**Killian Ronan – Student Number : 18328687**

**Telecommunications Assignment 2 : OpenFlow**

**Introduction:**

I began this assignment by researching the OpenFlow protocol and to achieve a greater understanding of what I was really being asked. With the use of the lecture slides, helpful Youtube videos and other online sources I acquired some knowledge on how an OpenFlow protocol is implemented and what goes on behind the scenes.

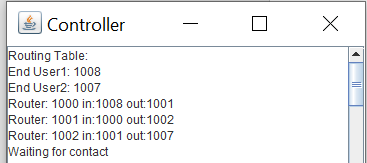
**My Approach:**

The approach I took to begin my version of the protocol was to use the code I had formed from assignment one. I used my previously made C&C class as a basis for the EndUser class, sending messages to be forwarded. The previous Broker class seemed like a good fit to use as a basis for the Routers, forwarding messages in two directions. Finally for the Controller class was initiated similar to the Broker and C&C, defining the routes to be taken by each router by assembling an overall flow table. As a result of only being able to complete the first part of this assignment, the flow table was created by the controller upon startup and does not change throughout runtime.

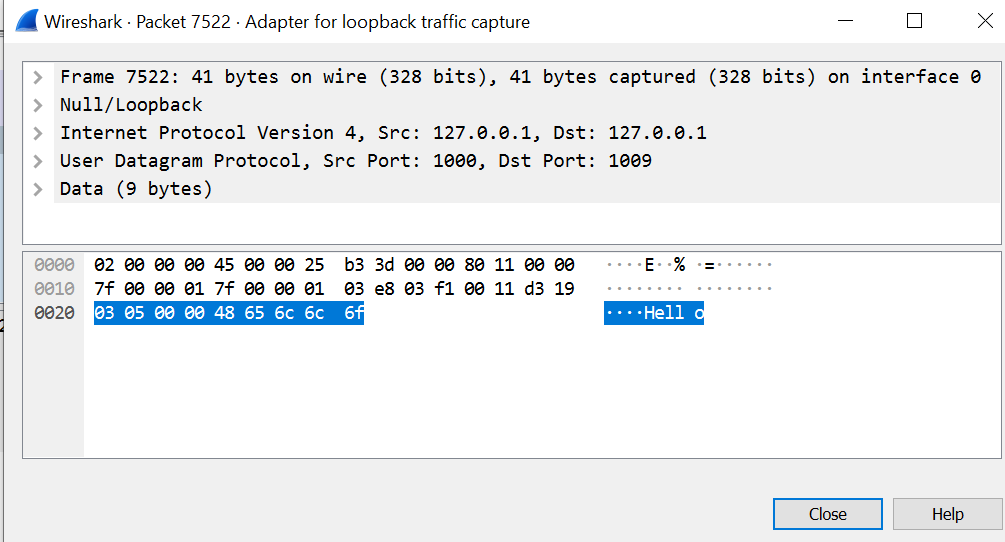
**Design and Implementation of Part One:**

The goal of the protocol is to allow an End User to communicate, through sending and receiving messages, with another End User. Hardcoded data was placed in the controller encapsulating the relevant data on how the OpenFlow network is connected. This routing data is implemented with the use of a flow table. It has all the information on how a packet sent by End User 1 is routed through each node. Upon startup of the controller the flow table/routing table is initialized. The approach I took was to declare a 2 dimensional array of integers, containing the port numbers of each router, along with their respective input and output ports. These input and output ports are represented with an array of integers. I thought this approach to be intuitive as I could easily switch the direction of the flow of the packets by swapping the input and output port of each router. Another approach I could have taken was to use a HashMap to store the information needed. I didn’t have a huge amount of experience using them and so decided on using simple arrays and 2D arrays instead.

