Network Layer and Protocols

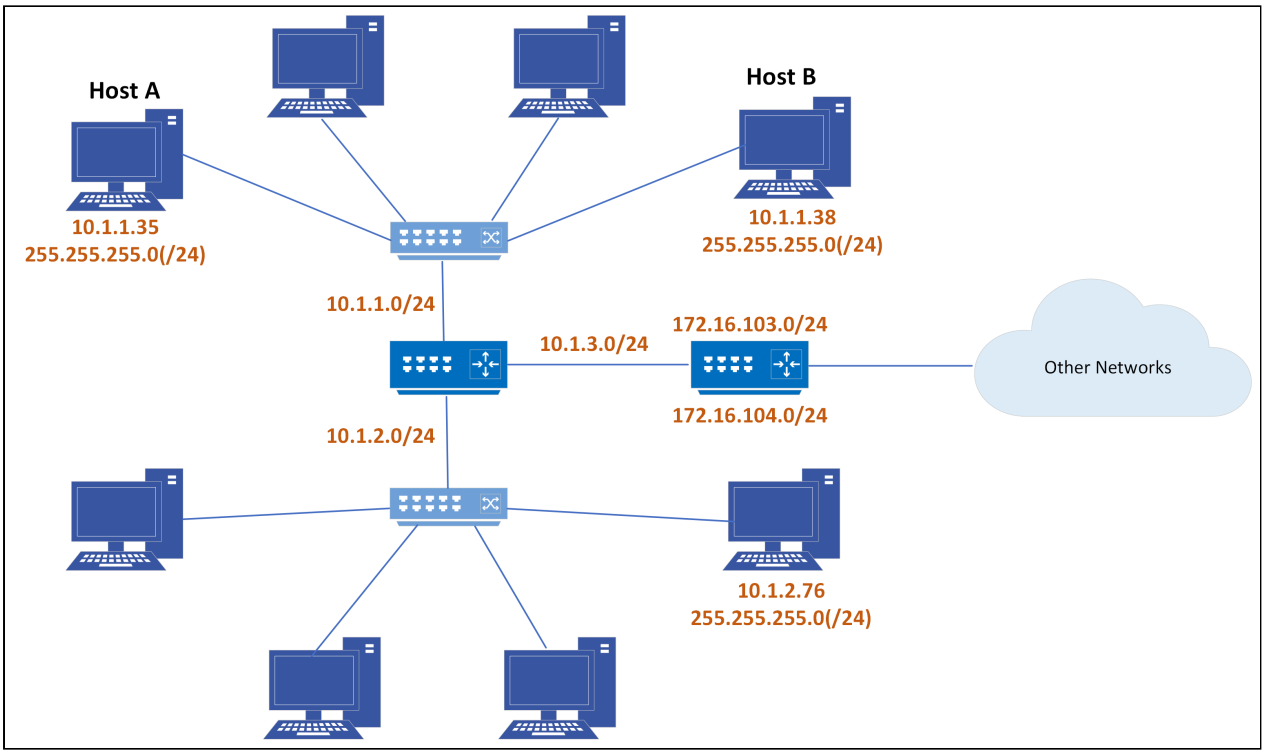
1. Network Layer Design Issues and Services

The **Network Layer** is a core part of computer networks, primarily focused on transporting packets from the source machine to the destination machine. This layer is concerned with issues like addressing, routing, and handling congestion.

1.1. Store-and-Forward Packet Switching

This is the fundamental operation used by routers in the network layer. When a packet arrives at an intermediate router, the router stores the entire packet temporarily. Once the router determines the best next-hop output line using its routing tables, it forwards the packet.

