Andrew Reyes
Dongjun Cho
CSE 160

Project 2: Link State Routing

Continuing to build upon our flooding and neighbor discovery implementation from project 1, we started by adding a new timer that fires and calls a new makeLSP() function. Here is where our link state packet is created and flooded out through AM_BROADCAST_ADDR with a PROTOCOL_LINKSTATE. Each LSP contains the ID of the node that created the LSP, a list of its directly connected neighbors, with the cost of the link, its sequence number and a TTL for the packet. In receive, we search for the PROTOCOL_LINKSTATE where we check to see if there is already a copy of that LSP stored. If it already has a copy then we compare the sequence numbers and store the LSP with the larger sequence number, meaning that it is a newer LSP. If doesn't have a copy then we simply store it in our RouteTable. After receiving the newest LSP the node then sends a copy of that LSP to all its neighbors. Eventually all the nodes in the network will have the newest information of the LSP with its cost. A node will generate an LSP under two conditions, either an expiry of a periodic timer or a change in topology such as a neighbor going down. To calculate our route table we use two lists of our LinkState struct, tentative and confirmed, and our current Route Table. Dijkstra's algorithm will be ran every time a timer fired event occurs. This will allow for each node to compute its routing table when its made. Our dijkstra algorithm follows the adjacency matrix version of the algorithm. We maintain two sets, one for the nodes in the shortest path and another set of nodes that are not yet included in our shortest path tree. Once our hashmap is completed we obtain our confirmed list which is our route table and can find the next hop to forward the packet to. Then the forwarding should continue until it reaches its destination node.