PROJECT TITLE:

IMPLEMENTATION OF LINK-STATE ROUTING PROTOCOL PROJECT OPERATIONS MANUAL

Name: Mayur Piyushkumar Mehta
CWID: A20405901
Subject: CS-542 Computer Networks-1
Date: 11/20/2017

IMPLEMENTATION OF LINK – STATE ROUTING PROTOCOL

- **INTRODUCTION**

Link State is a routing protocol which is used in packet switching networks for computer communication. In the network that uses link state, every node will perform link - state; they construct a map of the connection between nodes in network by showing which nodes connected with which other nodes. After that each nodes can independently calculate the best path from it to other nodes in network.

Dijkstra's algorithm is called as single source shortest path Algorithm. It computes the least-cost path from one node (Source) to all other nodes in the network. Dijkstra's algorithm is iterative and has the property that after the k^{th} iteration of the algorithm, the least-cost paths are known k-destination nodes, and among the least-cost paths to all destination nodes, these k-paths will have the k smallest costs.

In this project a program is developed to implement Link - State routing protocol. The program should simulate the process of generating routing tables for each router in a given network and compute optimal path with least cost between any two particular given routers. The Topology modification, by making a router down in the network is also implemented. Dijkstra’s algorithm will be used to calculate the direction as well as the shortest path between two routers.

Java is used to implement the Dijkstra’s algorithm. In the code, several two - dimensional arrays are used to store original routing table, the distance between routers, values of distance during shortest path calculation, final table after calculation. Some integer variables are used to keep a count on the routers. The program interface display a menu to user who can Create a Network Topology, by choosing to input matrix of routing table from file, the program also help users calculate the shortest path between any couple of routers and display it to on the screen. Simple Command Line Interface is developed to calculate shortest path using Dijkstra’s Algorithm.

Routing protocol and its classification:

Routing protocol is response to the demand for dynamic routing tables in network. A routing protocol is a combination of rules and procedures that let routers in the internet inform other routers of change. It helps routers in the network know about the internet of their neighbour by sharing information.

In routing protocol we have interior protocol which handles intra domain routing and exterior protocol which handles inter-domain routing protocol. Link-state is one of the intra-domain routing protocol.
Figure: Representation

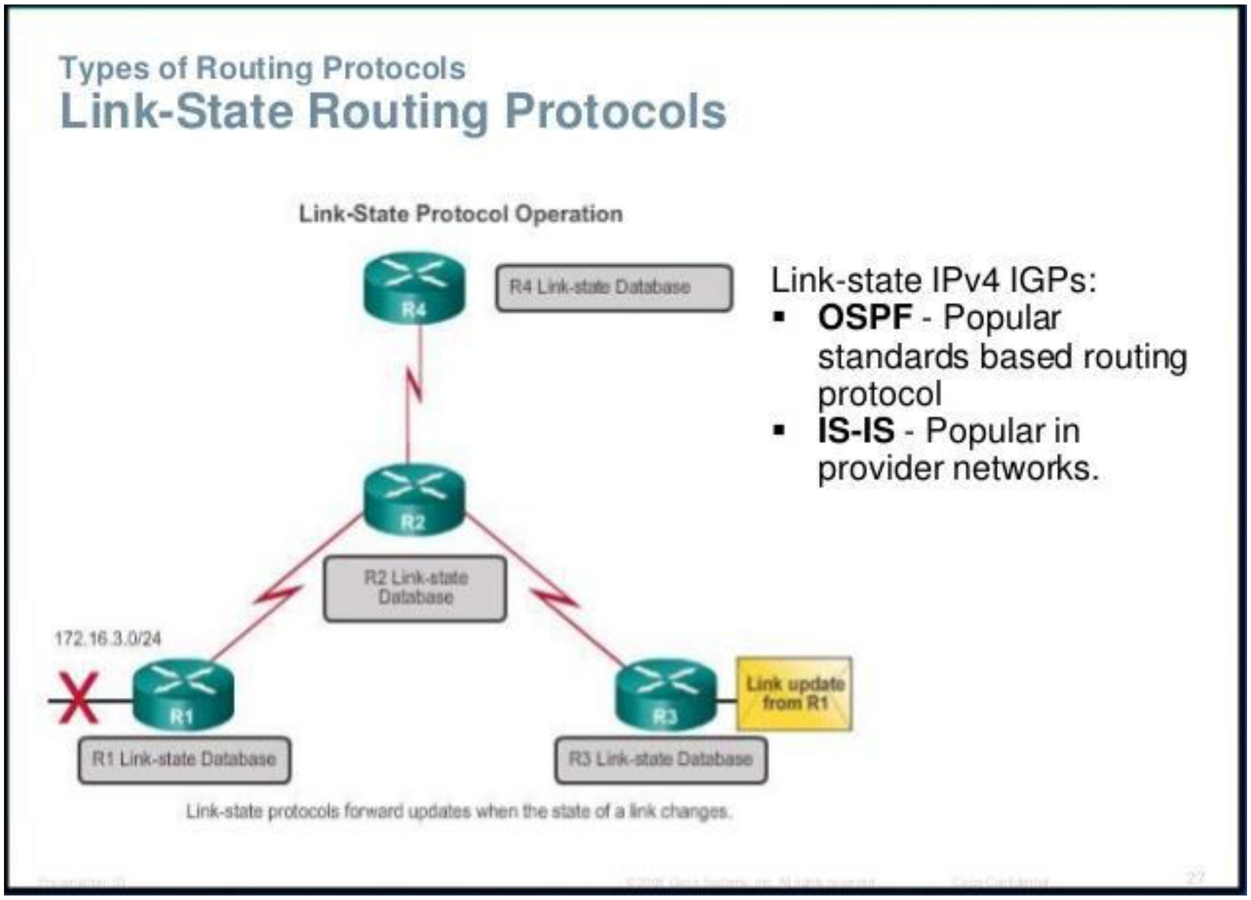

Types of Routing Protocols
Link-State Routing Protocols
Link-State Protocol Operation
R4
R4 Link-state Database
Link-state IPv4 IGPs:
▪ OSPF - Popular standards based routing protocol
▪ IS-IS - Popular in provider networks.
R2
R2 Link-state Database
172.16.3.0/24
R1
R1 Link-state Database
R3
Link update from R1
R3 Link-state Database
Link-state protocols forward updates when the state of a link changes.

