UCMerced Fall 2024 - CSE160 - Computer Networks
Project 1 - Flooding and Neighbor Discovery
Project Team: CJ Cox, Aaron Nam
Professor Cerpa

Design Process & Decisions

In our first stages, of figuring out our designs and decisions we spent numerous hours studying how the skeleton code operates, what functions are doing what, how to send a message through TinyOS, etc. In about two weeks, we were able to send messages and figure out how each module and interface worked. We learned how to modulate and wire nodes together which was a crucial step in learning how to operate these nodes and how to connect other necessary modules to forward our progress with making a Flooding implementation and a Neighbor Discovery. We created two separated python files, both will produce independently outcomes and printouts for now.

In Flooding, our idea was to track of the sequences of the packet when operating flooding. We also wanted to make sure to add an implementation of Time-to-live, so that the packet message does not need to stay any longer than it will AM_BROADCASTING to all other nodes. Since it's a flooding route it just wants to send to the package to everyone, so the idea for future design is to create a way to unravel the package and get the contents in it, but for now the idea was simple give the package to all the nodes (no nodes can discover whats in the packet). We want to display that after a node received a packet it will forward to the next one also displaying the TTL status along with it to ensure how fast nodes are passing the packet. Lastly, adding a way to check duplicates was important after a node receives a packet again it will drop that duplicate till the next sequence of flood takes place.

For Neighbor Discovery (ND), our basic idea was that each node would be running ND constantly on a timer. With that in mind, we decided that ND would rely on each node keeping track of the ND pings sent from other nodes instead of relying on other nodes replying to ND pings. This way, our process was simplified by removing the need for additional protocols and for reading payloads. Our neighbor discovery is currently just rudimentary, as it only keeps track of the last known ping from neighbors. We set up a hash map that would use the neighbor's node id as the key and the sequence number at the time of the last ping from said neighbor. From there any node who's most recent ping was at most 5 sequences ago would be considered a neighbor.

UCMerced Fall 2024 - CSE160 - Computer Networks
Project 1 - Flooding and Neighbor Discovery
Project Team: CJ Cox, Aaron Nam
Professor Cerpa

Discussion Questions

1. The benefit of event-driven programming is that it reduces the overall workload on the computer by allowing each function to occur asynchronously and allowing for more modular design, such as keeping functions on timers. The downside is that debugging can be harder to asynchronicity and possible cascading caused by events triggering other events.
2. To prevent packets from circulating indefinitely in a flooding implementation, track them by giving unique sequence numbers every time a pack is sent and set a static number for the maximum number of sequences it can go. This will prevent the packet from being sent to the same node too often. The TTL (Time to Live) field provides a way to keep a packet forwarding. By decrement every time a packet gets forwarded to a node. If it was just flood checking it will loop infinite and no TTL checks means that the packet can stay in the network for a lot longer than it probably needs to.
3. In the best-case situation, each node should send and receive one packet. In the case of Topology a line network would be an ideal scenario. The worst-case scenario would be a network with no priority of nodes, so it will rapidly flood responses to all other nodes.
4. Each node can use DFS from the information gathered by neighbor discovery. Whenever a package is sent, each node would then know the next neighbor to send a package to such that the package gets to its destination node in the shortest time, and it wouldn't require sending the "map" in the package itself.
5. There weren't any design compromises that were caused by the skeleton code, so there aren't any design decisions that we would make differently.