

**Final Project Report**

**ES-221**

**Group Members**

2023534 **– Umair Afridi**

2023572 **– Neshmiya Farooq**

2023528**- Taha Farooq**

**Final Report: Network Packet Transmission and Routing Simulation**

**1. Project Overview**

This project simulates a simplified computer network environment where packets are transmitted between routers using various data structures and algorithms. The simulation includes components like routing through graphs, handling packet transmission using queues and stacks, retransmission due to simulated packet loss, and packet organization through binary trees. It emphasizes how fundamental data structures and algorithmic concepts apply to real-world networking scenarios, making it a comprehensive educational tool to understand dynamic routing, sorting algorithms, and error handling mechanisms in a network.

**2. Implemented Data Structures and Algorithms**

Several essential data structures and algorithms were integrated into the project:

* **Graph (Adjacency List Representation)**: Each node in the graph represents a router, and edges represent direct connections (with latency) between routers. This structure allows efficient storage and access to connections.
* **Dijkstra’s Algorithm**: Used to compute the shortest path from a source router to all other routers in the network. It ensures that packets are routed optimally based on latency.
* **Queue (FIFO)**: Used to simulate real-world packet transmission order, where the first packet sent is the first to be processed.
* **Stack (LIFO)**: Simulates temporary packet storage or last-in-first-out data behavior, such as in protocol stacks or certain caching operations.
* **Linked List**: Used to manage the retransmission list, storing packet IDs that were lost during transmission for retrying later.
* **Binary Search Tree (BST)**: Stores and organizes packets using their ID. Supports insertion, deletion, and traversal operations to simulate real-time packet logging and querying.
* **Sorting Algorithms (Selection Sort & Bubble Sort)**: Demonstrated for performance comparison on a sample dataset, showcasing algorithm efficiency differences.

**3. Performance Analysis**

To evaluate algorithmic performance:

* **Dijkstra’s Algorithm** was used to calculate the shortest path in a graph of six routers. It efficiently found optimal routes in O(E log V) time using a priority queue.
* **Sorting Algorithms** were tested on a sample dataset:
  + *Selection Sort* showed slower performance on larger datasets due to its O(n²) complexity.
  + *Bubble Sort* performed similarly but was slightly slower due to more frequent swaps.

These tests highlighted the importance of choosing efficient algorithms based on data size and application needs.

**4. Challenges Faced and Solutions**

**Challenge 1**: Handling packet loss simulation realistically.  
**Solution**: Introduced a probabilistic packet loss model using C++’s random library with a uniform distribution, simulating a 10% chance of loss.

**Challenge 2**: Managing memory for dynamic packet objects.  
**Solution**: Implemented new to allocate PacketTreeNode objects and ensured proper deletion to avoid memory leaks after popping from the stack.

**Challenge 3**: Visualizing the binary search tree structure.  
**Solution**: Used inorder traversal to print packets in sorted order and demonstrate the hierarchical structure.

**Challenge 4**: Debugging segmentation faults in dynamic graph traversal.  
**Solution**: Added error checks (e.g., out\_of\_range, invalid\_argument) to catch exceptions and provide user-friendly error messages.

**5. Future Improvements**

* **Visual Representation**: Incorporate a GUI to visualize the network, packet routes, and BST structure dynamically.
* **Concurrency**: Add multithreading to simulate real-time packet transmission and handling.
* **Advanced Routing Protocols**: Implement Bellman-Ford or OSPF for comparative study with Dijkstra.
* **Packet Metadata**: Expand packets to include size, TTL, and flags for better simulation accuracy.
* **Logging and Analysis Tools**: Add tools to log performance metrics and visualize throughput, latency, and packet loss trends.

**6. Course Concepts Utilization**

This project extensively applied concepts from the **Data Structures and Algorithms** course:

| **Concept** | **Application in Project** |
| --- | --- |
| Graph Theory | Network modeled as a graph, routing using Dijkstra |
| Queues | Used for packet transmission order |
| Stacks | Used for temporary packet storage (LIFO) |
| Linked Lists | Used to manage lost packets for retransmission |
| Trees | Binary Search Tree to store and manage packet data |
| Sorting Algorithms | Compared selection and bubble sort performance |
| Dynamic Memory Allocation | Used new and managed pointers for packet objects |
| Exception Handling | Used try-catch blocks for robustness |
| Time Complexity Analysis | Compared sorting algorithm performance |
| Probabilistic Modeling | Simulated packet loss using random distributions |

**7. Summary**

This project is a hands-on integration of key computer science fundamentals into a realistic network simulation environment. By combining graph algorithms, classic data structures, and error simulation, it demonstrates how theoretical concepts power real-world networking systems. The project not only helped reinforce course content but also enhanced problem-solving and debugging skills essential for larger systems design.