Figure: Representation of Link – State Routing Protocol

ALGORITHM DESCRIPTION:

The method to calculate the optimal routing table and thus the shortest path using Dijkstra's algorithm and link state in project can be explained as follows:

We have a matrix of routing table that present the original value of metric between a router and its neighbour. At very first stage, each router does not know about connectivity of all others routers, each one just knows about its neighbour and its link, we use non-negative integer to present cost of a link and -1 for indirect connection. Router will uses the link-state routing table through four steps:

(i) Creating the state of link by each nodes,
(ii) Disseminating the link-state packet to other nodes,
(iii) Formatting shortest path for each node,
(iv)Calculating the routing table based on shortest path tree.

After each node has connectivity of other nodes it still does not have shortest path from itself to other one. At this time we use the Dijkstra's algorithm to calculate the shortest path:

(i) From table of n routers we will have n shortest path tree, each node will become the root in its shortest path tree.

(ii) Set the shortest path distance for all root neighbours to the cost between root and neighbour.

(iii) We continue search the nodes in the path select one with minimum shortest distances and add to the path.

(iv) Use the operation: $Di = minimu(Di\ , Dj + ci,j\ )$, we repeat above step until all nodes are added to the path.

(v) While having the shortest path tree we can create the routing table, we use 0 to represent the distance from root to root i.e. for same router say R1 → R1 has cost = 0. On the other hand it has positive cost values for two different nodes.

CODE:

➢ LinkStateRoutingPro.java

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.io.InputStreamReader;


public class LinkStateRoutingPro
{

                        static DijkstraPro d = new DijkstraPro();


                        public static int[][] ReadInput(String f1) throws Exception {

                              String RowText, row[];
                              int rCount = 0, cCount = 0, temp=0;

                              InputStream is = null;
     BufferedReader var1 = null;
                              try{
                                     is = new FileInputStream("C:/Users/Mayur
Mehta/Desktop/CS54204_2017F_Project_35_Mehta_Mayur/LinkStateRouting/topology.
txt");
        var1 = new BufferedReader(new InputStreamReader(is));
                                     while((RowText = var1.readLine())!=null)
                              {
                                     rCount++;
                                     row = RowText.split(" ");
                                     cCount = row.length;

                              }
                              var1.close();
                              }
                              catch (FileNotFoundException e) {
        e.printStackTrace();
     } catch (IOException e) {
        e.printStackTrace();
     }
```

```
                    int matrix[][] = new int[rCount][cCount];

                    InputStream is_new = null;
    BufferedReader var2 = null;
                    try{
                        is_new = new FileInputStream("C:/Users/Mayur
Mehta/Desktop/CS54204_2017F_Project_35_Mehta_Mayur/LinkStateRouting/topology.
txt");
      var2 = new BufferedReader(new InputStreamReader(is_new));
                        while ((RowText = var2.readLine()) != null)
                    {
                        row = RowText.split(" ");
                        int i=0;
                        while(i<cCount)
                        {
                            matrix[temp][i] = Integer.parseInt(row[i]);
                            i++;
                        }
                        temp++;
                    }

                    var2.close();
                    }
                    catch (FileNotFoundException e) {
      e.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
    }

                    System.out.println("Network Topology:");
                    return matrix;
                }

  public static void ShowMatrix(int matrix[][])
  {

                for(int i=0;i<matrix.length;i++)
                {
                    for(int j=0;j<matrix.length;j++)
                    {
                        System.out.print(matrix[i][j] + " ");
                    }
                    System.out.println();
                }
                System.out.println("Total Routers :"+matrix.length);
    }


                                public static void ShowForwardTable(int matrix[][])
                                {
                                        Scanner scnew=new Scanner(System.in);
                                        System.out.println("Enter the router value");

                                        int r = scnew.nextInt();
                                        System.out.println("Dest\tInterface");
                                        int i=0;
                                        while(i<matrix.length)
                                        {
                                                if(i!=r-1);
                                                {
                                                        System.out.print("R"+(i+1));
                                                        DijkstraPro.d(matrix, r-1, i, 1);
                                                }
                                                i++;
                                        }

                                }

                                public static void ShowShortestPath(int matrix[][]) throws
IOException
                                {
                                        System.out.println("Enter the source router");
                                        Scanner scnew1 =new Scanner(System.in);
                                        int source = Integer.parseInt(scnew1.nextLine()) - 1;

                                        System.out.println("Enter the destination router");
                                        int destination = Integer.parseInt(scnew1.nextLine()) - 1;
                                        DijkstraPro.d(matrix,source,destination,2);

                                }


                                public static void ModifyRouter(int[][] matrix) throws
IOException{
                                        Scanner scnew2 = new Scanner(System.in);
                                        System.out.println("Enter router to be deleted to modify
topology");
                                        int r = scnew2.nextInt();

                                        for(int i=0;i<matrix.length; i++)
                                        {
```

```
                if(i==r-1)
                {
                for(int j=0;j<matrix.length;j++)

                {
                        matrix[i][j] = -1;
                        matrix[j][i] = -1;
                }
                }
        }
        System.out.println("\nNew Topology will be:");
        for(int i=0;i<matrix.length;i++)
{
        for(int j=0;j<matrix.length;j++)
        {
                System.out.print(matrix[i][j] + " ");
        }
        System.out.println();
}
int h=matrix.length;
        int k=h-1;
        System.out.println("\nTotal Routers:"+k);
        System.out.println("\nTo calculate shortest path\n");
        ShowShortestPath(matrix);

}
public static void BestRouter(int[][] matrix) throws IOException
{
        int[] new_dist;
        int[] broadcast_matrix;
        int mini;
        int max=Integer.MAX_VALUE;
        int best_router=-1;
        for(int i=0;i<matrix.length;i++)
        {
                for(int j=0;j<matrix.length;j++)
                {
                        int broadcast_matrix;

broadcast_matrix=DijkstraPro.d(matrix,i,j,2);
                }

        }

        for(int k=0;k<broadcast_matrix;k++)
        {
```

```
                                        if(broadcast_matrix[k]<max)
                                        {
                                                new_dist=broadcast_matrix[k];
                                        }

                                }
                                System.out.println("\nBest Router to broadcast
is"+new_dist);
                                System.out.println("Cost is:"+best_router);
                        }

                        public static void main(String[] args) throws Exception
                        {