Figure 1.1: Store-and-Forward Packet Switching

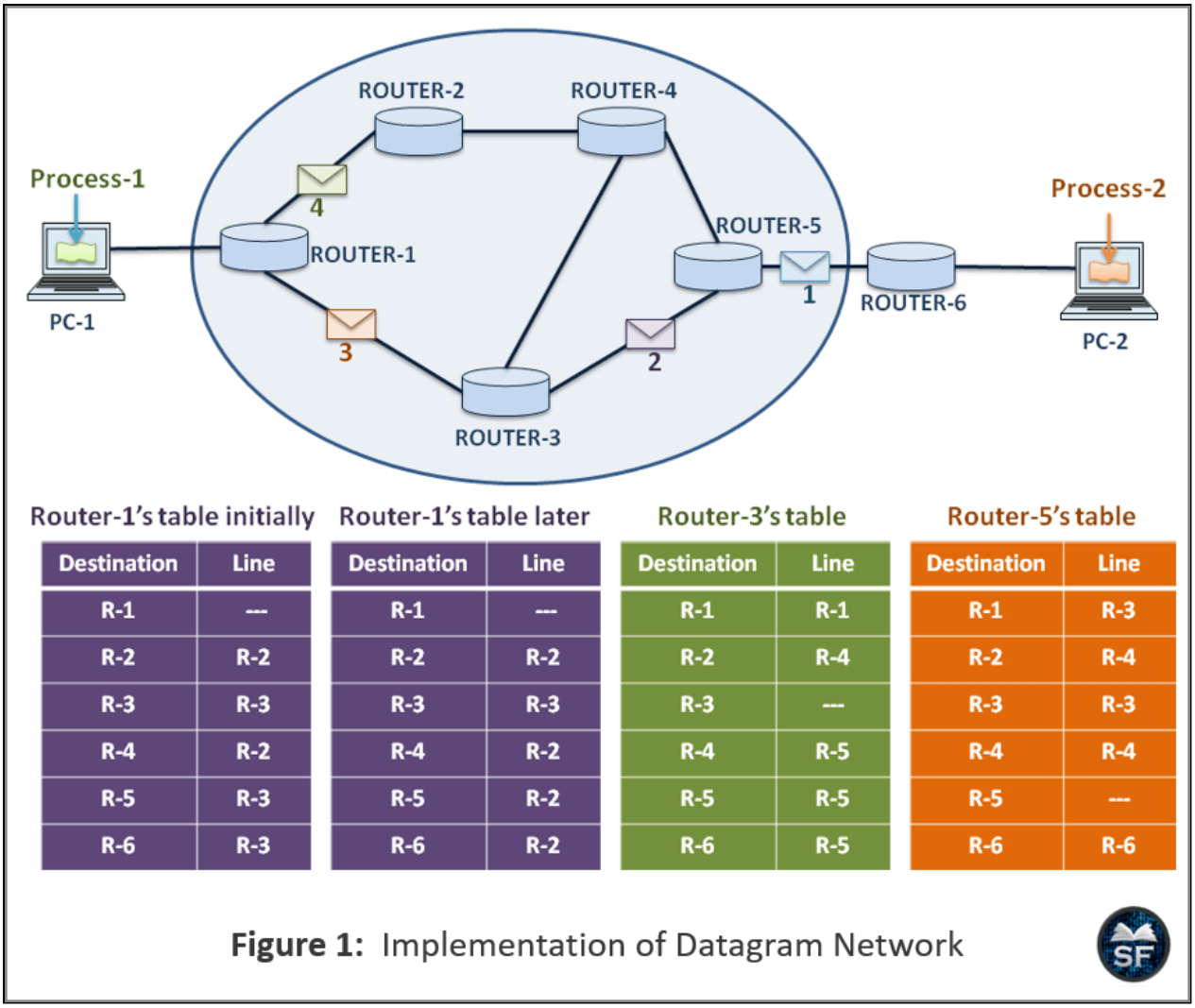


This diagram illustrates how a packet travels hop-by-hop between routers (A to B to D to F) until it reaches the destination host (H2).

1.2. Implementation of Connection-Oriented Service (Virtual Circuits)

In a **connection-oriented service**, the routing decision is made only once during the circuit setup time. A path, called a **Virtual Circuit (VC)**, is established from the source to the destination before data transfer begins. All packets belonging to this conversation follow the exact same path. This method often uses fixed tables in routers to map incoming virtual circuit identifiers (VCIs) and ports to outgoing VCIs and ports.

