Implementation of Link-State Routing Protocol

A PROJECT REPORT

CS 542 Project (Fall 2015)

Submitted by,

Nagaraj, Dhayalini

CWID-A20359686

Main Campus, Section -02

Seat No: 53

# Content

**ABSTRACT ............................................................................... 3**

**Introduction ............................................................................. 4**

**Algorithm ................................................................................. 4**

**Pseudocode .............................................................................. 6**

**Design of the program .............................................................. 6**

**Test Report for sample network file ……………………………………. 8**

**Instructions on compiling and running a file ……………………… 17**

**ABSTRACT**

**Link State Routing Protocol**

* Link-state routing protocols are one of the two main classes of [routing protocols](https://en.wikipedia.org/wiki/Routing_protocol) used in [packet switching](https://en.wikipedia.org/wiki/Packet_switching) networks for [computer communications](https://en.wikipedia.org/wiki/Computer_communication), the other being [distance-vector routing protocols](https://en.wikipedia.org/wiki/Distance-vector_routing_protocol).

**Examples**

[Open Shortest Path First](https://en.wikipedia.org/wiki/Open_shortest_path_first) (OSPF)

[Intermediate System to Intermediate System](https://en.wikipedia.org/wiki/IS-IS) (IS-IS).

* 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](https://en.wikipedia.org/wiki/Internet), these are called [routers](https://en.wikipedia.org/wiki/Router_(computing))).
* 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](https://en.wikipedia.org/wiki/Graph_theory), 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. The collection of best paths will then form the node's [routing table](https://en.wikipedia.org/wiki/Routing_table).

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Characteristic** | **Explanation** |
| Periodic updates | Only when changes occur. OSPF, for example, also sends all summary information every 30 minutes by default. |
| Broadcast updates | Only devices running routing algorithms listen to these updates. Updates are sent to a multicast address. |
| Database | A database contains all topological information from which an IP routing table is assembled. |
| Algorithm | Dijkstra Algorithm for OSPF. |
| Convergence | Updates are faster and convergence times are reduced. |
| CPU/memory | Higher CPU and memory requirements to maintain link-state databases. |
| Examples | OSPF and IS-IS. |

**Djikstra's Algorithm**

Djikstra's algorithm solves the problem of finding the shortest path from a point in a graph (the source) to a destination.

We can find the shortest paths from a given source to all points in a graph in the same time, hence this problem is sometimes called the single-source shortest paths problem.

1. **Introduction**

Dijkstra's algorithm is an [algorithm](https://en.wikipedia.org/wiki/Algorithm) for finding the [shortest paths](https://en.wikipedia.org/wiki/Shortest_path_problem) between [nodes](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) in a [graph](https://en.wikipedia.org/wiki/Graph_(abstract_data_type)). It was conceived by [computer scientist](https://en.wikipedia.org/wiki/Computer_scientist) [Edsger W. Dijkstra](https://en.wikipedia.org/wiki/Edsger_W._Dijkstra) in 1956 and published three years later.

The algorithm exists in many variants; Dijkstra's original variant found the shortest path between two nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a [shortest path tree](https://en.wikipedia.org/wiki/Shortest_path_tree).

* 1. **Algorithm**

For a given source node in the graph, the algorithm finds the shortest path between that node and every other.

1. Distance value is assigned to every node: Zero is set for the initial node and all other nodes are set to infinity.
2. The initial node should be marked as current and all the other nodes are marked as unvisited.
3. The initial node is the current node, the other unvisited nodes are considered and the tentative distance is calculated.
4. The newly calculated distance is compared and the smallest distance is assigned to the nodes.

**Example**

The current node A is marked with a distance of 4, and the edge connecting it with a neighbour B has length 2, then the distance to B (through A) will be 4 + 2 = 6. If B was previously marked with a distance greater than 6 then change it to 6. Otherwise, keep the current value.

1. If all the neighbours of the current node are considered then mark the current node as visited and remove it from the unvisited set. A visited node will never be checked.
2. If the destination node has been marked visited (when planning a route between two specific nodes) or if the smallest tentative distance among the nodes in the unvisited set is infinity (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes), then stop. The algorithm has finished.
3. Otherwise, select the unvisited node that is marked with the smallest tentative distance, set it as the new "current node", and go back to step 3.

For a graph,

G = (V,E) where V is a set of vertices and E is a set of edges.

Dijkstra's algorithm keeps two sets of vertices:

S- the set of vertices whose shortest paths from the source have already been determined V-S the remaining vertices.

The other data structures needed are:

d - array of best estimates of shortest path to each vertex

pi - an array of predecessors for each vertex

The basic mode of operation is:

1. Initialise d and pi,
2. Set S to empty,
3. While there are still vertices in V-S,

* Sort the vertices in V-S according to the current best estimate of their distance from the source,
* Add u, the closest vertex in V-S, to S,
* Relax all the vertices still in V-S connected to u

**Relaxation**

The relaxation process updates the costs of all the vertices, v, connected to a vertex, u, if we could improve the best estimate of the shortest path to v by including (u,v) in the path to v.