                                int matrix[][]=null;

                                System.out.println("------------CS542 Link State Routing
Simulator---------------");
                                System.out.println("------------Select from the following----
--------");
                                System.out.println("1. Create a Network Topology");
                                System.out.println("2. Build a Forward Table");
                                System.out.println("3. Shortest path to Destination router");
                                System.out.println("4. Modify a Topology -(Delete any
router)");
                                System.out.println("5. Best Router for Broadcast");
                          System.out.println("6. Exit");

                                Scanner sc = new Scanner(System.in);
                                int c = sc.nextInt();


                                while(c!=6)
                                {
                                        switch (c)
                                        {
                                        case 1:
                                                  System.out.println("Enter input filename
here");
                                           String f1 = sc.next();
                                           matrix = ReadInput(f1);
                                           ShowMatrix(matrix);

                                           break;

                                        case 2:
```

```
                                        ShowForwardTable(matrix);

                                        break;

                                case 3:
                                        ShowShortestPath(matrix);

                                        break;

                                case 4:
                                        ModifyRouter(matrix);

                                        break;

                                case 5:
                                        BestRouter(matrix);
                                        break;



                                default:System.out.println("Invalid. re-enter your
        choice");
                                }

                        System.out.println("------------CS542 Link State Routing
        Simulator---------------");
                        System.out.println("------------Select from the following----
        --------");
                        System.out.println("1. Create a Network Topology");
                        System.out.println("2. Build a Forward Table");
                        System.out.println("3. Shortest path to Destination router");
                        System.out.println("4. Modify a Topology -(Delete any
        router)");
                        System.out.println("5. Best Router for Broadcast");
                      System.out.println("6. Exit");

                                c=sc.nextInt();

                        }

                        System.out.println("Finished");
                        sc.close();

                }
        }
```

➢ DijkstraPro.java

```
public class DijkstraPro
{

	public static Object d(int[][] matrix, int source , int destination ,int c)
	{


		int[] dis = new int[matrix.length];					//
distance[] array will maintain shortest

				//distance between 2 routers
		int[] vis = new int[matrix.length];
	//visited[] array will keep track of visited

					//routers to avoid same router to be considered for shortest
path calculation
		int TempNext = source;
			// An index to predecessor array

			int j=0;
			while(j<matrix[0].length)
			{
				dis[j] = matrix[source][j];
				j++;
			}

		int[] pre = new int[matrix[0].length];				//initialize
predecessor that will help further to traverse path

			int i=0;
			while(i<pre.length)
			{
				pre[i] = source;
				i++;
			}
		vis[TempNext] = 1;
	// initially visited of source make it as true
```

```
                for(int x=0;x<matrix[0].length; x++)
        //sort distance array based on minimum path weight, to determine shortest path between
routers //calculation
                {
                        int min = Integer.MAX_VALUE;

                        for (int l = 0; l < dis.length; l++)
                        {

                                if (vis[l]!=1 && l != source && dis[l] != -1)
                                {

                                        if (dis[l] < min)
                                        {
                                                min = dis[l];
                                                TempNext = l;
                                        }
                                }
                        }

                        if (TempNext == destination)

                        {
                                break;
                        }

                        vis[TempNext] = 1;
                        int p=0;                        //int i=0;
                        while(p<dis.length)
                        {
                                if(vis[p] != 1 && dis[p] == -1 && matrix[TempNext][p] != -1)
                                //handle routers with -1 weight means routers having no
link/connectivity
                                        {
                                                dis[p] =  matrix[TempNext][p] + dis[TempNext];
                                                pre[p] = TempNext;
                                        }

                                        else if(matrix[TempNext][p] != -1 && dis[TempNext] >
min+matrix[TempNext][p])
                                        {
                                                dis[p] =  matrix[TempNext][p] + dis[TempNext];
                                                pre[p] = TempNext;
                                        }
                                        p++;
                        }
```

```
		}


		/*Hops in to this when user opts for creating shortest path option 3. User provided
source and
		 * destination values are passed further to output minimum path*/

		if(c==2 )
		{
			Traversal(pre, source, destination, dis.length,1);
			System.out.println();
			int total = dis[destination] - dis[source];
			if(total == -1)
									// when user request for the router
that is down or removed from the topology
			{
				System.out.println("Router is down, its not working. Please choose
different router");
			}
			else
			{
			System.out.println("Total cost will be =  "+ total);
			}
	    }

		/*To determine connection table for the router user has requested*/

		if(c==1)
		{
			Traversal(pre, source, destination, dis.length, 2);
		}
		else return null;

		return null;
  }


	/*This function output's shortest path between two routers and it also populates
connection router table */

	public static void Traversal(int[] pre, int source, int destination, int length, int c)
	{

		int current = 1;
		int[] edgePath = new int[length];
		int i = destination;
		edgePath[0] = i;


		boolean flag=false;

		while (pre[i] != source)				// aligns predecessor routers to
edgePath[] array in order to output the path
		{
			i = pre[i];
			edgePath[current] = i;
			current++;
		}
		edgePath[current] = source;

		if(c==1)
							// triggered when user wants to output the shortest
path between source and destination
		{
			System.out.print("Shortest path from " + (source + 1) + " to " +
(destination + 1));
			System.out.println();
			int k = current;
			while(k>0)
			{
				System.out.print("R" + (edgePath[k] + 1) + " to ");
				k--;
			}

			System.out.print("R" + (edgePath[0] + 1)) ;
		}


		if(c==2)
						// triggered when user wants to get connection table for
specified router
		{

			if(current>0)
			{
				int x = current-1;
				while(x>0)
				{
					System.out.print("\tR" + (edgePath[x] + 1) + "\n");
					flag=true;
					x--;
```

```
                        }

                }

                if(flag==false)
                {
                        System.out.print("\tR" + (edgePath[0] + 1)+"\n") ;
                }

