Simulation of Link-State Routing Protocol

Pooja Bhagwan Mankani

Illinois Institute of Technology

CWID A20354444

pmankani@hawk.iit.edu

Campus : Main Campus

Section No: 2

Kedar Kaushikkar

Illinois Institute of Technology

CWID A20355218

kkaushik@hawk.iit.edu

Campus : Main Campus

Section No: 2

Contents

Overview of Link State Routing Algorithm 3
Link State Routing algorithm: 3
Phases of the algorithm: 3
Reliable Flooding: 3
Challenges 3
Solutions 3
Flooding Example 4
Djiktra's shortest path algorithm 4
Steps implemented in the algorithm 4
Djiktra's Algorithm Example 5
Step 1 5
Step 2 5
Step 3 6
Step 4 6
Step 5 7
Complete 7
Modification of the topology 7
Implementation of the Link State Algorithm 8
Create a network topology 8
Build a connection table 8
Find Shortest Path 8
Modify Topology 8
Additional Features implemented 8

Overview of Link State Routing Algorithm

Link State Routing algorithm:

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).

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. The collection of best paths will then form the node's routing table.

This contrasts with distance-vector routing protocols, which work by having each node share its routing table with its neighbors. In a link-state protocol the only information passed between nodes is connectivity related.

Link-state algorithms are sometimes characterized informally as each router 'telling the world about its neighbors

Phases of the algorithm:

Link State Routing algorithm is implemented in two phases:

a. **Reliable flooding:** Tell all routers what you know about your local topology
b. **Path calculation (Dijkstra's algorithm):** Each router computes best path over complete network

Reliable Flooding:

a. Each router transmits a Link State Packet (LSP) on all links
b. A neighboring router forwards out all links except incoming
c. Keep a copy locally; don't forward previously-seen LSPs

Challenges

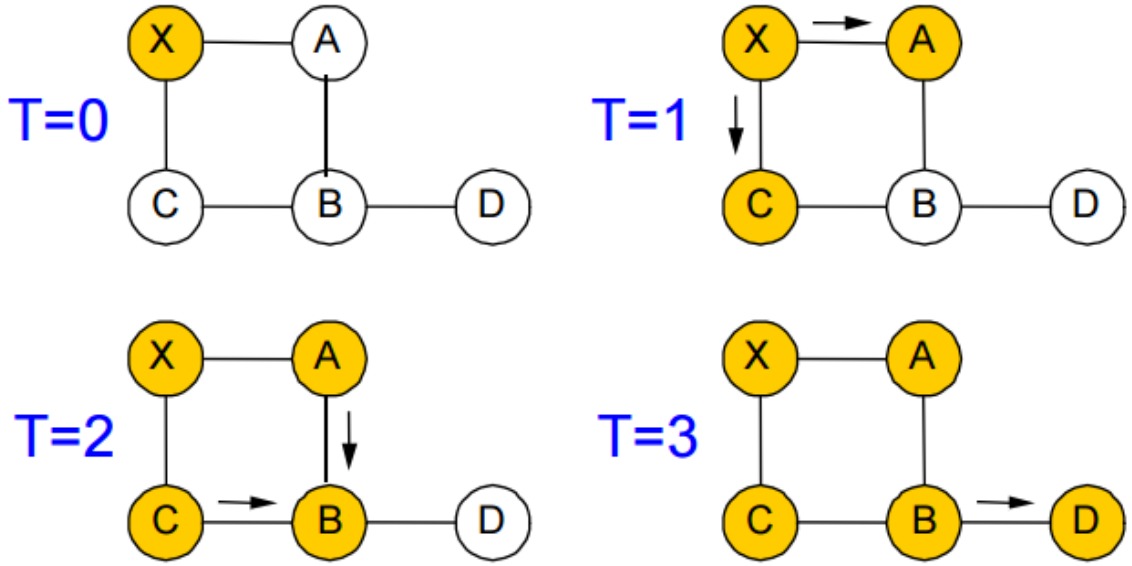
a. Packet loss
b. Out-of-order arrival

Solutions

a. Acknowledgments and retransmissions
b. Sequence numbers
c. Time-to-live for each packet

