Renee Helfert
John Villalvazo
CSE 160 - Fall 2020
Project 2
Due: 10/13/2020 @3pm

Project 2 Design Process and Decisions

Our goal for project 2 was to implement Link-State Routing in TinyOS that would work for networks of topologies of different sizes and would adjust for nodes that leave or join the network. In its current state, all the routing and neighbor discovery/flooding all happens in the same sending channel on Node.nc which may bring issues in future projects, however, we recently learned how to implement new sending channels and hope to move routing to a new module with its own sending channels to prevent clogging of the network.

We started the project by creating the necessary structs to hold the Link-State and Routing table for each node, as well as converting our Project 1 neighbor discovery to a hash for easier searching and recovery (packet checking was also converted to a hash). We also chose to implement Dijkstra's algorithm for finding the shortest path, since the connections between nodes are undirected, it is not the fastest algorithm, but we were able to implement it and it works for our intended solution. Based on our implementation, the efficiency of the time taken to both discover neighbors as well as find the link-state between them is not where we would like for it to be, but like mentioned in the previous line, we hope to implement a new sending channel within a new module that would allow this run time to be significantly reduced in the future implementation (Project 3). Along with the decision of hard coding Link-State Routing, for the time being, the reliability of our code is scalable to any topology that is presented to it. This is coupled with the addition of being able to handle adding/subtracting nodes in the topology.

Overall, we believe our project 2 submission is a solid attempt at being able to route packets throughout a given network. We have the reliability of our code being hard-coded to be able to handle any situation that is given, though we are aware of the lack of speed and efficiency that our code possesses. The choice to stray from modulating these new implementations was cut due to our lack of knowledge in how to implement a new sending channel. We did not want to risk having more collisions or issues if we had continued with a modular approach as we wanted to submit on time and focus on optimization at a later date.