**Controller printing flow table on startup:**

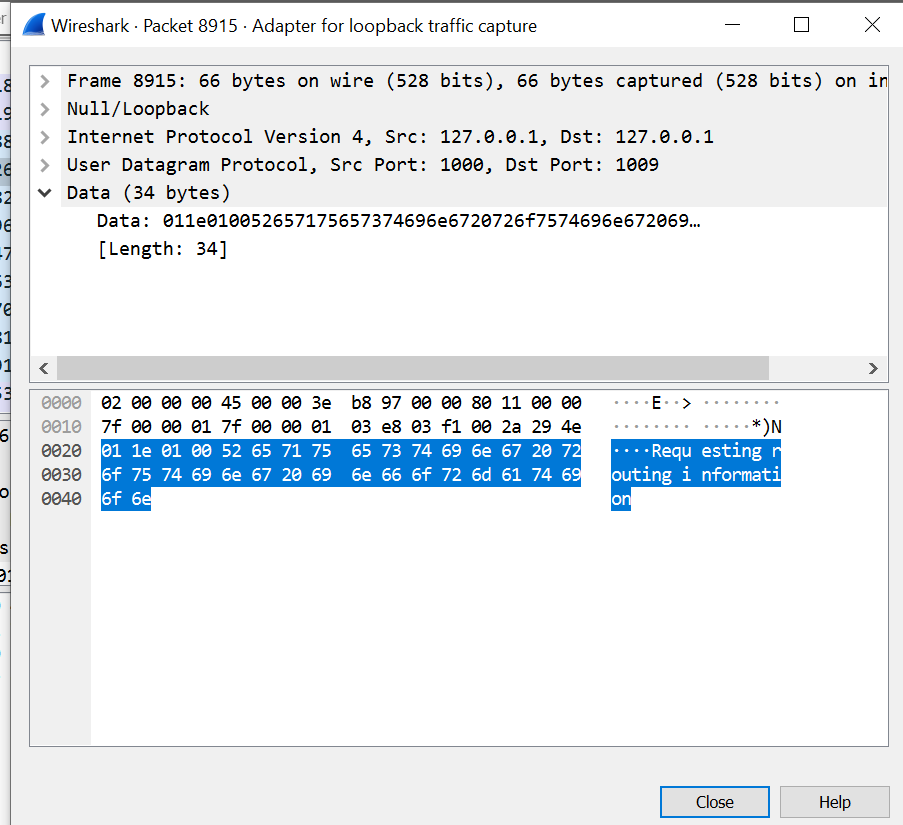


**Initial Hello sent to controller by Router:**

****

The Routers are initialized after the Controller. Each Router has its own routing table and each are established on a different port number with the prefix *localhost*. On startup of the program the routing table is empty, with placeholders for the End Users in the network. It must receive its routing information from the controller. The sendMessage() function is called in the main method of the class upon initialization of each router. Routers have an input and output port number which can be interchanged depending on which End User the received packet is being sent to. This is calculated through storing the destination port of the packet being sent in the header. Depending on the direction the packet is going, It will be routed to one port or another.

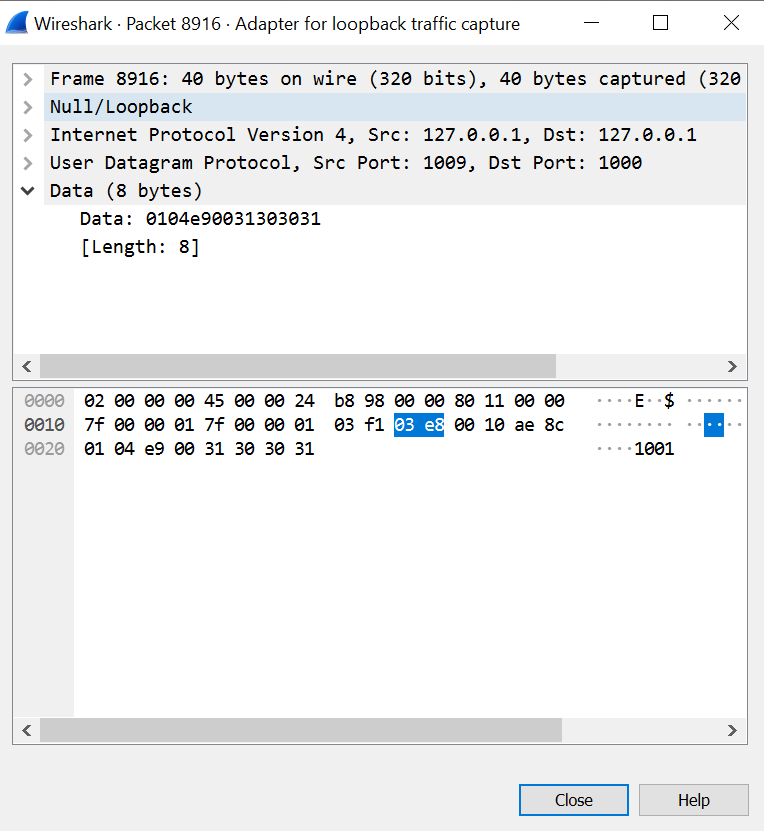
**Router Information Request:**



After the initialization of firstly the Controller, then the routers, the End Users are initialised. Upon startup the EndUser class has all the information it needs to send a message. The sendMessage() function is called and the port number of the End User sending the message is packetized along with the message. The user types a message to be sent from one End User to another and hits send. As it already knows which router to send the packet to, no more information needs to be attained. End users pose and allow the user to choose between the options of either sending or receiving a packet. If the End User is chosen to receive, it waits for contact and does nothing else. If the End User chooses to send a message, the user is prompted to enter a message. The destination port is set automatically as there are only 2 End Users in my version of the application. The message is packetised along with the destination port number and sent to the routers to be forwarded efficiently to the opposite End User.

