CS 542 Project Report
Link-State Routing Algorithm

By,

Patel Suchita Kiranbhai

CWID: A20410967

Seat No: 43

Computer Networks – I

Department of Computer Science

Illinois Institute of Technology

INDEX

AIM

The purpose of this project is to simulate Link-State Routing Algorithm.

INTRODUCTION

The Link-State Routing Simulator calculates the shortest distance between source and destination node and displays the minimum cost. It takes matrix file as input for the connection topology and displays forwarding table for selected router. The topology can be modified by deleting any router. It also finds the best router for broadcasting.

DESCRIPTION

- Routing is the process of directing data packets from source to destination and communication between routers is specified by routing protocols.
- There are two classes of routing protocols: Link-State Routing and Distance-Vector Routing.
- Link-State Routing is used in packet switching networks where every node takes route according to the forwarding table that is generated by algorithm.
- It finds the minimum distance graph between the switching nodes or routers.
- The graph is a connected structure that shows the shortest path in terms of cost from source to destination.
- This information can be directly used for moving in the network.

LINK-STATE ROUTING PROTOCOL

- Link-State Routing Algorithm models the network in the form of weighted graph where each node is a router and edge is the link.
- The link also specifies the cost of traversal between the routers.
- It can be applied to the networks in which the direct distance is known.
- It is based on Shortest Path First (SPF) algorithm also known as Dijkstra's Algorithm where it computes the best path between source and destination.
- It takes each node individually to calculate the optimal path to every other node in the network.
- For nodes that are not connected directly it first finds the neighbouring nodes by the information provided in forwarding table and connects to destination via intermediate nodes.

DIJKSTRA'S ALGORITHM

Dijkstra's Algorithm is the shortest path algorithm that computes the minimum distance path from one source node to all other destination nodes in the graph. The algorithm is as follows:

```
1.  Distance to source dist[s] ← 0                  (Distance from source to source)
2.  for each vertex v in Graph:                     (Initialization)
3.          dist[v] ← INFINITE                      (Distance is unknown from source to v)
4.          prev[v] ← UNDEFINED                     (Previous node in best path)
5.          add v to Q                              (Add initially all unvisited nodes in Q)
6.  while Q is not empty:
7.          u ← vertex in Q with min dist[u]        (To select least distance node first)
8.          remove u from Q                         (Remove node from to be traversed list)
9.          for each neighbour v of u:              (v is still in Q)
10.                 val ← dist[u] + length(u, v)    (Check if indirect distance is lesser)
11.                 if val < dist[v]:               (Path with lesser cost has been found)
12.                         dist[v] ← val
13.                         prev[v] ← u
14. return dist[], prev[]                           (Return shortest dist and prev node list)
```

IMPLEMENTATION OF DIJKSTRA'S ALGORITHM – Project Details

- The implementation is done in java.
- The application requires an input data file eg. Topology.txt which has the matrix topology.
- The program displays various choices for performing different operations.
- Assumptions: Distance of any node to itself is 0. And if nodes are not directly connected then distance is given by -1.

Following are the **main functions** implemented:

- The below function contains the logic to find the shortest distance and returns the array with minimum distance from source to other nodes:

private int[] shrtst_Path_Dijkstra_Algo(**int**[][] matrx, **int** sNode, **int**[] forward_tab_dist, **int**[] list_Prevoius_Nodes)

- The following function calculates the next node to which the packet needs to be forwarded:

private boolean display_Forward_Table(**int**[][] matrixTopo ,**int** source, **int**[] forward_tab_dist, **int**[] list_Prevoius_Nodes, ArrayList<Integer> del_router)

- The following function computes and displays the shortest path from source to destination:

private void source_to_dest_path(**int** source, **int** dest, **int**[] forward_dist_tab, **int**[] list_Prevoius_Node, ArrayList<Integer> del_router)

USER GUIDE

Executable jar: To run java file in command prompt

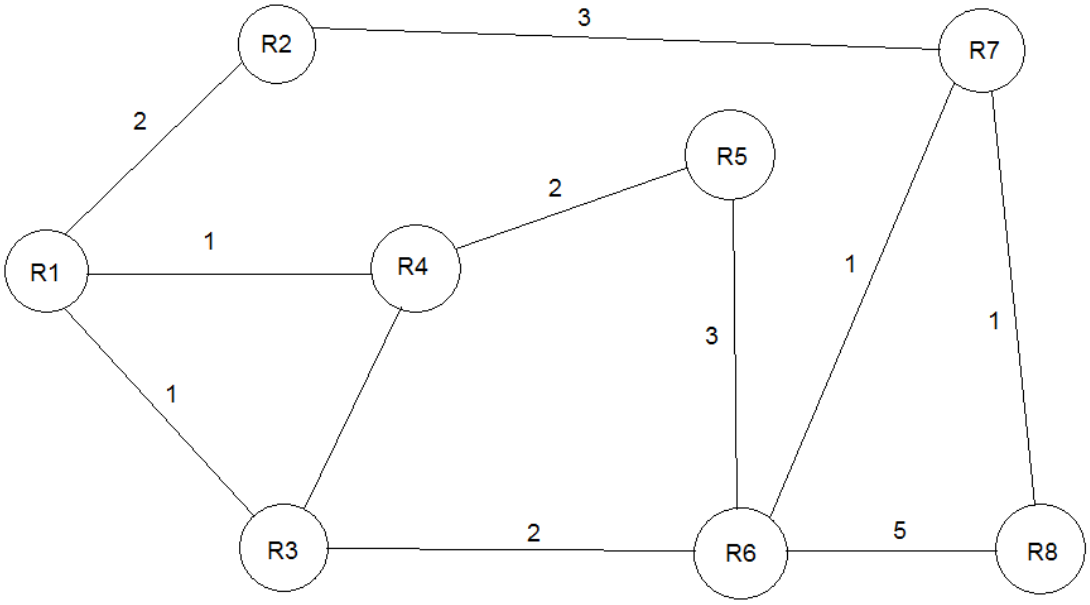
- Copy the input matrix file 'topology.txt' in LinkStateRoutingProtocol.jar file folder.
- Go to the address of the file in command prompt.
- Type the following command "java -jar LinkStateRoutingProtocol.jar" and press enter or run the 'runProject.bat' file.
- The application starts and displays main menu. Perform the operations as required.

