Aditya Lakkoju, Somya Shah
CSE 160
Mar 3, 2022

Discussion Questions:

1) We have used only bi-directional links in the calculations because the routing tables must be interchanged both ways. In other words, if the link was a directed link AB, but BA was not present, node A would not only be able to obtain neighbor search from B, but also would not receive the LSP. Therefore, A will not be able to use B as the next hop because it has no idea who has access to B.
2) Yes, because the "cost" is the same between each node, which means that the cost of getting from neighbor to neighbor is always of the same value. Hence, producing symmetric rates.
3) If a node advertises its neighbors, the shortest path count will most likely mark the node as the next hop. Therefore, a problem will arise, in which the node will receive messages intended for its neighbors, but will never forward them to those neighbors. This, those neighbors will never receive those messages. To deal with this, the algorithm would have to heed the ack messages (or lack of them) and, after some time, discard the node in question as the next hop, and fired a different hop.
4) Then the link state tables will not be updated. This means that some nodes will take longer to fill their tables, and this slows down the networking infrastructure.
5) Depending on the coordinate of the neighbor search, what might happen is that it aligns with the "withdrawn" term. Therefore the neighbor will time-out, and will no longer be marked as such. Thus, this link will cause the state table to change, and therefore cause the network to pretend that it does not exist. An easy way to combat this would be either a small update window, or a random update window.

Design:
We started by adding Project 1, neighbor discovery and network flooding capability, now we try to implement link state routing in our network. An LSP (Link State Packet) is sent only when a neighbor update is detected, either by removing or adding a neighbor, it is detected by the neighbor list and the neighbors list. Each LSP has a sequence number and TTL so that they do not flood the network forever and each LSP packet has a link state protocol and AM_BROADCAST_ADDR as its destination. Neighbor discovery also uses AM_BROADCAST_ADDR as the destination but using the LINKSTATE protocol allows any node to differentiate link status packets from neighbor discovery packets. Each node sends LSP until the network converges and the Update Neighbors timer allows us to know when the network changes. Upon receiving an LSP, each node checks to see if it is an old LSP, a new one, or the same one that we have already received. Compare the sequence number of the LSP to the most current LSP received from the LSP sender. Let's say, if it is a new LSP, we update lspHashMap. Finally, each node's lspHashMap will contain each node in the network with the neighbors of each node and the associated cost. We can use lspHashMap when trying to calculate the shortest path for each node. Once a ping event occurs, we run Dijkstra's algorithm using a temporary list, a confirmed list, and the full lspHashMap. Then we get a confirmed list which will be our routing table and we can choose the next hop to forward the packet. This forwarding continues, running Dijkstra for each node that receives this packet, until it reaches its destination node, this destination node then produces a ping answer using Dijkstra's algorithm to build its routing table. sends and then forwards it to the next hop. Now, instead of relying on the network to deliver a packet to its destination, each node sends a packet to the next hop according to the destination of the packet.