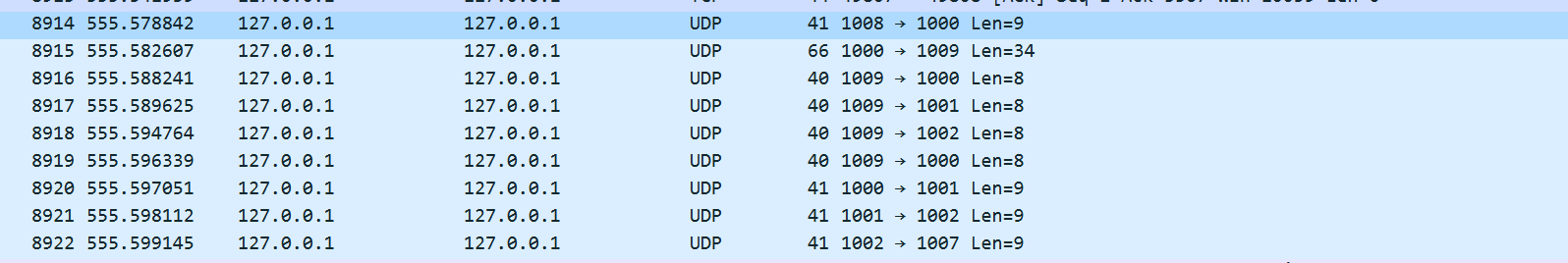
Upon receipt of the packetized message from the End User at the Router, the router depacketizes the packet in order to find the destination address(End User 2 Port) of the packet. It then cross-references this destination port with its own routing table to find out if it knows how to forward the message to the destination address correctly. If it has the information it needs, the packet is forwarded to the next port. However if the Router does not have the information it needs to forward the packet it must contact the Controller to update its flow table with the relevant information. This is implemented by the Router by sending an Information Request Packet to the Controller.

**Controller Response to Info Request by sending relevant port number:**



Upon receipt of this packet at the Controller class, it depacketizes the packet, finds which Router port is requesting information, and sends back the relevant information based on its hardcoded flow table, which contains all of the routing information for the network. As the controller does not receive any other packets, it knows this must be an Information Request from the Router and responds appropriately packetizing the destination and source nodes it has and sending the relevant data to each router, informing them of the next ‘Hop’ they must take. This hop is the port number where they must forward the packet. Every node affected by the transmission of the packet from End User 1 to End User 2 is informed on the route they must take whenever the Controller receives an Information Request. This updates each of the routers with the relevant routing information and allows them to forward the message to the receiving End User.

