Christian Doud
Phien Nguyen

For starters, we needed to implement the flooding function that would fill the network, so in node, we call a timer every “n” ms would send a message to the whole network with payload anything as we are not deciphering the payload from the node but looking for neighbors. We also created two protocols where 100 corresponds to the neighbor discovery request responsible for finding the nearby neighbor. The network itself would not forward any message as the neighbors are the nearby nodes therefore if the node furthest away can’t reach it, then it's not the neighbor node. In Node, for the event where the node receives the packet, it would check the protocol to see if the packet is asking for a request for a neighbor. There we would return a neighbor request acknowledgement back to the src given in the packet. This “ack” packet would contain a protocol specifically for neighbor requests acknowledged at 101. Back in node, it would check if the protocol similar to the request if it matches 101 and parse the packet to store the neighbor.

To store the neighbor, the node will check first if the neighbor is not known in the neighbor table with size 20 and each neighbor is given max ttl of the given network where the ttl doesn’t stop at 0, would go back to 255. Then it will save the 1 at the neighbor index in the neighbor list. The function would print the neighbor so to tell that the node is storing each neighbor. Of course the function would store one a time and every new neighbor would update the list and print the neighbor table rather than getting the same “ack” from the same neighbor.

All of this “req” and “ack” is queued within the node through event which calls the modules function so it doesn’t skip over important code. Everytime we send a packet, we make sure to decrease the ttl as to make the sure the package is not immortal.