README: Routing Protocols Assignment (CS425)

Kumar Kanishk Singh (210544)
Sunny Raja Prasad (218171078)
Tanmey Agarwal (211098)

April 21, 2025

Objective

This assignment aims to implement and simulate two core routing algorithms used in computer networks:

- Distance Vector Routing (DVR)
- Link State Routing (LSR)

The program takes an adjacency matrix as input and generates routing tables for each node using both algorithms. The goal is to better understand routing protocol mechanics and convergence behavior.

Files Included

- `routing_sim.cpp` – The complete simulation code for DVR and LSR.
- `Makefile` – Compilation instructions.
- `inputfile.txt` – Example input adjacency matrix.
- `README.pdf` – This document.

Compilation and Execution Instructions

Compiling the Code

To compile the simulation code, navigate to the directory containing `routing_sim.cpp` and run the following command:

```
g++ -std=c++17 -o routing_sim routing_sim.cpp
```

Running the Code

To execute the program:

```
./routing_sim <input_file>
```

Input File Format

The input file should be structured as follows:

- First line: an integer n indicating the number of nodes.
- Next n lines: each line contains n space-separated integers representing the adjacency matrix.

A value of 0 indicates no direct link (except for diagonal entries), and 9999 represents an infinite cost (unreachable).

Output Format

The program outputs:

- The Distance Vector Routing (DVR) table for each node.
- The Link State Routing (LSR) table for each node using Dijkstra's algorithm.

Each table displays: `Destination`, `Cost`, and `Next Hop`.

Explanation of the Code

- **Header and Constants:** The program uses standard headers, defines a constant `INF = 9999` for unreachable paths, and uses type alias `arr` for Dijkstra's priority queue.
- **Function:** `printDVRTable`
  Prints the final routing table for each node after Distance Vector convergence.
- **Function:** `simulateDVR`
  Implements the Distance Vector Routing protocol:
  - Initializes the distance and next hop matrices.
  - Repeatedly updates distances using the Bellman-Ford-like update rule until no changes occur.
- **Function:** `printLSRTable`
  Prints the shortest path cost and next hop for each destination as seen from the source node after Link State computation.
- **Function:** `simulateLSR`
  Implements the Link State Routing protocol:
  - Uses Dijkstra's algorithm with a priority queue.
  - Maintains arrays for distances and previous nodes to reconstruct paths.
  - Computes the next hop from the source to each destination.
- **Function:** `readGraphFromFile`
  Reads the input adjacency matrix from a given file. The matrix should be $n \times n$ integers.
- `main()`

  - Validates input arguments.
  - Reads the graph.
  - Runs both DVR and LSR simulations and prints routing tables.

This program provides a side-by-side comparison of Distance Vector Routing and Link State Routing using simple data structures and clear output for understanding routing behavior in networks.

Design and Implementation

Language and Structure

- Implemented in **C++**.
- Structured with modular functions for clarity:
  - `simulateDVR()` for Distance Vector Routing
  - `simulateLSR()` for Link State Routing (Dijkstra)
  - `printDVRTable()` and `printLSRTable()` to print the routing tables.
  - `readGraphFromFile()` to parse input from text file.

Individual Contribution

- **Kumar Kanishk Singh:** Implemented and debugged the DVR algorithm logic; worked on adjacency matrix handling.
- **Sunny Raja Prasad:** Implemented the LSR algorithm using Dijkstra's method and maintained the input-output handling.
- **Tanmey Agarwal:** Responsible for final integration, I/O formatting, edge-case testing, and preparing README and submission files.

Sources and References

- Lecture slides and notes from CS425.
- Standard C++ documentation: `https://en.cppreference.com`
- Dijkstra's Algorithm basics: GeeksforGeeks and Visualgo.net
- GitHub repository for input data and template: `https://github.com/privacy-iitk/cs425-2025`

Declaration

We hereby declare that the work submitted is our own and has been completed without any unauthorized collaboration. Any external references or code used are cited appropriately. This work adheres to the academic integrity policies of IIT Kanpur.