Source code: To compile and run source code

- Copy the input matrix file 'topology.txt' in LinkStateRoutingProtocol.java file folder.
- Go to the address of the file in command prompt.
- Set the environment variables by typing the command 'path="<specify the jdk path>"' and press enter.
- Now compile the file by typing "javac LinkStateRoutingProtocol.java".
- Type "java LinkStateRoutingProtocol" to run the program.

TEST CASES

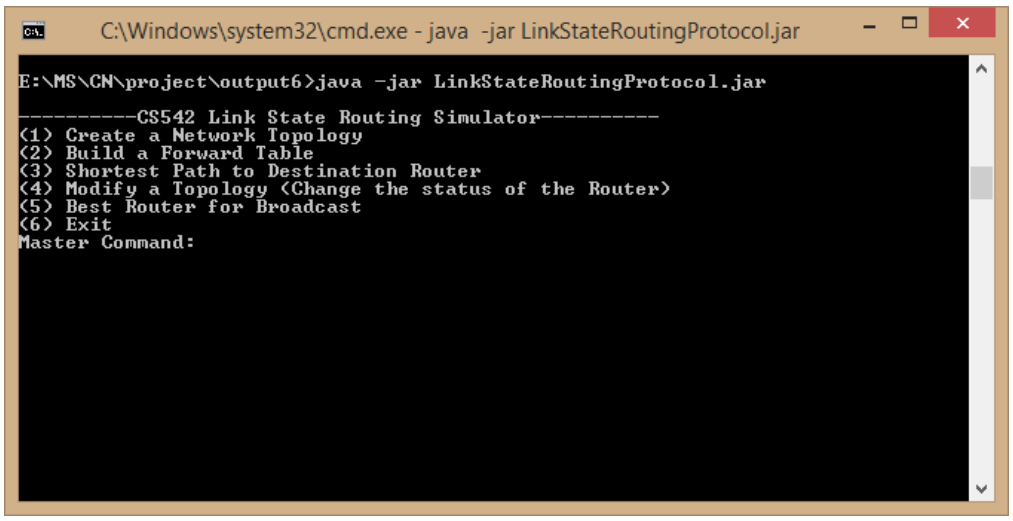
Test case 1:



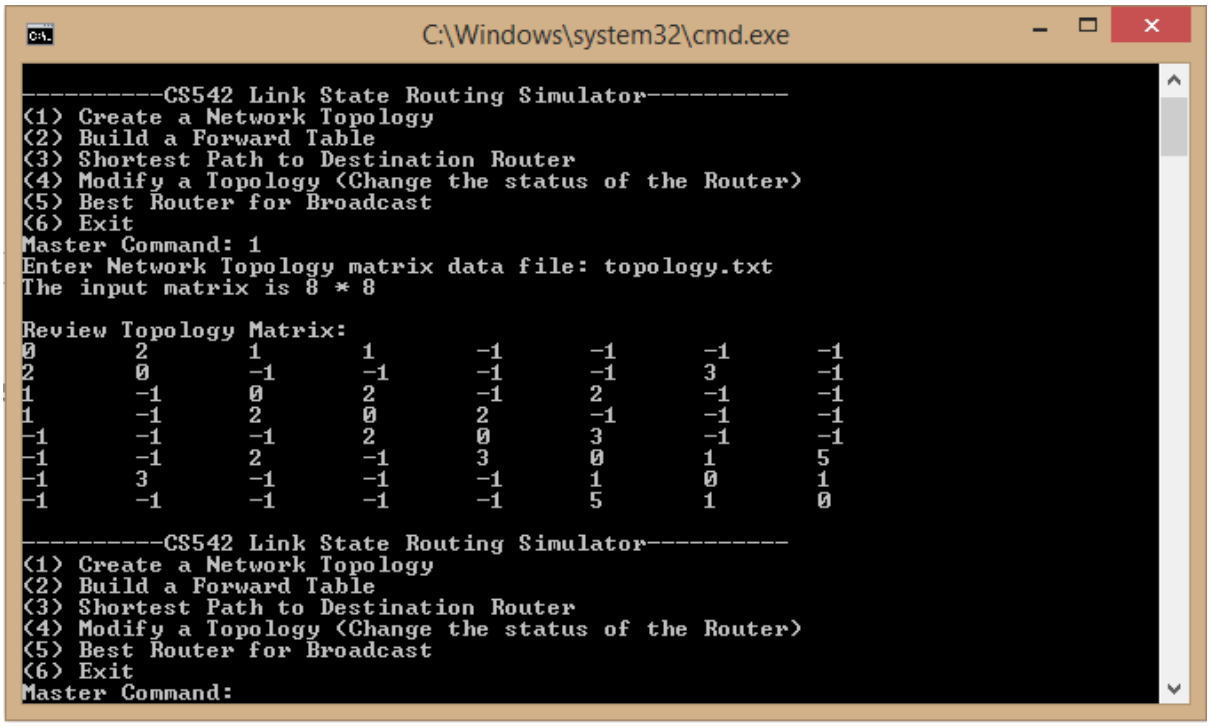
Topology Matrix:

```
 0  2  1  1 -1 -1 -1 -1
 2  0 -1 -1 -1 -1  3 -1
 1 -1  0  2 -1  2 -1 -1
 1 -1  2  0  2 -1 -1 -1
-1 -1 -1  2  0  3 -1 -1
-1 -1  2 -1  3  0  1  5
-1  3 -1 -1 -1  1  0  1
-1 -1 -1 -1 -1  5  1  0
```

