Renee Helfert
John Villalvazo
CSE 160 - Fall 2020
Project 2
Due: 10/13/2020 @3pm

Discussion Questions

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?

We would have missing packages and errors popping up due to nodes attempting to send messages through unavailable channels. Things would be lost and nodes might assume they have a neighbor to send to when they do not. This link is a form of "check" that prevents this error from happening, thus, this makes having a stable link between two nodes is vital to prevent this.

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?

Our routing algorithm produces symmetric routes between nodes. It most likely has to due to the use of Dijkstra's algorithm and how nodes that are active are fed into the algorithm in increasing numerical order, causing symmetrical routes between nodes.

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?

You could add a check between nodes that are neighbors to the non-forwarding node. If the "check" notes that a node is not forwarding your packet (aka, sending a response to the original node that its packet is being sent out again), then consider that node "active" ,but not able to be part of the routing table and send that info to all other nodes.

What happens if link state packets are lost or corrupted?

The routing table may be incorrect about whether or not a node exists and packets may not reach their desired destination or take a more costly route. Either way, a certain node may not have the most updated Link State info regarding its neighbor until it physically receives the Link State packet, or doesn't. At which point the routing of packets then comes to the choice of the TTL being out of cycles or taking a longer route within the topology to reach its intended destination.

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?

There might be two cases to this:

Case 1: The node broadcasting the “flickering” node is that node’s only neighbor, in this case, the code could be modified to assume the node is simply “offline” until the situation changes.

Case 2: The “flickering” node has multiple neighbors, in this case, nodes could be compared to see if the node itself is “flickering” or not, if there is a disagreement, then the neighbor node who has a more consistent broadcast could be taken as truth. If all neighbors say the same thing, then again, this node could be broadcasted as “offline” until the situation changes.