CSE Project 2, Linked-State Routing

UC Merced, CSE-160

Shrithik Saereddy and Dale Alabastro

Problem at hand

After finishing project 1, we are now given the task to implement linked-state routing into our code. Linked-state routing is a routing protocol used in packet switching networks for computer communications. In this lab, we are using the intermediate system versus the open-shortest path first system. We had to print out the link-state routing table specifically for this project on top of our neighbor and flood discovery.

Design Decisions

On top of making sure our neighbor and flood discovery functions correctly, we now had to implement link-state routing as mentioned above. In order to do so, we had to make new files in the interface as well as the modules in the "lib" section. We had to wire files together first of all in order for the program to function correctly. To do so, we wired two list interfaces. One is the neighbor list and the other has our linked-state protocol. Moving on, linked-state also required us to use dijkstra's algorithm, something we learned from CSE-100. To do so, we implemented that into "LinkstateP.nc". For background knowledge sake, dijkstra's algorithm is a greedy algorithm where it finds the shortest path starting from a starting node to an end node. Now, what makes dijkstra's algorithm special is that it can backtrack through the topological graph if it finds a path cheaper to take than the ones given. Also to note about our program, we had to use hop-by-hop pathing. In order to do so, dijkstra's algorithm was used to perform such a task.

Discussion Questions

1.)Why do we use a link for the shortest path computation only if it exists in our database in both directions? What would happen if we used a directed link AB when the link BA does not exist?

Answer: The reason why we use a link for the shortest path computation only if it exists in our database in both directions is because it saves on memory at the very least. It would be unnecessary to send a packet forward and then backward as well.

If we only used a directed link AB, the packet will only go forward and it will not go backwards (aka link BA). It has the benefit of being faster, but if we need to backtrack, issues can arise.

2.) Does your routing algorithm produce symmetric routes (that follow the same path from X to Y in reverse when going from Y to X)? Why or why not?

Answer: Due to the costs being the same, yes to an extent. It depends on the dijkstra's algorithm and how it is implemented. The algorithm, theoretically speaking, should take the same path in reverse due to the same costs.

3.) What would happen if a node advertised itself as having neighbors, but never forwarded packets? How might you modify your implementation to deal with this case?

Answer: Nodes only talk to their neighbors using messages. So, for this example, we will have node A, B and C. If node A wants to send to node C a message, it has to go through B. But, since B is, in this particular case, the bad neighbor, it holds the packet. However, node A is expecting a message back from C and if it does not get it, it will know that node B is bad. To modify our code, we would have to implement a case where node A has to go around node B somehow to send the packet.

4.) What happens if link state packets are lost or corrupted?

Answer: If the packets are not refreshed, then they will be dropped and forgotten. All the contents are basically cleared.

5.) What would happen if a node alternated between advertising and withdrawing a neighbor, every few milliseconds? How might you modify your implementation to deal with this case?

Answer: If a node is alternating, it would be considered slow in the regards of updating, thus packets could come slowly or even broken/corrupt. In order to fix this, we could implement a way to delay in order to give us more time to send packets.