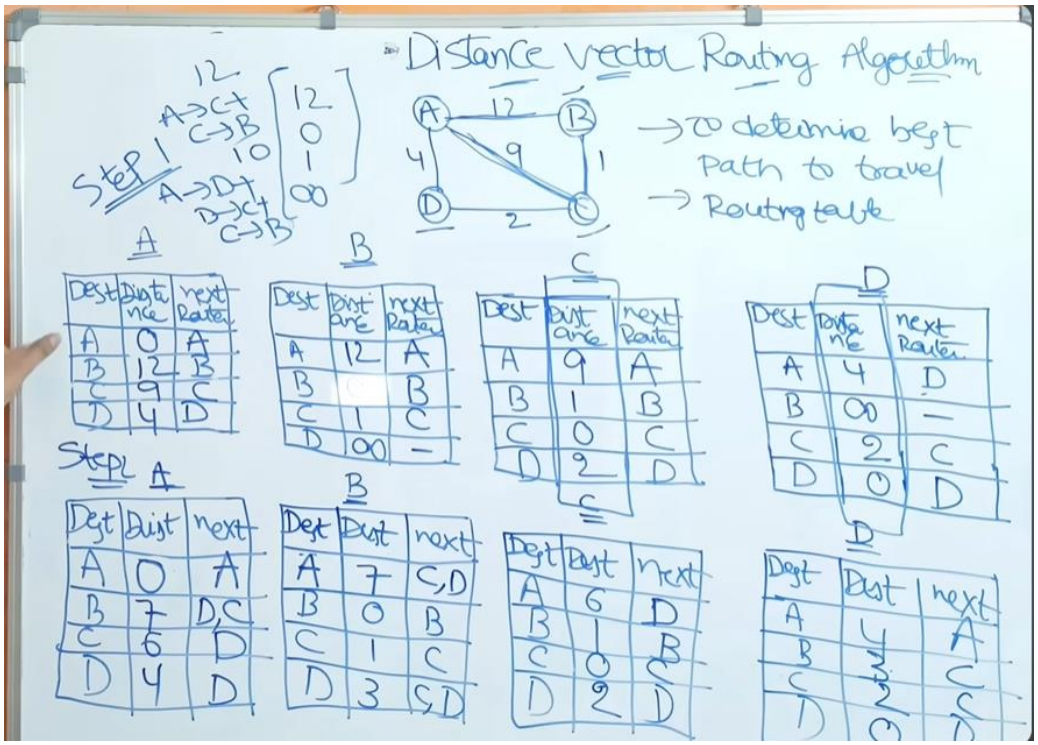
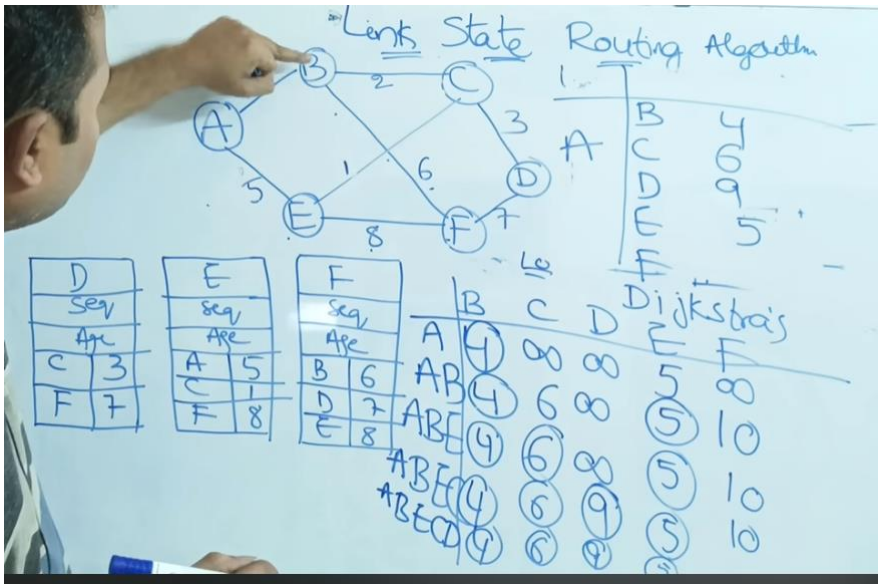
UNIT – 3

