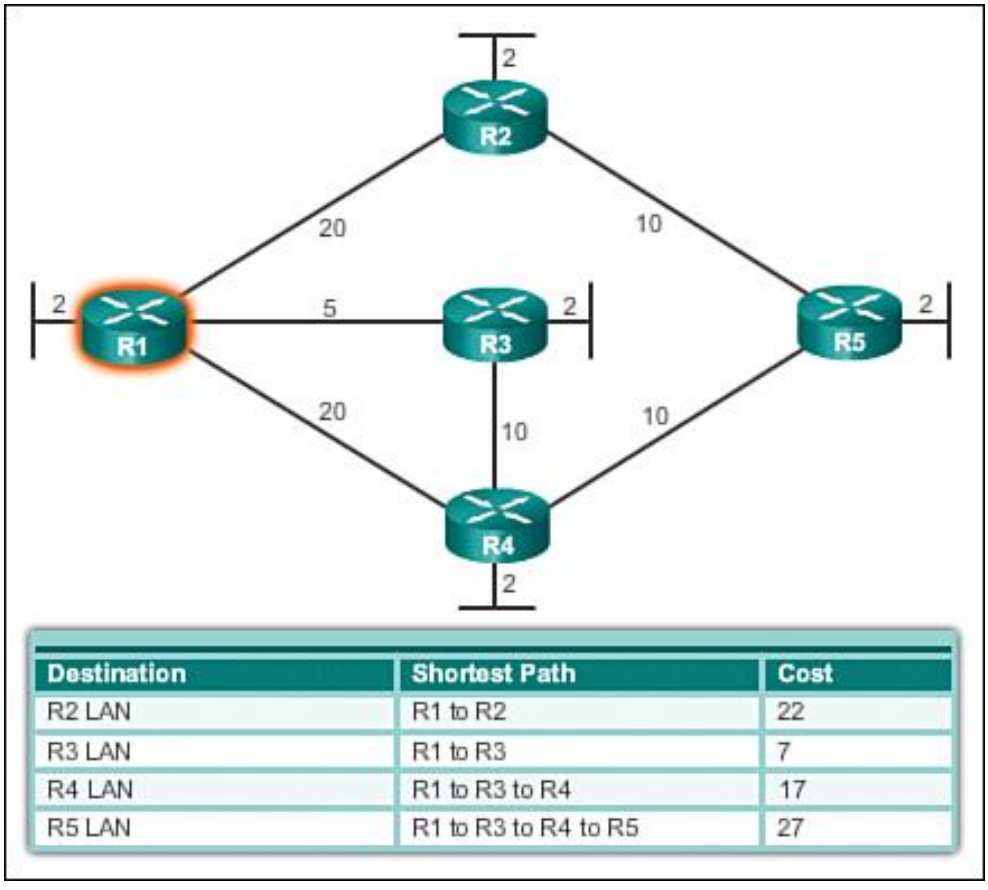
OBJECTIVE - 2

Implementation of Link – State Routing Protocol

Link state routing is a dynamic routing protocol in which each router shares knowledge of its neighbors with every other router in the network.

- The basic concept of link-state routing is that every node constructs a map of the connectivity to the network, in the form of a graph, showing which nodes are connected to which other nodes.
- Each node then independently calculates the next best logical path from it to every possible destination in the network.
- Each collection of best paths will then form each node's routing table.



Destination	Shortest Path	Cost
R2 LAN	R1 to R2	22
R3 LAN	R1 to R3	7
R4 LAN	R1 to R3 to R4	17
R5 LAN	R1 to R3 to R4 to R5	27

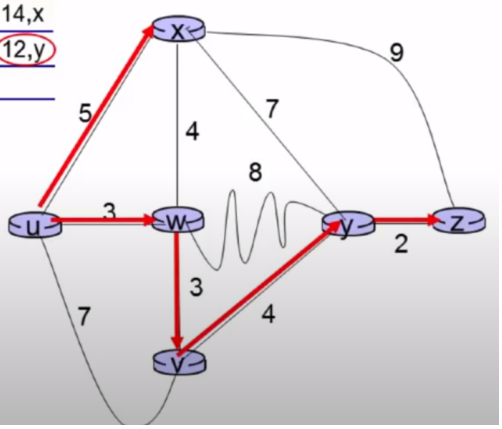
Dijkstra's algorithm is called a single source shortest path Algorithm. It computes the least-cost path from one node (Source) to all other nodes in the network. Dijkstra's algorithm is iterative and has the property that after the kth iteration of the algorithm, the least-cost paths are known k-destination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