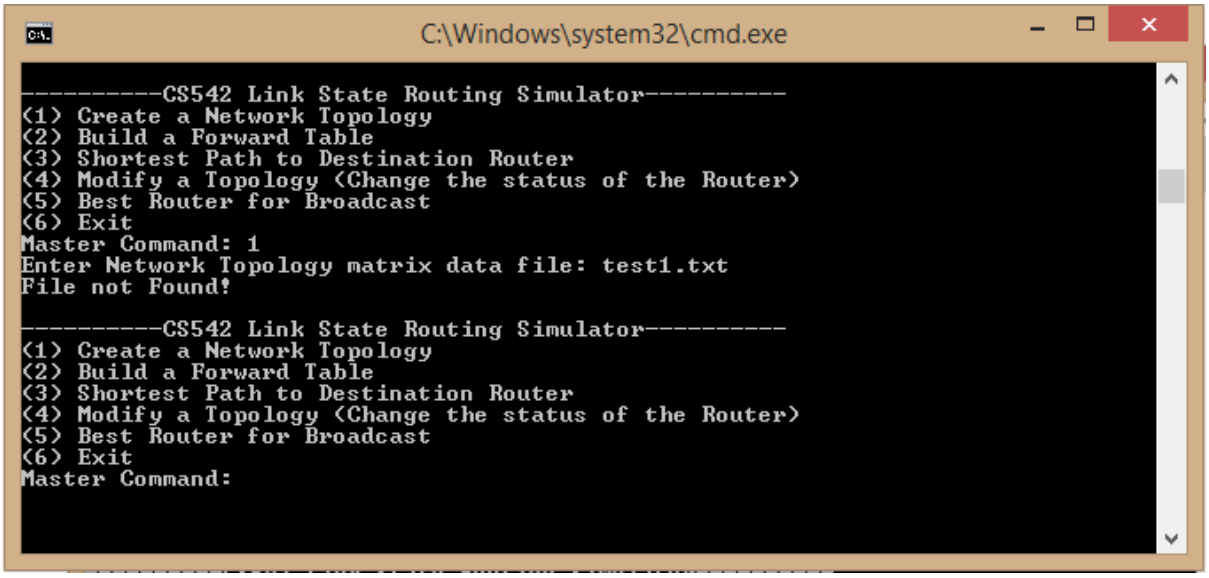
1. When program is executed the main menu appears as follows:



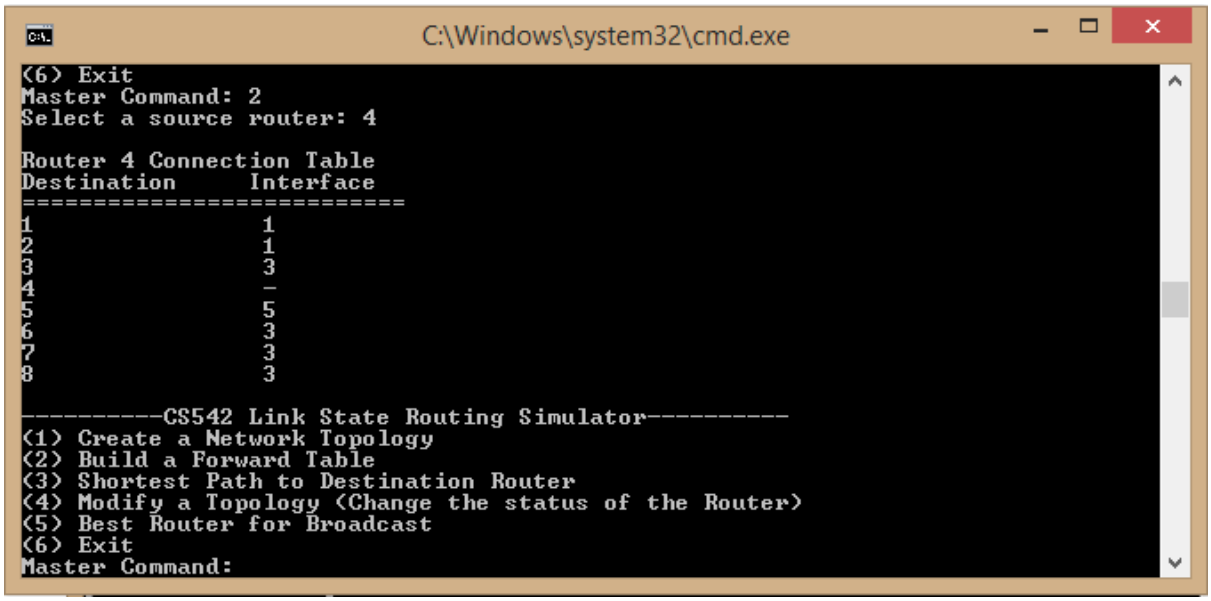
2. Selection of option 1 prompts for the data matrix input file. It displays the matrix topology. Source to source distance is shown by 0 and if node not reachable it is shown by -1.



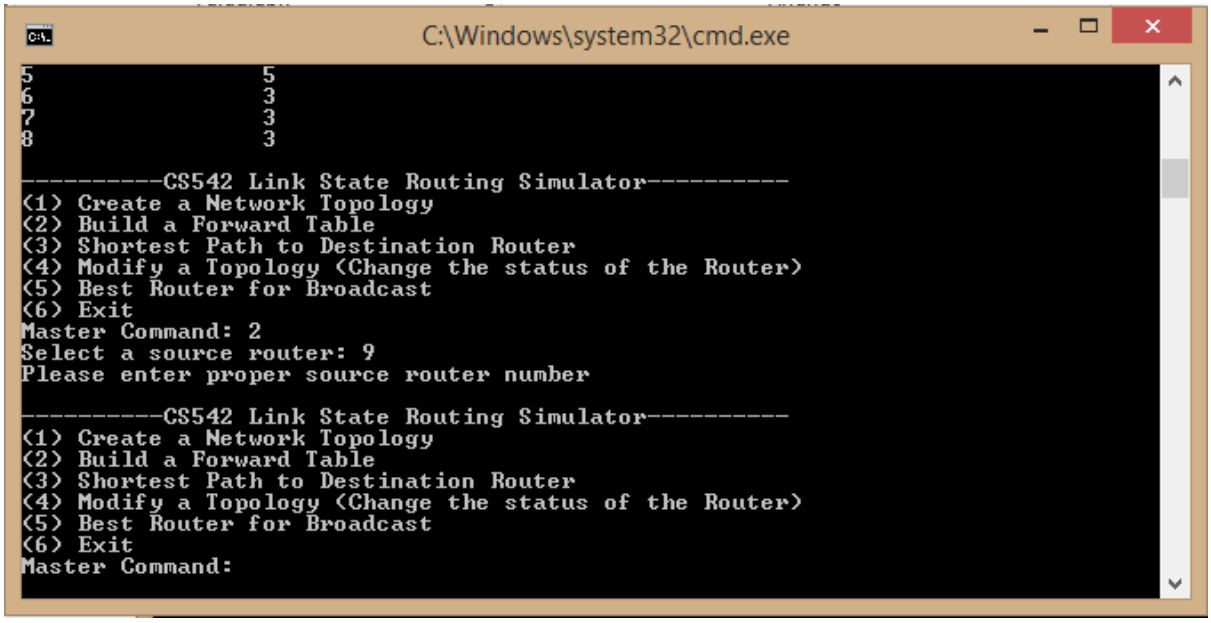
- If File is not found it displays an error message.



