Shyffer & English

Despite NeighborDiscovery being referred to as the “second part” of this project, we decided to tackle it before Flooding, because knowing your neighbors makes forwarding floods much more efficient. The biggest roadblock was effectively understanding the skeleton code and the proper syntax for passing information from place to place. Beyond that, progress was mostly smooth.

With neighbordiscovery, we had a timer set so that nodes would initiate discovery one after the other, and then they pack up a packet with a specified payload and broadcast it. Whenever a node receives a packet with that payload, it checks the sender, identifies it as a neighbor, and sends a specified response. Nodes then also identify neighbors based on that specified response, so every possible interaction between neighbors results in them identifying each other. Neighbors are tracked in an array, where the array is initialized to all 0s, and as a new neighbor is identified, the earliest zero-entry in the array is set to the neighbor’s ID number. We also could have accomplished this with a hash table, but the differences between the two are trivial with networks of this size.

Flooding was a more involved process. We originally created a flooding header, but we realized that all but one of the fields are redundant in that header: the original source must be tracked as well as the immediate source, but TTL and sequenceNumber are already present in the default packet header, so there was no need to repeat them. Therefore, the delivered version of the project just embeds that original source in the packet payload, and does not have a complete header. We do also have a version with a complete header at an earlier commit in our repository, but we decided to go with this one to minimize redundancy.

Each node keeps a list of the highest sequence numbers it’s seen from each other node in flooding, and will consider a flood to be a duplicate if it has a sequence number that is equal to or below the highest seen so far. We also used the PING and PINGREPLY protocols to distinguish between floods and ACKs. When a node receives a packet that is not used for neighborDiscovery, it considers it a flood, and sorts it based on whether it’s a flood or an ACK, and based on whether the node is the intended target or not, as well as checking the TTL and whether it’s a duplicate. If the packet needs to be forwarded, we have a function which checks the list of neighbors from neighborDiscovery, keeps track of the immediate sender, and forwards it to each neighbor other than the sender.

For sending ACKs, we create a new packet with the same sequence number but using the other protocol, reassign the originalSrc and the destination appropriately, and send it out. That way the original sender can know which node is sending them an ACK, and for which packet.