        }
    }
}
```

DESIGN AND WORKFLOW:

The Design of the Code is in Java. It has a Simple Command Line Interface.
The design of code is menu driven and works on user inputs.
The Application consists of 2 Java Source file, which contains the implementation of all the functionalities of the program.

The detailed design of the source code as follows:

When LinkStateRoutingPro.java is compiled and run, the application provides the user with the following functionalities, which the user has to select.

The Functionalities are:

1. Create a Network Topology
2. Build a Forward Table
3. Shortest Path to Destination Router

4. Modify a Topology – (Delete any Router)
5. Best Router for Broadcast
6. Exit

The user is prompted to select his option. Upon selecting the corresponding options, the code executes the corresponding functionality in the program based on the user input.

- ✓ When the user selects option 1, The **ShowMatrix()** Method is invoked and it is used to read the matrix inside the input Topology text file. The input Matrix which is read is then displayed on the Console.

- ✓ When the user selects option 2, The **ShowForwardTable()** method is invoked and the user is prompted to "Enter the Source Router :". Once the Source router id is invoked from the User, it computes the shortest path to every other node using Dijkstra's Algorithm using the **Object d()** method, and the Router Forward table is displayed on the console.

- ✓ When the user select option 3, The **ShowShortestPath()** method is invoked which implements the functionality of finding the Shortest path with the least cost available. This requires the user to "Enter the Destination Router ID", using which the method computes the cost and the complete shortest path using Dijkstra's Algorithm.

