Simulation of Link-State Routing Protocol

Vivek Pabani

Illinois Institute of Technology
CWID A20332117
vpabani@hawk.iit.edu

Abstract

The Simulation of Link-State Routing Protocol is a part of the project under course CS542 - Computer Networks. The program accepts the network topology details in terms of the cost of links, and provides the shortest path tree for the network. It uses Dijkstra's Algorithm to construct its routing table.

I. Introduction

A routing protocol specifies how routers communicate with each other, disseminating information that enables them to select routes between any two nodes on a computer network. Routing algorithms determine the specific choice of route. Each router has a priori knowledge only of networks attached to it directly. A routing protocol shares this information first among immediate neighbors, and then throughout the network. This way, routers gain knowledge of the topology of the network.

II. Link State Protocol

The basic concept of link-state routing is that every node creates 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 best logical path from it to every possible destination in the network. The collection of best paths will create a shortest path tree, which then forms the node's routing table.

Link state routing is based on the assumption that each node has partial knowledge: it knows the state (cost) of its links. In other words, the whole topology can be compiled from the partial knowledge of each node. In this way, it allows the topology to be dynamic.

In link state routing, four different tasks are required to ensure that each node has the routing table showing the least-cost node to every other node.

1. Creation of the states of the links by each node, called the link state packet or LSP.
2. Dissemination of LSP data to every other router, called flooding.
3. Formation of a shortest path tree for each node.
4. Calculation of a routing table based on the shortest path tree.

I. Creation of Link State Packet

A link state packet (LSP) can carry a large amount of information. For this project, we consider that it carries a minimum amount of data: the node identity and the cost of links. Both of these details are needed to make the topology. LSPs are generated either on a periodic basis or when there is a change in the topology of the domain.

II. Flooding of LSPs

After a node has prepared an LSP, it must be sent to all other nodes. This process is called flooding. The steps are as the following:

1. The creating node sends a copy of the LSP to each neighbour.
2. A node that receives an LSP compares it with the data it may already have. If the newly arrived LSP is same as the old one it has, it discards the LSP. If it is newer, then it updates the data, and sends a copy of it to each neighbour except the one it came from.

III. Formation of Shortest Path Tree

After receiving all LSPs, each node will have a copy of the whole topology. Dijkstra's Algorithm finds the shortest path to every node using this topology data. The algorithm is explained in detail in next section.

IV. Calculation of Routing Table from Shortest Path Tree

Each node uses the shortest path tree found by the Dijkstra Algorithm to construct its routing table. The routing table contains the cost of reaching each node from the root.

III. Dijkstra Algorithm

The Dijkstra algorithm is used to create a shortest path tree from a given graph. A shortest path tree is a tree in which the path between the root and every other node is the shortest. The algorithm uses the following steps:

1. Initialization : Select the node as the root of the tree and add it to the path. Set the shortest distances for all the rootâĂŹs neighbors to the cost between the root and those neighbors. Set the shortest distance of the root to zero.
2. Iteration : Repeat the following two steps until all nodes are added to the path:
   (a) Adding the next node to the path: Search the nodes not in the path. Select the one with minimum shortest distance and add it to the path.
   (b) Updating: Update the shortest distance for all remaining nodes using the shortest distance of the node just moved to the path in step 2.

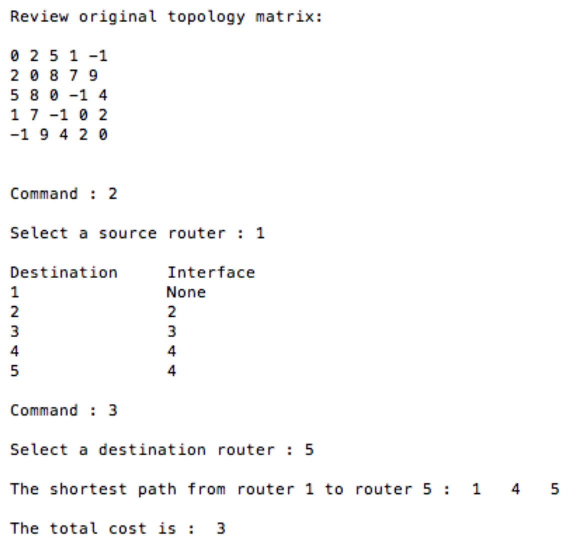
IV. Algorithm Implementation

The Dijkstra Algorithm to find the shortest path to from a source router to a destination router is implemented in Python for this project. The program follows below steps :

- The program first asks for a network topology file. It validates the data and store in matrix format.
- The next step is to create the connection table. The program takes the source router as input, and performs the Dijkstra Algorithm on it as explained in previous section. At every step, it keeps track of two type of nodes :
  1. The interface used to go to next router. (For connection table.)
  2. The parent node of last added node. (To create the final path.)
- Once both connection table and parent table are ready, the shortest path can be found from given source to destination by one of the two ways :
  1. Starting from the source node, follow the interface from the connection table to reach to the destination.
  2. Starting from the destination node, follow the parent node from the parent table to reach to the source, and provide the reverse path.
- In both the cases, the path and total cost is found and returned to the user.
- If there is no path from given source and destination, the program returns with such message.

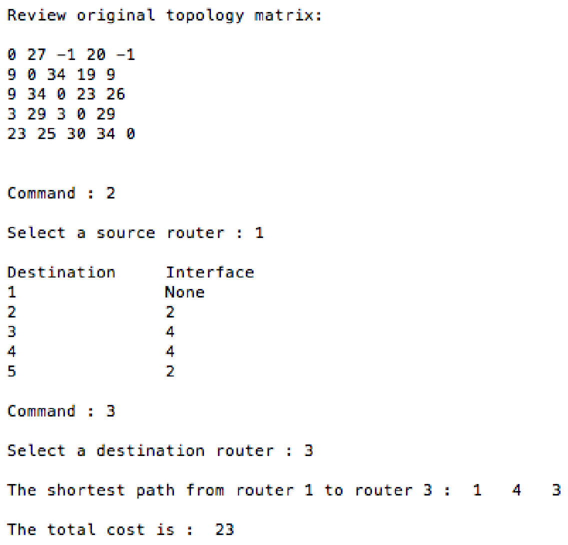
V. Test Data and Result

To test the program, I created several topology files with different network structure and varying number of routers. The program assumes that the given topology file will have one and only one distance/cost value for each pair of router.



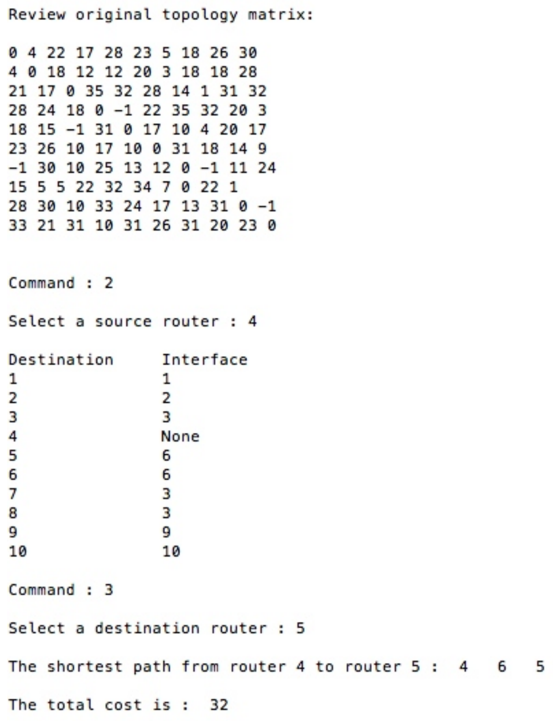
```
Review original topology matrix:

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


Command : 2

Select a source router : 1

Destination     Interface
1               None
2               2
3               3
4               4
5               4

Command : 3

Select a destination router : 5

The shortest path from router 1 to router 5 :  1   4   5

The total cost is :  3
```

Figure 1: *Sample test data with 5 routers provided with project definition.*



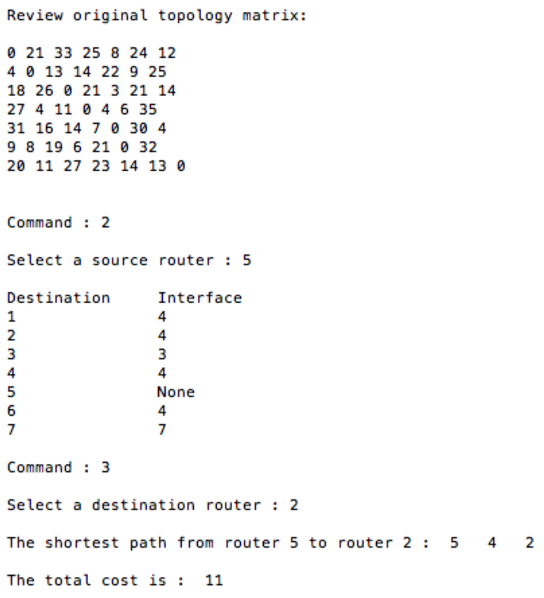
```
Review original topology matrix:

0 27 -1 20 -1
9 0 34 19 9
9 34 0 23 26
3 29 3 0 29
23 25 30 34 0


Command : 2

Select a source router : 1

Destination      Interface
1                None
2                2
3                4
4                4
5                2

Command : 3

Select a destination router : 3

The shortest path from router 1 to router 3 :  1   4   3

The total cost is :  23
```

Figure 2: *Test data with 5 routers.*



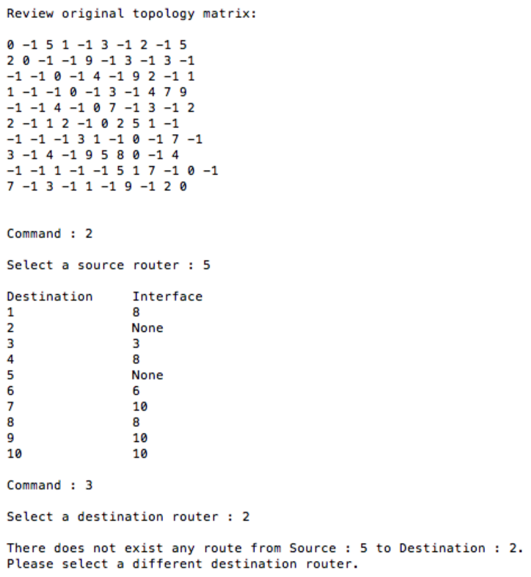
```
Review original topology matrix:

0 4 22 17 28 23 5 18 26 30
4 0 18 12 12 20 3 18 18 28
21 17 0 35 32 28 14 1 31 32
28 24 18 0 -1 22 35 32 20 3
18 15 -1 31 0 17 10 4 20 17
23 26 10 17 10 0 31 18 14 9
-1 30 10 25 13 12 0 -1 11 24
15 5 5 22 32 34 7 0 22 1
28 30 10 33 24 17 13 31 0 -1
33 21 31 10 31 26 31 20 23 0


Command : 2

Select a source router : 4

Destination      Interface
1                1
2                2
3                3
4                None
5                6
6                6
7                3
8                3
9                9
10               10

Command : 3

Select a destination router : 5

The shortest path from router 4 to router 5 :  4   6   5

The total cost is :  32
```

Figure 4: *Test data with 10 routers.*



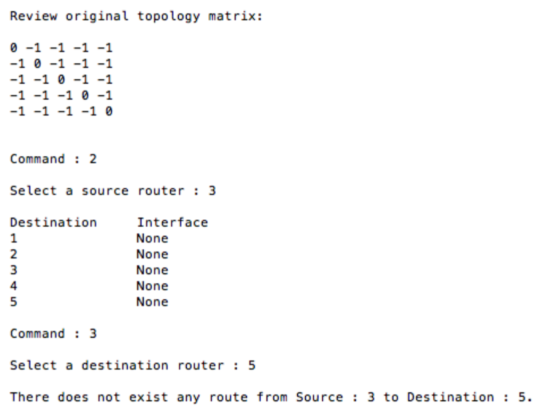
```
Review original topology matrix:

0 21 33 25 8 24 12
4 0 13 14 22 9 25
18 26 0 21 3 21 14
27 4 11 0 4 6 35
31 16 14 7 0 30 4
9 8 19 6 21 0 32
20 11 27 23 14 13 0


Command : 2

Select a source router : 5

Destination      Interface
1                4
2                4
3                3
4                4
5                None
6                4
7                7

Command : 3

Select a destination router : 2

The shortest path from router 5 to router 2 :  5   4   2

The total cost is :  11
```

Figure 3: *Test data with 7 routers.*



```
Review original topology matrix:

0 -1 5 1 -1 3 -1 2 -1 5
2 0 -1 -1 9 -1 3 -1 3 -1
-1 -1 0 -1 4 -1 9 2 -1 1
1 -1 -1 0 -1 3 -1 4 7 9
-1 -1 4 -1 0 7 -1 3 -1 2
2 -1 1 2 -1 0 2 5 1 -1
-1 -1 -1 3 1 -1 0 -1 7 -1
3 -1 4 -1 9 5 8 0 -1 4
-1 -1 1 -1 -1 5 1 7 -1 0 -1
7 -1 3 -1 1 -1 9 -1 2 0


Command : 2

Select a source router : 5

Destination      Interface
1                8
2                None
3                3
4                8
5                None
6                6
7                10
8                8
9                10
10               10

Command : 3

Select a destination router : 2

There does not exist any route from Source : 5 to Destination : 2.
Please select a different destination router.
```

Figure 5: *Test data with 10 routers having no path between two routers.*



```
Review original topology matrix:

0 -1 -1 -1 -1
-1 0 -1 -1 -1
-1 -1 0 -1 -1
-1 -1 -1 0 -1
-1 -1 -1 -1 0


Command : 2

Select a source router : 3

Destination     Interface
1               None
2               None
3               None
4               None
5               None

Command : 3

Select a destination router : 5

There does not exist any route from Source : 3 to Destination : 5.
```

Figure 6: *Test data with 5 routers having no path between any two routers.*

VI. Conclusion

The implented program for Link State Algorithm works for any network topology regardless of the size of network. With every node having partial information about the network topology, it can create shortest path tree for the network. Given valid totplogy data, it will provide you with the shortest path between source router and destination router.