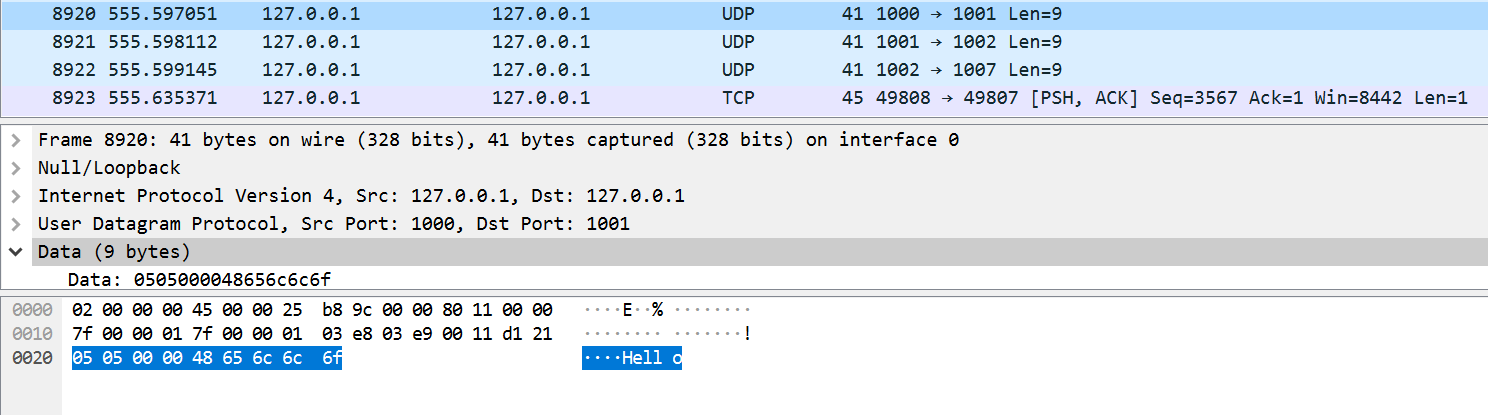
**EndUser1 -> EndUser2:**

****

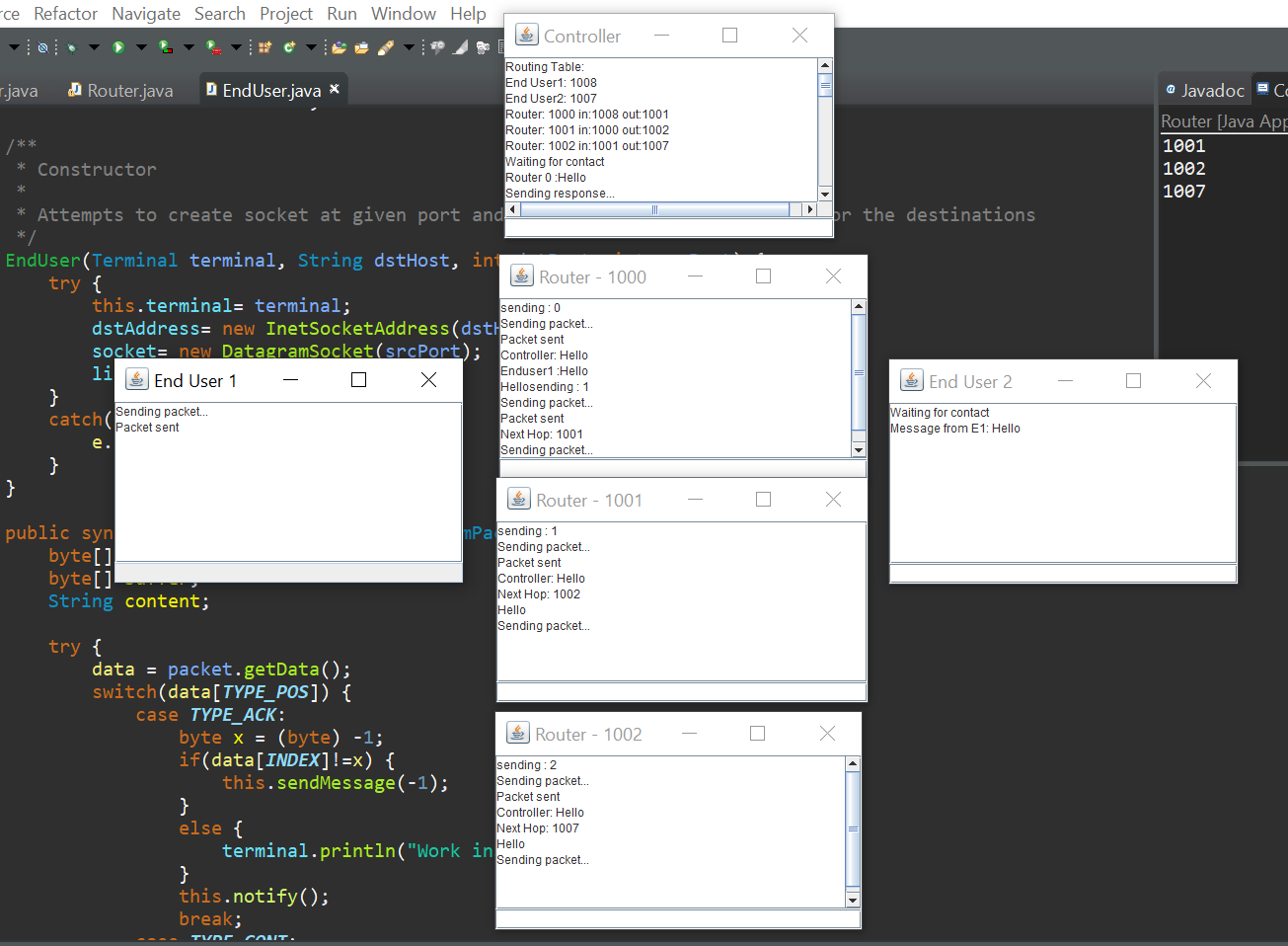
The controller does not have to be contacted each hop as each router is sent and retains the information on how to get from one End User to another, after one Information Request. The routing table for all routers is therefore considered complete once a packet has been sent from each End User.

If a router receives a packet, looks to its routing table and already has the information needed, there is no further data needed to forward the message and so it is forwarded to the next hop port number.

**Message final sending:**

****

**Full Program Flow:**

****

**Conclusion:**

This assignment was a completely new experience again for me. Although parts were very similar to the previous assignment, I found it difficult but still thoroughly enjoyed the entire process. The freedom to make my own decisions regarding my OpenFlow protocol helped a lot also. It was extremely beneficial for my understanding of Telecoms as a subject but also I learned a great deal of using Sockets and Threads and Networking in general with JAVA. The assignment helped me gain a deeper understanding of networking as it is so applicable in real life. It was a daunting challenge to begin with but, with a little help from the lecture notes, lab assistants and online sources, I feel entirely more confident in my understanding of networking and protocols related. This is sure to be beneficial next year when taking the next Telecommunications module.

**Estimation of time spent:**

30 hours roughly.