- ✓ When the user selects option 4, This implements the "Modify Topology" feature, where in the user is prompted to "Enter the Router ID to remove or make it Down". When the user enters the Router ID to be made down, then **ModifyRouter()** method is invoked and this makes the selected router down. i.e it make it unreachable in the network to other nodes. The value of this router will be "-1" to signify that it is unreachable.

- ✓ When the user selects option 5, This implements the "Best Router" feature, where the best router to broadcast from is displayed. When the user selects this option **BestRouter()** method is invoked and router that computes least cost with every router is computed.

- ✓ When user selects option 6, this is the Exit functionality, and it breaks the code flow and exits the application.

==

<u>USER MANUAL</u>

The Folder Structure of the Project is as Follows:
The "**CS542Project_35_Mehta_Mayur.zip**" contains the Entire Project Folder "LinkStateRouting".

1) LinkStateRouting → This folder contains **2** subfolders →
a) "Src" → This contains the Source code file "**LinkStateRoutingPro.java**" and "**DijikstraPro.java"** and also the Compiled Java Bytecodes
"**LinkStateRoutingPro.class**" & "**DijkstraPro.class**" files within it.

b) "Topology" → This contains the Text file "**Topology.txt".**

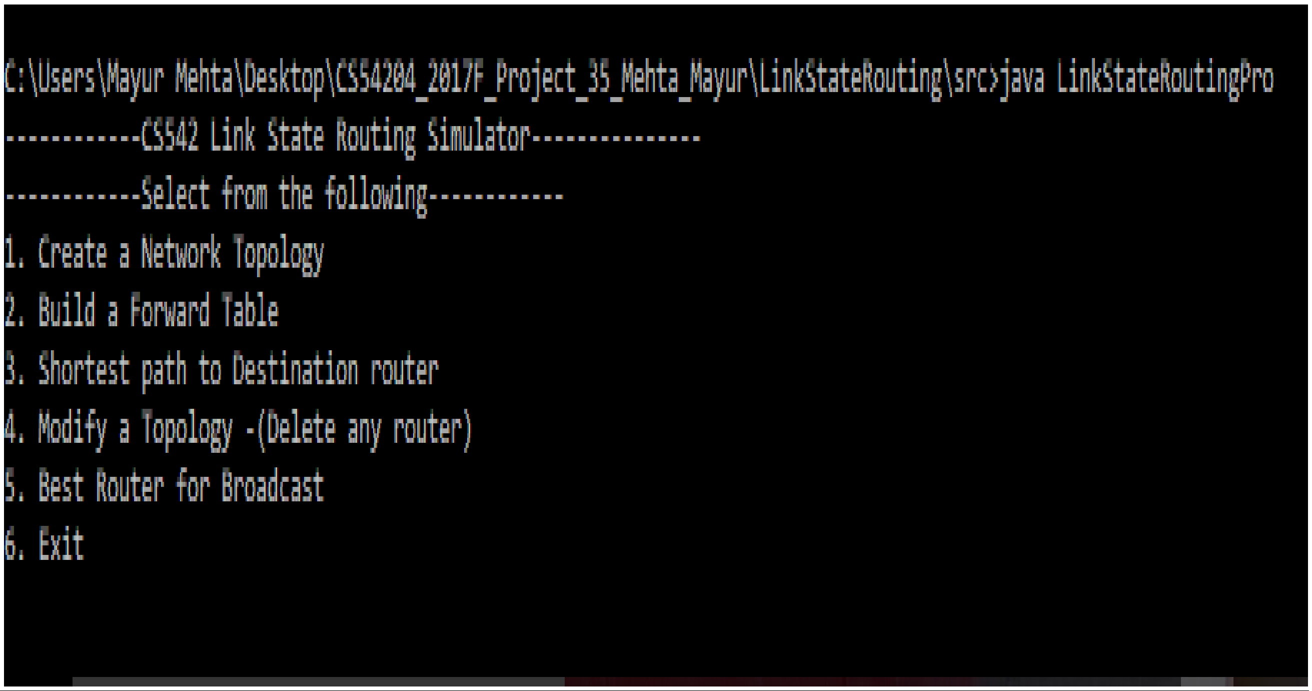
c) Link_State Routing_PPT → Project Presentation File

d) Link_State_Routing_REPORT → Project Report and Operations manual.

Instructions to run:

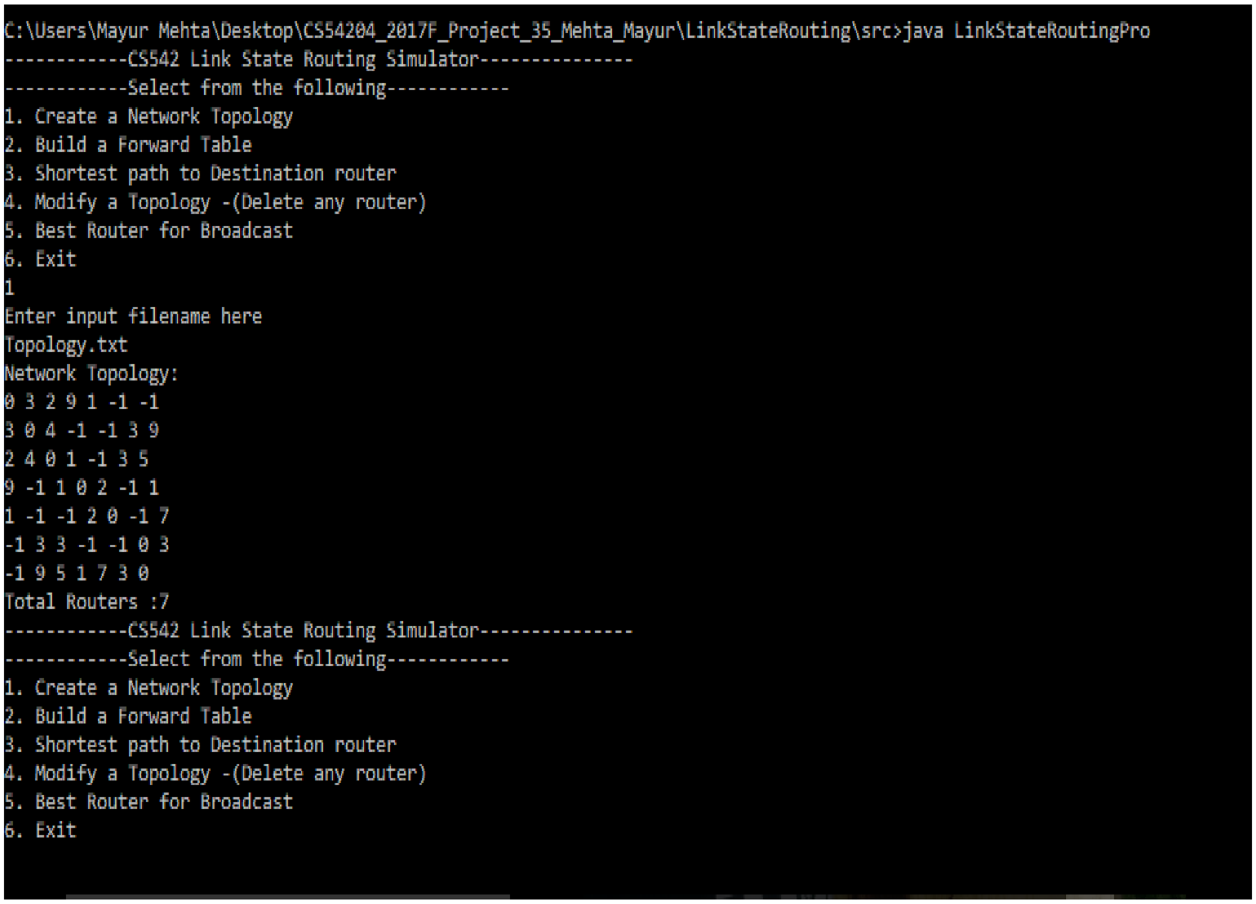
- ✓ From the Command prompt/Terminal, Navigate into the respective folder.
- ✓ Compile both java files using Javac *.java command.
- ✓ Run the LinkStateRoutingPro.java application by using following command – java LinkStateRoutingPro
