Department of
Computer Science and Engineering

University of Dhaka

Title: Implementation of Distance Vector Routing Algorithm

CSE 3111: Computer Networking Lab

Batch: 28/3rd Year 1st Semester 2024

Course Instructors

Dr. Ismat Rahman (IsR)
Mr. Jargis Ahmed (JA)
Mr. Palash Roy (PR)

1 Objective(s)

- To gather understanding of the working principle of the Distance Vector (DV) Routing Algorithm.
- To simulate Distance Vector Routing using Bellman-Ford algorithm.
- To implement Poison Reverse to prevent routing loops.
- To analyze convergence and visualize routing updates in a distributed network.

2 Background Theory

Distance Vector (DV) Routing is a dynamic, distributed routing protocol where each router shares its knowledge of the network with its immediate neighbors. It is based on the Bellman-Ford algorithm and is used in many early routing protocols, such as RIP (Routing Information Protocol). DV is a dynamic routing protocol in which routers periodically share their routing tables with immediate neighbors. Each router maintains a vector of distances (costs) to every other router. It uses the Bellman-Ford Algorithm to update routes as follows

$$D_x(y) = min_v c(x, v) + D_v(y),$$

where, $D_x(y)$ is cost from node x to node y. $c(x, v)$ indicates cost from x to its neighbor v. $D_v(y)$ denotes v's distance to y. Key concepts of the DV routing are as follows

- Periodic Updates: Each node sends its table to neighbors.
- Incremental Updates: Triggered by a change in topology.
- Route Loop: Occurs when incorrect routes circulate endlessly.
- Poison Reverse: To avoid loops, if router A uses B to reach D, A advertises the distance to D as ∞ to B.

Each router maintains a routing table where each row represents a known destination in the network. Each entry in the routing table has the following components:

- Destination address
- Distance (cost or metric) to the destination
- Next hop (neighbor to use for forwarding)

3 Lab Task: Please implement yourself and show the output to the instructor in the next lab.

You have to design and implement a program that simulates a network of routers using the Distance Vector (DV) Routing Protocol. Each router will operate independently, maintaining its own routing table and periodically exchanging distance vectors with its immediate neighbors. The goal is to study how routing tables are updated based on the Bellman-Ford algorithm and how dynamic changes in link costs impact convergence. You must

- Implement this simulation in Java or Python.
- Display routing tables on every change and cost update.
- Simulate edge cost change at regular intervals (30 seconds).
- Show the final shortest path from each router to all others.

3.1 Problem Analysis

1. **Network Topology Configuration:** At first, the initial network topology must be read from a configuration file (topology.txt or similar). Each router should be initialized with:
   - A unique router ID (e.g., A, B, C, etc.)
   - A mapping of ports and neighboring routers
   - The initial cost to its direct neighbors
2. **Routing Table Initialization:** Upon reading the topology:
   - Each router should know the cost to its immediate neighbors.
   - The cost to other routers should be initialized to ∞ (infinity).
   - The routing table should store:
     - Destination
     - Cost to destination
     - Next hop
3. **Distance Vector Exchange and Update:** Routers should periodically exchange their routing tables (distance vectors) with neighbors. Upon receiving a vector from a neighbor, a router must:
   - Apply the Bellman-Ford update formula:

$$D_x(y) = \min_{v \in N(x)} \{c(x, v) + D_v(y)\}$$

   - If a shorter path is found, update its routing table and mark the table as changed.
   - **Use Poison Reverse:** if router A uses router B to reach destination C, then A must advertise the route to C as having cost ∞ to B.
4. **Dynamic Cost Changes**
   - Every 30 seconds (or a configurable time interval), randomly update the cost of one link in the network.
   - Routers should detect the change and trigger vector updates and route recalculation.
5. **All-Pair Shortest Paths and Logs:**
   - Each router must print its routing table after any change.
   - The system should maintain logs of:
     - Total number of routing updates/messages exchanged
     - Number of iterations for convergence after any cost change
   - Optionally, the simulation can produce a full network-wide shortest path table for analysis.

.

3.2 Example Scenario: Initial Network Topology

topology.txt:

```
A B 2
A C 5
B C 1
B D 3
C D 2
```

This configuration describes the following undirected graph:

$$A \leftrightarrow B(2), \quad A \leftrightarrow C(5), \quad B \leftrightarrow C(1), \quad B \leftrightarrow D(3), \quad C \leftrightarrow D(2)$$

3.3 Sample Output Console (Router A)

```
[Time = 0s] Initial Routing Table at Router A:
Dest | Cost | Next Hop
----------------------
A    | 0    | A
B    | 2    | B
C    | 3    | B
D    | 5    | B

[Time = 30s] Cost updated: B <-> D changed from 3 to 7
[Time = 31s] Routing Table at Router A:
Dest | Cost | Next Hop
----------------------
A    | 0    | A
B    | 2    | B
C    | 3    | B
D    | 5    | C

[Time = 32s] Convergence complete. Total messages exchanged: 8
```

3.4 Implementation Guidelines

To implement the Distance Vector Routing Protocol as described, follow the structured modular approach below. This helps you keep your code organized, flexible for topology changes, and allows easy integration of periodic updates and logging.