Figure 1.2: Connection-Oriented Service



Router-1's table initially

Destination	Line
R-1	---
R-2	R-2
R-3	R-3
R-4	R-2
R-5	R-3
R-6	R-3

Router-1's table later

Destination	Line
R-1	---
R-2	R-2
R-3	R-3
R-4	R-2
R-5	R-2
R-6	R-2

Router-3's table

Destination	Line
R-1	R-1
R-2	R-4
R-3	---
R-4	R-5
R-5	R-5
R-6	R-5

Router-5's table

Destination	Line
R-1	R-3
R-2	R-4
R-3	R-3
R-4	R-4
R-5	---
R-6	R-6

Figure 1: Implementation of Datagram Network

1.3. Implementation of Connectionless Service (Datagrams)

In a **connectionless service**, the routing decision is made for **each datagram** individually. Each packet (datagram) is treated independently and may travel a different path to reach the destination. The router uses a routing table that maps the destination address to the next output line, and this decision is made anew for every incoming packet. This is similar to how the postal system treats individual letters.

Key Design Issues and Services Provided to the Transport Layer:

- **Store-and-Forward:** The basic method where routers receive and temporarily hold a packet before forwarding it.
- **Connectionless Service (Datagrams):** Each packet is routed independently, and no setup phase is required.
- **Connection-Oriented Service (Virtual Circuits):** A fixed path is established before data transfer, and all packets follow it.
- **Addressing:** The network layer must define a unique way to identify hosts (not explicitly detailed, but implied by routing tables).

2. Routing Algorithms

The **routing algorithm** is the crucial software component that decides the path a packet will take, meaning it chooses the next output line or node. The network layer's main responsibility is to route packets from the source to the final destination.

2.1. Requirements for Routing Algorithms

Effective routing algorithms must meet several common requirements to ensure network efficiency and reliability.

Common Requirements:

- **Correctness:** The algorithm must prevent issues like deadlocks and unreachable states.
- **Simplicity:** Simple algorithms allow for fast handling of packets and result in fewer failures.
- **Robustness:** The system must be able to handle failures, changes in network topology, and variations in traffic load.
- **Stability:** The algorithms should remain stable and functional under all possible network circumstances.
- **Optimality:** The best path should be chosen to maximize throughput and minimize the mean packet delay.

Total Path Cost (C) Calculation: $C = \sum_{i=1}^{N} c_i$

This represents a simple model where the total cost (C) of a path is the sum ($\sum$) of the individual costs (c) of each link (i) in the path (N) (simplified concept based on Shortest Path algorithms).

2.2. Types of Routing Algorithms

Routing algorithms are broadly classified into two main types: static and dynamic (or adaptive).

Comparison of Dynamic and Static Routing Algorithms

Feature	Dynamic Routing (Adaptive)	Static Routing (Non-Adaptive)
Decision Basis	Changes routing decisions based on network **topology and traffic**.	Does **not** base routing decisions on network topology or traffic.
Information Source	Uses routing information provided by adjacent or all routers.	Routing information is downloaded to routers only when the network boots up.
Optimization Parameters	Optimizes based on hop count, distance, and estimated transit time.	Uses predetermined paths regardless of current network conditions.

