Aaron Romero

CSE 160

To complete the routing project, I started by refactoring the flooding module to be able to send a packet with an input protocol. Then I just made a switch case statement over the input protocol, and for this project the only one that matters is PROTOCOL_LINKSTATE. All it does is use the input protocol as the protocol header for the packet. Also, I made it so that AM_BROADCAST_ADDR can be used to make all nodes receive packets, read them, and then pass them on via flooding. Once this was finished, I had to use the NeighborDiscovery.neighbors command to help the Linkstate module know what neighbors it has. I compressed this information into one bit per neighbor, so if there are 40 nodes then 5 bytes will be stored with each bit representing whether Node X is a neighbor. Then I just used these 5 bytes as the message for the flood packet, and set each node to start sending out neighbor information packets 100 seconds after start() is called using a timer. Once the packet is received, it goes into Receive.receive in node.nc, which reads the protocol in a switch case. In the case of protocol_linkstate, we call Flooding.handleFlood, which will see that it has a linkstate packet, and send it to the linkstate module with linkstate.receiveUpdate, which will then update each node's 2D array of neighbor values. This array is length N * ceil(N / 8), where N is the number of nodes. Note that ceil(N/8) is the number of bytes required to store neighbor values for a singular node. Once all neighbor values are computed, we call djikstra's algorithm and create a routing table which simply dictates which neighbor packets are sent to on the next hop, and this logic is managed in LinkState.handleRoutingPacket. I also use a second protocol called DIRECTROUTE_PROTOCOL for packets which will need to be directly routed rather than flooded.