Notations used:

- **c(i, j):** Link cost from node i to node j. If i and j nodes are not directly linked, then c(i, j) = ∞.
- **D(v):** It defines the cost of the path from source code to destination v that has the least cost currently.
- **P(v):** It defines the previous node (neighbor of v) along with the current least-cost path from source to v.
- **N:** It is the total number of nodes available in the network.

```
Initialization
N = {A}     // A is a root node.
for all nodes v
if v adjacent to A
then D(v) = c(A,v)
else D(v) = infinity
loop
find w not in N such that D(w) is a minimum.
Add w to N
Update D(v) for all v adjacent to w and not in N:
D(v) = min(D(v) , D(w) + c(w,v))
Until all nodes in N
```

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	7,u	3,u	5,u	∞	∞
1	uw	6,w		5,u	11,w	∞
2	uwx	6,w			11,w	14,x
3	uwxv				10,v	14,x
4	uwxvy					12,y
5	uwxvyz					



Java Code:

Dijkstra.java:

```
/* Implementation of Link State Routing Protocol - Dijkstra's Algorithm */

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.Scanner;

import javax.swing.JFileChooser;


public class Dijkstra {

    //Initialization and Declarations of variables
    static int Max_routers;

    static int DistGrph[][] = null;
    static int DistRow[] = null;
    static int source = -1, destination = -1;
    class ConTableEntry {
        boolean flag;
        int length;
        int []ids;
        int depth;
    }

    //Connection Table initialization
    static ConTableEntry []conTable = null;

    // Main Method
    public static void main(String[] args) {

        Dijkstra psFrame = new Dijkstra();

    }
```

```
    public int nodeid(int in){
        return (in+1);
    }

    /* ReadTextfileToBuildGraph Method  */
    public void ReadTextfileToBuildGraph() {
        try
        {

            System.out.println("Enter Text File name to read Network
Topology Matrix:");
            Scanner in1 = new Scanner(System.in);              //User
input - the file name
            String fiename = in1.nextLine();

            FileReader fr = new FileReader(fiename);           //Creation
of FileReader object
            String val=new String();
            BufferedReader br = new BufferedReader(fr);        //Creation
of BufferedReader Object
            String[] temp;
            val=br.readLine();

            temp = val.split(" ");                             //To find
size of the matrices or number of routers
            DistGrph = new int [temp.length][temp.length];
            br.close();
            fr.close();
            System.out.println("\n<-----Graph Size Read----->\t:
"+temp.length);

            fr = new FileReader(fiename);                      //Read the
content of file into the FileReader object
            val=new String();
            br = new BufferedReader(fr);
            int i = 0;
            while((val=br.readLine())!=null)
            {
                String[] temp1;
```

```
                temp1 = val.split(" ");                    //Splitting with
space as a Delimiter
                for (int dist =0; dist < temp1.length; dist++ ){
                    DistGrph[i][dist] =  Integer.parseInt(temp1[dist]);
                }
                i++;
            }
            br.close();
            fr.close();
            System.out.println("<-----Graph Table Initialized----->");
            Max_routers = i;


            String imageName = "%3d" ;                        //Formatting
to display the Content in the Matrix format
            System.out.println();
            System.out.print("ID|");
            for (int j =0; j < Max_routers; j++ ){
                System.out.print(String.format( imageName,nodeid(j)));
            }
            System.out.println();

System.out.println("-----------------------------------------------------
-------------");
            for (int j =0; j < Max_routers; j++ ){
                imageName = "%2d|" ;                    //Formatting to
display the Content in the Matrix format
                System.out.print(String.format( imageName,nodeid(j)));
                imageName = "%3d" ;
                for (int k =0; k < Max_routers; k++ )
                    System.out.print( String.format( imageName,
DistGrph[j][k]));
                System.out.println();
            }

System.out.println("-----------------------------------------------------
-------------");
        } catch(Exception e){
            System.out.println("File Not Found, Please Enter a Valid File
!");    //To Handle File Not Found Exception
```

```
        }
    }
    /* ComputeConnectionTable Method*/
    public void ComputeConnectionTable(){

        if (DistGrph[source][source] == 0) {
            conTable = new ConTableEntry[Max_routers];    //Creates a
ConTable Object with the Number of Routers

            for (int j = 0;j<Max_routers;j++) {
                ConTableEntry ce = new ConTableEntry();
                //Initializing the variables to start processing
                ce.flag = true;
                ce.length = -1;
                ce.ids = new int[Max_routers];
                ce.ids[0] = source ;
                ce.depth = 1;
                for (int i = 1;i<Max_routers;i++) ce.ids[i] = -1;
                conTable[j] = ce;
            }

            //initializing source in working row
            int tmpsource = source;
            conTable[tmpsource].length = 0;
            conTable[tmpsource].ids[0] = source;
            conTable[tmpsource].flag = false;
