Jose Herrera

09/24/2017

CSE 160-02L

Proj 1 Questions

The biggest modifications are done in the Node.nc and NodeC.nc files, specifically the Receive event in Node.nc. For flooding to work, we needed the nodes to know what to do when they receive a packet, there are two cases for this, they receive a packet meant for them and they receive a packet not meant for them. When a node receives a packet not meant for them, they should simply repackage that packet with TTL-1, the node will then push this packet onto its seen packet list and rebroadcast that packet using the broadcast channel. My code will decrement TTL each time a packet is repackaged and check to see if a packet has a TTL of 0 and drop it if it does. It will resend this package on the broadcast channel and the nodes that hear this message will also figure out whether the message was meant for them or not and act accordingly, allowing the functionality of multi-hopping. My code will first check if the packet coming in has a protocol ping or not, if it does have a protocol ping then it will check to see if the packet received was meant for this node or not. If it was meant for this node then we print out a dbg statement saying that the message has arrived and add it to our list of seen packets. Other wise, it resends the packet and adds to seen list. checkPacket(pack Package) will check and add to seenPackList.

For neighbor discovery, I wired a timer to NodeC.nc and had it starting periodically at 100 s intervals. This periodic timer triggers a fire event which called my neighborDiscovery() function and allows for periodic checks and updating a node’s neighbor list. For the neighbors, I created a struct that held a neighbors node ID and its so called life. Every time a node received a ping from a neighbor trying to discover them, the node will go through its NeighborList to see if this neighbor that is pinging them is on the list or not. If neighbor node is not on the list, then the node is added and that neighbor nodes life = 0. Every other neighbor node in the NeighborList will get there life updated by 1 each time a new ping for neighbor discovery to a node is received. So the higher the value of a neighbors life is, corresponds to the amount of pings it has been since we have last heard from this neighbor in the NeighborList. I picked a Neighbor life greater than 5 to be considered enough pings to justify a node to dropping that neighbor from their neighborList as that neighbor could have been taken off the network. Everytime a neighbor is heard from again, their life is resetted to 0. In order to determine who's a neighbor and who's not, each node needs to be able to respond correctly to the packet type or protocol type. If it got a ping, then it simply packages a ping reply with its own id as the source and sends it back to the sender without flooding the network though. If the node receives a ping reply, that means that it has heard back from a neighbor and if the node finds the neighbor already in its neighbor list, then it will reset the neighbor’s life. Otherwise, it's a new neighbor and is pushed to neighbor list.