**DESIGN PROCESS AND DECISIONS**

For project 2 we added three files linkstateP.nc, linkstateC.nc and linkstate.nc, we also needed to make some additions in the node.nc file to have the link state functional. In linkstateP I implemented dijkstra’s shortest path algorithm. This was to ensure the most optimal shortest path in the network with uniform link costs. Using a routing table data structure to store destination next-hop with all the entries have the dest node id, the next hop neighbor, the path cost and validity flag. This allows for O(1) lookup time with forwarding packets. In the computingroutingtable dijkstras algorithm, has distance and predecessor arrays to find the shortest path from the current node to all possible destinations. It iterates through the LSDB to find the nodes that have not been visited yet with the shortest path and updates distances to their neighbors and then goes back through the predecessor array to find the first hop neighbor for each node. The table is recomputed every 10 seconds and it broadcasts link state advertisements and when the new ads are received from the other nodes. We also added getNextHop() to query the routing table for specific destinations.

The changes that we made in node.nc was that some of the packet forwarding was changed from flooding to routing, but this is only for unicast packets. When a node gets a ping or pingreply packet it will query the getnesthop to the find the correct and most efficient neighbor. We did this in order to avoid just full flooding packets to all the neighbors to have a more efficient system. So if a valid/ good route exists it will send the packet directly there otherwise our program will just the use the flooding we previously implemented in project 1.