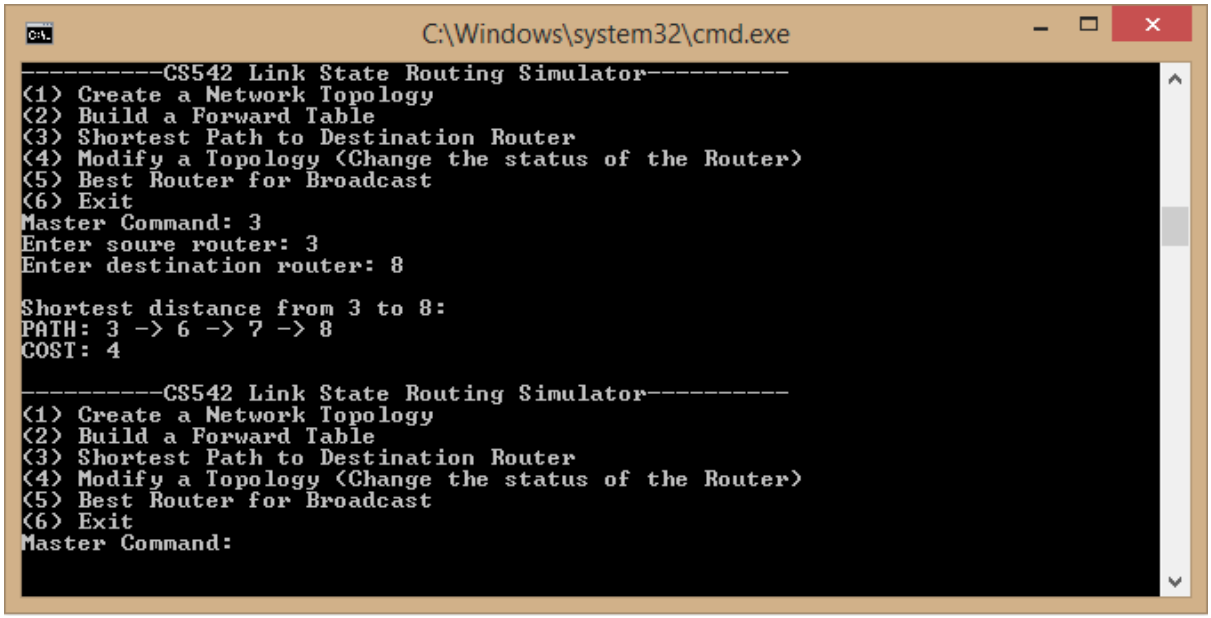
3. Option 2 will display the Connection or Forwarding table for specified router. The program displays '-' if is the source node itself.



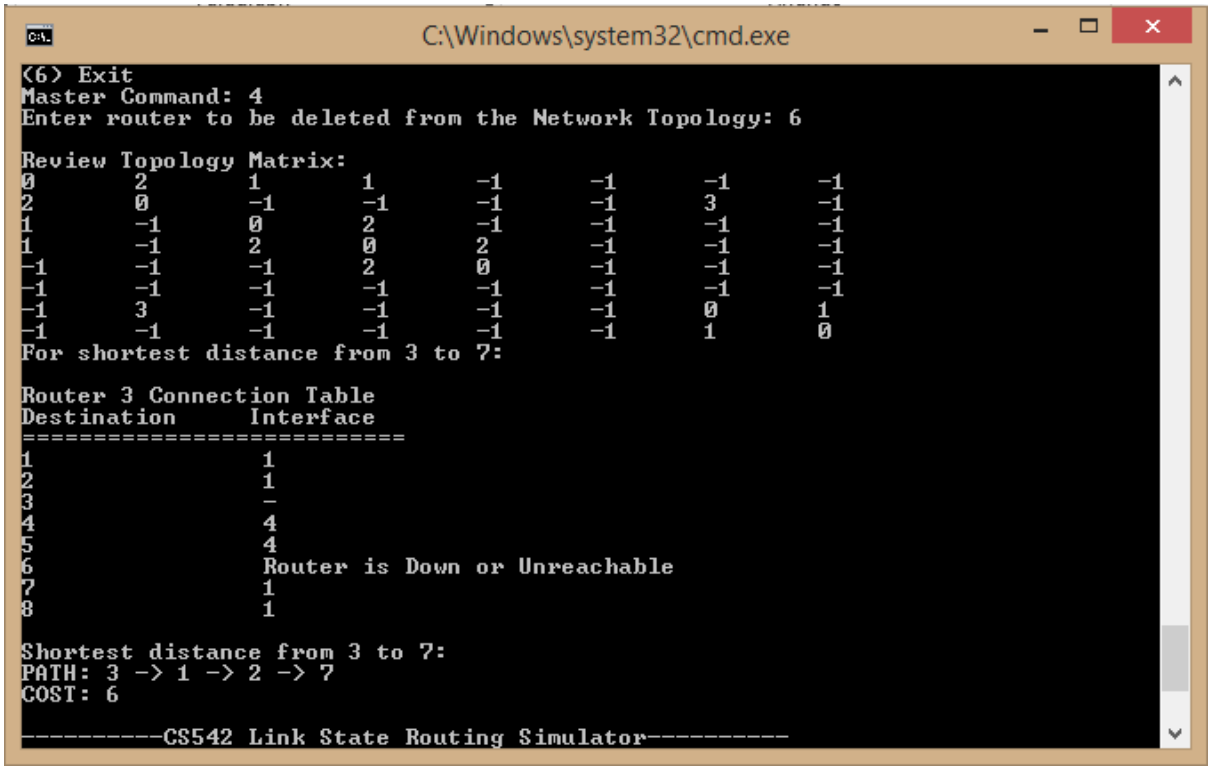
- If the router number other than existing is specified then an error message is displayed. For example the input matrix is 8*8 and router 9 is selected as source router then following message is displayed.



