Link State Routing Simulator

Project Operations Manual

CS 542

Saba Ghauri
A20336256

INTRODUCTION

Routing is the process of moving packets across a network from one host to another. It is usually performed by dedicated devices called routers. Packets are the fundamental unit of information transport in all modern computer networks, and increasingly in other communications networks as well, therefore the need of effective and efficient routing protocols arises.

In this report, the concept of link state routing and operational aspects and functionality of Link state routing protocol simulator is explained. Further, it includes a user manual for Link state simulator. In the end couple of test cases are presented in order to verify working of link state routing simulator.

LINK STATE ROUTING PROTOCOL

Link-state routing protocols are one of the two main classes of routing protocols used in packet switching networks for computer communications, the other being distance-vector routing protocols. Examples of link-state routing protocols include Open Shortest Path First (OSPF) and intermediate system to intermediate system (IS-IS).

OSPF is an interior gateway protocol (IGP) for routing Internet Protocol (IP) packets solely within a single routing domain, such as an autonomous system. It gathers link state information from available routers and constructs a topology map of the network. The topology is presented as a routing table to the Internet layer which routes packets based solely on their destination IP address. OSPF supports Internet Protocol Version 4 (IPv4) and Internet Protocol Version 6 (IPv6) networks and supports the Classless Inter-Domain Routing (CIDR) addressing model.

OSPF detects changes in the topology, such as link failures, and converges on a new loop-free routing structure within seconds. It computes the shortest-path tree for each route using a method based on Dijkstra's algorithm.

WORKING OF LINK STATE ROUTING PROTOCOL:

The link-state protocol is performed by every switching node in the network (i.e., nodes that are prepared to forward packets; in the Internet, these are called routers). 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.

LINK-STATE ROUTING PROCESS

1. Each router learns about its own links (directly connected networks)
2. Find directly connected neighbors
3. Builds a Link-State Packet (LSP) with the state of each directly connected link
4. Floods the LSP to all neighbors, who stores the received LSPs in a database
5. Each router uses the database to construct a complete map of the network topology
6. Computes the best path to each destination network

DIJKSTRA ALGORITHM

```
1  function Dijkstra(Graph, source):
2      dist[source] ← 0                                  // Initialization
3
4      create vertex set Q
5
6      for each vertex v in Graph:
7          if v ≠ source
8              dist[v] ← INFINITY                        // Unknown distance from source to v
9              prev[v] ← UNDEFINED                       // Predecessor of v
10
11         Q.add_with_priority(v, dist[v])
12
13
14     while Q is not empty:                             // The main loop
15         u ← Q.extract_min()                           // Remove and return best vertex
16         for each neighbor v of u:                     // only v that is still in Q
17             alt ← dist[u] + length(u, v)
18             if alt < dist[v]
19                 dist[v] ← alt
20                 prev[v] ← u
21                 Q.decrease_priority(v, alt)
22
23     return dist[], prev[]
```

Explanation – Shortest Path using Dijkstra's Algorithm

The idea of the algorithm is very simple.

1. It maintains a list of unvisited vertices.
2. It chooses a vertex (the source) and assigns a maximum possible cost (i.e. infinity) to every other vertex.

3. The cost of the source remains zero as it actually takes nothing to reach from the source vertex to itself.
4. In every subsequent step of the algorithm it tries to improve(minimize) the cost for each vertex. Here the cost can be distance, money or time taken to reach that vertex from the source vertex. The minimization of cost is a multi-step process.
5. For each unvisited neighbor (vertex 2, vertex 3, vertex 4) of the current vertex (vertex 1) calculate the new cost from the vertex (vertex 1).
6. For e.g. the new cost of vertex 2 is calculated as the minimum of the two ((existing cost of vertex 2) or (sum of cost of vertex 1 + the cost of edge from vertex 1 to vertex 2))
7. When all the neighbors of the current node are considered, it marks the current node as visited and is removed from the unvisited list.
8. Select a vertex from the list of unvisited nodes (which has the smallest cost) and repeat step 4.