            //int nodedepth = 1;

            for (int loopcnt = 0; loopcnt < Max_routers; loopcnt++) {

                for (int k = 0;  k < Max_routers ; k++)
                {
                    if (conTable[k].flag)
                    {
                        if (DistGrph[tmpsource][k] != -1){

                            if ((conTable[k].length != -1) ) {
                                // smaller ( selected node length +
tableentry, previous entry path)
```

```
                                if (conTable[k].length >
conTable[tmpsource].length + DistGrph[tmpsource][k]) {
                                    conTable[k].length =
conTable[tmpsource].length + DistGrph[tmpsource][k];
                                    for (int idx = 0; idx<
conTable[tmpsource].depth ;idx ++)
                                        conTable[k].ids[idx] =
conTable[tmpsource].ids[idx];
                                    conTable[k].depth =
conTable[tmpsource].depth ;
                                    conTable[k].ids[conTable[k].depth] =
k;
                                    conTable[k].depth++;
                                }
                            }
                            else
                            {  //selected node length is added to length
table entry for new length
                                conTable[k].length =
conTable[tmpsource].length + DistGrph[tmpsource][k];

                                for (int idx = 0; idx<
conTable[tmpsource].depth ;idx ++){
                                    conTable[k].ids[idx] =
conTable[tmpsource].ids[idx];
                                }



                                conTable[k].depth =
conTable[tmpsource].depth ;
                                conTable[k].ids[conTable[k].depth] = k;
                                conTable[k].depth++;
                            }
                        }

                    }
                }

                System.out.println();
                //initalize smallest dist
```

```
                int small = 0;
                int indx_small = 0;

                for (int i = 0; i<Max_routers; i++){

                    if (conTable[i].flag){
                        if(conTable[i].length !=-1 ){
                            small = conTable[i].length;
                            indx_small = i;
                            break;
                        }
                    }
                }
                //find source for next iteration
                for (int i = 0; i<Max_routers; i++){
                    if (conTable[i].flag){
                        if(conTable[i].length != -1 ){
                            if (small > conTable[i].length){
                                small = conTable[i].length;
                                indx_small = i;
                            }
                        }
                    }
                }
                tmpsource = indx_small;
                conTable[tmpsource].flag = false;

            }

            System.out.println("Router [" + nodeid(source) + "] "+
"Connection Table:");
            System.out.println("============================");
            System.out.println("Destination        Interface");
//Printing the Connection Table
            for (int i = 0; i<Max_routers; i++){
                String tmp = String.valueOf(conTable[i].ids[1]+1);
                if (conTable[i].ids[1] == -1) tmp = "-1";
//Check the Router ID if it is -1
                if (i == source) tmp = "-";
//Source to Source Router will be "-"
```

```
                System.out.print("      "+  nodeid(i) + "
"+ tmp);
                System.out.println();
            }
        }
        else {
            System.out.println("Router [" + nodeid(source) + "] "+
"Connection Table:");
            System.out.println("============================");
            System.out.println("Destination        Interface");
//If there is no Interface to the router then assign -1
            for (int i = 0; i<Max_routers; i++){
                System.out.print("      "+  nodeid(i) + "
-1");
                System.out.println();
            }

        }
    }
    /* PrintConnectionTable Method */
    public void PrintConnectionTabel() {
        System.out.println("Enter Source Router Id < 1 - "+
(Max_routers)+" >:");
        Scanner in1 = new Scanner(System.in);       //Takes input from the
User for the Source Router ID
        String str_source = in1.nextLine();
        source = Integer.parseInt(str_source);
        source--;                                   //Decrements the
Source ID by 1
        ComputeConnectionTable();                   //Invoke
ComputeConnectionTable Method
    }
    /* PrintShortPathToDestination Method */
    public void PrintShortPathToDestination() {

        System.out.println("Enter Destination Router Id < 1 - "+
(Max_routers)+" >:");
        Scanner in1 = new Scanner(System.in);       //Takes from the user
the Destination Router ID as input
        String str_dest = in1.nextLine();
```

```
        destination = Integer.parseInt(str_dest);
        destination--;                                  //Decrements
Destination Router ID
        if (DistGrph[source][source] == 0) {
            if (DistGrph[destination][destination] == 0) {

                System.out.print("Shortest Path from Router:
["+nodeid(source) +"] to ["+ nodeid(destination) + "] is: ");

                if (conTable[destination].length > 0) {
