Design Process

Neighbor Discovery

The first step in having each node discovery its neighbors was to initiate a timer after each node is finished booting. This means adding something like timer.periodic(10000); and implementing a timer.fired() method; once a node's timer has fired it will call neighborDiscovery() which will first empty the node's NeighborList. The reason for this is to account for neighbors that have dropped out of the neighbors range. After the list is emptied, the node sends a broadcast message to all nodes in its range. Each node upon receiving a broadcast message adds the source to its Neighbor list, because if it was able to receive a message then that means they are Neighbors. This can run continuously but I added a boolean flag after each neighborDiscovery() call from timer.fired() so that it only runs once. This is correct under the assumption that the topography does not change. I also added a command to the neighbor module which will return a boolean on whether or not a node is a Neighbor of the node requesting it; This is useful later. The packets that are sent via Neighbor discovery are sent with a protocal of 255 to identify them as discovery packets so that they are not flooded.

Flooding

when the timer is fired it will also call the flooding if the TOS_NODE_ID is 1 which will start the flooding at source 1. the sequence is going to be used to tell if a node has already received a packet before. Upon receiving a packet who's protocol is not 255 (this would mean it is a discovery packet) and who's TTL is greater than 0, then flooding.flood(pack) is called which forwards the pack to all of its neighbors if the previous pack it had received does not have the same sequence. Flooding can also be called using ping in the TestSim.py with the use of the NeighborList. inside of ping(pack, uint8_t) it will confirm that the destination is a neighbor before sending it directly. if the destination is not a neighbor then it will broadcast and begin the flooding protocol. A list could be used to store the previous sequences that a node has received but I used a uint8_t because for this application and with the TTL to ensure messages don't circulate forever the trade off of memory vs sending packs when they do not need to be seemed worth it.