CS2031 Telecommunications II

Assignment #2: OpenFlow

Chloe Conneely 17323080

Contents

1. Introduction
2. Components

- Source

- Destination

- Controller

- Router

- The Packet

1. Running the Program
2. Reflection

- Closing

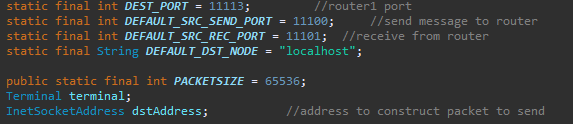
1. Introduction

I have been tasked with designing a version of OpenFlow implementing the controller- and router-side of the protocol. The implementation must contain flow tables for each router and the interaction with the controller will change each routers flow table. This report describes the components and processes of my implementation of OpenFlow.

2. Components

# Source:

The source class contains the port to where the packet is sent. The port from where the packet is sent in the source and the port from where the source receives messages from the router. The terminal allows for a terminal to appear on the screen display the outputs. The address is where the packet to send will be constructed.



Methods:

sendPacketIn gets the timestamp of when the packet is sent. Then it creates the contents of the packet: A type indicator ‘Source’, the port and the timestamp along with the message. The packet is then sent to the router.

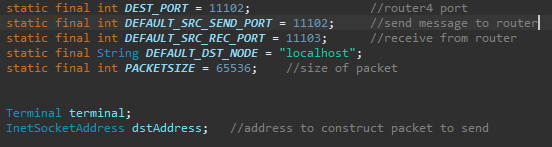
start prints “source sending packet” in the console and calls sendPacketIn.

main creates a terminal through which a packet can be sent along with threads for sending and receiving.

run waits for the source to receive contact with other classes.

# Destination:

The destination class contains the port to where the packet is sent. The port from where the packet is received in the destination and the port from where the source sends messages to the router. The terminal allows for a terminal to appear on the screen to display the outputs. The address is where the packet to send will be constructed.



Methods:

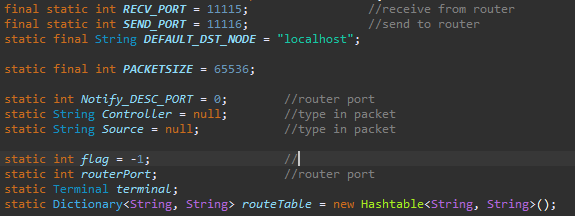
onReceipt receives the packet from the router and creates a buffer to extract the data. The terminal then displays a message stating the destination has received the packet from the router.

main creates a terminal to display the message received and a thread to receive packets.

run waits for the destination to receive contact with other classes.

# Controller:

The controller class contains the port from where packets are sent and the port where packets are received. There is also a variable to store the number of the router and the type indicator of the controller and source. The terminal allows for a terminal to appear on the screen display the outputs. There is also a flag variable used to determine the type of message sent. The routeTable contains the flow table for the controller.



Methods:

onReceipt receives the packet and calls parsePacket.

parsePacket creates a buffer to extract the data. The message type is stored in a variable called messageType as the message type is needed to determine what the controller will do after receiving a packet. The variable ‘content’ contains the message type. A switch statement is used for each path of action. When the message type is a hello message, ‘content’ is given “Hello”. A timestamp is created and the *‘Controller’* variable is given the controller type indicator ‘C- ‘, the port number of the controller and the timestamp. Then, the router port is found along with the source port. When the message type is a feature reply, ‘content’ is given “Feature Reply”. When the message type is packetIn, ‘content’ is given “PacketIn”. The source port is found along with the router port. Finally, the flag is set to 10. A message is then printed in the terminal stating the message was received from the either the source or the router.

sendFlowTable searches the controller’s flow table to find the routing information of the router in question. Then, it puts the routing information into a packet and sends it to the router.

sendFeatureRequest creates a packet to send a feature request to the router. A message is displayed in the terminal stating the feature request is sent to the router.

sendHelloMessage creates a packet to send a hello message to the router. A message is displayed in the terminal stating the hello is sent to the router. The flag is set to -1 and sendFeatureRequest is called.

start contains a switch statement which is controlled by how the flag is set. When the flag is set to 0, sendHelloMessage is called. When the flag is set to 10, sendFlowTable is called.