Flooding Example

a. LSP generated by X at T=0
b. Nodes become orange as they receive it



Djiktra's shortest path algorithm

Graph algorithm for single-source shortest path tree (find best route to all nodes)

Steps implemented in the algorithm

a. Assign to every node a tentative distance value: set it to zero for our initial node and to infinity for all other nodes.
b. Set the initial node as current. Mark all other nodes unvisited. Create a set of all the unvisited nodes called the unvisited set.
c. For the current node, consider all of its unvisited neighbors and calculate their tentative distances. Compare the newly calculated tentative distance to the current assigned value and assign the smaller one. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B (through A) will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, keep the current value.
d. When we are done considering all of the neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again.
e. 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.

f. 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 c.

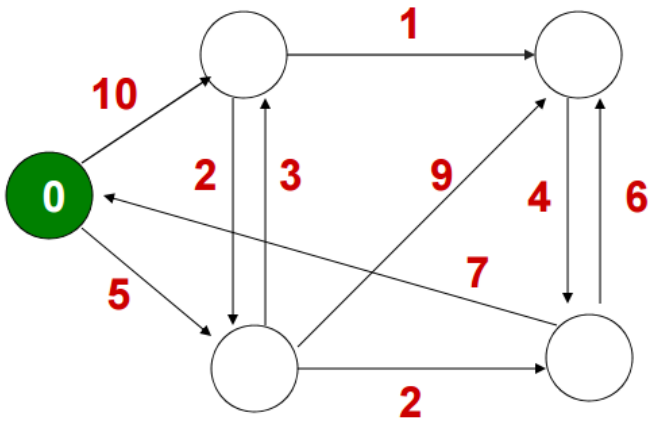
```
S ← {}
Q ← <remaining nodes keyed by distance>
While Q != {}
        u ← extract-min(Q)   u = node with lowest cost
        S ← S plus {u}                    ← u is done
        Within Q:
            for each node v adjacent to u
                "relax" the cost of v  is it cheaper to go
                                       through u?
```

5

Djiktra's Algorithm Example

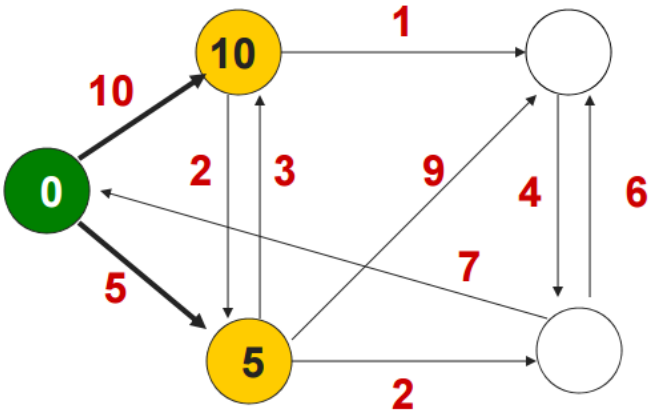
a. Green nodes are "confirmed"
b. Yellow nodes are "tentative"
c. We can add ourselves to "confirmed"

Step 1



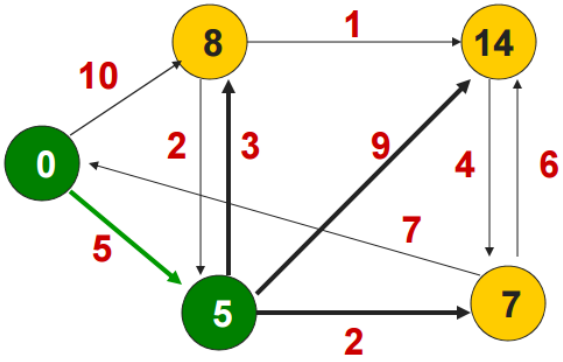
Step 2

a. First look at neighbors
b. "5" is cheaper than "10"
c. We can confirm path with cost "5"