Part-1 pdf

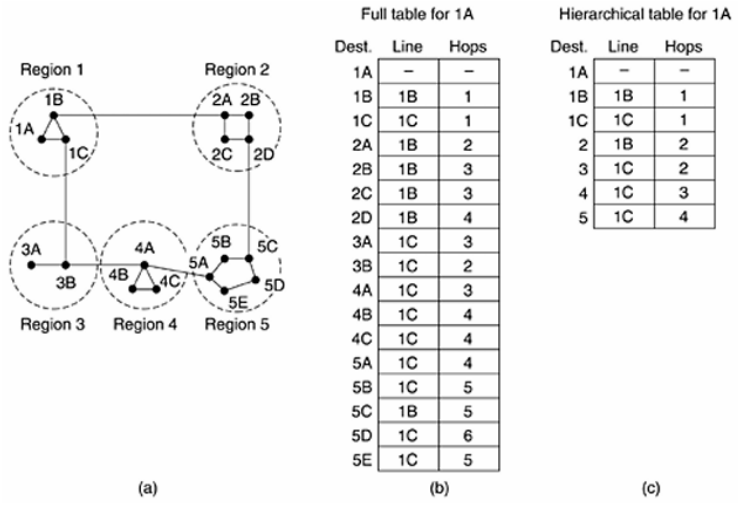




Hierarchical Routing

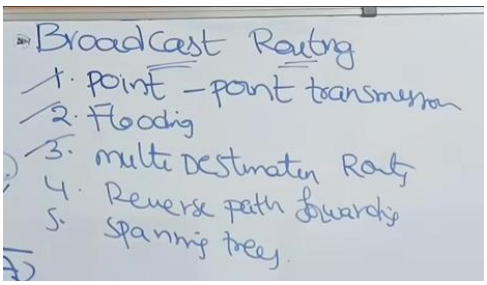
- **Why Needed**: Large networks make routing tables too big and inefficient.
- **How It Works**: Divide routers into **regions**.
    - Routers know details of their region but only general paths to other regions.
- **Example**: A packet from California to Kenya passes through major hubs (Los Angeles → New York → Nairobi → Malindi).
- **Benefits**: Smaller routing tables.
- **Drawbacks**: Routes may be longer.

Figure 5-15. Hierarchical routing.



Broadcast Routing

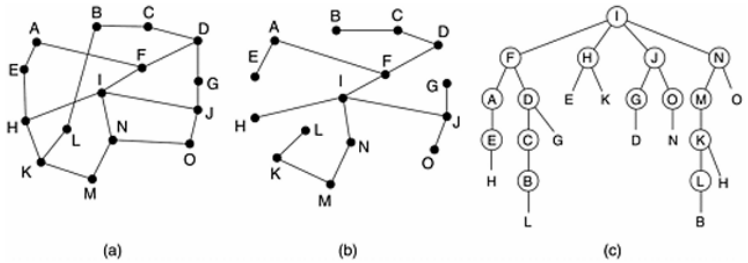
- **Purpose**: Send a message to **all devices** in the network.
- **Methods**:



1. **Send to all destinations individually**:
    - Simple but wasteful of resources.
2. **Flooding**:
    - Each router sends packets to all its neighbors. Very resource-intensive.
3. **Multidestination routing**:
    - A packet lists all destinations, and routers split the list among appropriate paths.
4. **Using a spanning tree**:
    - A subset of network links connects all routers without loops.
    - Routers send broadcast packets only along tree paths, reducing redundancy.
5. **Reverse path forwarding**:
    - Routers check if the packet arrived via the best route to the sender.
    - If yes, forward it to others; if no, discard it as a duplicate.

- **Best Choice**: Reverse path forwarding is efficient, simple, and avoids duplication without extra data like spanning tree knowledge.

Figure 5-16. Reverse path forwarding. (a) A subnet. (b) A sink tree. (c) The tree built by reverse path forwarding.



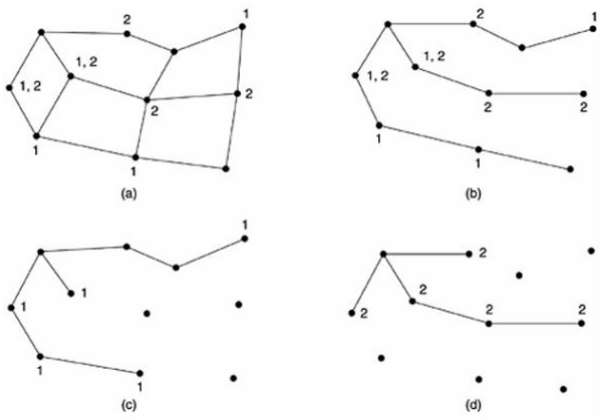
Multicast Routing (Simple Explanation)

- **Purpose**: Send messages to a specific **group of devices**, not all.
- **Why Multicasting?**
  - Broadcasting to everyone wastes bandwidth.
  - Sending individually to each group member is inefficient for large groups.
- **How It Works**:
  - **Group management**: Routers know which devices are in which groups.
  - **Spanning tree**: A tree structure connects only group members, ensuring efficient routing.
  - Routers "prune" the tree to exclude parts not needed for the group.
- **Methods**:

1. **Using link-state routing**:
   - Routers know the network topology and prune the tree based on group membership.
2. **Using distance vector routing**:
   - Routers send "PRUNE" messages to indicate they don't need certain group data, recursively shrinking the network.

- **Challenges**:
  - Storing multiple trees for many groups can require a lot of memory.
  - **Solution**: Use a **core-based tree**, where one central router (core) handles group data, reducing memory requirements.

Figure 5-17. (a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.



Routing in Ad Hoc Networks

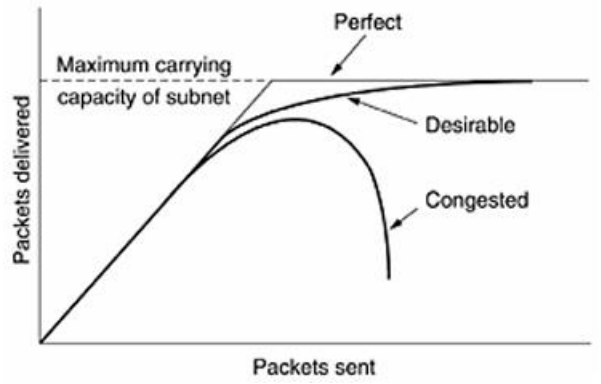
- **What Are Ad Hoc Networks?**
  Networks with mobile routers and no fixed infrastructure, e.g., military vehicles, emergency workers, or devices in areas without Wi-Fi.
- **Challenges:**
  - Constantly changing topology.
  - No fixed relationship between IP addresses and locations.
  - Paths may become invalid quickly.
- **AODV Algorithm (Ad hoc On-demand Distance Vector):**
  - A routing method designed for mobile environments.
  - **On-demand**: Finds routes only when needed.
  - Optimized for limited bandwidth and low battery life.