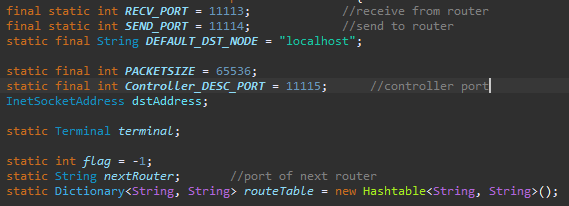
main creates the terminal and calls InitPreConfiguration. Then, threads for sending and receiving messages are created.

InitPreConfiguration fills the controller’s flow table with the in and out ports for each router

# Router:

Each router works in the same way, so I will discuss how the router works using “Switch1” as an example.

The router contains a port for sending packets along with a router for receiving packets. It also contains the controller port, a terminal to allow for a terminal to appear on the screen to display the outputs, a flag used to determine the type of message sent. The routeTable contains the flow table for the controller.



Methods:

onReceipt calls parsePacket.

parsePacket creates a buffer to extract the contents of the packet. The message type is stored in a variable called messageType as the message type is needed to determine what the controller will do after receiving a packet. The variable ‘content’ contains the message type. A switch statement is used for each path of action. When the message type is a hello message, ‘content’ is given “Hello” and a message is displayed in the terminal stating the message was received from the controller. When the message type is a feature reply, ‘content’ is given “Feature Reply”, a message is displayed in the terminal stating the message was received from the controller and the flag is set to 5. When the message type is packetIn, ‘content’ is given “PacketIn”, a message is displayed in the terminal stating the message was received from the source and the flag is set to 10. When the message type is flowMod, ‘content’ is given “FlowMod”, a message is displayed in the terminal stating the message was received from the source and the routing information is found. Finally, the routing information for the router is placed into the routing table for the router.

forwardPacketIn creates a packet that will be sent to the next router. The terminal displays a message stating the router is forwarding the packet. A timestamp is created which follows the type indicator ‘R1-’ and the port number of the router. The packet also contains the message type OFPT\_REQUESTFORWARD. The packet is then sent to the next router.

sendPacketIn creates a packet that will be sent to the controller. The terminal displays a message stating the router sending a packetIn message to the controller. A timestamp is created which follows the type indicator ‘R1-’ and the port number of the router. The packet also contains the message type set to OFPT\_PACKET\_IN. The packet is then sent to the controller.

sendFeatureReply creates a packet that will be sent to the controller. The terminal displays a message stating the router sending a featureReply message to the controller. A timestamp is created which follows the type indicator ‘R1-’ and the port number of the router. The packet also contains the message type set to OFPT\_FEATURES\_REPLY. The packet is then sent to the controller.

start contains a switch statement which is controlled by how the flag is set. When the flag is set to 5, sendFeatureReply is called and the flag is set to -1. When the flag is set to 10, sendPacketIn is called and the flag is set to -1. When the flag is set to 14, forwardPacketIn is called and the flag is set to -1.

Main creates a terminal and threads for sending and receiving messages.

contactController sends a hello message to the controller. A timestamp is created which follows the type indicator ‘R1-’ and the port number of the router. The message type is OFPT\_HELLO. The router information and the message type are put into the packet and then the packet is sent.

# The Packet

This section will describe the structure of the packet used for information exchange between the controller and the routers.

### Controller

The packet for the controller consists of a type indicator ‘C-‘, the origin port from where the packet was sent, a timestamp noting the time at which the packet was sent and the message type indicating what type of message was sent

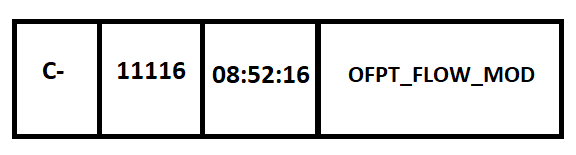


Figure 2.1. Example of the Controller packet

### Router

The packet for the router consists of a type indicator ‘R’, the number of the router followed by a ‘-‘, the origin port from where the packet was sent, a timestamp noting the time at which the packet was sent and the message type indicating what type of message was sent.