                    for (int n = 0;n< conTable[destination].depth; n++ ) {
                        if (-1 != conTable[destination].ids[n])
System.out.print(" "+ nodeid(conTable[destination].ids[n]));
                    }
                    System.out.println();
                    System.out.println("The total cost is "+
conTable[destination].length);
                }  else System.out.println("Path Not Available");
            }  else System.out.println("Destination Router is Down");
//If Destination Router is down
        } else System.out.println("Source Router is Down");
//If Source Router is down

    }
    /* Turn off a Router */
    public void ChangeRouterDown(){

        System.out.println("Enter Router Id < 1 - "+ (Max_routers)+" > to
Down:");
        Scanner in1 = new Scanner(System.in);          //Takes from the
user the Router ID to Down as input
        String str_delt = in1.nextLine();
        int delid = Integer.parseInt(str_delt);
        delid--;

        for (int j =0; j < Max_routers; j++ ){
            DistGrph[j][delid] = -1 ;             //Assigns -1 to the Down
Router row
        }
```

```
        for (int l =0; l < Max_routers; l++ ){
            DistGrph[delid][l] = -1 ;            //Assigns -1 to the Down
Router column
        }
        System.out.println("Modified Topology:");

        //insert
        String imageName = "%3d" ;                  //Formatting the
content in Matrix format
        System.out.println();
        System.out.print("ID|");
        for (int j =0; j < Max_routers; j++ ){
            System.out.print(String.format( imageName,nodeid(j)));
        }
        System.out.println();

System.out.println("-----------------------------------------------------
-------------");
        for (int j =0; j < Max_routers; j++ ){
            imageName = "%2d|" ;               //Formatting the content in
Matrix format
            System.out.print(String.format( imageName,nodeid(j)));
            imageName = "%3d" ;
            for (int k =0; k < Max_routers; k++ )
                System.out.print( String.format( imageName,
DistGrph[j][k]));
            System.out.println();
        }

System.out.println("-----------------------------------------------------
-------------");

    }

    /* MENU */
    public Dijkstra() {

        while (true){
```

```
System.out.println("\n=====================================================
=");
            System.out.println("Dijkstra's Algorithm - Link State Routing
Simulator:");

System.out.println("=====================================================\
n");
            System.out.println("Enter The Option :\n=================\n1.
Upload a Network Topology\n \n2. Choose the Source Router \n \n3. Find
Shortest Path to Destination Router \n \n4. Turn a Router Down \n \n5.
Exit\n");
            System.out.println("Command:");
            Scanner in = new Scanner(System.in);
            String regmessage = in.nextLine();

            if (regmessage.equals("1")){
                ReadTextfileToBuildGraph();
//ReadTextFiletoBuildGraph method call
                for (int n = 0;n<Max_routers;n++ ) {    //EXTRA FEATURE
IMPLEMENTATION -> TO DISPLAY CONNECTION TABLE FOR ALL NODES
                    source = n;
                    ComputeConnectionTable();
//ComputeConnectionTable method call
                    System.out.println();
                }
            }
            if (regmessage.equals("2")){
                PrintConnectionTabel();
//PrintConnectionTable method call
            }
            if (regmessage.equals("3")){
                PrintShortPathToDestination();
//PrintShortPathToDestination method call
            }
            if (regmessage.equals("4")){
                ChangeRouterDown();                //ChangeRouterDown
method call
                if ((source >-1) && (source < Max_routers)){
```

```
                    ComputeConnectionTable();
//ComputeConnectionTable method call
                    if (DistGrph[source][source] == 0) {
                        if ((destination >-1) && (destination <
Max_routers)){
                            if (DistGrph[destination][destination] == 0) {
                                System.out.print("Shortest Path from
Router: ["+nodeid(source) +"] to ["+ nodeid(destination) + "] is: ");
                                if (conTable[destination].length > -1) {
                                    for (int n = 0;n<Max_routers;n++ ) {
                                        if (-1 !=
conTable[destination].ids[n]) System.out.print(" "+
nodeid(conTable[destination].ids[n]));
                                    }
                                    System.out.println();
                                    System.out.println("The total cost is
"+ conTable[destination].length);
                                }
                                else System.out.println("Not Available");

                            } else System.out.println("Destination Router
is Down");
                        } else System.out.println("Destination node is not
selected");
                    } else System.out.println("Source Router is Down");
//Router Check conditions

                }else System.out.println("Source node is not selected");
            }
            if (regmessage.equals("5")){
                System.out.println("Exiting LinkStateRouting project...
Good Bye!.");
                System.exit(0);     //Exit system call
            }
        }
    }
}
```

