Link State Routing Protocol Simulator Design Report

Link State Routing:

- There are mainly two categories of routing protocol.
  1.Link State Routing 2. Distance vector routing
- Link-state routing protocols are one of the main classes of routing protocols used in packet switching networks for computer communications.
- The basic concept of link-state routing is that every router constructs a map of the connectivity to the network, in the form of a graph.
- In Distance vector routing routers use the distributed algorithm to build the routing table for each router i.e. each node shares its routing table with its neighbors.
- In Link State Routing routers build the routing table using message passing and it includes list of routers and links, and how they are connected to each other, including cost of the links.
- Each router then uses Dijkstra Algorithm to calculate its own routing table. Every router in the network is having same topology but the routing table for each router is different.

Dijkstra Algorithm:

- Dijkstra's algorithm is an algorithm for finding the shortest paths between routers in a graph of the network.
- Dijkstra's original algorithm is for the shortest path between two nodes but a more common different fix a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph.
- For a given source node in the graph, the algorithm finds the shortest path between that node and every other. Also, it can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined.
- This shortest path algorithm is widely used in network routing protocols, most particularly in Open Shortest Path First (OSPF) and IS-IS (Intermediate System to Intermediate System).

Implementation of Dijkstra's Algorithm:

- We are using the 'calDijkstra' method for calculating the shortest path.This method is having input parameters as inputMatrix as the original topology matrix, matrixSize as the size of original matrix size, Source as initial router, Destination as destination router in case of specific path calculation, Choice for printing the result of various options.
- Algorithm Steps:
    1. Declare the all the variable required for the calculation of shortest path.
        a) paths (for storing path from one router to other).
        b) totalCost (to store the cost of all other routers from the source router).
        c) distance (to calculate the optimal distance from the source router to each router)
        d) preDet (to store the intermediate router if there is no direct link between the source router to destination router)
        e) visited (to keep the track for visited routers while finding the distance and path)
        f) min (for calculating the minimum distance)
    2. Initialize the variables:
        a) paths to blank.
        b) totalCost to zero.
        c) distance to distance of the source router to all other routers using original topology matrix.
        d) preDest to the source router.
        e) visited of source to true.
        f) Replace the -1 values in the original topology matrix to max value of short.

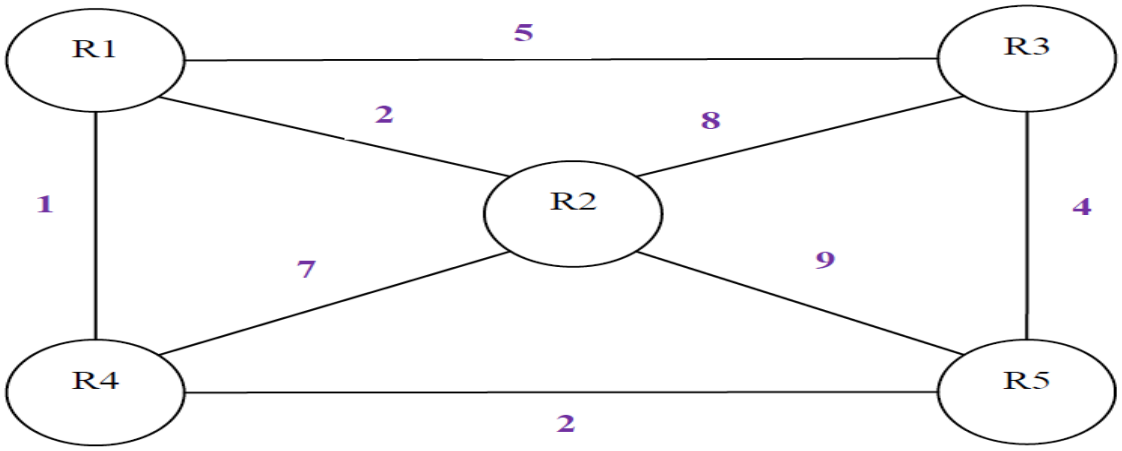
    3. Repeat till matrixSize
        a) Assign max value of Short to min.
        b) Repeat till matrixSize
            - Find the min distance from the distance array of the source router and assign that router to the nextNode variable.
            - Mark visited of nextNode to true.
            - Repeat till matrixSize
                - If there is minimum distance to any other destination router from the nextNode and that node is node visited, then update the distance array for that destination router and insert that router into the preDet array.
        c) Repeat till matrixSize
            - Assign path of source to source.
            - Check if distance of source is not max value of short