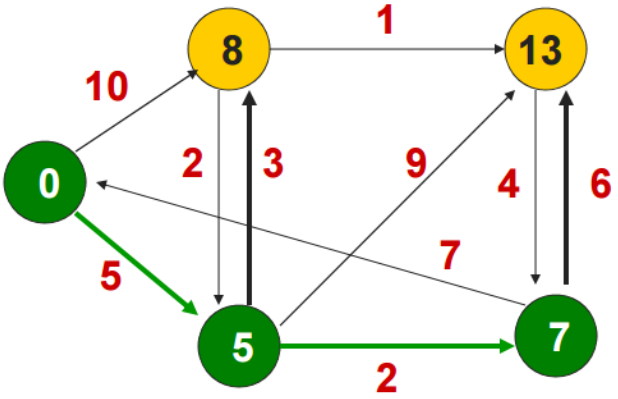
Step 3

a. Update costs
b. Look at 5's neighbors 7 is cheapest
c. We can confirm path with cost 7



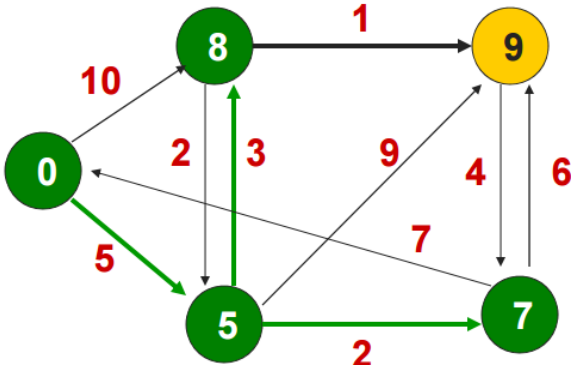
Step 4

a. Update costs
b. 7 has no new neighbors
c. 8 is cheapest
d. We can confirm 8

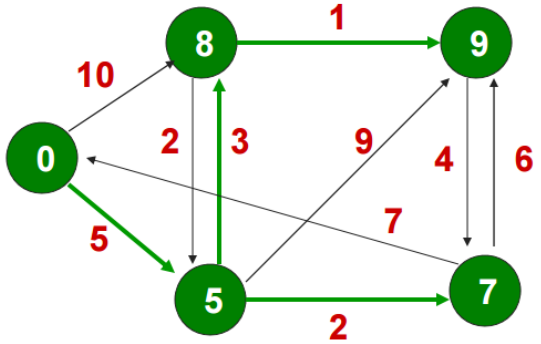


Step 5

a. Update costs
b. No new neighbors
c. 9 is cheapest
d. We can path with cost 9



Complete



Modification of the topology

a. LSPs carry sequence numbers to distinguish new from old
b. Routers only accept (and forward) the "newest" LSP
c. Send a new LSP with cost infinity to signal a link down

Implementation of the Link State Algorithm

The below steps are used to implement the algorithm:

Create a network topology

a. Load the file which has the adjacency matrix of the link state router topology.
b. The links which are not directly connected have a distance of -1.
c. In case there exists a link between routers then the cost is represented in the adjacency matrix

Build a connection table

a. The distances are checked between the links.
b. Based on the distances a connection table is created for each of the nodes
c. The minimum link of the node is checked and that is updated in the connection table
d. In case multiple links exist with the same minimum cost each of the links are updated in the connection table

Find Shortest Path

1. The previous nodes to navigate through the shortest path are calculated in the previous array.
2. In case there are multiple previous nodes that reach the shortest paths the multiple nodes are added to the array.
3. Based on the connection table and previous nodes the shortest path between the source and the destination is calculated. This is calculated using the Djiktra's algorithm .

Modify Topology

1. In case a router fails the distances to and from the router to other neighboring routers are marked as infinity.
2. Once this is done the connection table and the shortest path between source and destination routers are calculated

Additional Features implemented

Along with the basic link state algorithm the below features are implemented:

1. The user can input a topology file only with 8 or more than 8 number of nodes. In case the nodes are lesser than 8 the user is prompted to upload a file with 8 or more number of nodes.
2. Djiktra's algorithm is used to calculate the shortest path between the source node to the destination node
3. Incase multiple paths are present with the same cost the algorithm calculates multiple paths and displays all of them with the cost associated to it.
4. In case a router fails the user has the ability to remove the router. Once the router is removed the shortest path(s) is recomputed between the source and the destination router.
5. In case the source or the destination router is deleted, the user is asked to input either the source or the destination router and the shortest path(s) are recomputed.
6. A complete GUI is implemented to help the user simulate the link state algorithm