet.
Destination Address	128	The IPv6 address of the intended recipient of the packet.

Key Points

- **Version**: Always set to 6 for IPv6.
- **Traffic Class**: Provides information for QoS, helping prioritize traffic.
- **Flow Label**: Used to label sequences of packets for special handling.
- **Payload Length**: This field allows routers to determine how much data to expect.
- **Next Header**: It enables the use of various transport protocols and extensions by indicating the protocol in the next layer.

- **Hop Limit**: Ensures that packets do not circulate indefinitely by limiting the number of hops.
- **Source and Destination Addresses**: Each is a 128-bit address, allowing for a vastly larger number of unique IP addresses compared to IPv4.



This format allows for greater efficiency in routing, better support for modern networking applications, and provides a much larger address space.

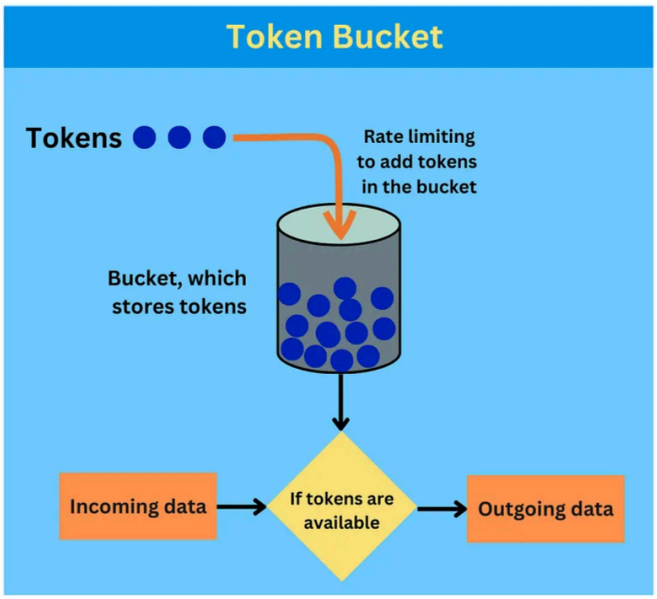
Q.9) NOTE ON TOKEN BUCKET AND LEAKY BUCKET ALGORITHM

The Token Bucket and Leaky Bucket algorithms are both used for traffic shaping and rate limiting in networking. They help manage data flow in a network by controlling the amount of data that can be transmitted over time. Here's a detailed note on each algorithm:

Token Bucket Algorithm

Overview

The Token Bucket algorithm is designed to allow bursts of traffic while enforcing an average rate limit over time. It uses tokens to control the flow of data packets.



How It Works

- **Tokens**: Tokens are generated at a fixed rate and stored in a bucket. The bucket has a maximum capacity (i.e., it can hold a limited number of tokens).
- **Sending Packets**: To send a packet, a token must be available. If a token is present, it is removed from the bucket, and the packet is sent. If no tokens are available, the packet must wait until tokens are replenished.
- **Burstiness**: The algorithm allows for bursts of traffic up to the capacity of the bucket. This means that if several tokens are generated before sending packets, the sender can transmit more data quickly.

Characteristics

- **Average Rate**: The average rate is controlled by the token generation rate.
- **Burst Handling**: It accommodates bursts up to the maximum bucket capacity.
- **Token Replenishment**: Tokens are added to the bucket at a fixed rate, up to the maximum capacity.