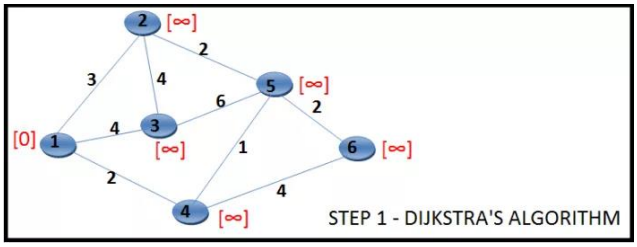
At the end, there will be no possibilities to improve it further and then the algorithm ends

```
1 S ← empty sequence
2 u ← target
3 while prev[u] is defined:                    // Construct the shortest
path with a stack S
4 insert u at the beginning of S          // Push the vertex onto the stack
5 u ← prev[u]                            // Traverse from target to source
6 insert u at the beginning of S              // Push the source onto the
stack
```

Step Wise Execution

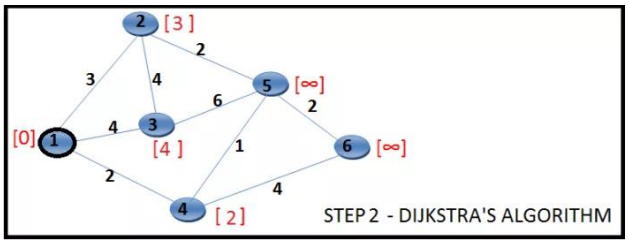
Step 1:

Mark Vertex 1 as the source vertex. Assign a cost zero to Vertex 1 and (infinite to all other vertices). The state is as follows:



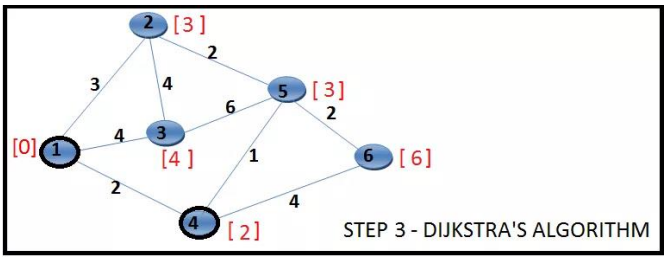
Step 2:

For each of the unvisited neighbors (Vertex 2, Vertex 3 and Vertex 4) calculate the minimum cost as min (current cost of vertex under consideration, sum of cost of vertex 1 and connecting edge). Mark Vertex 1 as visited, in the diagram we border it black. The new state would be as follows:



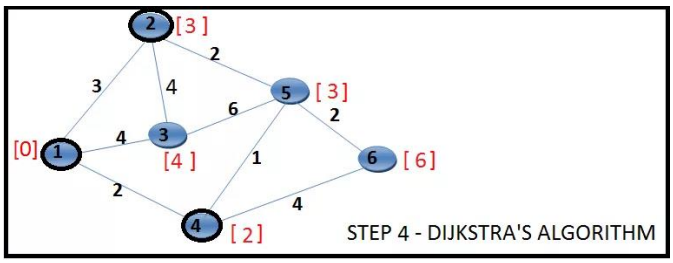
Step 3:

Choose the unvisited vertex with minimum cost (vertex 4) and consider all its unvisited neighbors (Vertex 5 and Vertex 6) and calculate the minimum cost for both of them. The state is as follows:



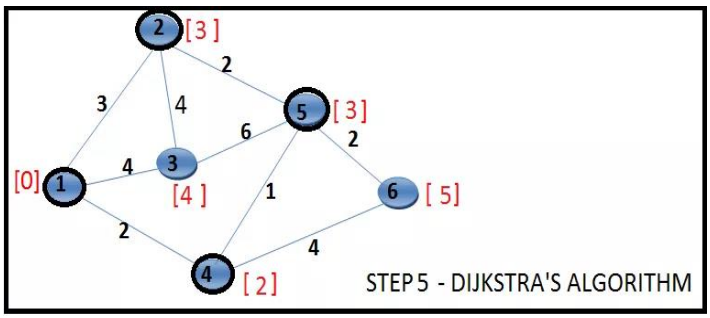
Step 4:

Choose the unvisited vertex with minimum cost (vertex 2 or vertex 5, here we choose vertex 2) and consider all its unvisited neighbors (Vertex 3 and Vertex 5) and calculate the minimum cost for both of them. Now, the current cost of Vertex 3 is [4] and the sum of (cost of Vertex 2 + cost of edge (2,3)) is 3 + 4 = [7]. Minimum of 4, 7 is 4. Hence the cost of vertex 3 won't change. By the same argument the cost of vertex 5 will not change. We just mark the vertex 2 as visited, all the costs remain same. The state is as follows:



