Network Simulator Report

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

May 19, 2025

1 Introduction

This document presents a Network Simulator that implements key networking concepts at the Data Link Layer and Physical Layer. The simulator models frame transmission, collision detection using CSMA/CD, and flow control mechanisms such as Stop-and-Wait ARQ. The simulation also includes switching functionality.

2 Language used

Python

3 Project Structure

```
network-simulator/
 data_link_layer/         # Implements Data Link Layer components
    __init__.py
    access_control.py    # CSMA/CD implementation
    bridge.py            # Bridge functionality
    end_device.py        # End devices in the network
    error_control.py     # Error detection mechanisms (parity/CRC)
    frame.py             # Frame structure definition
    switch.py            # Switch with MAC learning functionality

 physical_layer/          # Implements Physical Layer simulation
    __init__.py
    physical_layer.py    # Physical layer logic

 tests/                   # Contains test scripts
    __init__.py
    test_data_link.py    # Tests for data link layer
```

```
general/                # Environment setup files
   bin/
   lib/
   .gitignore
   pyvenv.cfg

main.py                 # Entry point of the simulation
README.md               # Project documentation
```

4 Features

- Dedicated Link Simulation: Simulates direct communication between two devices.
- Star Topology Simulation: Models hub-based and switch-based star topologies.
- CRC Error Detection: Implements Cyclic Redundancy Check (CRC) for error detection.
- Bridge Simulation: Simulates bridges to divide networks into smaller segments.
- Stop-and-Wait ARQ: Implements an automatic repeat request protocol.
- CSMA/CD Testing: Tests Carrier Sense Multiple Access with Collision Detection (CSMA/CD) for Ethernet.

4.1 Setting Up a Virtual Environment

```
python -m venv venv
source venv/bin/activate  # On Windows use `venv\Scripts\activate`
```

5 Running Tests

To test the Data Link Layer implementation:

```
python tests/test_data_link.py
```

6 Running the Simulation

To run the complete network simulation:

```
python main.py
```

The following menu will be displayed:

```
========== 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. Exit
============================================
Enter your choice (1-9):
```

7 Extended Functionality (Network Layer)

The network simulator has been extended to include key Network Layer features. The following files implement the routing, IP assignment, ARP resolution, and packet forwarding logic:

Updated Project Structure

```
network-simulator/
 host.py            # Defines Host class with IP, MAC, ARP handling
 router.py          # Defines Router class with static routing logic
 switch.py          # Switch logic with MAC learning + ARP handling
 rt.py              # Utilities for IP matching (e.g., prefix match)
 serialLink.py      # (Optional) Serial link support if required
 testcase1.py       # Basic ARP + packet delivery demonstration
 testcase2.py       # Static Routing: A → Router → D
```

Host (host.py)

- Configures IP and MAC addresses for each host.
- Sends ARP requests and handles ARP replies.
- Maintains a per-host ARP table.
- Sends data using gateway MAC via switch.

Router (router.py)

- Supports multi-interface configuration.
- Maintains ARP and routing tables.
- Receives packets and uses static routing via:
  - Longest prefix matching
  - Next-hop forwarding
- Responds to ARP requests and replies.

Switch (switch.py)

- Learns MAC-to-port mappings upon packet/ARP receipt.
- Floods ARP requests to connected interfaces.
- Forwards packets to the correct destination MAC port.

Routing Utility (rt.py)

- Implements the `longest_prefix_match()` function.
- Parses subnet mask length and performs binary comparison.
- Used by the router to find the correct outgoing interface.

Test Cases

- `testcase1.py`: Validates ARP discovery from Host A → Router and Host D → Router.
- `testcase2.py`: Implements full static routing:
  - Host A → Switch X → Router → Switch Y → Host D
  - Forwarding via static routes on the router

Console Logging

All components include detailed ‘print()‘ logs:

- ARP send/receive logs with resolved MACs
- Switch forwarding and learning logs
- Router routing decisions and packet flow trace

These enhancements provide a minimal but functional simulation of IP routing with ARP and forwarding, forming a foundational layer for implementing RIP in the next phase.

8 References

- Collision Detection in CSMA/CD - GeeksforGeeks
- CSMA with Collision Detection (CSMA/CD) - TutorialsPoint
- Stop and Wait ARQ - GeeksforGeeks
- Network Packet Sniffer: Process an Ethernet frame (MAC src & dest address + protocol) using Python - StackOverflow
- Introduction to Ethernet - NetworkLessons
- The Data Link Layer and the Local Area Networks - Computer Networking: Principles, Protocols and Practice
- A Network Simulator Implementing Entire Protocol Stack - GitHub