Understanding Congestion Control

What is Congestion?

- **Congestion** happens in a network when there are **too many packets (data)** being sent through it, leading to performance issues.
- Imagine a busy road with many cars. When there are too many cars, traffic slows down and can come to a standstill. In a network, when too many data packets are sent at once, **routers** can't handle them all, and packets get delayed or even lost.

Figure 5-25. When too much traffic is offered, congestion sets in and performance degrades sharply.



Why Does Congestion Happen?

1. **Too Many Packets Needing the Same Output Line:**
   - When multiple data streams try to use the same path (output line) at the same time, a **queue** forms, just like cars at a traffic signal. If the queue becomes too long, some packets may be lost.

2. **Insufficient Memory in Routers:**
    - Routers store packets in memory while processing them. If the memory is full, new packets can't be stored, leading to packet loss.
    - Surprisingly, adding more memory doesn't always help. If routers had unlimited memory, packets could take too long to be processed and get "timed out," causing duplicates. This would **increase congestion** further, as more unnecessary packets clog the network.
3. **Slow Processors in Routers:**
    - If a router's CPU is slow at handling packets (updating tables, queueing), it can't keep up, causing delays even if the network lines have enough capacity.
4. **Low-Bandwidth Lines:**
    - If the network lines themselves are slow, they can't handle high amounts of traffic, creating a bottleneck.

Congestion Control vs. Flow Control

These two terms are often confused but are different:

1. **Congestion Control**:
    - Focuses on managing the **overall network traffic**.
    - It aims to **prevent the entire network** from being overwhelmed by too much data.
    - Example: If many users are using a network, congestion control tries to ensure the network can handle the load without collapsing.
2. **Flow Control**:
    - Deals with traffic between **specific sender-receiver pairs**.
    - It ensures that a **fast sender** doesn't send data faster than what the **receiver** can handle.
    - Example: If a supercomputer is sending data to a personal computer, flow control prevents the supercomputer from overwhelming the personal computer.

Key Takeaways:

- **Congestion** is about **too much traffic** in the network, causing performance issues.
- **Flow Control** prevents **one sender** from overwhelming **one receiver**.
- **Congestion Control** prevents the **entire network** from being overloaded.

General Principles of Congestion Control

Congestion in a network happens when **too much traffic** exceeds the network's capacity, causing delays and packet losses. To manage this, two main approaches are used: **open-loop** and **closed-loop** congestion control.

1. Open-Loop Congestion Control

- **Preventive Approach:**
    - The idea is to **prevent congestion before it happens** through good network design.

- Once the system is running, it doesn't make mid-course adjustments.

- **Techniques Used:**
  - **Traffic Admission:** Controlling when to accept or reject new traffic.
  - **Packet Discarding:** Deciding which packets to drop when the network is busy.
  - **Scheduling Decisions:** Prioritizing certain packets over others without checking the current network state.

2. Closed-Loop Congestion Control

- **Feedback-Based Approach:**
  - Uses a **feedback loop** to detect and respond to congestion in real-time.
- **Steps Involved:**
  - **Monitoring:** Detect where and when congestion is happening (e.g., by checking packet loss, queue lengths, or delays).
  - **Reporting:** Send feedback about congestion to the places where action is needed (like sending a signal to the traffic source).
  - **Adjusting:** Make changes based on the feedback to reduce congestion (e.g., slow down the traffic source).

Feedback Methods in Closed-Loop Control:

- **Explicit Feedback:**
  - Routers directly inform the source about congestion using special packets.
- **Implicit Feedback:**
  - The source **infers congestion** by observing delays or missing acknowledgments.

How to Handle Congestion:

- **Increase Resources:**
  - Add more bandwidth, use backup routes, or bring spare routers online to handle the extra load.
- **Decrease Load:**
  - Reduce the traffic by limiting service, scheduling demands, or dropping some low-priority packets.

Summary:

- **Open-Loop**: Prevents congestion through good design but doesn't adjust once running.
- **Closed-Loop**: Detects and fixes congestion as it happens using feedback.

- The goal is to either **increase the network capacity** or **reduce the traffic load** to manage congestion effectively.

Congestion Prevention Policies

1. Data Link Layer (Close Range Communication):

This layer controls how data is sent between two directly connected devices (like two buildings).

- **Retransmission Policy:**
  Explanation: Decides when to resend a lost packet. If a packet is lost, sending it again too quickly can create too much traffic, causing congestion.
  Example: Imagine a car gets stuck on the road. Should we send another one immediately, or wait for it to clear? Waiting a bit before sending the next car avoids creating a traffic jam.
- **Out-of-Order Caching Policy:**
  Explanation: When packets arrive in the wrong order, this policy allows holding onto them in a buffer rather than asking for them to be resent.
  Example: If cars arrive in a wrong order at a destination, we can hold them in the parking lot instead of sending them back for reordering, avoiding extra traffic.
- **Acknowledgement Policy:**
  Explanation: Determines when and how packets should acknowledge receipt. Sending too many acknowledgments immediately can cause traffic, so delayed acknowledgments can help reduce congestion.
  Example: If every car (packet) immediately sends a message saying, “I arrived safely,” it creates more traffic. Waiting for some cars to arrive together reduces this.
- **Flow Control Policy:**
  Explanation: Controls how fast data is sent to prevent overwhelming the receiver. By limiting the rate of packet transmission, it prevents congestion.
  Example: Setting a speed limit on a road ensures there aren't too many cars arriving at once, preventing traffic jams.

