**Dept. of Computer Science**
**Illinois Institute of Technology, Chicago, IL 60616**

# LINK STATE ROUTING PROTOCOL SIMULATOR

## CS-542 Computer Networks I – Project

Submitted To
Professor Dr. Michael Y Choi

Priyank P Shah
Pshah104@hawk.iit.edu

# Index:

# 1. Instructions:

**System Requirements:** Java JDK /JRE 1.7

**How to run project?**

You can run the program using the following command on the command line. You must have a JRE or JDK installed on the machine.

**java –jar CS542Project_Sec03_SHAH_PRIYANK.jar**

Once Program starts, Project Operation Walkthrough will help you regarding the operation sequence.

**Note:** Put the Topology file and .jar file in the same folder/drive to run successfully.

## Walkthrough:

Select Option 1 and input the file name.
Select Option 2 and Input the Source Node -> Connection Table will be generated.

Select Option 3 and enter the Destination Node -> Print the Best Possible path between source and destination node with the minimum cost of routing.

Select Option 4 to Modify topology(i.e. Down any Router) -> If Source and destination are already selected then it will print the alternate path for the same source and destination else it will return back to menu.

Select Option 5 to Exit the program.

## 2. Project Design:

Project Consist of 4 java file named as:

1. Driver.java
2. CreateTopology.java
3. ConnTable.java
4. ShortPathRoute.java

**Driver.java** file handles the basic user interaction with the program regarding the user input and then call the appropriate class and its respective function. In this class I have provided validation for proper sequence of user input/
**i.e.** User must input the topology file first in order to access the other feature of the programs. (User need to select option 1 before option 2 , else it will produce error message).

**CreateTopology.java** file will take input of file name from user and store the data as an array form in program.

**ConnTable.java** will take the user input for source node and for that node it will generate the routing table .
It will also be responsible for 4$^{th}$ Option of modifying the topology(Router Down)

**ShortPathRoute.java** will take the user input of destination node and display the shortest path from between source node and destination node.

**Q:** How did I convince myself that program works correctly?

**A:** First of all, I studied the link state routing algorithm and use of dijkstra's algorithm. I watched video of actual working of it. Then I implemented it and checked the application output with the expected output. Each time through the different test cases, Application's output matched the expected output which convinced me that the program works correctly.

# 3. Detail Description of Algorithm I applied:

**Project works In the following flow:**

**Option 1:**

**Operations:** Take Filename as an input from user and generate array of routing values.
**Description:**

First of all user takes the input for the file name in Option 1. It is mandatory to go through this option in order to gain access to other options. If user enters the invalid file name then it will generate massage for invalid file name error

**Option 2:**

**Operations:** Generate routing table.
**Description:**

In the next step, it will ask user for the source node number, at that time it will check if the node number within the range of array. It will store the value of distance to each node from that source node in separate array. In each iteration it checks the distance to each node with corresponding link with minimum cost. In addition to this it will also store the best path to each node in another matrix(next[][]) and min cost to each node in another matrix(distance[]).

When displaying the next best node, It will display the value of very first node from the (next[][]), which in other words traversal back in the path.

**Option 3:**

**Operations:** Print the Best available/possible path with its minimum cost
**Description:**

If Operation 2 in not performed then this operation is not accessible. Once connection table is successfully build then, Enter the Destination node. It will fetch the value as next[destination_node][i] until it finds the value 0 in next field.

Program will automatically consider minimum path cost of 9999 as infinite and display the message regarding the “Path not found/Routing is not possible.”

**Option 4:**

**Operations:** Router Down , If previously path is generated then shows alternate path with its cost.
**Description:**

This Operation is accessible once user input the file in program. This operation will simply turns the value of router[down_routernumber][i] and router[i][ down_routernumber] as -1. -1 is used to indicate that the there is no path between two nodes if the value is -1.