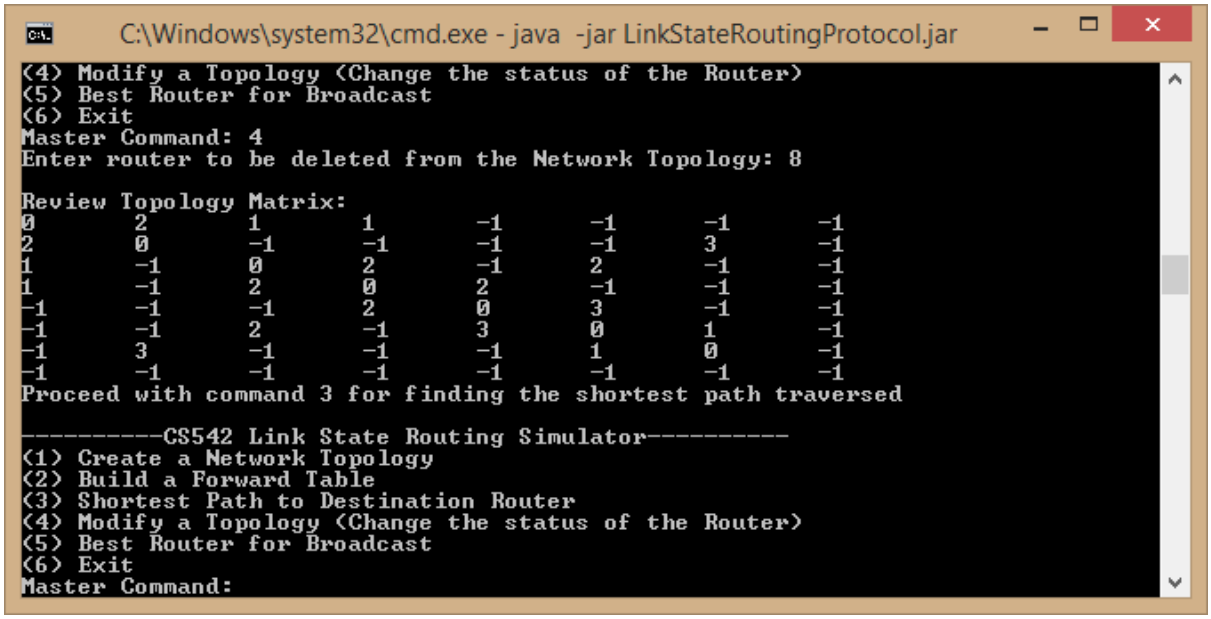
4. Option 3 computes the optimal path with minimum cost from specified source to destination node. It displays the path along with the minimum cost to traverse that path.



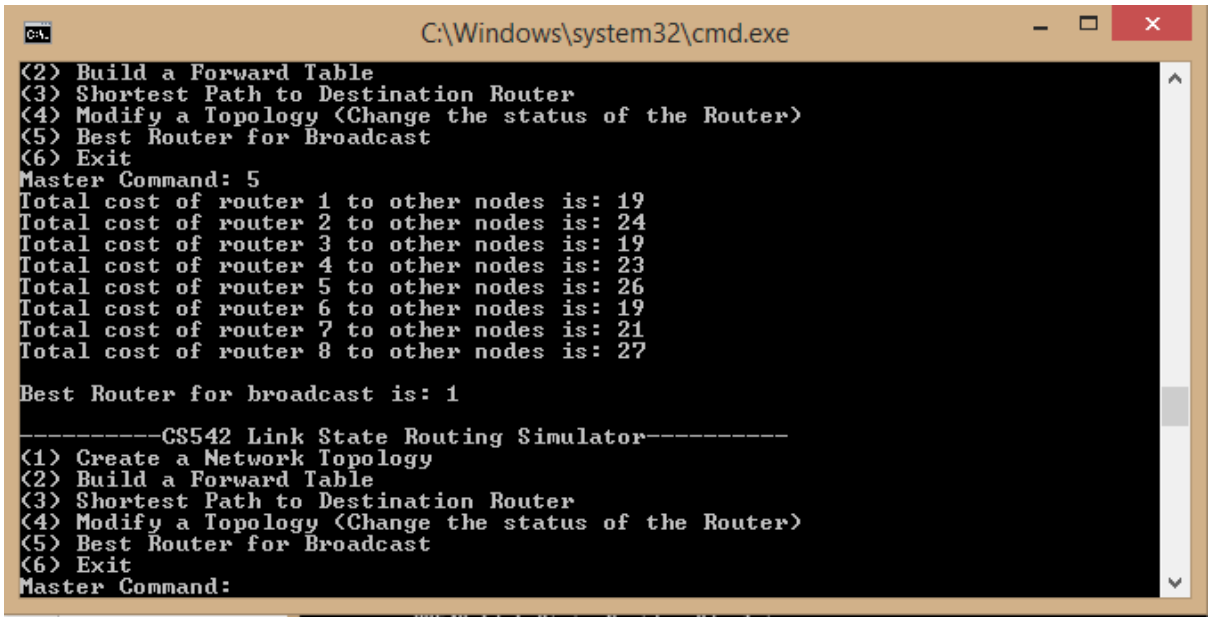
5. Network topology can be modified if any router is down. Option 4 prompts for router to be deleted. Now it will recalculate the forwarding table and optimal path from source to destination if they were specified in previous step otherwise it will just display the modified matrix and from main menu modified optimal path can be found. Also when a router is deleted it displays 'Router is down or unreachable' message.



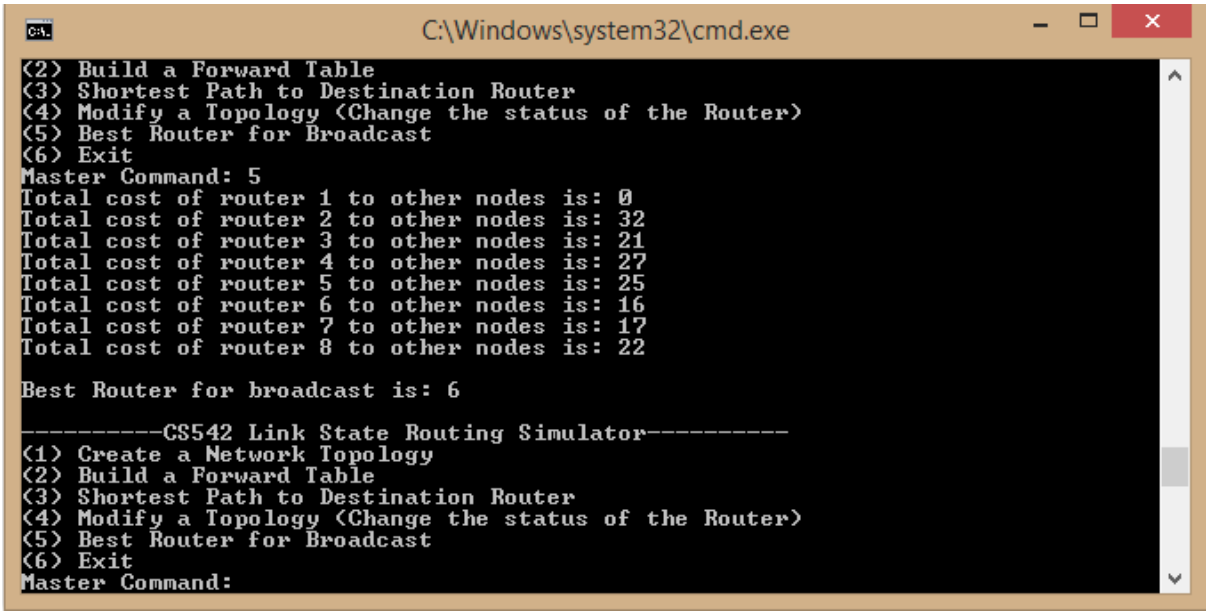
- If in previous step source and destination nodes were not provided it will display a message to select menu 3 for finding the shortest path.