This sets up the graph so that each node has no predecessor (pi[v] = nil) and the estimates of the cost (distance) of each node from the source (d[v]) are infinite, except for the source node itself (d[s] = 0).

The relaxation procedure checks whether the current best estimate of the shortest distance to v (d[v]) can be improved by going through u (i.e. by making u the predecessor of v):

**1.2 Pseudocode**

initialise\_single\_source ( Graph g, Node s )

for each vertex v in Vertices (g)

g.d[v] := infinity

g.pi[v] := nil

g.d[s] := 0;

relax (Node u, Node v, double w[][])

if d[v] > d[u] + w[u,v] then

d[v] := d[u] + w[u,v]

pi[v] := u

The algorithm itself is now:

shortest\_paths (Graph g, Node s)

initialise\_single\_source ( g, s )

S := { 0 } /\* Make S empty \*/

Q := Vertices( g ) /\* Put the vertices in a PQ \*/

while not Empty(Q)

u := ExtractCheapest( Q );

AddNode( S, u ); /\* Add u to S \*/

for each vertex v in Adjacent( u )

relax( u, v, w )

1. **Design of the program**

The program uses the java source file to read the input file, compute the connection table for a given router, calculate the shortest path for a given source and destination router, update the connection table and shortest path if a router has been removed.

The algorithm of the program is as follows:

1. Display the options for the user to enter.

(1)Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the Command:

1. If the choice is 1, ask the user to enter the file name along with the path.

* Read the file line by line
* Print each line
* Split the string with space as the delimiter and store it in tokens array.
* Each row value is stored as source router and column is considered as destination router and the edge values are stored in Edge object.
* If the file is not loaded properly, error message is displayed and asks the user to enter the command.
* If the file is loaded properly then it goes back to step 1.

1. If the user s choice is 2, ask the user to enter the source router.

* The router name should be in the format of R(number). Eg: R1 or R2 else error message is thrown and asks the user to give a proper router name.
* If the input file is not loaded it notifies the user to load the file and goes to step 1.

1. If the source router is given as input the connection table for given source router is displayed.

* The link from source router to all the other router is checked and the weight of the edges is also checked.
* Another copy of edge object is created and compare the edge values with the original one.
* Minimum distance is set as 0
* If the weight is 0 the source router and destination router is same.
* If the weight is -1, there exists no link between the source router and destination.
* If the weight of the edge is other than that, the weight is compared with the minimum weight.
* If the edge weight is less than the minimum weight, then the edge weight is updated as minimum weight.
* If the minimum weight is not updated, then it checks for the router which has the minimum distance from the source router and traverses to that router.

**Dist[ToNode]= minimumweight +Distance from current node to ToNode**

* The minimum distance is updated and the current node is marked as visited.
* This goes in loop till all the nodes are visted.
* The path from source router to other nodes are store in direct nodes if the source router has a direct link to the other router else the path is stored in indirect nodes.
* The first hop from the source router to other router is updated in the connection table.the connection table is printed and goes to step 1.

1. If the user choice is 3, ask for the destination router.

* If source router already exists then destination router is given as input .
* If there is no source router, the user is asked to enter the source router and then the destination router.
* The shortest path cost is calculated as in step 4 and the path is stored.
* The path and the path cost is displayed and goes to step 1.

1. If the user choice is 4, ask the user enter a router to be removed.

* If there is no network available, prompt the user to load the file and goes to step 1.
* If the file is already loaded,the router to be removed is given as input and the connection table of the router to be removed is displayed
* If the source and destination router is not available and if the user tries to give the source or destination router as the removed router, message is displayed to notify the user that the router has been removed.
* If the source router and destination router are already present and another router is removed then the connection table of source and destination router is displayed and the path and its cost is also displayed.
* If the destination router is same as the router to be removed then the source router connection table and the router to be removed connection table will be updated and printed. The path and its cost will not be displayed since the destination router is removed. The path from source router will be calculated only if the user again gives a destination router using the menu options.
* If the source router is same as the router to be removed then the destination router connection table and the router to be removed connection table will be updated and printed. The path and its cost will not be displayed since the source router is removed. The path from source router will be calculated only if the user again gives a source router using the menu options. If the tries to give the destination router again instead of source router then the message is displayed to the user that the source router has been removed.
* If the user tries to change the topology again by removing another node then a message is displayed to notify that the topology has been modified.
* To modify the topology again the input file has to be loaded again and it goes to step 1.