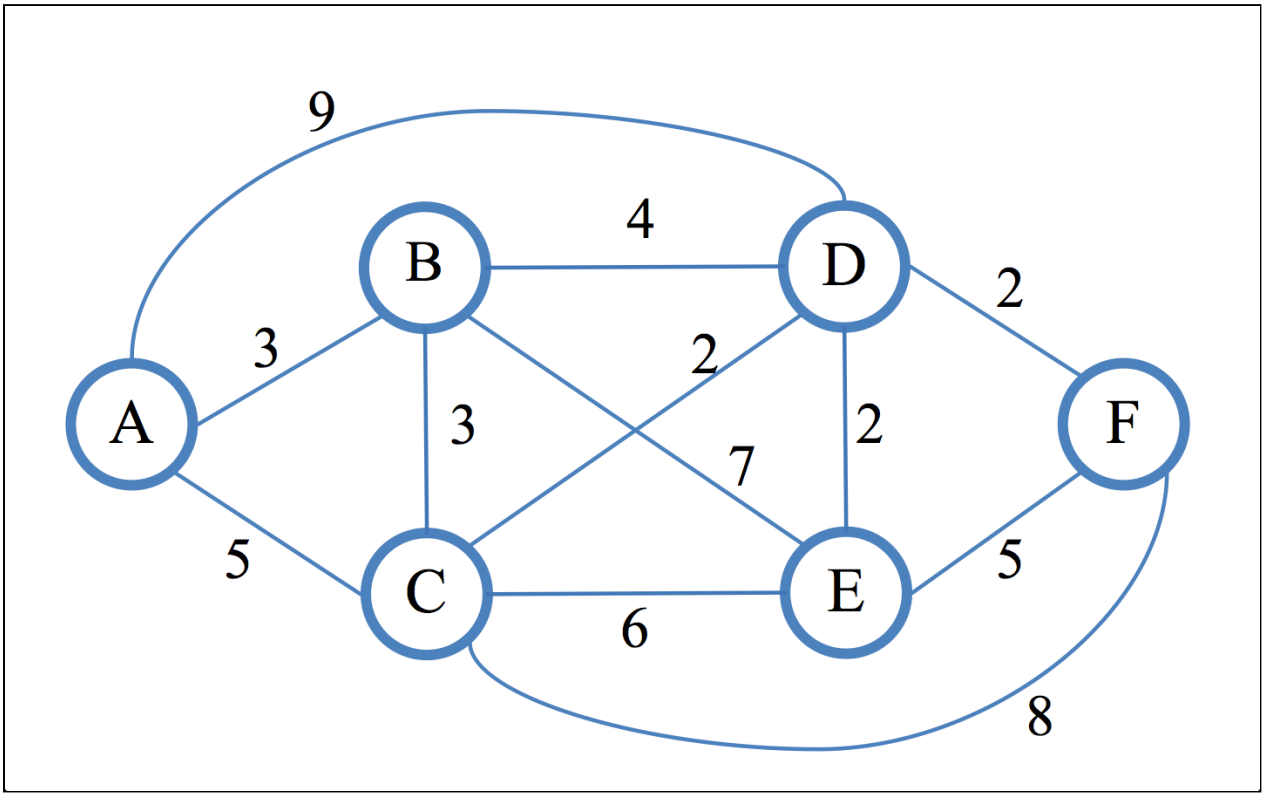
2.3. Specific Routing Algorithms

Various algorithms are employed in networks, falling under the static or dynamic categories.

Shortest Path Routing Algorithm (e.g., Dijkstra's Algorithm): This algorithm is a static routing method designed to find the path with the minimum cost between two nodes.

Link State Routing Algorithm: A dynamic algorithm where each router must follow a series of steps: discover its neighbors, measure the delay/cost to neighbors, construct a packet with this information, send the packet to all other routers, and finally compute the shortest path to every other router based on the collected information.

Figure 2.1: Example of Shortest Path Calculation



This figure shows a network graph used to find the shortest path between nodes, such as from A to F, based on link costs.

3. Congestion Control and Traffic Shaping

Congestion occurs when a part of the subnet, such as one or more routers in an area, becomes overloaded. Routers are receiving packets faster than they can forward them.

3.1. Congestion and Its Causes

When congestion occurs, the subnet must either prevent new packets from entering the congested region or the congested routers must discard queued packets to make room.

Primary Factors that Cause Congestion:

- **Excessive Packet Arrival Rate:** The rate at which packets arrive exceeds the capacity of the outgoing link.
- **Insufficient Memory:** There is not enough buffer memory to store the arriving packets.
- **Burst Traffic:** Sudden, high-volume traffic bursts overwhelm the network resources.
- **Slow Processor:** A slow router processor cannot handle the processing load quickly enough.

3.2. Congestion Control Mechanisms

Congestion control involves techniques and mechanisms to either prevent congestion before it happens or remove it after it has occurred.

Mechanisms are divided into two categories:

1. **Open-Loop Congestion Control (Prevention):** Policies are applied to prevent congestion before it starts, handled by either the source or destination. Examples include Retransmission Policy, Window Policy (e.g., using Selective Repeat instead of Go-back-N), Discarding Policy, Acknowledgment Policy, and Admission Policy.
2. **Closed-Loop Congestion Control (Removal):** Techniques that remove congestion after it has happened, such as **Backpressure** (a node-to-node technique that propagates congestion signals opposite to the data flow direction) and **Choke Packets** (a packet sent by a congested router directly to the source to reduce traffic).

