COMP/ELEC416

Project 3 – Part 2
ModivSim

KOÇ
UNIVERSITY

Disclaimer

These PS slides are provided only for the sake of better understanding of the project structure. You should see this a complementary resource and implement your project based on the uploaded project description in the course website.

Do not ignore the implementation details asked in the project description by solely relying on these slides.

Description

Goal: investigate the network layer routing algorithms.

Focus on:

Modified Distance Vector Routing algorithm

Flow Routing simulation

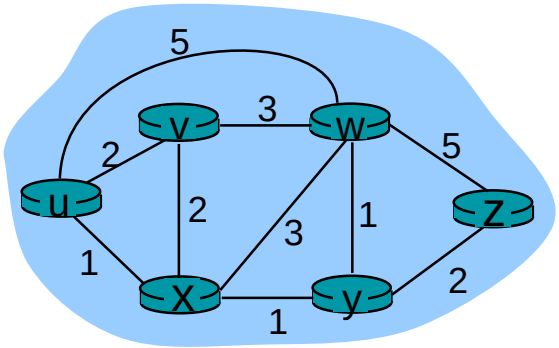
Distance Vector Routing

- Let dx(y) := cost of least-cost path from x to y

- Then **dx(y) = min {c(x,v) + dv(y) }**
  - **min:** taken over all neighbors v of x
  - **c(x, v):** cost to neighbor v
  - **dv(y):** cost of neighbor v to destination y

- The node achieving minimum is the next hop in the shortest path, used in forwarding table

Bellman-Ford example



source: u, destination: z
neighbors of u: v, x, w
clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$d_u(z) = \min \{ c(u,v) + d_v(z), \; c(u,x) + d_x(z), \; c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\} = 4$$

node achieving minimum is the next hop
in the shortest path, used in forwarding table.
Example: The next hop is x.

Distance Vector

- $D_x(y)$ = estimate of the least cost from x to y
  - x computes distance vector $\mathbf{D}_x = [D_x(y): y \in N]$

- node x:
  - knows cost to each neighbor v: c(x,v)
  - maintains its neighbors' distance vectors. For each neighbor v, x maintains

    $\mathbf{D}_v = [D_v(y): y \in N]$

- Distance table → distance vector → forwarding table

ModivSim: Distance Vector Routing Simulator

- Classes: Node, Message, and ModivSim
- Node: a single Node
- Message: messages with distance vector estimates
- ModivSim: main class, reads input, simulates, prints outputs

ModivSim: Initialization

- Read node-i.txt -> Each node has its own txt file containing the information.
- Each line corresponds to a Node object
- Create Node(s) objects from the input file
- Keep the set of nodes in the ModivSim class
- Each node should run in different process/thread
- Each node should periodically send updates to the neighbors
- Each node will keep two next best hops

ModivSim

-Implements rounds as a loop variable

-Performs distance vector at each round for every node:

- Node n1, n2
- boolean n1.sendUpdate() //does n1 have anything new to send?
- //n1 should create a new message m with distance vector information
- void n2.receiveUpdate(Message m) //n1 updates n2 with the message containing distance vector
- How to invoke receiveUpdate of a node from another node? //Up to you, for example: n1.updateNeighor(Node neighbor)
- Display the requested information at each round (see the project description)

In each node

- Store costs to all directly-connected neighbors
- Store and maintain node’s own distance vector
- Store all directly-connected neighbors’ distance vectors
- A distance vector maintains the minimum cost to every nodes in the network over all possible paths to reach it.

Execution Scenarios:

Static Link Cost Scenario: Fixed Link Cost

<Node Number>,<(Neighbour Number, Link to Neighbour's Cost, Link Bandwidth)>,....., <(Neighbour Number, Link to Neighbour's Cost, Link Bandwidth)>

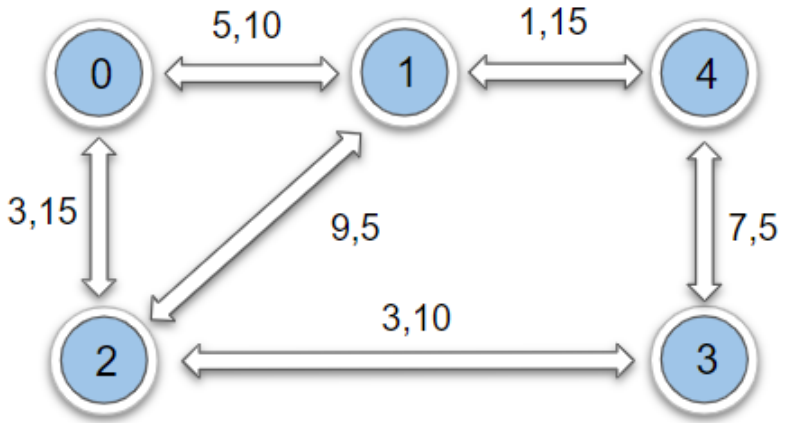
Node0.txt: 0,(1,5,10),(2,3,15)