- ✓ The input file has to be in the same location as the source class" and jar files are present. If new input file has to be tested, it has to be copied to the same location as the source, class and jar files.

SNAPSHOTS



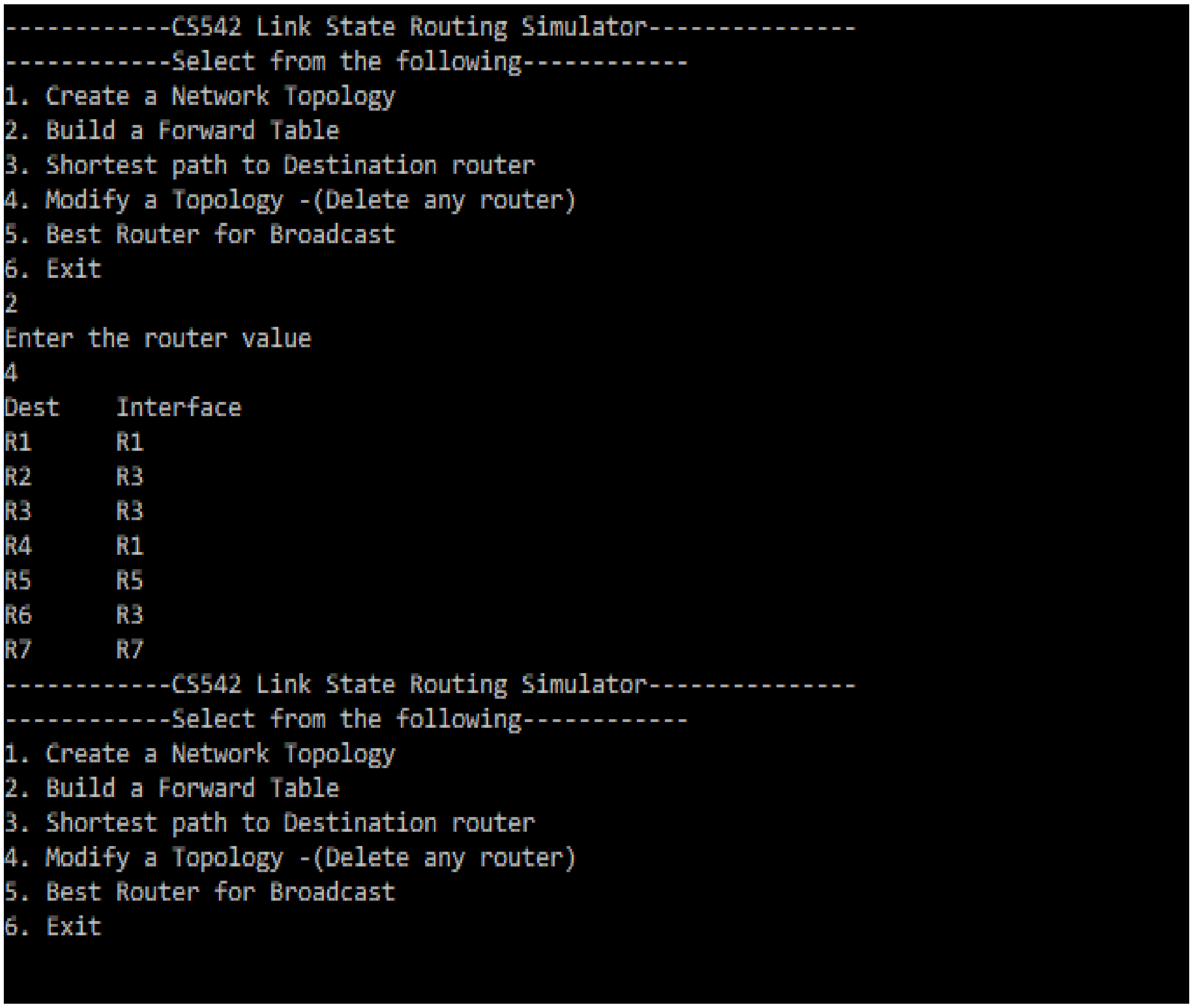
```
C:\Users\Mayur Mehta\Desktop\CS54204_2017F_Project_35_Mehta_Mayur\LinkStateRouting\src>java LinkStateRoutingPro
-----------CS542 Link State Routing Simulator--------------
-----------Select from the following-----------
1. Create a Network Topology
2. Build a Forward Table
3. Shortest path to Destination router
4. Modify a Topology -(Delete any router)
5. Best Router for Broadcast
6. Exit
```

Interface Design Link State Routing Simulator



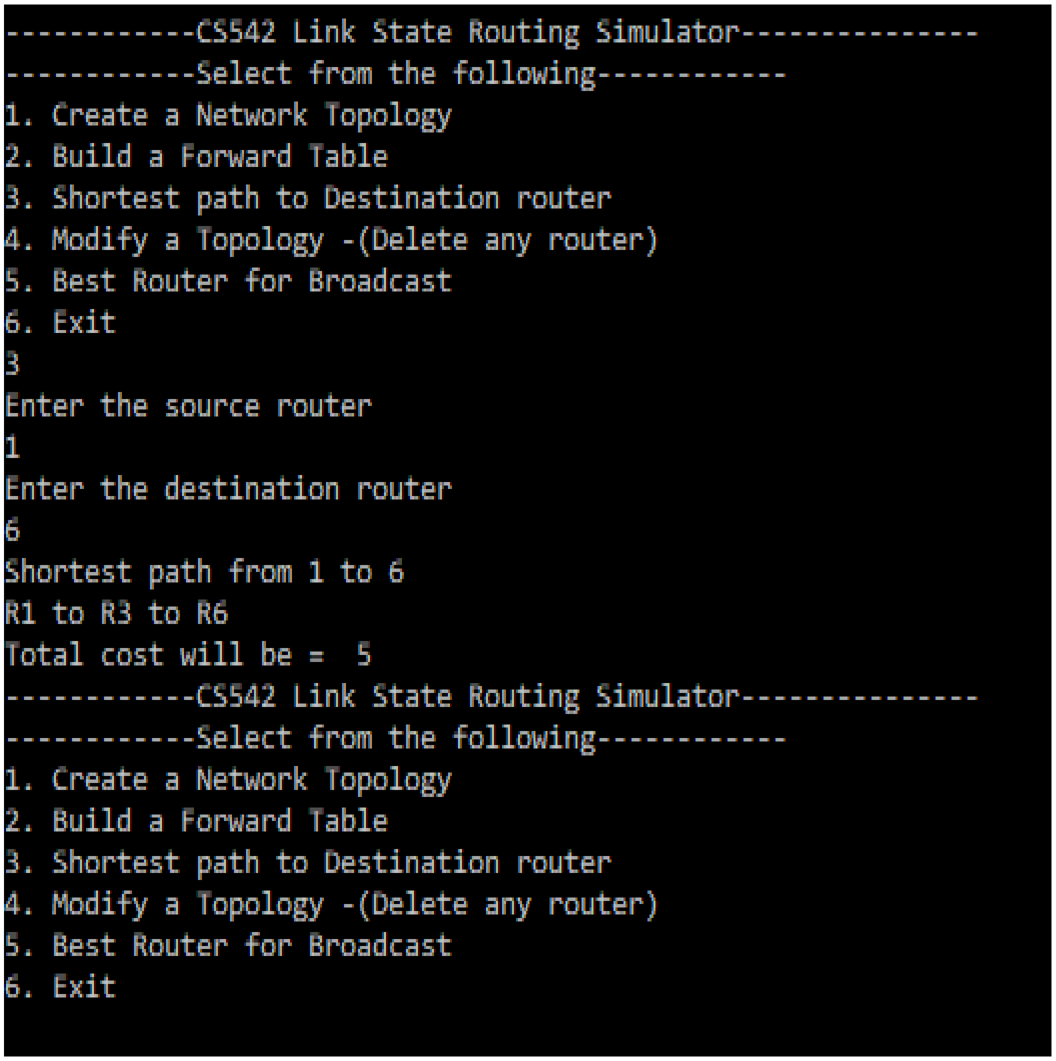
```
C:\Users\Mayur Mehta\Desktop\CS54204_2017F_Project_35_Mehta_Mayur\LinkStateRouting\src>java LinkStateRoutingPro
------------CS542 Link State Routing Simulator---------------
------------Select from the following------------
1. Create a Network Topology
2. Build a Forward Table
3. Shortest path to Destination router
4. Modify a Topology -(Delete any router)
5. Best Router for Broadcast
6. Exit
1
Enter input filename here
Topology.txt
Network Topology:
0 3 2 9 1 -1 -1
3 0 4 -1 -1 3 9
2 4 0 1 -1 3 5
9 -1 1 0 2 -1 1
1 -1 -1 2 0 -1 7
-1 3 3 -1 -1 0 3
-1 9 5 1 7 3 0
Total Routers :7
------------CS542 Link State Routing Simulator---------------
------------Select from the following------------
1. Create a Network Topology
2. Build a Forward Table
3. Shortest path to Destination router
4. Modify a Topology -(Delete any router)
5. Best Router for Broadcast
6. Exit
```

When User Selects option 1:
Topology.txt is called
Matrix is printed.



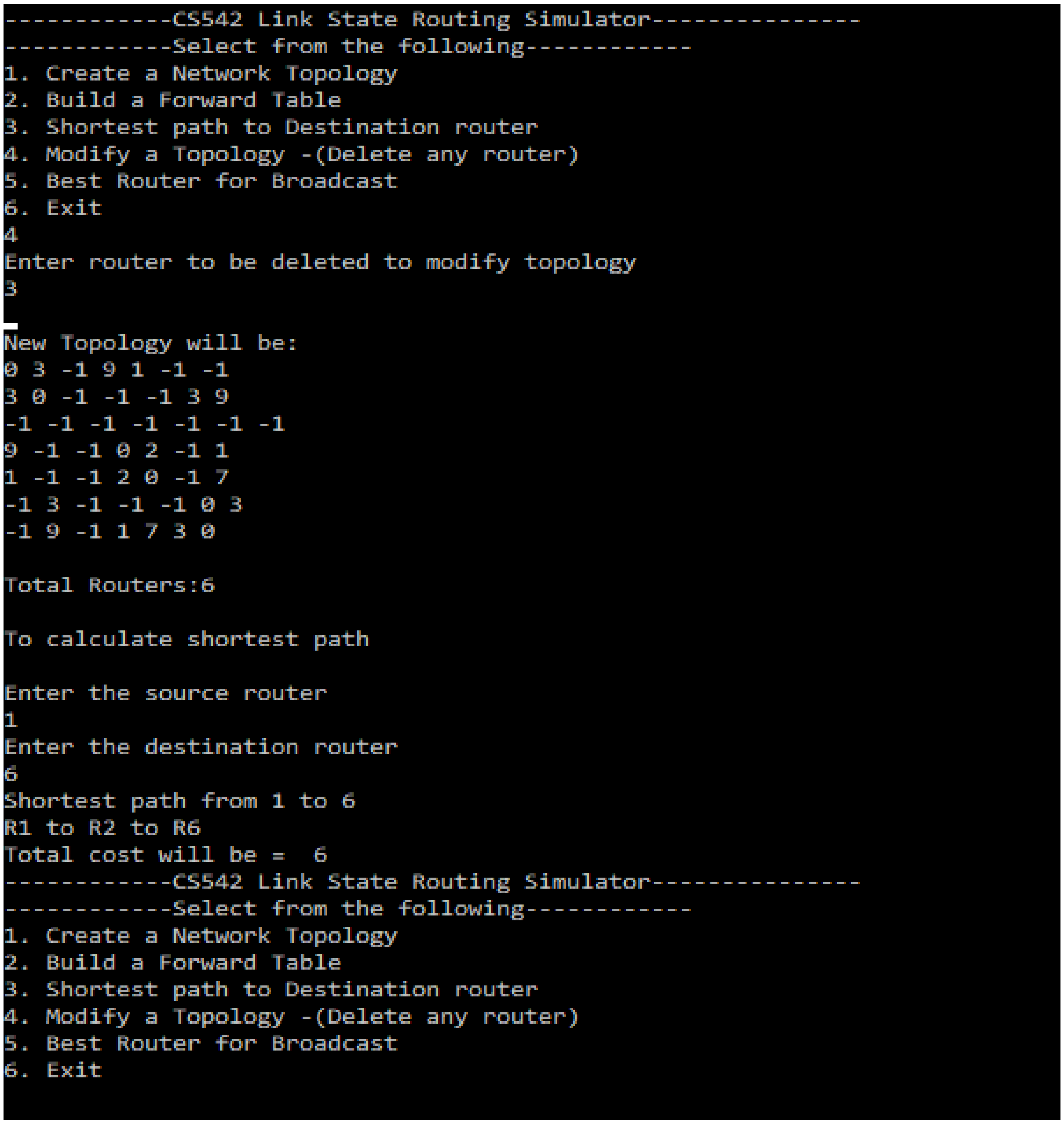
```
------------CS542 Link State Routing Simulator---------------
------------Select from the following------------
1. Create a Network Topology
2. Build a Forward Table
3. Shortest path to Destination router