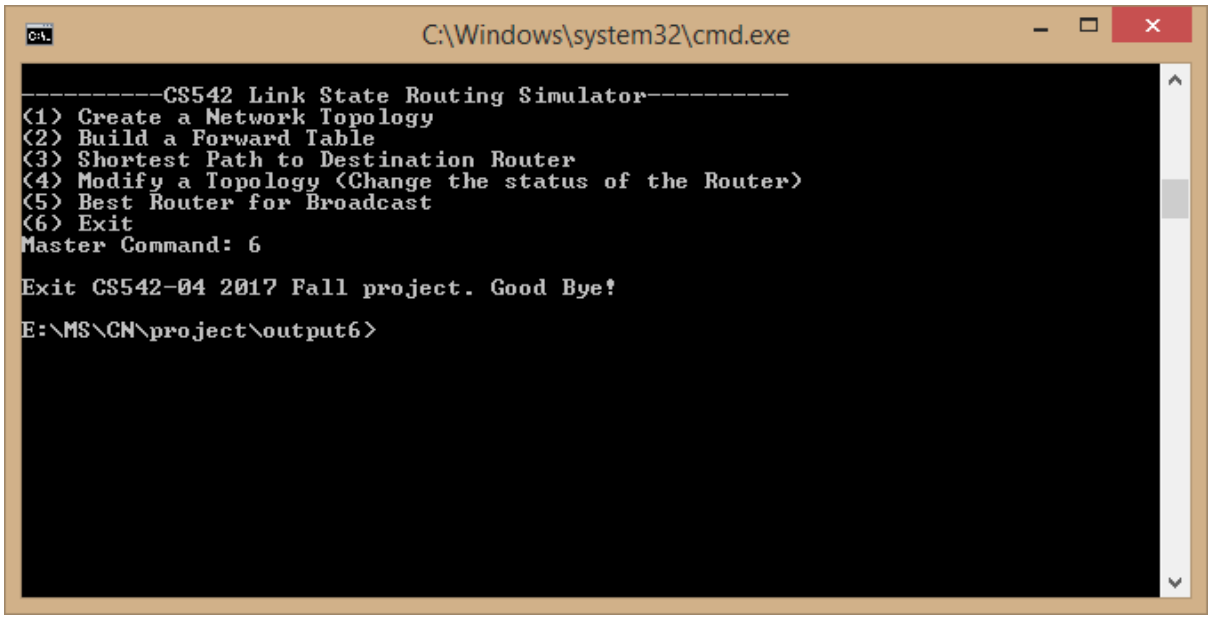
6. Option 5 will provide best router for broadcasting. First it displays total minimum cost from each router to other routers and then it selects the node with minimum cost.



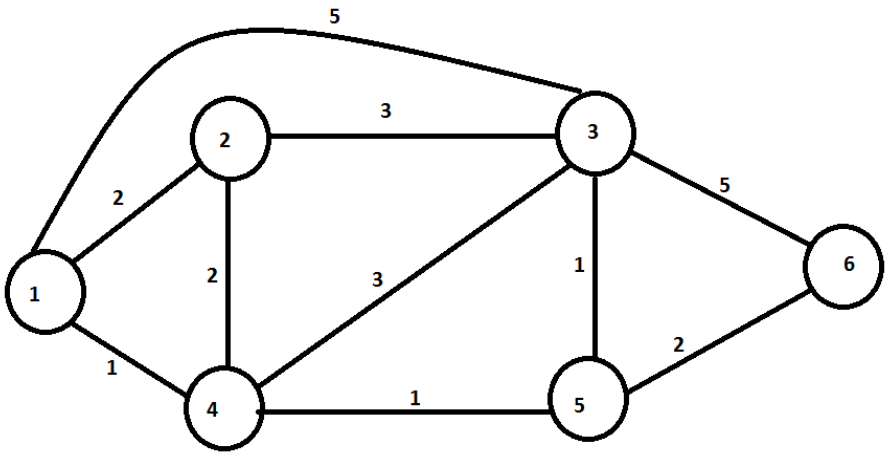
- Additionally, if node 1 is deleted, it is not considered in the network calculations. Hence cost shortest path changes and now node 6 is selected as the best router for broadcasting.



