For Project 2, me and my teammate implemented a link state routing module that consolidates all the routing tables received from each node into a LSDB. We used neighborDiscovery to discover neighbors and fill in the link state routing tables. We then used flooding to flood the packets to each node. After receiving link state tables from each node and putting it into a LSDB, we then run Dijkstra's shortest path algorithm to produce a new optimized routing table, which is then flooded to all other nodes, so every node has the optimized routing table. Link state payloads contain a sequence so that nodes can update their link state if the origin is already recorded and the sequence is higher than the one stored, then it can replace it and update it. The payload also contains a neighbor count which records the number of neighbors as well as an array holding all the neighbor IDs. I also send out linkstate packets every 10 seconds, to ensure that my LSDB is up to date, but isn’t updating too frequently. We also implemented a packetpayload struct for link state, using a new struct instead of the provided pack struct, so we could put an array of the neighbor nodes into the packet. Our design process also ensures bidirectional forwarding, so node A must be able to hear from node B and vice versa. We also decided to set the cost of sending a packet as just the number of hops until it reached its destination, this was the simplest method to implement and keep things organized for us.