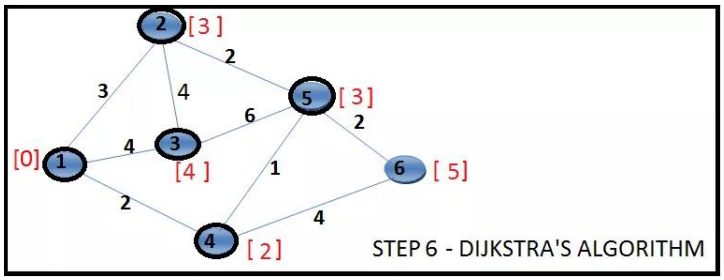
Step 5:

Choose the unvisited vertex with minimum cost (vertex 5) and consider all its unvisited neighbors (Vertex 3 and Vertex 6) and calculate the minimum cost for both of them. Now, the current cost of Vertex 3 is [4] and the sum of (cost of Vertex 5 + cost of edge (5,3)) is 3 + 6 = [9]. Minimum of 4, 9 is 4. Hence the cost of vertex 3 won't change. Now, the current cost of Vertex 6 is [6] and the sum of (cost of Vertex 5 + cost of edge (3,6)) is 3 + 2 = [5]. Minimum of 6, 5 is 45. Hence the cost of vertex 6 changes to 5. The state is as follows:



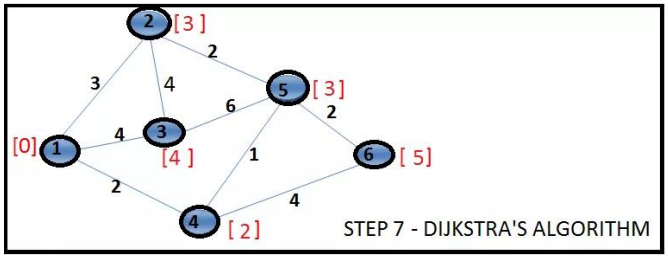
Step 6:

Choose the unvisited vertex with minimum cost (vertex 3) and consider all its unvisited neighbors (none). So mark it visited. The state is as follows:



Step 7:

Choose the unvisited vertex with minimum cost (vertex 6) and consider all its unvisited neighbors (none). So mark it visited. The state is as follows:



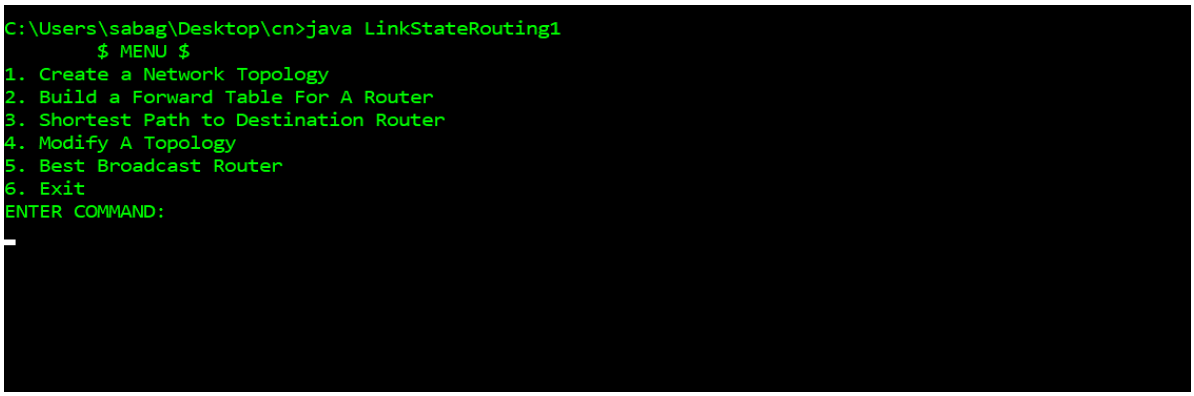
Now there is no unvisited vertex left and the execution ends. At the end we know the shortest paths for all the vertices from the source vertex 1. Even if we know the shortest path length, we do not know the exact list of vertices which contributes to the shortest path until we maintain them separately or the data structure supports it.

DESIGN OF STATE ROUTING PROTOCOL SIMULATOR

State Routing simulator has following operations:

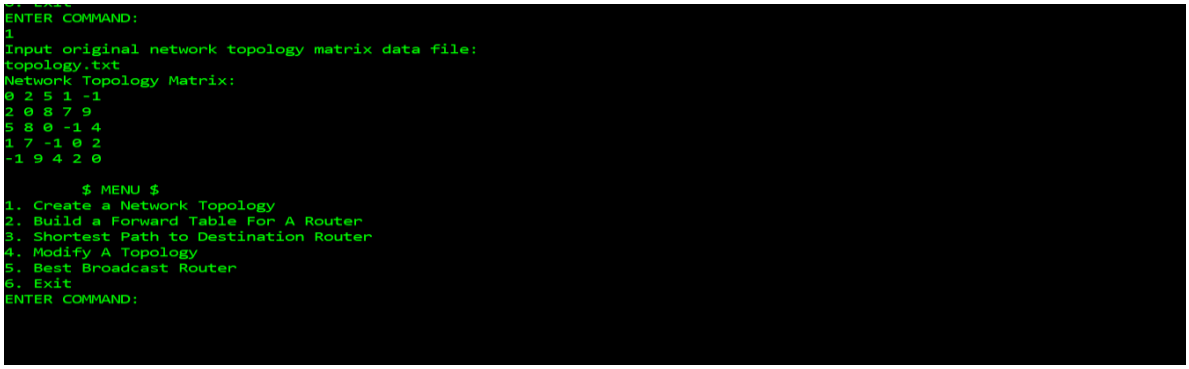
1. Main Menu

   The Simulator starts with a Main Menu, which all the different functionalities offered by the simulator, user can select any function from the menu.



2. Create a Network Topology

   The link state routing simulator inputs a user defined network topology and stores it as an adjacency matrix for future use.



3. Build a Forward Table

After storing the network topology, user can generate forwarding table at any particular node. This forwarding table show the next hop for at destinations/ other nodes.

4. Shortest Path to Destination Router

User can also see the shortest path between any source destination pair within the network topology. Besides this once a router becomes down it can reroute for the shortest path. This function is based on Dijkstra's Algorithm for shortest path.

5. Modify a Topology (Change the status of the Router)

User can also manually select a router to remove (declare it a s a down router). And can then create forward table or calculate the shortest path using modified network topology. Once a router is down it is represented as "-1" which means unreachable/down.

```
ENTER COMMAND:
4
Enter router to be deleted:
2
0 -1 5 1 -1
-1 -1 -1 -1 -1
5 -1 0 -1 -1
1 -1 -1 0 -1
-1 -1 -1 -1 -1

To calculate shortest path

Enter the source router
2
Enter the destination router
3
Shortest path from 2 to 3
R2 to R3
Total cost =  0
```

6. Best Router for Broadcast

This Simulator also selects the best broadcast router within the network, by using Dijkstra's Algorithm. Also, is a router is removed from network it re calculates the best broadcast router.

```
      $ MENU $
1. Create a Network Topology
2. Build a Forward Table For A Router
3. Shortest Path to Destination Router
4. Modify A Topology
5. Best Broadcast Router
6. Exit
ENTER COMMAND:
5
Best Router = R1

      $ MENU $
1. Create a Network Topology
2. Build a Forward Table For A Router
3. Shortest Path to Destination Router
4. Modify A Topology
5. Best Broadcast Router
6. Exit
ENTER COMMAND:
```

7. Exit

Finally, at any point if user wants to leave it can select Exit and the Simulator will exit safely.

```
      $ MENU $
1. Create a Network Topology
2. Build a Forward Table For A Router
3. Shortest Path to Destination Router
4. Modify A Topology
5. Best Broadcast Router
6. Exit
ENTER COMMAND:
6
Exit CS542-04 2017 Fall project. Good Bye!

C:\Users\sabag\Desktop\cn>
```

User Manual for Link State Routing Simulator

1) To run the link state routing simulator via command line prompt type
   java LinkStateRouting1

2) User can also use Eclipse or any other Java Compiler, import the Project and run its LinkStateRouting1.java file

3) Add the input network topology file in the same directory as the source code

4) In the input file, all the elements should be separated by single space.

5) Enter the desired choice to get the required output.

References

http://techieme.in/shortest-path-using-dijkstras-algorithm/

https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

https://en.wikipedia.org/wiki/Open_Shortest_Path_First