7. Option 6 can be used to exit the program.



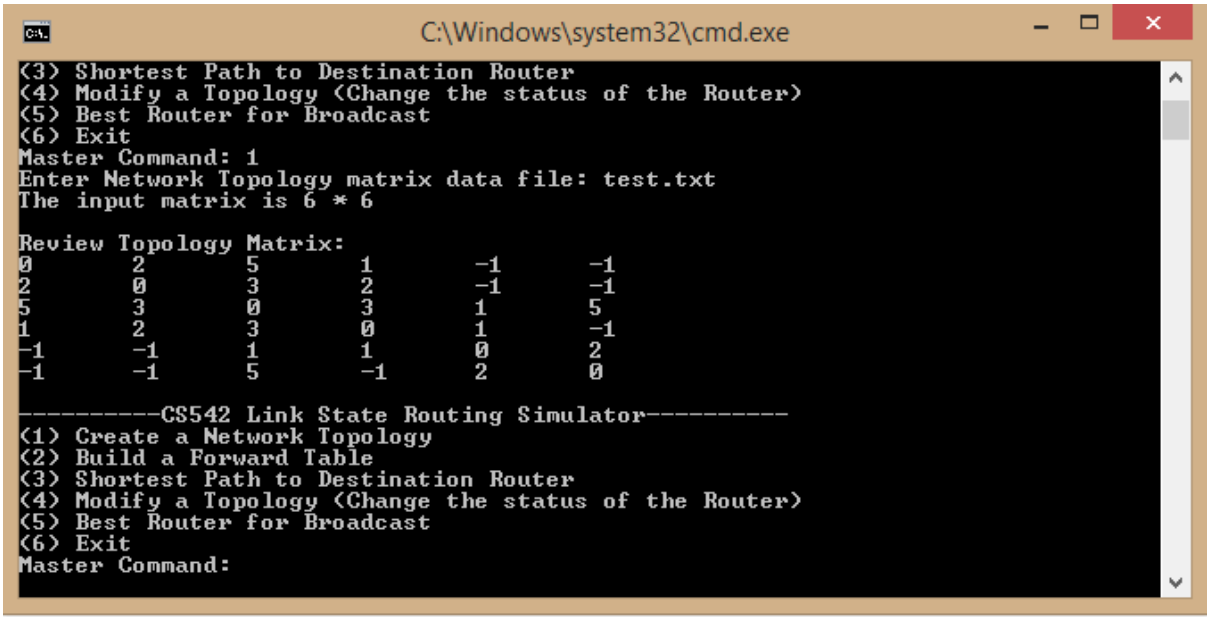
Test case 2



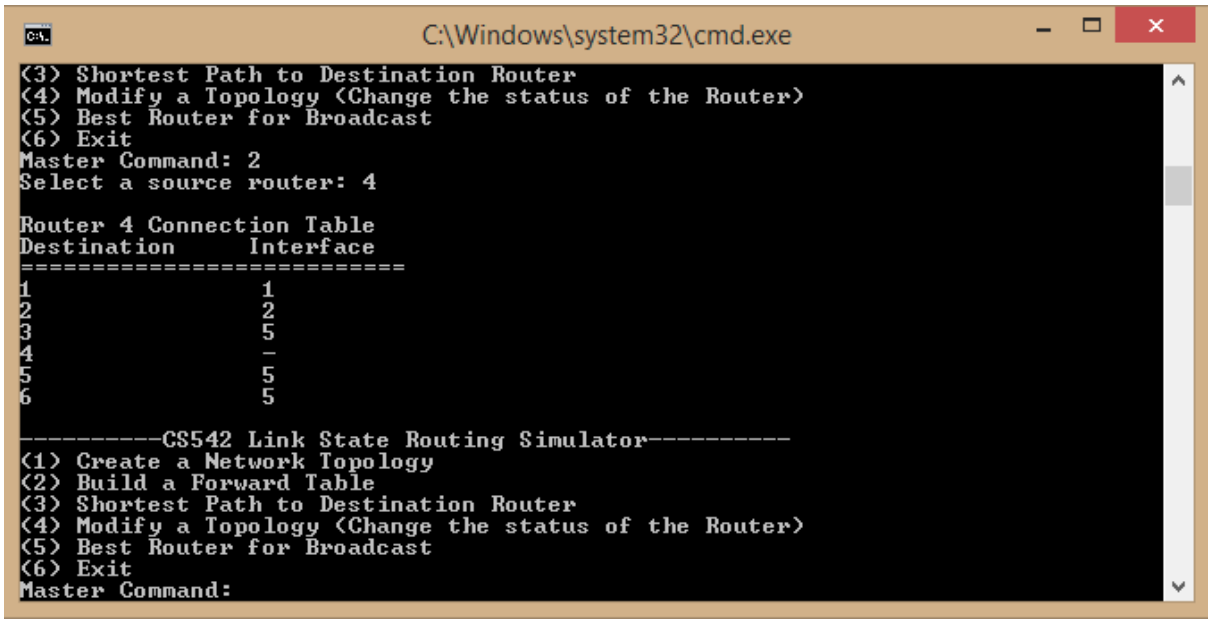
Topology Matrix:

```
 0  2 5  1 -1 -1
 2  0 3  2 -1 -1
 5  3 0  3  1  5
 1  2 3  0  1 -1
-1 -1 1  1  0  2
-1 -1 5 -1  2  0
```

