COMP4322: Internetworking Protocols, Software and Management

Course Project: Link State Routing

April 10, 2022

YOON Sun Ho (17086679D)

CHOUDHARY Muhammad Haris (18086809D)

Table of Contents

1. Usability Guide

Clone from github https://github.com/harisch1/Link-State-Routing.git

Make sure system is using JDK-18 and

cd src

Make sure test files are in the same folder.

Run “java LSRCompute routes.Isa A SS” for a step by step pathfinder.

Run “java LSRCompute routes.Isa A CA” to display only the final routes.

Execution modes

1. Single Step Function
    a. Uses an SS flag as an input argument for Step by Step print.
    b. Executes whenever a new closest node is found.
2. Compute all Function
    a. Uses a CA flag to ensure the SS mode is not executed.
    b. It displays the whole routing table at the end of the program.

2. Summary of LSR algorithm

The Link state routing method is a means of creating an optimal network topology of all routers in the internetwork. Each router perceives the network then makes a routing table depending on the topology from the information from its neighbors. Each router in the network shares data about its connections to its neighbors in the form of *flooding*.

Link State Routing has two phases. The first stage of reliable flooding tells all the routers about the local topology perceived. The second phase of path calculation finds the optimally short path from the current neighborhood information. The Link State Routing algorithm iterates till all routers have the same information and there are no updates in the routers.

2.2 Pseudo-Code

```
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
```

Figure 1. Dijkstra Algorithm

Figure 1 shows the methodology of finding the shortest path from a source node. There are two stages, initialization and looping to find the fastest path. The algorithm is run independently on each node to find the shortest path from the source to every other node in the network. In the initialization stage, the root node is initialized. Henceforth, the distance/weight

of the edge and the neighboring nodes are put into an equation to find the minimum among the other known paths. The table iterates till the shortest known path is achieved. This link state is then propagated to other nodes on the network.

2.3 Time Complexity

The Dijkstra algorithm has a big O of n^2. This is because there can be cycles in the graph and the shortest path algorithm has to go through E edges and V vertices. However, by using a min-priority queue, time complexity can be improved to (V + ElogV) [1].

2.4 Weaknesses

Link state routing is disadvantageous in that it has higher memory requirements than distance vector protocols because it creates and maintains a database and SPF tree [2]. Second, LSR protocols require more CPU processing because the shortest path finding algorithm finds the complete map of the topology. Lastly, the bandwidth requirements for LSR is high during the flooding phase or event-driven update phase. LSA packets are flooded through the network and this affects the available bandwidth on the network.

2.5 Strengths

The first advantage is fast network convergence. Once a Link State Packet is received, the packet is broadcasted/flooded throughout the whole network. Second, each node individually determines the shortest path to every node in the network. The topological map stores each node's mapping. Lastly, LSR does not send periodic updates. This means changes in the topology will trigger LSP sending throughout the network based on the change. This reduces unnecessary computation and updating [1].

2.6 Route Change Update

Since Link State Routing algorithm is event based, updates to a node triggers a subsequent flood of information to the segments in the network excluding the one which it was received [3]. All routers process the updates concurrently, hence, convergence of the network is reached faster as routers can receive and process updates in parallel.

3. Implementation

This section explains the java functions that implement the Link-State Routing. The two phases of reliable flooding and path calculation are successfully implemented.

3.1 Data Types and Variables

The LSR network topology is stored as a 2 dimensional integer array (Matrix). This stores the weight/cost of each link. The labels are converted to ascii to get the indices for the array.

A Character array is used to store the router labels. This is used for mapping the final routing results.

A Boolean flag to toggle between Single Step and Calculate All modes.

Integer Arrays are used to store the shortest path cost and the previous node/router in the shortest path.

A Boolean array is used to flag each visited node. This prevents any node from being checked multiple times.

Integers are used to mark the indices of the source and next closest node/router to help track each step.

A Stack is imported from java Arrays. This is used to push the path trace from the destination using the previous node and then pop back to print the correct path.

3.2 Functions

The main function takes in a string of arguments from the command line, parses the arguments and initializes the network matrix, and runs the dijkstra's algorithm on the initial

table. Based on the command line argument, SSmode is toggled. The program prints the output and ends the program once the result is found.

The buildMatrix function takes in a string datatype of the file path and inputs the network configuration into a 2 dimensional integer array. This datatype is then used globally for the later computations.