Output:

A Network topology which has 5 Routers is selected. The Input matrix is read from the input file, *5_routers.txt* and the Network topology matrix is displayed on the Console along with the connection table of all routers.

```
=====================================================
Dijkstra's Algorithm - Link State Routing Simulator:
=====================================================

Enter The Option :
==================
1. Upload a Network Topology

2. Choose the Source Router

3. Find Shortest Path to Destination Router

4. Turn a Router Down

5. Exit

Command:
1
Enter Text File name to read Network Topology Matrix:
5_routers.txt

<-----Graph Size Read----->     : 5
<-----Graph Table Initialized----->

ID|  1  2  3  4  5
-------------------------------------------------------------------
 1|  0  2  5  1 -1
 2|  2  0  8  7  9
 3|  5  8  0 -1  4
 4|  1  7 -1  0  2
 5| -1  9  4  2  0
-------------------------------------------------------------------
```

Connection table of all routers in console:

```
Router [1] Connection Table:
============================
Destination        Interface
     1                -
     2                2
     3                3
     4                4
     5                4



Router [2] Connection Table:
============================
Destination        Interface
     1                1
     2                -
     3                1
     4                1
     5                1



Router [3] Connection Table:
============================
Destination        Interface
     1                1
     2                1
     3                -
     4                5
     5                5
```

```
Router [4] Connection Table:
============================
Destination        Interface
     1                1
     2                1
     3                1
     4                -
     5                5



Router [5] Connection Table:
============================
Destination        Interface
     1                4
     2                4
     3                3
     4                4
     5                -
```

The Source Router is selected to be 1 and the connection table of that router is displayed in the console.

```
=====================================================
Dijkstra's Algorithm - Link State Routing Simulator:
=====================================================

Enter The Option :
==================
1. Upload a Network Topology

2. Choose the Source Router

3. Find Shortest Path to Destination Router

4. Turn a Router Down

5. Exit

Command:
2
Enter Source Router Id < 1 - 5 >:
1




Router [1] Connection Table:
============================
Destination        Interface
      1                -
      2                2
      3                3
      4                4
      5                4
```

The Destination Router is selected to be 5 and the shortest path cost from router 1 to router 5 is calculated and displayed in the console along with the shortest path taken.

```
==================================================
Dijkstra's Algorithm - Link State Routing Simulator:
==================================================

Enter The Option :
================
1. Upload a Network Topology

2. Choose the Source Router

3. Find Shortest Path to Destination Router

4. Turn a Router Down

5. Exit

Command:
3
Enter Destination Router Id < 1 - 5 >:
5
Shortest Path from Router: [1] to [5] is:  1 4 5
The total cost is 3
```