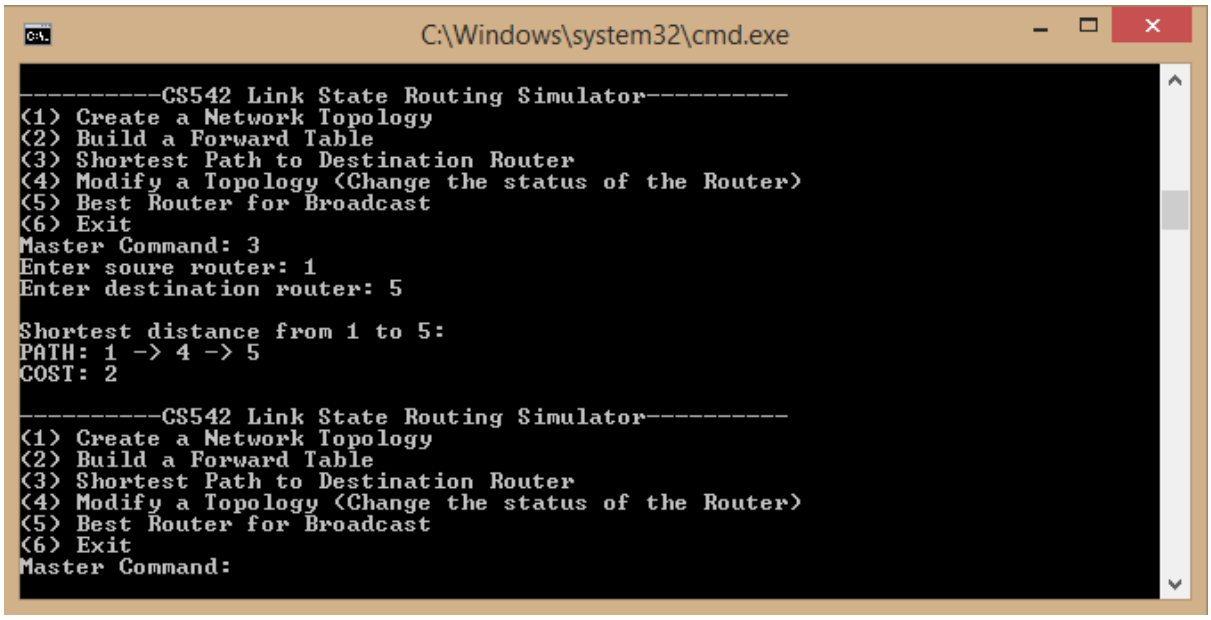
1. Selects an input matrix file



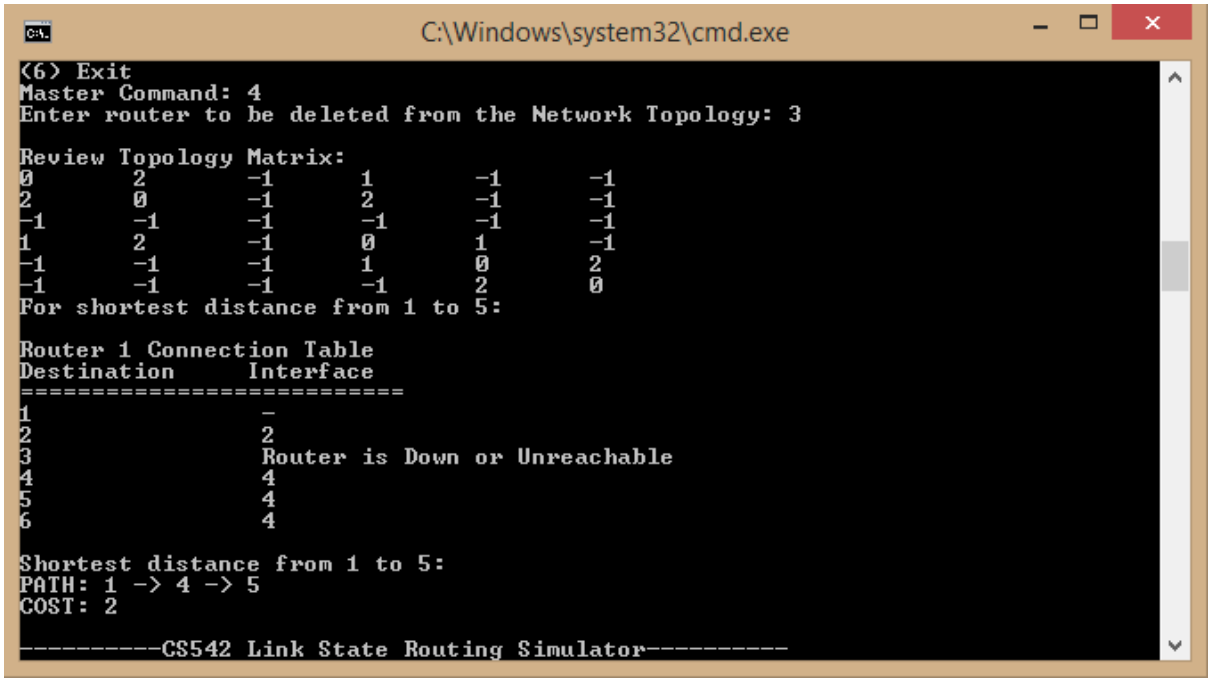
2. Builds a forwarding table for the specified router



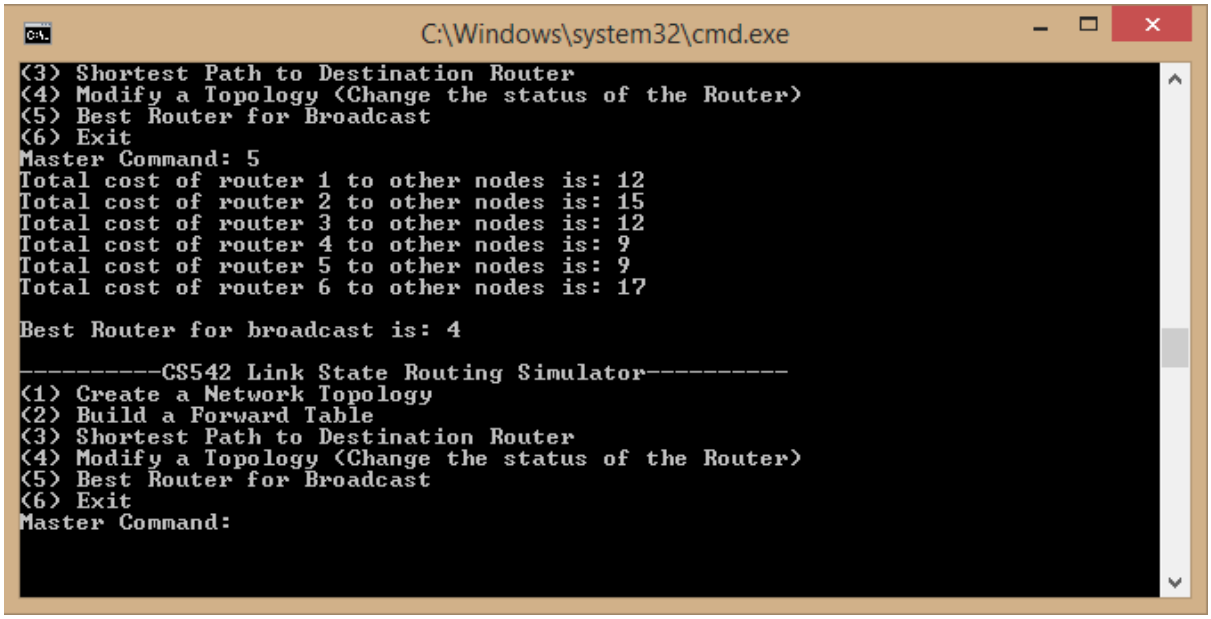
3. Finds the shortest distance path from source to destination router



4. Topology can be modified by deleting a router



5. Selects the router with minimum cost to other nodes for broadcasting.



CONCLUSION

The Link-State Routing protocol to find shortest distance is implemented successfully using Dijkstra's Algorithm.

REFERENCES

- Kurose and Ross, "Computer Networking a Top-Down Approach", Addisen-Wesley, 6th Edition.
- https://en.wikipedia.org/wiki/Link-state_routing_protocol
- https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm