Alexicia Richardson, Thomas Baker, Nirmit Patel

Lab 2

Due: April 3, 2017

Parallel Flow Connection for Supporting Reliable

and Mobile Communication Hub for First Responders

* Project Overview and Requirements
  + For this project, we were tasked with developing a communication system using Parallel Flow Connections. The project will be able to send data from the First Responders to the fire chief using parallel communication. Data being delivered is critical so in case Ethernet fails to send the data, a backup form of sending data to the fire chief, such as Manet with flooding.
* Architecture
  + For the gremlin functions, the user is prompted for a value ranging from 0 to 100. For Ethernet, if the user enters 100, the Ethernet fails and the program will be executed via Manet. Overall, our project has six files, being the 4 sensors files, a client file, and a sever file. However, there are multiple classes. For example, the client file has the client class along with the Node class. The Node class allows us to set up connection via Manet, and the client class inherits that file. To simulate the sensors, we have sensor classes that are ran, the sensor data is sent, each with a different rate, to the client, the client sends that data to the server, via Ethernet and, or via Manet.
* Design
  + We decided to complete our code in Java. We implemented 4 sensor classes for specific data that would then be sent at different rates to the sensor hub. In the client class, it handles all the information received from the sensors along with reading the configuration file for the MANET network configuration. The MANET is used to transfer data from one node to another until it reaches its destination, whereas Ethernet travels directly from client to server. In the client file, there is a Node class that take the configuration file and stores the node objects in an array. Before sending a packet, the client checks the configuration file to check for any modification along the network. If there has been any modification, we relink the nodes appropriately and attempt to send across the network. However, this only occurs if communicating via ethernet fails. Since we have two ways of sending data, we now require two Gremlin functions that will allow us to simulate data loss. If the data is completely loss through ethernet, i.e. it fails, then we switch to MANET. Once the Ethernet has been restored, and MANET fails, the communication switches back to Ethernet. When the server is executed, it opens the 4 sensor windows along with a main window for the Fire Chief. The server receives the packets from the client and displays the data received appropriately.
* Issues
  + The biggest issue of the lab was trying to figure out the best way to set up the MANET communication and having the ability to switch from one form of communication to the other if failure was to occur. Another issue we encountered, was knowing what was needed to simulate packet dropping via Manet. As we read, once a node is out of range of another node, their link must be dropped and they are no longer connected. We learned that via Manet, if the Gremlin function parameter was 100, then Manet fails, also the packet drop rate can be effected by the distance between the nodes.