1. If the users’ choice is 5, then the message ‘Exit CS542 project. Good Bye!’ is displayed and program is terminated.
2. **Test Report for sample network file**

Input File:

0 -1 5 1 -1

-1 0 -1 7 9

5 -1 0 -1 4

1 7 -1 0 2

-1 9 4 2 0

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

1

Enter the filename:

sddas

Unable to load Input file

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

2

Please enter 1 to load the file

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

1

Enter the filename:

src/dijkstra/square.txt

0 -1 5 1 -1

-1 0 -1 7 9

5 -1 0 -1 4

1 7 -1 0 2

-1 9 4 2 0

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

2

Select a source router:

R0

The given router is not available

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

2

Select a source router:

R1

R1 Connection Table

Destination Interface

------------------------

------------------------

R1 -

R2 R4

R3 R3

R4 R4

R5 R4

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

3

Select a destination router:

R5

The shortest path from router R1 to R5 is R1 - R4 - R5 the total cost is3

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

4

Select a router to be removed:

R2

R1 Connection Table

Destination Interface

------------------------

------------------------

R1 -

R3 R3

R4 R4

R5 R4

R5 Connection Table

Destination Interface

------------------------

------------------------

R1 R4

R3 R3

R4 R4

R5 -

The shortest path from router R1 to R5 is R1 - R4 - R5 the total cost is3

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

2

Select a source router:

R2

The source router is not available

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

2

Select a source router:

R4

R4 Connection Table

Destination Interface

------------------------

------------------------

R1 R1

R3 R1

R4 -

R5 R5

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

3

Select a destination router:

R2

The destination router is not available

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

3

Select a destination router:

R1

The shortest path from router R4 to R1 is R4 - R1 the total cost is1

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

5

Exit CS542 project. Good Bye!

Input file

0 20 1 -1 -1 1

5 0 3 4 -1 -1

1 2 0 8 15 -1

-1 1 6 0 11 -1

-1 -1 2 10 0 9

7 -1 -1 -1 1 0

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

1

Enter the filename:

C:\Users\Dell\Desktop\input.txt

0 20 1 -1 -1 1

5 0 3 4 -1 -1

1 2 0 8 15 -1

-1 1 6 0 11 -1

-1 -1 2 10 0 9

7 -1 -1 -1 1 0

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

3

Select a source router:

R2

Select a destination router:

R5

The shortest path from router R2 to R5 is R2 - R3 - R6 - R1 - R5 the total cost is6

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

4

Select a router to be removed:

R2

R2 Connection Table

Destination Interface

------------------------

------------------------

R1 -

R3 -

R4 -

R5 -

R6 -

R5 Connection Table

Destination Interface

------------------------

------------------------

R1 R3

R3 R3

R4 R4

R5 -

R6 R3

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

2

Select a source router:

R3

R3 Connection Table

Destination Interface

------------------------

------------------------

R1 R1

R3 -

R4 R4

R5 R6

R6 R1

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

3

Select a destination router:

R4

The shortest path from router R3 to R4 is R3 - R4 the total cost is8

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

4

Topology is already modified...

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

5

Exit CS542 project. Good Bye!

Input File

0 1 3 -1 4 3 6 2

1 0 -1 3 -1 6 5 4

2 3 0 5 -1 3 5 7

1 1 3 0 5 -1 5 4

-1 4 5 7 0 3 4 4

2 -1 -1 3 2 0 3 5

3 -1 4 5 -1 4 0 6

2 4 5 3 7 -1 -1 0

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

1

Enter the filename:

src/dijkstra/matrix.txt

0 1 3 -1 4 3 6 2

1 0 -1 3 -1 6 5 4

2 3 0 5 -1 3 5 7

1 1 3 0 5 -1 5 4

-1 4 5 7 0 3 4 4

2 -1 -1 3 2 0 3 5

3 -1 4 5 -1 4 0 6

2 4 5 3 7 -1 -1 0

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

4

Select a router to be removed:

R7

R7 Connection Table

Destination Interface

------------------------

------------------------

R1 -

R2 -

R3 -

R4 -

R5 -

R6 -

R8 -

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

3

Select a source router:

R1

Select a destination router:

R8

The shortest path from router R1 to R8 is R1 - R8 the total cost is2

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

2

Select a source router:

R5

R5 Connection Table

Destination Interface

------------------------

------------------------

R1 R6

R2 R2

R3 R3

R4 R6

R5 -

R6 R6

R8 R8

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

4

Topology is already modified...

(1) Create a Network Topology

(2) Build a Connection Table

(3) Shortest Path to Destination Router

(4) Modify a topology

(5) Exit

Enter the command:

5

Exit CS542 project. Good Bye!

1. **Instructions on compiling and running the file**
2. The source code can be imported as java project in Eclipse IDE and can be run.
3. The jar file can be run in command prompt using the following command:

java –jar filename.jar. If the file is not found, go to the file path in command prompt and run the command.

1. Provide the full path for the input file.
2. In the input file, separate the elements only by single space. There should not be any extra spaces in the input file.
3. The router value should be entered as R1 starting from router 1(i.e.: R1 for 1, R2 for 2, etc.)
4. The topology matrix can be modified only once and to modify it again load the input file again. (The removed router is stored in a string and not in array so it will not store the values and update the topology again and again).