The topology is changed by turning off the router 4 and the changed topology is displayed in the console.

```
=====================================================
Dijkstra's Algorithm - Link State Routing Simulator:
=====================================================

Enter The Option :
==================
1. Upload a Network Topology

2. Choose the Source Router

3. Find Shortest Path to Destination Router

4. Turn a Router Down

5. Exit

Command:
4
Enter Router Id < 1 - 5 > to Down:
4
Modified Topology:

ID|  1  2  3  4  5
-----------------------------------------------------------------------
 1|  0  2  5 -1 -1
 2|  2  0  8 -1  9
 3|  5  8  0 -1  4
 4| -1 -1 -1 -1 -1
 5| -1  9  4 -1  0
-----------------------------------------------------------------------
```

Again the same Source Router is selected (router 1) and the connection table of that router is displayed in the console.

There is a change in the connection table, because the router 4 is now down.

```
====================================================
Dijkstra's Algorithm - Link State Routing Simulator:
====================================================

Enter The Option :
=================
1. Upload a Network Topology

2. Choose the Source Router

3. Find Shortest Path to Destination Router

4. Turn a Router Down

5. Exit

Command:
2
Enter Source Router Id < 1 - 5 >:
1




Router [1] Connection Table:
============================
Destination        Interface
      1                -
      2                2
      3                3
      4                -1
      5                3
```

Now when the Destination Router is selected to be 5, the shortest path cost from router 1 to router 5 is changed to 9 and the shortest path taken is 1 → 3 → 5 instead of 1 → 4 → 5 because the router 4 which was an intermediate router in the shortest path is now down.

```
====================================================
Dijkstra's Algorithm - Link State Routing Simulator:
====================================================

Enter The Option :
=================
1. Upload a Network Topology

2. Choose the Source Router

3. Find Shortest Path to Destination Router

4. Turn a Router Down

5. Exit

Command:
3
Enter Destination Router Id < 1 - 5 >:
5
Shortest Path from Router: [1] to [5] is:  1 3 5
The total cost is 9
```

Now, if we select some router as source and router 4 as destination, we can see that it says “Destination Router is Down” when asked for the shortest distance.

```
==================================================
Dijkstra's Algorithm - Link State Routing Simulator:
==================================================

Enter The Option :
=================
1. Upload a Network Topology

2. Choose the Source Router

3. Find Shortest Path to Destination Router

4. Turn a Router Down

5. Exit

Command:
2
Enter Source Router Id < 1 - 5 >:
1




Router [1] Connection Table:
============================
Destination        Interface
     1                -
     2                2
     3                3
     4                -1
     5                3
```

```
==================================================
Dijkstra's Algorithm - Link State Routing Simulator:
==================================================

Enter The Option :
=================
1. Upload a Network Topology

2. Choose the Source Router

3. Find Shortest Path to Destination Router

4. Turn a Router Down

5. Exit

Command:
3
Enter Destination Router Id < 1 - 5 >:
4
Destination Router is Down
```

Similarly, if we select router 4 as the source, we can see that it says "Source Router is Down" when asked for the shortest distance.

```
==================================================
Dijkstra's Algorithm - Link State Routing Simulator:
==================================================

Enter The Option :
==================
1. Upload a Network Topology

2. Choose the Source Router

3. Find Shortest Path to Destination Router

4. Turn a Router Down

5. Exit

Command:
2
Enter Source Router Id < 1 - 5 >:
4
Router [4] Connection Table:
============================
Destination        Interface
      1               -1
      2               -1
      3               -1
      4               -1
      5               -1
```

```
==================================================
Dijkstra's Algorithm - Link State Routing Simulator:
==================================================

Enter The Option :
==================
1. Upload a Network Topology

2. Choose the Source Router

3. Find Shortest Path to Destination Router

4. Turn a Router Down

5. Exit

Command:
3
Enter Destination Router Id < 1 - 5 >:
5
Source Router is Down
```

The Link State Routing Protocol implemented in this objective using Dijkstra's algorithm has been tested under various cases and is found to be successful. This project has been tested for various number of routers like 5 routers, 10 routers, 12 routers, 15 routers, 20 routers etc.

Result:

- The Link State Routing Protocol is implemented using Dijkstra's algorithm.
- It is tested against different cases including Link-Cost Change and path change, Link Failure.