Building upon what was already built from project 1, neighbor discover and ability to flood a network, we now try to implement link state routing into our network. A LSP(link state packet) is only sent out whenever a neighbor update is detected, either by a removal or addition of a neighbor, this is detected by neighborList and listofNeighbors. Each LSP carries a sequence number and TTL so they do not flood network forever and each LSP packet has a LINKSTATE protocol and AM\_BROADCAST\_ADDR as its destination. Neighbor discover also uses AM\_BROADCAST\_ADDR as destination but using protocol LINKSTATE allows any node to differentiate a link state packet from a neighbor discovery packet. Every node sends out LSPs until the network converges and the updateNeighbors Timer allows us to know when the network changes. Upon receiving an LSP, each node checks to see if this is an old LSP, a new one, or the same one we already seen by comparing the sequence number of the received LSP with the most current LSP received from whoever sent LSP, if this is a new LSP, then we update lspHashMap. Eventually, every node’s lspHashMap will contain every node in the network with the neighbors to each node and respective costs. We can then use the lspHashMap when trying to calculate shortest path to each node. Once a ping event occurs, we run Dijkstra algorithm using a tentative list, a confirmed list and the completed lspHashMap. We then obtain a confirmed list which will be our routing table and we can pick the next hop to forward packet to. This forwarding continues, running Dijkstra for every node that receives this packet, until it reaches its destination node, this destination node then sends a ping reply using Dijkstra’s algorithm to construct its routing table and then forward it to the next hop. Now, instead of relying on flooding the network to get a packet to a destination, each node sends a packet to next hop according to destination of the packet.