The dijkstraAlgorithm function takes in a 2 dimensional integer array and outputs the converged configuration of the input array. The distance and cost of the network is stored in respective local variables, visited, shortest and previous arrays. The local variables are first initialized to null and with each iteration through the nodes, the visited, shortest and previous values are stored for each source node.

The pressEnterKeyToContinue function takes in void and functions as an interface function to improve the usability of the command line interface. It moves the cursor down to the next line during argument input.

The SingleStepPrint function is only called if the SSmode flag is set. This function checks for each new found node and traces the new shortest path using a stack.

The printTable function takes in void and iteratively goes through the 2 dimensional array and prints out the path and the cost of the link state of the router.

4. Test Results

Following the usability guide above, 3 arguments need to passed in for the code to execute. We create 2 routing files with different number of nodes and distances. Following are the 2 tests for each output mode.

File ‘routes2.lsa’ contains the following data:

A: B:2 C:7
B: A:2 C:4 D:4
C: A:7 D:1 F:5
D: B:6 C:1 E:4
E: D:4 F:1 G:1
F: C:5 E:1 G:3
G: E:1 F:3

We execute ‘java LSRCompute routes.lsa E SS’ the output is as follows:

```
PS C:\Users\haris\Downloads\PolyU\Internetworking\Project\Link State Routing\src> java LSRCompute routes.lsa E SS
Found F: E>F Cost = 1           [Press Enter key to continue...]

Found G: E>G Cost = 1           [Press Enter key to continue...]

Found D: E>D Cost = 4           [Press Enter key to continue...]

Found C: E>D>C Cost = 5         [Press Enter key to continue...]

Found B: E>D>B Cost = 10                [Press Enter key to continue...]

Found A: E>D>C>A Cost = 12              [Press Enter key to continue...]


Source: E
A: Path = E>D>C>A Cost = 12
B: Path = E>D>B Cost = 10
C: Path = E>D>C Cost = 5
D: Path = E>D Cost = 4
F: Path = E>F Cost = 1
G: Path = E>G Cost = 1
```

File ‘routes2.lsa’ contains the following data:

A: B:20 C:22 D:6
B: A:20 C:2 D:3 E:3
C: A:22 E:6 F:6
D: A:6 B:3 E:8 F:25
E: B:3 C:6 D:8 F:12
F: C:6 D:25 E:5

We execute ‘java LSRCompute routes2.lsa D CA’ the output is as follows:

```
PS C:\Users\haris\Downloads\PolyU\Internetworking\Project\Link State Routing\src> java LSRCompute routes2.lsa D CA

Source: D
A: Path = D>A Cost = 6
B: Path = D>B Cost = 3
C: Path = D>B>C Cost = 5
E: Path = D>E Cost = 8
F: Path = D>B>C>F Cost = 11
PS C:\Users\haris\Downloads\PolyU\Internetworking\Project\Link State Routing\src>
```

5. Team Roles

Sun Ho YOON	1. Code - Dijkstra, CA Print, File Parsing 2. Documentation - LSR summary
Haris CHOUDHARY	1. Code - Dijkstra, SS Print, File Parsing 2. Documentation - Implementation 3. Test Results

6. References

[1] *Computer Network: Link State Routing Algorithm - javatpoint*. www.javatpoint.com. (n.d.). Retrieved April 10, 2022, from https://www.javatpoint.com/link-state-routing-algorithm#:~:text=The%20Link%20state%20routing%20algorithm%20is%20also%20known%20as%20Dijkstra's,other%20node%20in%20the%20network.

[2] *Informit*. InformIT. (n.d.). Retrieved April 10, 2022, from https://www.informit.com/articles/article.aspx?p=26129&seqNum=10

[3] Libretexts. (2020, July 7). *9.6: Link-State Routing-update algorithm*. Engineering LibreTexts. Retrieved April 10, 2022, from https://eng.libretexts.org/Bookshelves/Computer_Science/Networks/Book%3A_An_Introduction_to_Computer_Networks_(Dordal)/09%3A_Routing-Update_Algorithms/9.06%3A_Link-State_Routing-Update_Algorithm#:~:text=Link%2Dstate%20routing%20is%20an,such%20as%20those%20of%20ISPs.

[4] Wikimedia Foundation. (2022, January 2). *Link-state routing protocol*. Wikipedia. Retrieved April 11, 2022, from https://en.wikipedia.org/wiki/Link-state_routing_protocol#Calculating_the_routing_table