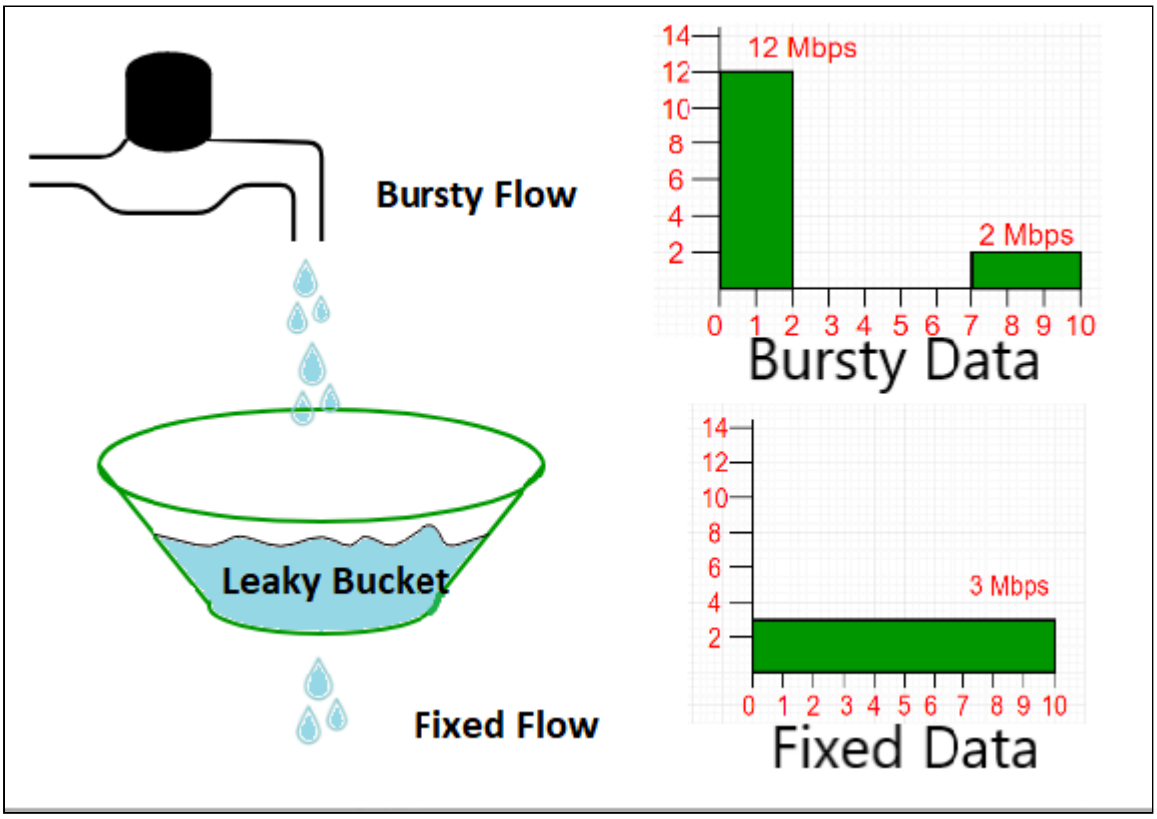
There are also signaling methods:

- **Implicit Signaling:** The source guesses there is congestion (e.g., due to a lack of acknowledgments) without explicit communication from the congested node.
- **Explicit Signaling:** A node informs the source or destination about congestion directly. Unlike a choke packet, the signal is often included in the data-carrying packets.

3.3. Traffic Shaping Algorithms

Traffic shaping is a method of congestion control that "shapes" the traffic before it enters the network, controlling the **rate** at which packets are sent. Two main algorithms are used for this:

Figure 3.1: Leaky Bucket Mechanism



The leaky bucket converts burst traffic into a uniform, constant flow of packets.

The Leaky Bucket Algorithm: This algorithm enforces a **constant output rate** regardless of how bursty the input traffic is. It acts like a single-server queue with a constant service time. If the buffer (bucket) overflows, packets are discarded. The host injects one packet per clock tick, resulting in a uniform flow.

Token Generation Rate (R): $R = 1 / \Delta t$

In the Token Bucket algorithm, tokens are generated at a rate of one token every Δt seconds.

The Token Bucket Algorithm: In contrast to the Leaky Bucket, this algorithm allows the output rate to **vary** depending on the size of the burst. The bucket holds tokens, and a host must capture and destroy one token to transmit a packet. Idle hosts can save up tokens (up to the bucket's max capacity) to send larger bursts later, providing more flexibility than the Leaky Bucket. If there is no token in the bucket, the packet cannot be sent.

Study hard! This note covers all the key concepts.