Node1.txt: 1,(0,5,10),(2,9,5),(4,1,15)

Node2.txt: 2,(0,3,15),(1,9,5),(3,3,10)

Node3.txt: 3,(2,3,10),(4,7,5)

Node4.txt: 4,(1,1,15),(3,7,5)



Execution Scenarios:

Dynamic Link Cost Scenario: Time-variant Link Cost

<Node Number>,<(Neighbour Number, Link to Neighbour's Cost, Link Bandwidth)>,....., <(Neighbour Number, Link to Neighbour's Cost, Link Bandwidth)>

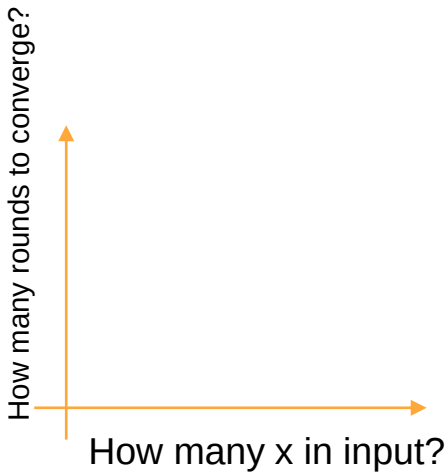
Node0.txt: 0,(1,5,10),(2,3,15)

Node1.txt: 1,(0,5,10),(2,x,5),(4,1,15)

Node2.txt: 2,(0,3,15),(1,x,5),(3,3,10)

Node3.txt: 3,(2,3,10),(4,7,5)

Node4.txt: 4,(1,1,15),(3,7,5)



Assumptions

- Costs to non-neighbors are assumed to be infinite
- Infinity is represented by the integer 999
- Cost of a node to itself is zero, best path to a node is itself
- Packets are sent one by one and received with the order they were sent

Creating Forwarding Table

- Based on the distance vector, generate routing table.
- Should contain 2 next hops arranged according to the cost.

Key	Value
1	(1, 2)
2	(2,1)
4	(1, 2)
3	(2, 1)

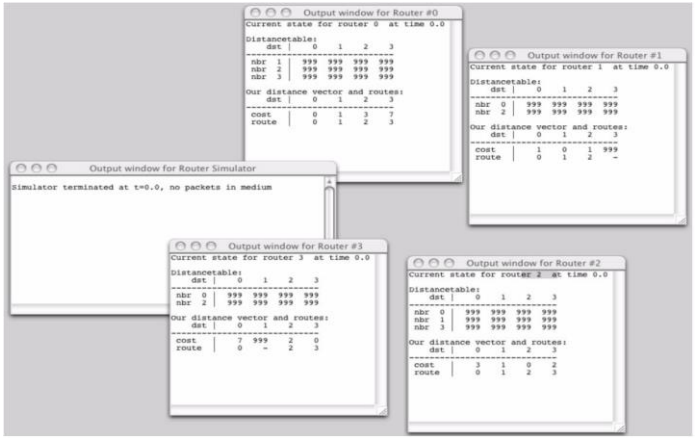
Example Forwarding Table for Node 0.

Execution

- The ModivSim class is the main class of the simulator that starts by initializing each router.

- Several windows appear on the screen, one for the simulator output and one for each router node that is initialized

- As an alternative, you can start several terminals manually (instead of main class creating windows) and then detect the convergence of the algorithm.



Example of how the project will execute.

Flow Routing Simulation

- Keep link bandwidth at nodes.
- Store bottleneck bandwidth.
- Send the flows

Example:

- Flow A: 0 -> 3, 100Mbits
- Flow B: 0->3, 200Mbits
- Flow C: 1->2, 100Mbits

Demo Scenario

Your submission is tested against:

- Topologies of different sizes
- Static/dynamic scenarios
- Different numbers of dynamic links in dynamic topologies
- Different topologies for each group
- Different packet flows

Demo Organization

All groups should register their group members within the specified time.

Attending the demo session is required for your project to be graded.

All group members must attend the demo session.

The on-time attendance of all group members is considered as a grading criteria.