Valentinno Cruz

CSE: 165 Spring 2022

02/08/2022

Discussion Questions

1. Describe a pro and a con of using event driven programing.

**Nodes in a given system are permitted by event driven programs to be independent without any type of specified code written to link groups in a system. therefore, a node could put out a packet without the need of a function on another node to trigger the packet to be processed. However, even driven programming makes things more difficult to digest specially the code within a system. Its hard to tell if a node is simply reacting to another node or not.**

1. Flooding includes a mechanism to prevent packets from circulating indefinitely, and the TTL field provides another mechanism. What is the benefit of having both? What would happen if we only had flooding checks? What would happen if we had only TTL checks?

**Having both TTL and flooding checks helps stop packets from continuously looping infinitely. These two work together in all network sizes, TTL for big networks and flooding for smaller ones. If we only used flooding checks it would work great for smaller networks but very slow for larger ones. This is because it visits it individual node twice prior to the packet even disappearing. If we only used TTL it would be the reversed. Good for large networks and smaller ones would not. So having both is best for all scenarios**

1. When using the flooding protocol, what would be the total number of packets sent/received by all the nodes in the best case situation? Worse case situation? Explain the topology and the reasoning behind each case.

**Best case scenario would be one packet sent outward and only one inward. Worst case would be so many packets that each node would flood the next neighbor, this means all the nodes would get the packet. For example. If node A wanted to send a packet to node B and they are neighbors. Node A would send a message to node B, then it would send a packet to its neightbor node B. this is the best case scenario. Worst case scenario is if node A wanted to send a message to node B, node A would send and receive the same number of packets as the number of nodes in that system.**

1. Using the information gathered from neighbor discovery, what would be a better way of accomplishing multi-hop communication?

**Taking neighbor discovery into consideration, we can make an algorithm to calculate the shortest distance, if the nodes tell one another who their neighbors are. This make multi-hopping possible and makes the network run more efficiently. This is a much better option than flooding. It would allow far less packet transfers and increase the data used per node.**

1. Describe a design decision you could have made differently given that you can change the provided skeleton code and the pros and cons compared to the decision you made?

**I would have changed the CommandHandler.ping() from the skeleton code. It could have been used to ping any or all packets which were sent to AM\_BROADCAST\_ADDR. It would fllod by allowing nodes that got that ping packet to immediately return it.**

Design write up

Neighbor Discovery/Flooding

To find the surrounding neighbors, I used timers in m.s. which sent out a command to the next door nodes. Keeping in mind AM\_BROADCAST\_ADDR would make sure all neighbor nodes would receive this command and resend it with the appropriate identifier as a list. Nonetheless, after looking into this, hash mapping seemed could be implemented much easier for better utilization in other projects. The network would have been familiar with the packets and neighbors it encountered previously, and wouldn't need to look for it again.

The list of identifiers continuously checks if the neighbors are still alive. This is done within 3 unresponsive Reponses from said neighbor, if unresponsive the neighbor is taken off the list. Hash mapping could have been better when searching through this list of available neighbors. This could of had a much better RT. Nonetheless, this was much easier said than done.

To check if packets were lived for too long I used timeouts to determine if the packets had circulated long enough. Keeping in mind that TTL would keep the network maintained of unwanted information.

Building this project correctly in order to have a smooth transition to other projects is of the utmost importance. Other implementations will build on this project so improper execution would make it harder to include other additions. Lastly, NesC is unlike any programming language I have ever used before, so certain implementations available in other languages like C++, and python arent used in NesC.