- Assign the distance of that router to totalCost.
- If ith router is not equal to source router
  - Repeat till jth router is not equal to source router
    - Get router from the preDet array and assign store it in paths variable.
- Else part: Assign totalCost to max value of the Short. Paths value to Not reachable. Distance of path equal to zero.

d) Check if choice is equal to 2 then print the interface table for the source router.

e) Check if the choice is equal to 3 then print the shortest path between source to destination using paths and totalCost array.

Example of Dijkstra's Algorithm:



Original Topology:

0 2 5 1 -1

2 0 8 7 9

5 8 0 -1 4

1 7 -1 0 2

-1 9 4 2 0

Step 1: Source=0 i.e. Actual router selected is 1

Declare Paths[], visited[], distance[],

preDet=[],totalCost= []

Step 2:

Paths [, , , ,], visited=[false, false, false, false, false], distance=[0, 2, 5, 1, 32767]

preDet=[0, 0, 0, 0, 0],totalCost= [0, 0, 0, 0, 0]

Step 3:

For i=0:

a)Visited= [true, false, false, false, false],distance=[0, 2, 5, 1, 32767],nextNode=3

Min=1

b)finding the distance from the 3rd node to other nodes and checking whether it is minimum than the existing distance.

As there is link for the 4th node from 3rd node with cost of 3. So, updating the distance array distance= [0, 2, 5, 1, 3]. Also for route from 0th to 4 th we have to go through 3rd node so update the preDet= [0, 0, 0, 0, 3].

Repeat above step till we cover the all the nodes.

c) print the interface table if choice is 2: Interface table for router 1

Destination Interface

```
==========================

R1            -
R2            R2
R3            R3
R4            R4
R5            R4
```

d) print the shortest path if choice is 3: source is 5 and destination 1

```
Shortest Path from R5 to R1 is: R5->R4->R1, the total cost is 3.
```

Software Interfaces

- This Software Application is built using Java 1.8.
- Operating System: Windows 10.

Software Design Description

- Below we mentioned a different classes and methods used for development.
  1. LinkStateRouteSim.java
     a. Variables
        i. static Integer inputMatrix : static variable to store the data in matrix read from topology file
        ii. static Integer matrixSize : static variable to store the size of topology matrix
     b. Methods
        i. public static void main(String[] args) : main method for displaying the Menu options and calling related functions
        ii. void readFile(String filename) : method for reading the topology input file from project folder
  2. DijkstraAlgorithm.java
     a. Variables
        i. static String[] paths : Static variable to store the paths from one router to all other router
        ii. static Integer[] totalCost : Static variable to store the distance from one router to all other router
     b. Methods
        i. public static int[][] calDijkstra(Integer[][] Mat, Integer Matsize, Integer source, Integer dest, Integer choice) : This function is calculating the shortest path from source node to all other nodes. Also it display the connection table of selected router and shortest path from source to destination router
        ii. public static void shortestPath(Integer[][] inputMatrix, Integer matrixSize) : method calculating the shortest from source to destination and taking the user inputs for source and destination
        iii. public static void bestBroadcast(Integer[][] Mat, Integer Matsize) : To find the best router for the broadcasting which is having least distance from other routers
        iv. public void updateTopology(Integer[][] Mat, Integer Matsize, int delNode, boolean option) : for modifying the topology.User can delete the router from existing topology
        v. public static void printMatrix(Integer[][] inputMatrix) : function is used for printing the topology matrix