CN- Network Simulator Project

Kavya Verma(031), Raynav Kumar(032)

Submitted to – Dr. Iqra Altaf Gillani

Physical Layer

This layer focuses on how devices are physically connected and how signals are transmitted across the network. It includes actual hardware components like cables, connectors, and devices.

- **End Devices** such as computers or mobile phones send and receive data.
- A **Hub** is used to connect multiple devices within a network. When one device sends data, the hub replicates it and forwards it to all connected devices. This creates a **collision domain**, where multiple signals can interfere with each other.

Implementation-
Two devices are initially connected directly, and data is exchanged between them. Later, a hub is added to connect multiple devices, and the code demonstrates how data sent from one device gets broadcasted to all others. A visual graph is generated to show how all devices are linked together, making the physical structure of the network easy to understand.

Data Link Layer

This layer ensures reliable communication between devices on the same network using **MAC addresses**.

- Each device has a **MAC address**, which acts as a unique identifier.

- A **Switch** is introduced that learns the MAC addresses connected to its ports and forwards data only to the intended destination if known. Otherwise, it broadcasts the data.
- A **Bridge** is used to connect different network segments and forwards frames based on MAC addresses.

Additional Features Implemented:

- **Parity Check:** Before sending a frame, it checks if the number of 1s in the data is even. If not, an error is assumed.
- **CSMA/CD:** Devices check if the medium is free before sending. If a collision occurs (simulated), the device waits and retries.
- **Sliding Window Protocol:** Ensures reliable delivery of frames. It sends multiple frames and waits for acknowledgments, resending if necessary.

Implementation- Devices are connected to a switch, and data transmission occurs through it. Before sending, parity checks and CSMA/CD are applied. If everything is successful, data is sent using the sliding window technique. The switch's learning behaviour and frame forwarding are displayed in the output. A visual graph shows device and switch connections.

Network Layer

This layer handles the movement of packets across different networks using **IP addresses** and **routers**. It determines the best path for each packet.

- Each **Router** has interfaces with unique IP and MAC addresses.
- **Static Routing** is implemented using manually defined routes and **Longest Prefix Matching** to choose the best one.

- **IP Packets** carry source/destination IPs and **TTL** to avoid infinite loops.
- **ARP** is used to find the MAC address when only the IP address is known.

Dynamic Routing with OSPF:

- Routers exchange **LSAs (Link State Advertisements)** to share their connectivity.
- Each router builds a **Link State Database** and uses **Dijkstra's Algorithm** to compute shortest paths.
- Routing tables are updated automatically based on network topology.

Implementation-
Routers are created with IP/MAC interfaces. Hosts are configured with IPs and gateways. Static routes are set, and packet forwarding is demonstrated, including TTL expiry behaviour.
Then, OSPF is implemented where routers dynamically build their routing tables. Hosts send packets using these dynamic paths, and all steps—routing decisions, ARP lookups, and TTL handling—are shown in the output.

Transport Layer

The transport layer ensures end-to-end, reliable data transmission between processes using **ports**, **segmentation**, and **acknowledgment** techniques.

- A **Port Manager** is used to assign port numbers.
- **TCP Segments** are created with fields like source port, destination port, sequence number, acknowledgment number, flags, and data.

- A **Go-Back-N Sliding Window** technique is implemented to improve efficiency in transmission. It sends a group of segments and waits for acknowledgment. If acknowledgment is lost or missing, it retransmits the window.

Implementation-

The sliding window handles sending messages from one application to another using TCP segments. The sender and receiver are assigned ports through the Port Manager. The message is broken into smaller segments and sent in windows of size 3. Segment information like ports, sequence numbers, and acknowledgments is printed for each transmission. The acknowledgment logic is deterministic here (always succeeds) to keep the simulation simple and clean.

Application Layer

This is the topmost layer where actual user-level applications like chat or file transfer are simulated.

1. **ChatApp**: Receives text messages and prints them as chat.

2. **FTPMock**: Simulates FTP file transfers and displays each data chunk received.

Implementation-
ChatApp and FTPMock act as endpoints. They receive TCP segments and print the contained data. This demonstrates how data sent at the application layer travels through transport mechanisms and reaches the correct application at the receiving end.

Output

NetworkX and **matplotlib** are used to draw network topology graphs showing how hubs, switches, routers, and devices are connected.

The graph shows:

- Data transmissions and frame deliveries
- MAC/IP address resolutions
- Collision detection and CSMA/CD
- Routing table generation
- TCP segment flow and acknowledgments
- Application-level message

These outputs provide a complete view of how data moves and how each protocol contributes to reliable and efficient communication in a computer network Simulator.