4. Modify a Topology -(Delete any router)
5. Best Router for Broadcast
6. Exit
2
Enter the router value
4
Dest    Interface
R1      R1
R2      R3
R3      R3
R4      R1
R5      R5
R6      R3
R7      R7
------------CS542 Link State Routing Simulator---------------
------------Select from the following------------
1. Create a Network Topology
2. Build a Forward Table
3. Shortest path to Destination router
4. Modify a Topology -(Delete any router)
5. Best Router for Broadcast
6. Exit
```

When User selects Option:-2
Source router is asked
Forward Table of respective router is displayed.



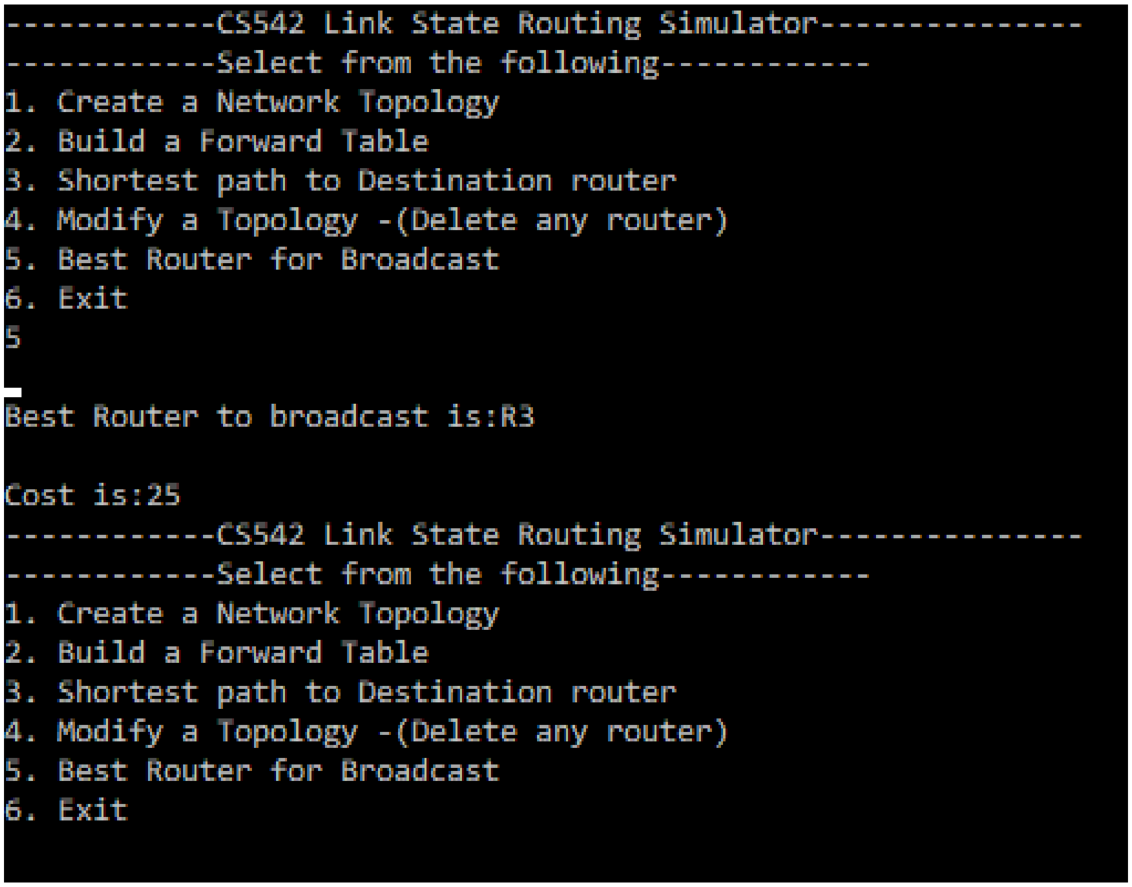
```
------------CS542 Link State Routing Simulator---------------
------------Select from the following------------
1. Create a Network Topology
2. Build a Forward Table
3. Shortest path to Destination router
4. Modify a Topology -(Delete any router)
5. Best Router for Broadcast
6. Exit
3
Enter the source router
1
Enter the destination router
6
Shortest path from 1 to 6
R1 to R3 to R6
Total cost will be =  5
------------CS542 Link State Routing Simulator---------------
------------Select from the following------------
1. Create a Network Topology
2. Build a Forward Table
3. Shortest path to Destination router
4. Modify a Topology -(Delete any router)
5. Best Router for Broadcast
6. Exit
```

When User selects option:- 3
Source and Destination routers are entered
Shortest path is displayed between respective routers.



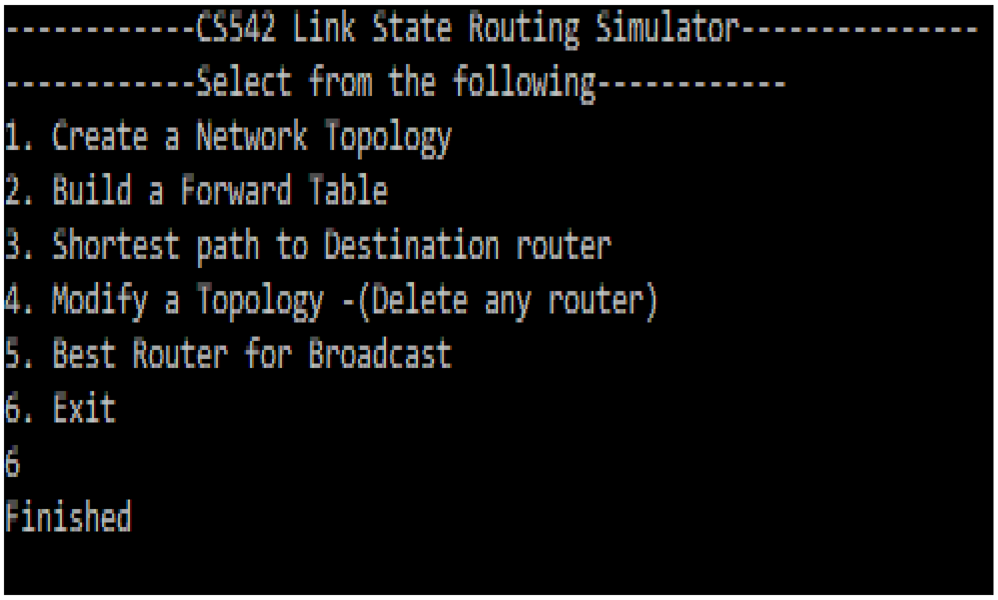
```
------------CS542 Link State Routing Simulator---------------
------------Select from the following------------
1. Create a Network Topology
2. Build a Forward Table
3. Shortest path to Destination router
4. Modify a Topology -(Delete any router)
5. Best Router for Broadcast
6. Exit
4
Enter router to be deleted to modify topology
3