2. Network Layer (Larger Range Communication):

This layer deals with traffic management across the entire network, like highways across cities.

- **Virtual Circuits vs. Datagrams:**
  Explanation: Virtual circuits have a fixed route for traffic, which can prevent congestion by knowing the best path. Datagrams route traffic dynamically, which may cause more congestion if paths aren’t optimized.
  Example: A fixed road route (virtual circuit) is easier to manage, while deciding on the route while driving (datagram) might lead to unexpected jams.
- **Packet Queueing and Service Policy:**
  Explanation: Determines how packets are queued and processed. The way packets are handled in queues affects how quickly they move through the network.
  Example: If cars are waiting at a toll booth, should we serve the first one that arrives or prioritize emergency vehicles? The order impacts how quickly traffic flows.

- **Packet Discard Policy:**
  Explanation: If a network is congested, some packets need to be dropped. The policy decides which ones to drop to minimize the impact on the network.
  Example: If a parking lot is full, which cars (packets) should we send back? It's better to remove less important cars to make space for more critical ones.

- **Routing Algorithm:**
  Explanation: Efficient algorithms help distribute traffic evenly by directing packets to less congested paths.
  Example: A good GPS system helps drivers avoid traffic jams by rerouting them to less crowded roads.

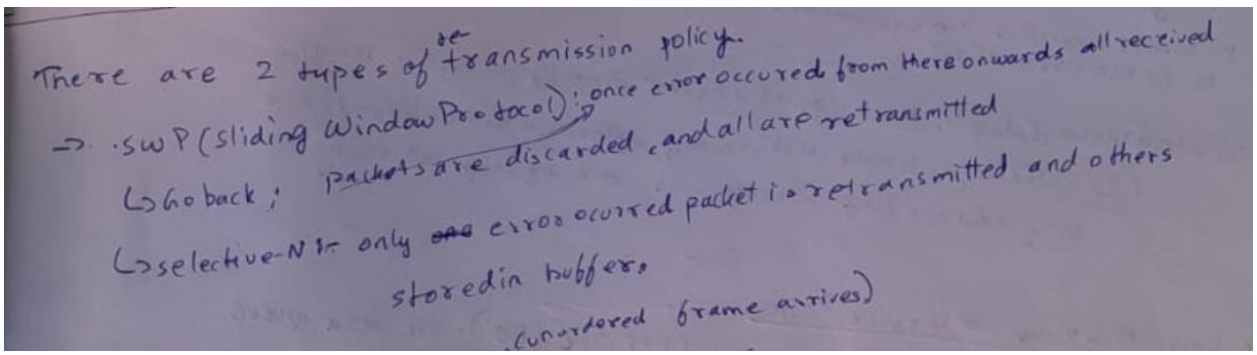
- **Packet Lifetime Management:**
  Explanation: Defines how long a packet should stay in the network. If a packet is too old, it's discarded to prevent clogging.
  Example: If a car is stuck on a road too long, it should be removed to avoid further congestion.

3. Transport Layer (Long-Distance Communication):

This layer manages data between end systems, ensuring efficient transmission even with unpredictable delays.

- **Timeout Determination:**
  Explanation: This policy sets how long to wait before retransmitting a lost packet. If we wait too short a time, we may send unnecessary packets, increasing traffic. If we wait too long, the response will be delayed.
  Example: Imagine a delivery truck (packet) that's running late. How long should we wait before sending another truck? If we wait too little, there'll be unnecessary trucks; if we wait too long, the delivery is delayed.

Layer	Policies
Transport	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy • Timeout determination
Network	• Virtual circuits versus datagram inside the subnet • Packet queueing and service policy • Packet discard policy • Routing algorithm • Packet lifetime management
Data link	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy



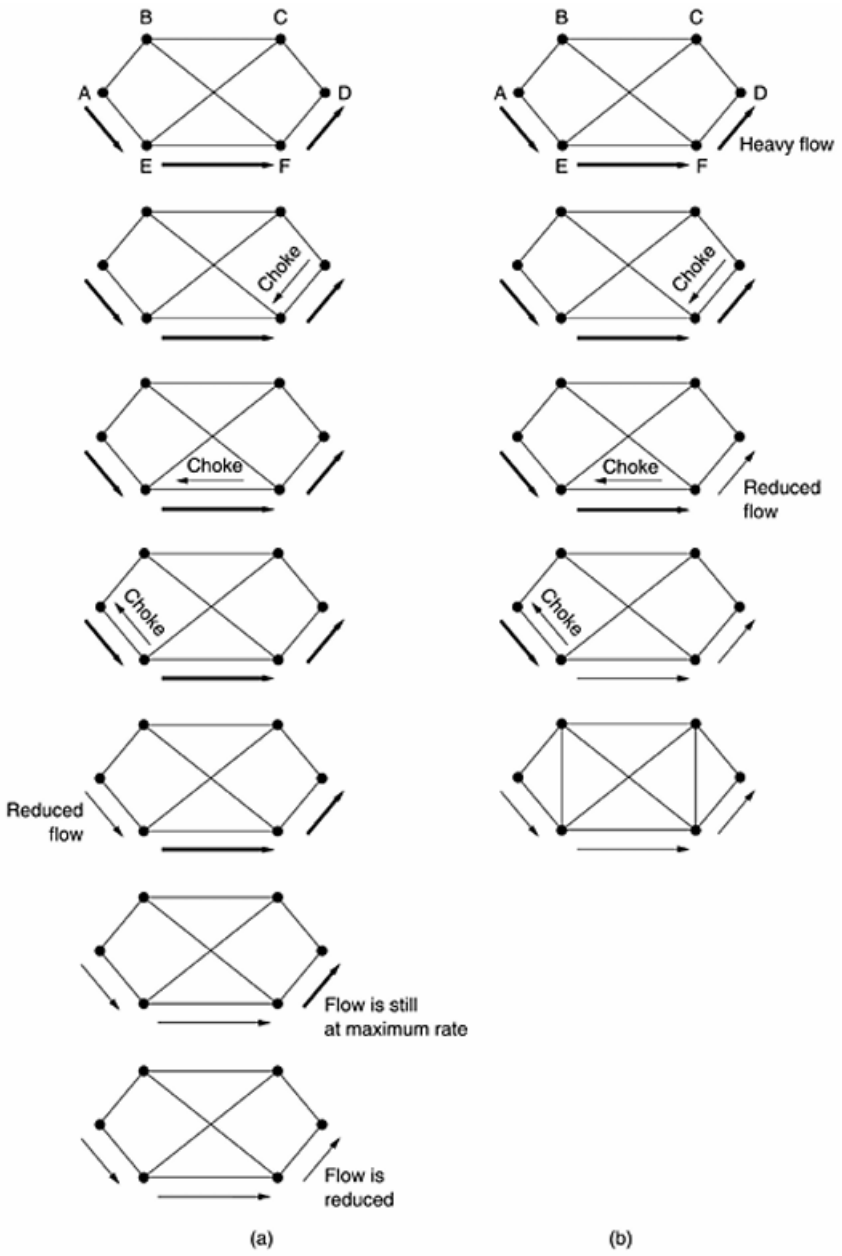
Virtual-Circuit Subnets Congestion Control:

1. **Admission Control:**
   - **How it works:** When congestion is detected, no new connections (virtual circuits) are allowed until the issue is resolved.
   - **Example:** Like a full concert hall that stops selling tickets until people leave.
2. **Routing Around Congestion:**
   - **How it works:** New connections are routed to avoid congested routers.
   - **Example:** If a road is blocked, traffic is rerouted through a different route.
3. **Resource Reservation:**
   - **How it works:** Resources (like bandwidth) are reserved for each new connection to ensure no congestion occurs.
   - **Example:** Booking a seat ensures you have a place, but it might waste space if not fully used.

Datagram Subnets Congestion Control:

1. **Warning Bit:**
   - **How it works:** Routers mark packets with a "warning bit" when a line is congested. The source adjusts its transmission rate based on these warnings.
   - **Example:** If traffic is heavy on the road, drivers get a warning to slow down.
2. **Choke Packets:**
   - **How it works:** A router sends a choke packet to the source to directly tell it to reduce traffic to a specific destination.
   - **Example:** Like a traffic light telling cars to slow down at a specific intersection.
3. **Hop-by-Hop Choke Packets:**
   - **How it works:** Choke packets affect each router along the path, slowing traffic at each hop to prevent congestion faster.
   - **Example:** Each step of the way, a driver is told to slow down, preventing a traffic jam from spreading.

(a) A choke packet that affects only the source. (b) A choke packet that affects each hop it passes through.



Load Shedding:

When a router is overwhelmed with too many packets and can't process them all, it starts **dropping packets**. This is called **load shedding**. Think of it like power companies turning off power to certain areas to prevent a blackout.

- **Smart Dropping:** Routers try to drop less important packets first. For example, for a file transfer, it's better to drop older packets, while for live video, dropping newer packets is better.
- **Priority Marking:** Applications can mark their packets based on importance, allowing routers to prioritize which packets to drop.
- **Excess Traffic:** Hosts can send extra traffic marked as low priority, which can be dropped when congestion occurs, but this uses resources efficiently when the network is not congested.

Random Early Detection (RED):

- Instead of waiting for congestion to be severe, **RED** tries to prevent it by dropping packets early.
- Routers **monitor the queue size** and when it gets too large, they randomly drop packets.
- This encourages sources to **slow down** before congestion becomes a bigger problem.
- **Feedback Without Overloading:** Instead of sending extra messages about congestion, RED simply drops packets and the source slows down when it notices lost packets.

Jitter Control:

- Jitter refers to the variation in packet arrival times, which can affect real-time applications like audio and video streaming.
- **Low jitter** means packets arrive at nearly the same time, while **high jitter** can cause uneven playback (e.g., skipping in audio).
- **Control Jitter:** Routers manage jitter by adjusting packets' delivery times. If a packet is early, it's held a bit longer; if it's late, it's sent out quickly.
- **Buffering:** In some cases, receivers can store packets in a buffer to smooth out jitter, but for real-time communication, like video calls, buffering is not ideal.

In short, these congestion control methods (load shedding, RED, and jitter control) help keep networks running smoothly by managing traffic efficiently and preventing delays or disruptions.

Quality of Service (QoS)

Definition:

- Quality of Service (QoS) in networking ensures that data is transmitted with reliability, acceptable delay, jitter (variations in delay), and sufficient bandwidth based on the application's requirements.
- QoS refers to techniques and mechanisms used in networking to guarantee specific performance levels (reliability, delay, jitter, and bandwidth) for different applications.

QoS Parameters

- **Reliability**: Ensures data accuracy by retransmitting damaged packets.
- **Delay**: Measures how long it takes for data to travel. Real-time apps need minimal delay.
- **Jitter**: Variations in packet arrival times. Streaming apps require consistent timing.
- **Bandwidth**: Amount of data transmitted per second. Video and gaming need high bandwidth.

Application Requirements

- **Stringent Reliability**: File transfer, emails.
- **Stringent Delay & Jitter**: Videoconferencing, telephony.
- **Low Bandwidth Needs**: Emails, browsing.
- **High Bandwidth Needs**: Streaming, video.

QoS Categories in ATM Networks

1. **Constant Bit Rate (CBR)**: Fixed bandwidth (e.g., telephony).
2. **Real-Time Variable Bit Rate (rt-VBR)**: Dynamic bandwidth for real-time video.
3. **Non-Real-Time Variable Bit Rate (nrt-VBR)**: Streaming services like movies.
4. **Available Bit Rate (ABR)**: For delay-insensitive data like file transfers.

Techniques for QoS

1. **Overprovisioning**:
    - Extra capacity for routers and bandwidth.
    - Simple but expensive.
2. **Buffering**:
    - Stores data temporarily to manage jitter but increases delay.
3. **Traffic Shaping**:
    - **Leaky Bucket Algorithm**: Ensures steady data output by discarding excess.
    - **Token Bucket Algorithm**: Allows bursts by storing tokens to match packet size.
4. **Resource Reservation**:
    - Allocates bandwidth, buffer space, and CPU for specific flows.
5. **Admission Control**:
    - Accepts new flows only if the network has spare capacity to handle them.

Flow-Based QoS

- Flow refers to packets between a source and destination.
- **Resource Reservation Protocol (RSVP)**: Sets up paths for flows, ensuring dedicated bandwidth.

Class-Based QoS (Differentiated Services)

- Simpler than RSVP, uses predefined service classes:
    - **Expedited Forwarding (EF)**: Low delay for premium packets.
    - **Assured Forwarding (AF)**: Four priority classes with discard levels (low, medium, high).

Algorithms for Packet Scheduling

- **Fair Queueing**: Each flow gets an equal share of bandwidth.

- **Weighted Fair Queueing**: Allocates bandwidth based on importance (e.g., video gets more than file transfers).

Other QoS Mechanisms

1. **Integrated Services (IntServ)**:
    - Provides guarantees for each flow but requires setup and tracking (complex to scale).
2. **Multi-Protocol Label Switching (MPLS)**:
    - Labels packets for fast routing and resource management.

Token Bucket vs. Leaky Bucket

Both algorithms are traffic shaping techniques used in networking to manage how data flows through a network. Here's a comparison and explanation of each:

---

1. Leaky Bucket Algorithm

Concept:

- Imagine a bucket with a small hole at the bottom.
- Water (data packets) leaks out at a constant rate, no matter how much water is poured in.
- If the bucket fills up (too much data arrives), any extra water (packets) overflows and is discarded.

How It Works:

- A fixed output rate is maintained, smoothing bursts of incoming traffic.
- Excess data is dropped if the input rate exceeds the bucket's capacity.

Key Features:

- **Constant Rate**: Packets leave the bucket at a fixed rate.
- **No Burstiness**: Even if bursts of data arrive, the output remains steady.
- **Packet Loss**: If the bucket is full, excess packets are discarded.

Example:

- A video streaming server sends packets at a steady rate of 5 Mbps. If data arrives faster than that, it’s dropped to maintain the steady output.

Use Case:

- Suitable for applications that require a steady and predictable data flow.

---

2. Token Bucket Algorithm

Concept:

- Imagine a bucket that fills with tokens at a fixed rate.
- Each token represents permission to send one packet (or a specific number of bytes).
- Packets can only be sent if there are enough tokens in the bucket.

How It Works:

- Tokens are added to the bucket at a constant rate.
- If there's a burst of data, packets can use saved tokens, allowing for bursts up to a maximum capacity (bucket size).
- If there are no tokens left, incoming packets must wait until new tokens are generated.

Key Features:

- **Allows Bursts**: Unlike the leaky bucket, it permits bursts of traffic as long as tokens are available.
- **No Packet Loss**: Packets are not discarded but may be delayed if tokens are unavailable.
- **Flexible Rate**: Adapts to varying traffic patterns.

Example:

- A server transmits packets at an average of 2 Mbps, but bursts up to 10 Mbps are allowed as long as saved tokens are available.

Use Case:

- Suitable for applications where occasional bursts of data are acceptable, like file transfers or web browsing.