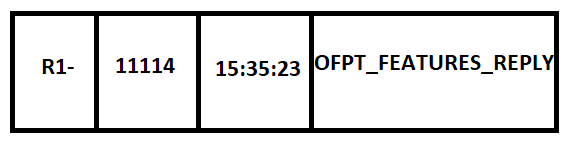


Figure 2.2. Example of the Router packet

3. Running the Program

The program starts with the controller waiting to connect with the routers as seen in figure 3.1. The controller then connects with the first router by switch 1 sending a hello message. The controller then sends a hello message back and then a feature request. Router 1 then sends a feature reply and a connection is established. This is illustrated in figure 3.2. The controller connects with second and third routers in the same manner as with the first router. This is illustrated in figures 3.3. and 3.4. The packet is sent from the source as shown in figure 3.5. Router 1 receives the packet from the source and then sends a PacketIn message to the controller informing the controller of the router’s receiving of the packet from the source. The controller then sends a FlowMod message containing the routing information to the router. The router now knows where to send the packet and sends it to router 2. The interaction between router 1 and the controller is shown in figures 3.6 and 3.9. The second and third routers obtain their routing information in the same manner as router 1 as shown in figures 3.7 and 3.8 respectively along with figure 3.9. The only difference occurring in the third router as this is the final router which sends the packet to the destination. The destination receiving the packet from the third router is shown in figure 3.10.