## 4. Link State Routing Algorithm:

Link state routing protocol is used to maintain the map of the network for each router running with the link state routing protocol. Each router originates the information about the router, Its directly connected links and state of those links. This information is sent to all routers in network as multicast message.

Link state routing keeps its routing table updated, by updating itself whenever changes occurs to it. Each router maintains its own routing table for best path and next hop for each destination. Meanwhile with each changes it also updates.

Link state routing protocol is based on the Shortest Path First(SPF) algorithm to find the best path to a destination. Shortest Path first algorithm is nothing but Dijkstra's algorithm. This algorithm is used to find the shortest path between a source and a destination with the minimum cost either by time, by hops or by weight.

In shortest path first algorithm whenever link changes its status, routing update called a link-state advertisement is exchanged between routers. When a router receives an LSA routing update, the link state algorithm is used to recalculate the shortest path to affected destinations. Each router has a complete map of a network.

# 5. TEST REPORTS:

**INPUT FILE:**

**0 2 5 -1 -1 -1 -1 1**
**2 0 2 -1 4 -1 -1 2**
**5 2 0 2 6 -1 2 -1**
**-1 -1 2 0 -1 3 -1 -1**
**-1 4 6 -1 0 1 -1 -1**
**-1 -1 -1 3 1 0 8 -1**
**-1 -1 2 -1 -1 8 0 5**
**1 2 -1 -1 -1 -1 5 0**

| | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 |
|---|---|---|---|---|---|---|---|---|
| R1 | 0 | 2 | 5 | -1 | -1 | -1 | -1 | 1 |
| R2 | 2 | 0 | 2 | -1 | 4 | -1 | -1 | 2 |
| R3 | 5 | 2 | 0 | 2 | 6 | -1 | 2 | -1 |
| R4 | -1 | -1 | 2 | 0 | -1 | 3 | -1 | -1 |
| R5 | -1 | 4 | 6 | -1 | 0 | 1 | -1 | -1 |
| R6 | -1 | -1 | -1 | 3 | 1 | 0 | 8 | -1 |
| R7 | -1 | -1 | 2 | -1 | -1 | 8 | 0 | 5 |
| R8 | 1 | 2 | -1 | -1 | -1 | -1 | 5 | 0 |

**Routing Tables:**

| **R1:** | | **R2:** | | **R3:** | |
|---|---|---|---|---|---|
| R1 | - | R1 | R1 | R1 | R2 |
| R2 | R2 | R2 | - | R2 | R2 |
| R3 | R2 | R3 | R3 | R3 | - |
| R4 | R2 | R4 | R3 | R4 | R4 |
| R5 | R2 | R5 | R5 | R5 | R5 |
| R6 | R2 | R6 | R5 | R6 | R4 |
| R7 | R8 | R7 | R3 | R7 | R7 |
| R8 | R8 | R8 | R8 | R8 | R2 |

**Result:**

These Routing tables are generated from the program are same as expected output.

**Shortest Path with the Minimum Cost:**

1. 1 -> 5 : Path: 1->2->5
   Cost: 6
2. 2 -> 7: Path: 2->3->7
   Cost:4
3. 2 -> 6: Path:2->5->6
   Cost:5
4. 5->1: Path:5->2->1
   Cost: 6
5. 1->7: Path: 1->2->3->7
   Coat: 6

**Router Down:**

Router 3 Down:
Path:1->8->7
Cost: 6

Router 8 Down:
Path: 1->2->5->6->7
Cost:15

Router 6 Down:
Path: -
Cost: -
Message: Router is Down, Routing is not Possible !!!

**Result:**

The Path and Cost that are generated from the program are same as expected output even while modifying the topology.

**Extra Credit Note:**

I have Implemented Functionality 2 of checking min 8 nodes are required, not sure though whether what I thought is correct or not, so I **commented it** in **driver.java** file in **Case 1.**

**Logic** that I applied is, meanwhile taking input from file, I checked if the array on input length is more than 8 or not. If it is then program will move else it will display error message.