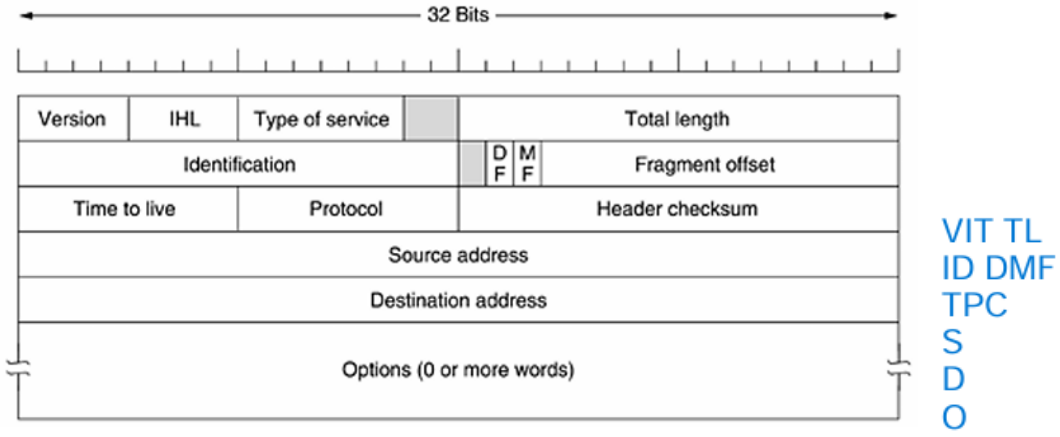
Feature	Leaky Bucket	Token Bucket
Output Rate	Fixed	Variable (based on token availability)
Burst Handling	No burst allowed	Allows bursts within limits
Packet Dropping	Drops excess packets	No drops (packets wait for tokens)
Use Case	Real-time apps needing steady flow	Apps needing flexibility in bursts

The IP Protocol IPV4

The **IP Protocol** governs how data travels across networks using **datagrams**, which have two parts:

1. **Header**: Contains control information (minimum size: **20 bytes**, maximum: **60 bytes**).
2. **Data/Text**: The actual content being sent (total datagram size: **up to 65,535 bytes**).

Key Points About the IPV4 Header:



- **Version (4 bits)**: Indicates the protocol version (e.g., IPv4 or IPv6). Allows gradual updates over time.
- **Header Length (IHL - 4 bits)**: Specifies the length of the header in 32-bit words. Minimum = **5 (20 bytes)**, maximum = **15 (60 bytes)**.
- **Type of Service (1 byte)**: Indicates priority, such as speed vs. reliability. Often ignored by routers.
- **Total Length (2 bytes)**: Includes header and data, with a maximum size of **65,535 bytes**.
- **Identification (2 bytes), DF, MF (1 bit each), and Fragment Offset (13 bits)**:
    - Manage fragmentation when large packets are split into smaller pieces.
    - **DF (Don't Fragment)**: Ensures packets stay whole.
    - **MF (More Fragments)**: Indicates more parts are coming.
    - Fragment Offset tells the position of the fragment in the original data.
- **Time to Live (TTL - 1 byte)**: Limits the number of hops (routers) a packet can travel (maximum = **255 hops**). Prevents infinite loops.
- **Protocol (1 byte)**: Indicates the transport layer protocol (e.g., TCP = 6, UDP = 17).
- **Header Checksum (2 bytes)**: Verifies header integrity and detects errors. Recomputed at every router.
- **Source and Destination Addresses (4 bytes each)**: Specify the sender's and receiver's IP addresses.
- **Options (Variable, up to 40 bytes)**: Adds optional features:
    - **Strict Source Routing**: Forces a specific path.
    - **Record Route**: Logs the route taken by the packet.
    - **Timestamp**: Adds time data at each hop for debugging.

The **minimum header size** is **20 bytes**, while the maximum with options is **60 bytes**. Total datagram size, including data, is capped at **65,535 bytes**. This design ensures flexibility and efficiency for network communication.

IPV6

IPv6 was developed to address the limitations of IPv4, such as running out of IP addresses. It aims to improve scalability, security, and flexibility for future Internet growth, especially with the increasing number of connected devices like phones, TVs, and computers. Some key goals of IPv6 include:

1. **Support billions of devices** – IPv6 provides a much larger address space than IPv4.
2. **Simplified protocol** – IPv6 has fewer fields, making it easier for routers to process packets faster.

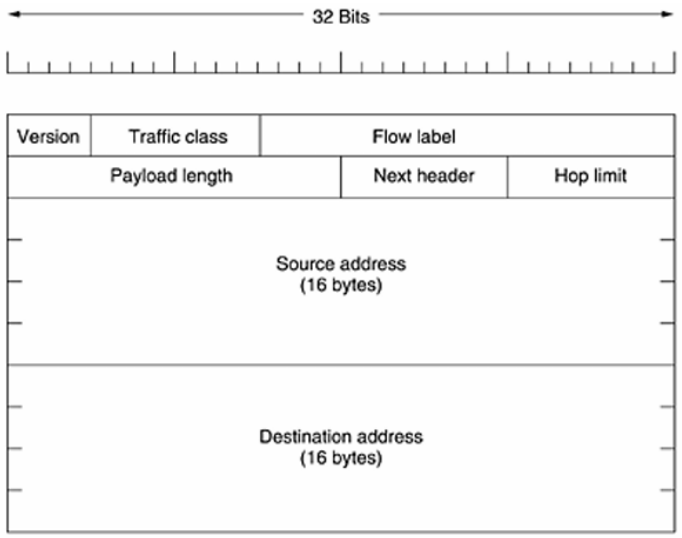
3. **Improved security** – Authentication and encryption are built into IPv6.
4. **Better support for multimedia** – It supports real-time data and multimedia delivery.
5. **Addressing mobile devices** – IPv6 allows devices to keep their IP addresses as they move between networks.

