Network Simulator Report

Khushboo (2022BITE002)
Afsheen (2022BITE048)
Sibgat (2022BITE010)

June 14, 2025

1 Introduction

This report outlines the implementation of a full-stack Network Simulator that models the complete protocol stack—from Physical Layer to Application Layer. Each layer simulates real-world network functionalities including frame transmission, switching, routing, flow control, reliable delivery, and basic application services like Telnet and FTP.

2 Language Used

Python

3 Project Structure

```
network-simulator/

 physical_layer/
    physical_layer.py         # Dedicated link and star topology

 data_link_layer/
    access_control.py         # CSMA/CD
    bridge.py                 # Bridge simulation
    end_device.py             # End device definition
    error_control.py          # CRC error detection
    frame.py                  # Frame structure
    gbn.py                    # Go-Back-N (initial Layer 2 implementation)
    stop_n_wait.py            # Stop-and-Wait ARQ
    switch.py                 # Switch with MAC learning

 NetworkLayer/
    host.py                   # Hosts with IP/MAC, ARP table
    router.py                 # Router with static routing
    rt.py                     # Routing utility (longest prefix match)
    serialLink.py             # Optional serial link logic
    testcase1.py              # ARP + Basic packet delivery
    testcase2.py              # Static Routing with multiple routers

 TransportLayer/
    sliding_window.py         # GBN protocol (Transport layer)
    transport.py              # Send/receive logic for GBN

 ApplicationLayer/
    echo_app.py               # Echo message app
    ftp_app.py                # Simulated file transfer
    telnet_app.py             # Simulated Telnet communication

 tests/
    test_data_link.py         # Tests for Data Link Layer
    __init__.py

 test_transport_app.py         # Transport Layer test
 main.py                       # Main simulation menu
 README.md                     # Project documentation
```

4 Simulator Features

- **Physical Layer:** Simulates dedicated and star topology links.
- **Data Link Layer:** Frame structuring, CRC error detection, ARQ protocols, CSMA/CD, switch, and bridging.
- **Network Layer:** Implements IP addressing, ARP, and static routing using longest prefix match.
- **Transport Layer:** Implements Go-Back-N (GBN) protocol using a sliding window mechanism.
- **Application Layer:** Echo, Telnet, and FTP-style apps simulating client-server behavior.

5 Simulator Menu Interface

The user can interact with the simulator via a command-line menu in ‘main.py‘:

```
========== NETWORK SIMULATOR MENU ==========
1. Dedicated Link (End-to-End Connection)
2. Simulation through Hub | STAR TOPOLOGY
```

```
3. CRC Error Detection Simulation
4. Bridge Simulation
5. Stop and Wait Simulation
6. Switch with 5 Devices
7. Two Star Topologies with Hubs + Switch
8. Testing CSMA/CD
9. Network Test Case 1 (Basic Router)
10. Network Test Case 2 (Three Routers with RIP)
11. GBN Simulation test
12. Exit
============================================
```

6 Transport Layer

The Transport Layer functionality is implemented in:

- `sliding_window.py`: Implements Go-Back-N (GBN) logic using a sliding window.
- `transport.py`: Integrates sender and receiver logic and connects to application layer.

Key features:

- Reliable, in-order delivery of packets.
- Cumulative acknowledgments.
- Timeout-based retransmission.
- Custom tests provided in `test_transport_app.py`.

7 Application Layer

Application-level services simulate interactive communication on top of the transport layer.

Implemented Apps

- `echo_app.py`: Receives a message and echoes it back (used for connectivity testing).
- `ftp_app.py`: Transfers text-based files using the simulator.
- `telnet_app.py`: Simulates text-based command/response over a pseudo-terminal.

Each app triggers end-to-end data encapsulation:

> Application Layer → Transport Layer → Network Layer → Data Link Layer → Physical Layer

8 Network Layer

The Network Layer modules simulate IP-based routing and ARP resolution.

Key Files

- `host.py`: Assigns IP/MAC, performs ARP resolution, manages ARP table.
- `router.py`: Handles routing using static table, performs next-hop forwarding.
- `rt.py`: Provides

  $$longestprefixmatch()$$

  function for route selection.

Test Cases

- `testcase1.py`: Host Router ARP + packet forwarding.
- `testcase2.py`: Multi-router topology with RIP-style static routing.

9 Data Link Layer

This layer handles framing, MAC addressing, error detection, flow control, and switching.

Modules

- `frame.py`: Defines structure of a data frame.
- `switch.py`: MAC learning and packet forwarding.
- `bridge.py`: Divides broadcast domains.
- `stop_n_wait.py`: Implements basic ARQ.
- `gbn.py`: Layer-2 level Go-Back-N protocol.
- `access_control.py`: Implements CSMA/CD.

10 Physical Layer

Implemented in

$$physicallayer.py$$

The physical layer simulates raw bit transmission for two topologies:

- Dedicated Link: Simulates point-to-point wired transmission.
- Star Topology: Uses hub/switch to simulate broadcast/bus-based connectivity.

11 Logging and Debugging

Each layer provides console-based logs:

- ARP request/reply logs
- Routing decisions and next-hop info
- GBN window tracking and retransmissions
- Application send/receive events
- MAC learning and switch forwarding

12 Conclusion

This simulator provides an educationally rich, layered model of network behavior:

- Encapsulation from Application to Physical Layer.
- Modular codebase with reusable components.
- Debug-friendly console logs and test cases.
- Can be extended with Selective Repeat, congestion control, or DNS simulation.

13 References

- CSMA/CD – GeeksforGeeks, TutorialsPoint
- Go-Back-N – GeeksforGeeks
- ARP, IP Routing – Computer Networking: Principles, Protocols and Practice
- Socket Programming – Python Docs, StackOverflow
- Telnet/FTP – RFCs 854, 959 and Python implementations