Can Olmezoglu - 2269236
Bugra Veysel Yildiz -2104601

INTEGRATION PROJECT REPORT

What did we change after the demo?

In the demo, we were informed that our multihop broadcasting implementation (despite working otherwise) did not account for changes. Hence, we implemented a quick fix. We added a 120 second timer that triggers the sending of a dedicated DATA_SHORT message, which causes all the receiving nodes to restart their routing systems. As a consequence, if a change happens in the routing topology, the reboot will be able to catch it.

We worked on this change because after the decision changed, we had an almost working system which was better than the minimal pass and really wanted to get an extra half point in our challenges. We think that it would be really nice if one unaccounted issue in the demo (which was fixed before the deadline) would not prevent us from this.

What was the aim of the project?

The aim of the integration project was designing and implementing a fully distributed multi-hop ad-hoc chat application using wireless sound communication. Mobile or wireless ad hoc networks are decentralized types of network which can work without wires. It does not rely on a specific setup such as routers or access points in managed wireless networks. This project uses wireless communications using sound waves which utilized a physical layer by document that is located on Canvas.

First of all, when we want to send messages to other nodes with a default emulator, we realize that the string messages that we were sending, sent as bytes. We provided a line of code which converts bytes into strings with UTF-8. This helped us to broadcast text messages instead of bytes. Then we had to implement and design a mechanism for addressing. We were able to see the “transmit ranges” of the nodes with the Integration Tuner and that helps us to detect which nodes can communicate and which nodes can not communicate with specific nodes. The addressing method that we used to communicate outside of the transmitting range is effective and simple. We assigned random numbers between 0 and 128 to every node. These were called “Node IDs”. Then, we provided a code that creates a routing table for each node. In order to process the messages correctly, current state information has to be maintained in the routing table entries for the destinations of interest. When nodes joined the network, first, they mention their node ids. This is important to detect which node is transmitting or listening. After we assigned id’s to the nodes, we determined a code that mentions the nodes’ destination nodes and hop counts to their destinations. All nodes do the same process to complete their setup. That’s why, the setup process takes some When they finish their setup connection, they are ready to communicate with each other.

While nodes are transmitting data at the same time, a collusion might occur. This may happen both at the setup and while they are transmitting data. To avoid these possible collisions, we used threads to maintain the order of packets to send. It checks the sending queue before it puts the message into the queue and if it is empty, it directly pushes the message that it has. However, if the queue has a message to deliver, our thread sleeps for a specific amount of time and then it sends its message. This is the medium access control mechanism that we use to avoid packets from possible collusion.

.