Renato Millan

Abdias Tellez Benitez

Design Decisions

The design process for our Project #1 started with analysis the skeleton code and referencing the Tos-nesC-Programming document. We studied the data structures and the includes files, then we decided that we need to add a couple of functions. In data structures, we change the listC.nc to include a push back drop; the push back drop is a function that is the same as push back with the ability to drop the element held the longest if the container is full. This function is used to aid us in the Flooding process. In includes, we add a new file called packet_id.h that is a packet that just has a sequence number as an ID. In the lib's interface, we created the interface file for flooding and Neighbor Discovery that contains command/calls we can use in node.nc.

Neighbor Discovery was a bit challenging to implement because our program needs to differentiate between a normal PING from a PING sent. We created the function a ping handler that has a Ping and a Ping reply. We created a protocol handler that would change a neighbor discovery packet's protocol (PING to PINGREPLY). If the node was ping it would add to a Hash map called Neighbors and change packet to have a ping rely. Furthermore, If the node has already reply, but was not added to the hash map. Then, the protocol ping reply would just add the node to the hash map. One implementation we had trouble with was a way to determine if a neighbor node has failed/die. We decided to have a timeout integer variable that would be stored at each node in the hash map. In addition, we created a method that would decrease the time out value by until it reaches 0 and removes the node. The node is removed because it has failed to reply to the last 5 neighbor discovery packets. Furthermore, we added a method would turn packets into a neighbor discovery packet and would broadcast the neighbor pack with its address. The last part of the implementation was easier than the protocol handler because I just need methods to called when a node receives a neighbor discovery packet, return list of neighbors, returns the number of neighbors, and prints the list of neighbors at the node n. Finally, we included a component to introduce randomness to our Neighbor Discovery timer so each node would not broadcast at the same time and cause collisions.

In flooding, we had trouble with how to implement and we are in the process of finishing the channel. We decide to include in the skeleton code was a duplicate check, valid check, the send flood method, and the flood method. The duplicate check is the packet is not flooded indefinitely and the valid check call duplicate check to verified there is no duplicates in nodes. The send flood broadcasts the destination. The Flooding.flood floods packets into its neighbor and calls the push back drop with pack ID. We must finish implementing a TTL field to avoid loop forever, a node table, use neighbor discovery to get neighbors (connecting), and a link layer module. We don't have flooding working correctly yet, however, after the added methods it should run and flood the packets.