Key Features of IPv6:

- **Longer addresses**: IPv6 uses 128-bit addresses (16 bytes), compared to IPv4's 32-bit addresses. This allows for an enormous number of IP addresses, ensuring we won't run out.
- **Simplified header**: The IPv6 header has only 7 fields (compared to 13 in IPv4), which makes packet processing faster.
- **Better security**: IPv6 includes built-in security features, like authentication and encryption, to protect data.
- **Quality of service**: IPv6 pays more attention to the type of service, such as providing priority for real-time data like video and voice.
- **No need for NAT**: With a larger address space, NAT (Network Address Translation) is no longer necessary, simplifying network management.

Address Format:

Figure 5-68. The IPv6 fixed header (required).



- **Version:** Always set to 6 for IPv6.
- **Traffic class:** Defines the priority of the packet (TOS-type of service).
- **Flow label**: Identifies data streams requiring special handling – QOS(quality of service).
- **Payload length :** Specifies the length of the data (payload) following the header.
- **Next header** : : Identifies the type of the next header or the transport protocol.
- **Hop Limit**: Limits the number of hops (routers) a packet can pass through.
- **Source Adress**: 128bits
- **Destination Address:** 128bits

IPv6 addresses are written in 8 groups of 4 hexadecimal digits, separated by colons (e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334). Leading zeros can be dropped, and consecutive zeros can be replaced with "::".

Main Changes from IPv4:

- **No fragmentation by routers**: In IPv6, only the source host can fragment packets, not routers, which reduces the load on routers.
- **Extension headers**: IPv6 allows additional information through optional extension headers, such as routing or security options.
- **Flow label**: This helps in managing special types of traffic that need certain performance guarantees.

In short, IPv6 addresses the limitations of IPv4 by offering a larger address space, simpler header structure, better security, and improved support for modern Internet requirements.

Feature	IPv4	IPv6
Address Length	32-bit (4 numbers, e.g., 192.0.2.1)	128-bit (8 groups, e.g., 2001:0db8::1)
Address Format	Decimal, separated by dots	Hexadecimal, separated by colons
Number of Addresses	About 4.3 billion	Virtually unlimited (3.4×10^{38} addresses)
Header Size	20-60 bytes	Fixed 40 bytes
Header Complexity	More complex, with optional fields	Simplified, with extension headers
Security	Limited, optional (IPSec can be added)	Built-in security features (like IPSec)
Fragmentation	Routers and hosts handle fragmentation	Only the source handles fragmentation
Quality of Service	Limited support	Better support for real-time data (via Traffic Class and Flow Label fields)
Configuration	Manual or DHCP	Auto-configuration (stateless or DHCPv6)
Broadcast Support	Yes	No (uses multicast instead)
Transition Support	Does not support IPv6 directly	Compatible with IPv4 using dual-stack or tunneling methods

SAQ

- **IP Addresses**:
  - Unique 32-bit addresses identifying devices on a network.
  - Divided into **Network Number** and **Host Number**.
  - Written as dotted decimals (e.g., 192.41.6.20).
- **Special Addresses**:
  - 0.0.0.0: Refers to the current network.

    - 255.255.255.255: Broadcasts to all devices.
    - 127.x.x.x: Used for testing within the device.
- **Subnetting**:
    - Splits a large network into smaller **subnets** for efficient management.
    - Uses a **subnet mask** to separate network and host parts.
    - Reduces router table size and simplifies routing.

CIDR - Classless Inter Domain Routing

- **Problem of Address Exhaustion**: Fixed IP address classes (A, B, C) led to inefficient use of addresses, especially with class B networks.
- **CIDR Solution**: Allocates variable-sized blocks based on needs, avoiding rigid class limits.
- **Efficient Routing**: Introduces subnet masks to match and forward packets, using the longest prefix match for accuracy.
- **Route Aggregation**: Combines multiple routing entries into one to save space in router tables and reduce complexity.
- **Extended IPv4 Usage**: CIDR optimizes address distribution and routing, delaying the exhaustion of IPv4 addresses.

NAT—Network Address Translation

- **NAT Purpose**: Shares one public IP among multiple private devices, saving IP addresses.
- **How It Works**: Translates private IPs to public IPs using ports for mapping.
- **Issues**: Breaks IP uniqueness, connection independence, and some applications.
- **Criticism**: Slows the shift to IPv6, the proper solution for IP scarcity.

Internet Control Protocols

ICMP: Internet Control Message Protocol monitors network issues and sends messages like *Destination Unreachable*, *Time Exceeded*, and *Echo Request/Reply* for diagnostics and error reporting.

ARP: Address Resolution Protocol maps IP addresses to hardware (MAC) addresses, enabling devices on a LAN to communicate. It uses broadcasting to find the hardware address of the destination device.

RARP: Maps Ethernet to IP but requires a server per network.

BOOTP: Adds UDP support and extra info but needs manual setup.

DHCP: Automates IP assignment with leasing and relay agents.

OSPF - The Interior Gateway Routing Protocol

- Computes shortest paths using a network graph.
- Supports hierarchical routing with areas and dynamic updates.
- Maintains routes via link state exchanges.

BGP (Border Gateway Protocol) is used to route data between different networks (ASes) on the Internet. It allows organizations to set routing policies based on their needs and uses TCP for reliable communication between routers. BGP shares full paths to destinations, helping choose the best routes.