1. Class and Data Structure Design

- **Router (Class):** Represents a single router/node.
  - `id: String` – Unique Router ID (e.g., A, B)
  - `neighbors: Map<String, Integer>` – Direct neighbors and cost
  - `routingTable: Map<String, (cost, nextHop)>`
  - `distanceVector: Map<String, Integer>` – Current known costs
- **Network (Class):** Represents the global network.
  - `routers: Map<String, Router>`
  - `topologyFile: String` – Input file name (e.g., `topology.txt`)
  - `messageCount: Integer` – Total messages exchanged
  - `log: List<String>` – Records of all routing changes

2. Essential Functions to Implement

- `readTopology(fileName)` – Read the file and build the graph by:
  - Adding each router as a node
  - Populating neighbors and their link costs
  - Initializing routing tables (cost to self = 0; neighbors = cost; others = ∞)
- `initializeRoutingTables()` – For each router, populate:

$$routingTable[router][destination] = (cost, nextHop)$$

- `sendDistanceVector()` – Each router sends its vector to all neighbors.

- `receiveVector(routerId, fromNeighbor, vector)` – Perform Bellman-Ford:

$$D_x(y) = \min\left(D_x(y), c(x,v) + D_v(y)\right)$$

  Update routing table if better path is found.

- `applyPoisonReverse()` – If a route to destination D uses neighbor N as next hop, advertise cost $= \infty$ to N for D.
- `updateCostRandomly()` – Every 30s, pick a random edge and update its cost (increase or decrease). Propagate changes to neighbors.
- `printRoutingTable(routerId)` – Print current routing table for a router.
- `logChange(message)` – Append change information to system logs.

3. Timer and Convergence Logic

- Use a timer or thread to:
  - Trigger periodic `sendDistanceVector()` every 5 seconds.
  - Invoke `updateCostRandomly()` every 30 seconds.
- After any routing table change:
  - Print the updated routing table.
  - Log the event and increment message counter.
  - Continue until convergence (no table changes after N cycles).

3.5 Evaluation Criteria and Final Output Expectations

You should demonstrate the following output.

- Each router's routing table (at each major change)
- Final network-wide all-pair shortest paths
- Total number of messages exchanged
- Total convergence time (in seconds or iterations)
- Optional: A visualization of the final topology.

You will be evaluated based on the following criteria.

- Correctness of Bellman-Ford updates and Poison Reverse
- Functional periodic cost updates and convergence
- Accurate routing table outputs
- Performance analysis (message count, iterations)

4 Submission as a Lab Report

Run simulations of DV routing using different network topologies (Small-scale to large-scale) and analyze different performance metrics such as the total number of messages exchanged, the total convergence time (in seconds or iterations). Include the following sections in your Lab Report.

1. **Introduction** Provide a brief introduction of DV routing
2. **Objectives:** Mention major 3-4 key objectives.
3. **Detailed Description of the Program**

- Explain the structure of your code.
- Describe the data structures used for:
  - Router representation (class structure)
  - Distance vector table
  - Neighbor mappings
- Describe the use of the **Bellman-Ford algorithm** for routing table updates.
- Detail how **Poison Reverse** is implemented to prevent routing loops.
- Explain how periodic updates are scheduled, and how dynamic link changes are introduced randomly. How cost changes (triggered every 30s) propagate through the network to achieve convergence.

4. **Network Topology Used**
   - Attach or reproduce the contents of `topology.txt`.
   - Include information on link costs and connected routers.
5. **Screenshots or Console Output**
   - Include at least 3 output samples from different routers showing:
     - Initial routing table
     - Table after an update
     - Final converged table
   - Highlight any changes in shortest path routes after link cost updates.
6. **Routing Table Update Logs**
   - For at least one router, include a full log of routing table updates over time.
   - Represent in a tabular format:

```
[Router A] Table Update at Time = 10s
Dest | Cost | Next Hop
-----------------------
A    | 0    | A
B    | 2    | B
C    | 3    | B
D    | 5    | C
```

7. **Performance Analysis**
   - Evaluate and comment on:
     - Time complexity of each Bellman-Ford update step and Memory usage for storing per-router entries.
     - Number of messages exchanged per update.
     - Convergence time (in seconds or iterations).
8. **Challenges and Debugging**
   - Discuss any problems you faced during the implementation.
   - Example issues:
     - Infinite loops in routing
     - Incorrect table updates
     - Timing/thread synchronization
   - Mention what you did to fix these issues.
9. **Conclusion**

   Summarize your implementation results and what you have learned from this experiment.

- Discuss the successful convergence of routing tables using Bellman-Ford.
- Reflect on the importance of loop prevention via poison reverse.
- Comment on the impact of dynamic link changes and how the protocol adjusts to it.
- State why Distance Vector routing remains foundational despite modern advances.
- Conclude with your understanding of DV's practicality, advantages (simplicity), and limitations (slow convergence, loops).

5 Policy

Copying from the Internet, classmates, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.