New Topology will be:
0 3 -1 9 1 -1 -1
3 0 -1 -1 -1 3 9
-1 -1 -1 -1 -1 -1 -1
9 -1 -1 0 2 -1 1
1 -1 -1 2 0 -1 7
-1 3 -1 -1 -1 0 3
-1 9 -1 1 7 3 0

Total Routers:6

To calculate shortest path

Enter the source router
1
Enter the destination router
6
Shortest path from 1 to 6
R1 to R2 to R6
Total cost will be =  6
------------CS542 Link State Routing Simulator---------------
------------Select from the following------------
1. Create a Network Topology
2. Build a Forward Table
3. Shortest path to Destination router
4. Modify a Topology -(Delete any router)
5. Best Router for Broadcast
6. Exit
```

When User selects option 4:
User inputs router to be deleted
New modified path is displayed of respective routers.



```
------------CS542 Link State Routing Simulator---------------
------------Select from the following------------
1. Create a Network Topology
2. Build a Forward Table
3. Shortest path to Destination router
4. Modify a Topology -(Delete any router)
5. Best Router for Broadcast
6. Exit
5

Best Router to broadcast is:R3

Cost is:25
------------CS542 Link State Routing Simulator---------------
------------Select from the following------------
1. Create a Network Topology
2. Build a Forward Table
3. Shortest path to Destination router
4. Modify a Topology -(Delete any router)
5. Best Router for Broadcast
6. Exit
```

When User selects option 5:
Best Router to Broadcast with minimal cost is displayed.



```
------------CS542 Link State Routing Simulator---------------
------------Select from the following------------
1. Create a Network Topology
2. Build a Forward Table
3. Shortest path to Destination router
4. Modify a Topology -(Delete any router)
5. Best Router for Broadcast
6. Exit
6
Finished
```

When user selects option 6:
Program gets exit.

REFERENCES:

https://en.wikipedia.org/wiki/Link-state_routing_protocol
https://en.wikipedia.org/wiki/Dijkstra's_algorithm
https://en.wikipedia.org/wiki/Routing
http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/dijkstraAlgor.htm
http://www.danscourses.com/CCNA-2/link-state-routingprotocols.html
http://www.ciscopress.com/articles/article.asp?p=2180210&seqNum=7

===