Applications

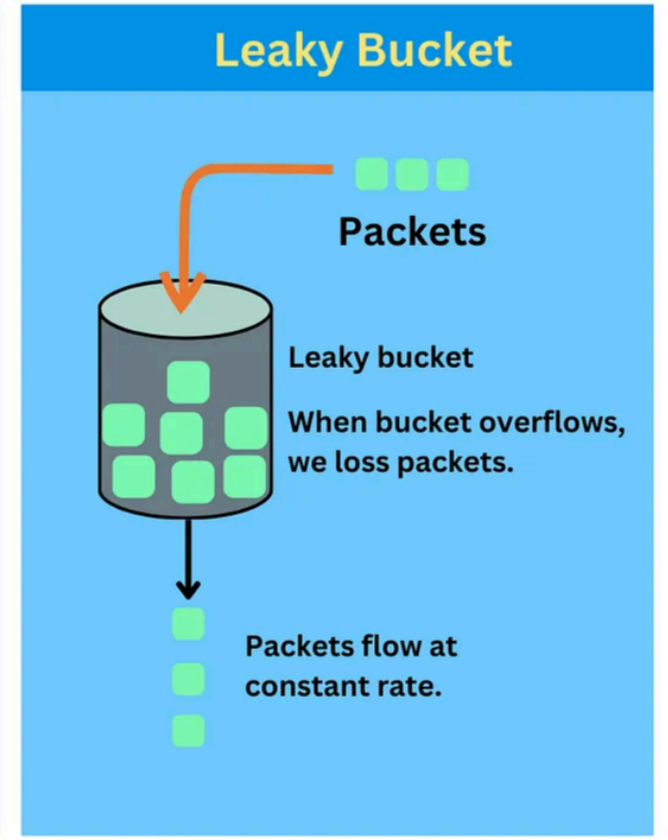
- **Network Traffic Shaping**: Used to manage bandwidth in networks where traffic flows can be bursty.

- **Quality of Service (QoS)**: Ensures that applications requiring a steady flow of packets can get the necessary bandwidth.

Leaky Bucket Algorithm

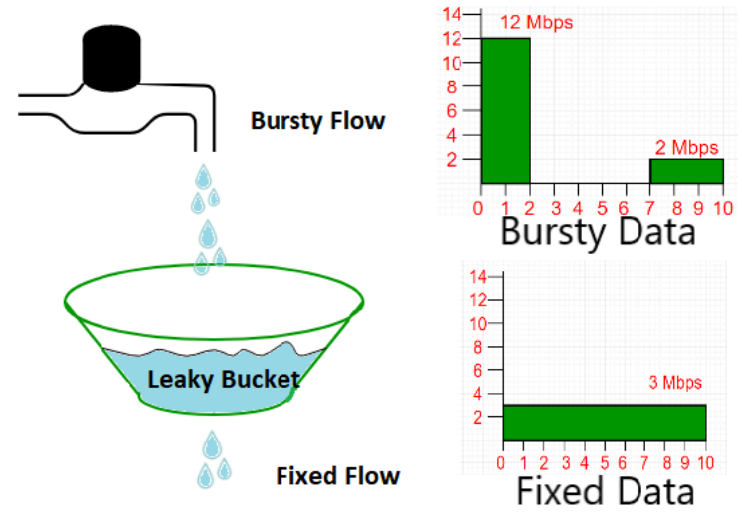
Overview

The Leaky Bucket algorithm enforces a strict output rate for packets. It simulates a bucket with a small hole that leaks water at a constant rate, regardless of how much water (data) is poured into it.



How It Works

- **Bucket Capacity**: The bucket has a fixed capacity, which represents the maximum amount of data that can be held at any time.
- **Leaking Rate**: Data is sent (or leaks out) at a constant rate. This means that packets are sent out steadily over time.
- **Sending Packets**: Incoming packets are added to the bucket until it reaches its maximum capacity. If the bucket is full and new packets arrive, those packets are discarded (or dropped).



Characteristics

- **Constant Rate**: Guarantees a fixed rate of data transmission, providing predictable network behavior.
- **No Bursting**: Unlike the Token Bucket, it does not allow bursts; the data output is smoothed out.
- **Packet Discarding**: Incoming packets are discarded if they arrive when the bucket is full.

Applications

- **Traffic Shaping**: Useful for applications where a constant output rate is critical.
- **Network Congestion Control**: Helps to prevent network congestion by regulating the amount of data sent.