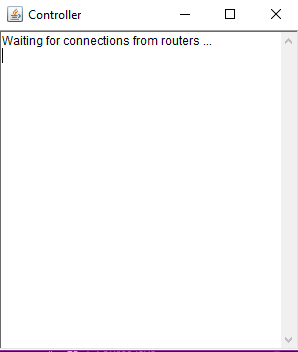


Figure 3.1. Controller waiting for connections

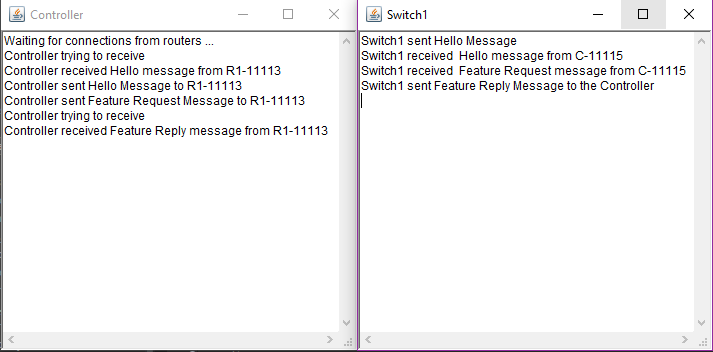


Figure 3.2. Controller and switch 1 establishing a connection

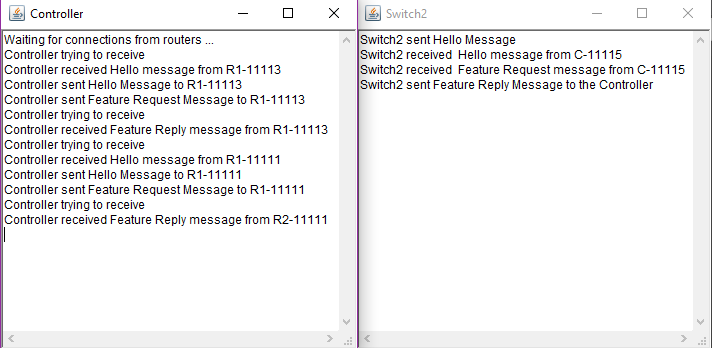


Figure 3.3. Controller and switch 2 establishing a connection

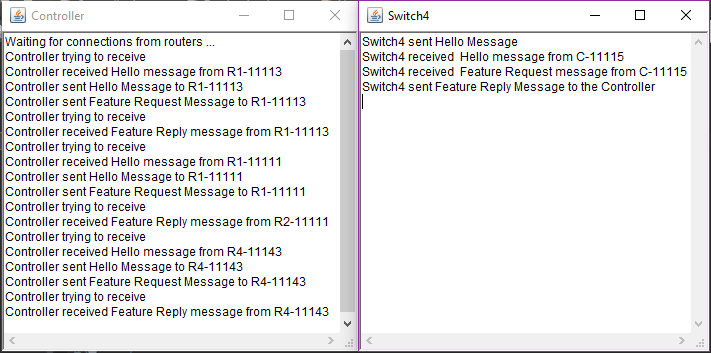


Figure 3.4. Controller and switch 4 establishing a connection

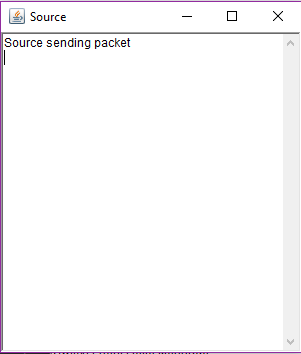


Figure 3.5. Source sending packet

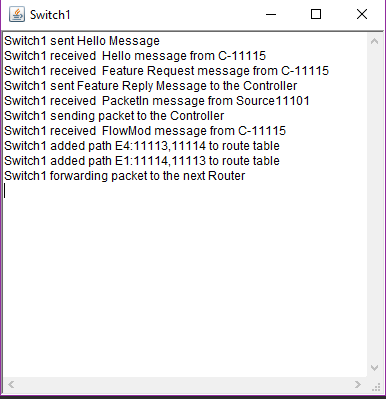


Figure 3.6. Router 1 obtaining routing information from the controller

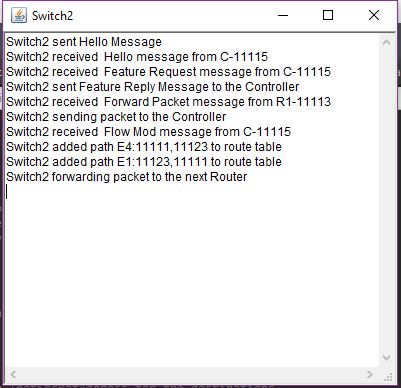


Figure 3.7. Router 2 obtaining routing information from the controller

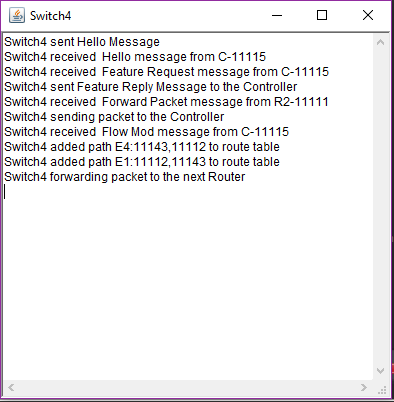


Figure 3.8. Router 3 obtaining routing information from the controller

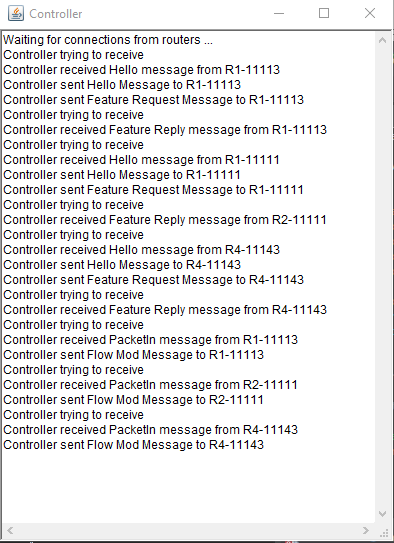


Figure 3.9. Controller interacting with the routers

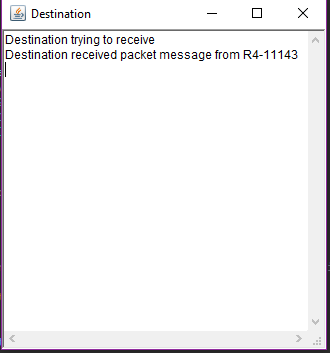


Figure 3.10. Destination receiving packet

4. Reflection

# Closing

I have designed an implementation of OpenFlow that consists of a controller interacting with multiple routers to send a packet from a source to a destination. The controller establishes a connection to the routers and directs them to where they must send the packet. The source sends the packet to the first router in the network. The destination receives the packet from the last router in the network. I used the sample files provided as a starting point for the protocol which, while it was helpful in terms of finding a direction for the implementation, it means the protocol may be obsolete in its design. I did not make use of containers as I felt I would not understand how it worked in time for completing this assignment. I did not add any extra components to this assignment as I did not start this assignment in good time. I estimate I spent about 70 hours on this assignment between the implementation and the documentation.