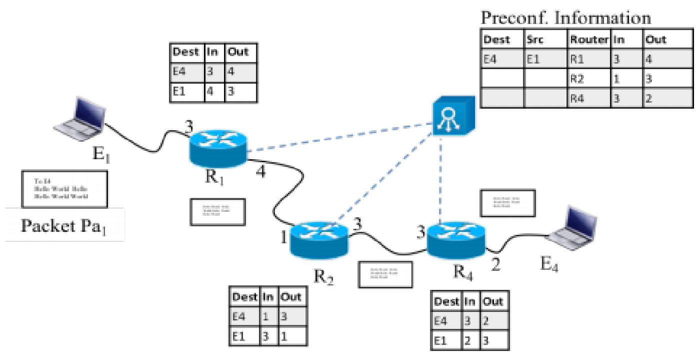
CS2013 - Telecommunications II Assignment II

Anton Yamkovoy.

December 27th 2018

1 OpenFlow Routing



Preconf. Information

Dest	Src	Router	In	Out
E4	E1	R1	3	4
		R2	1	3
		R4	3	2

Dest	In	Out
E4	3	4
E1	4	3

Dest	In	Out
E4	1	3
E1	3	1

Dest	In	Out
E4	3	2
E1	2	3

Figure 1: OpenFlow scenario.

1.1 Requirements for the OpenFlow Routing implementation

- EndPoint users who have the ability to send a receive packets, coming from a router or other endpoint.
- EndPoint users need to be able to accept and send a packet.
- Router class implementation, which is able to forward packets, that it intercepts from another router, or a EndPoint.
- Routers, must be able to communicate with a controller, who will change their routing table and feed them information on how to route the incoming packets, to the next router in the network
- The controller should be able to contact routers and send them information from a pre-configured/generated routing table for the current scenario.

1.2 Additional features implemented:

- The controller, EndPoints and Routers, all have visual and input devices if needed, they are launched through a terminal, with possibilities to input data or send some kind of request.
- When packets are sent in between the different terminal types, there is visual confirmation and details of information transferred and routing tables are updated.

2 EndPoint Analysis.

2.1 EndPoint components:

- EndPoint class:
  The EndPoint class is an extension of the Node class, it has the functionality to send and receive packets incoming from routers / other EndPoints, it uses the EndPoint terminal, and the DatagramPacket classes for it's operation
- EndPoint terminal class:
  The EndPoint terminal class is a jframe implementation of the terminal, needed in this scenario, it uses swing/awt components to achieve the ability to send a receive packets, containing a test message, with visual confirmation of packet arrival, and a button input mechanism.

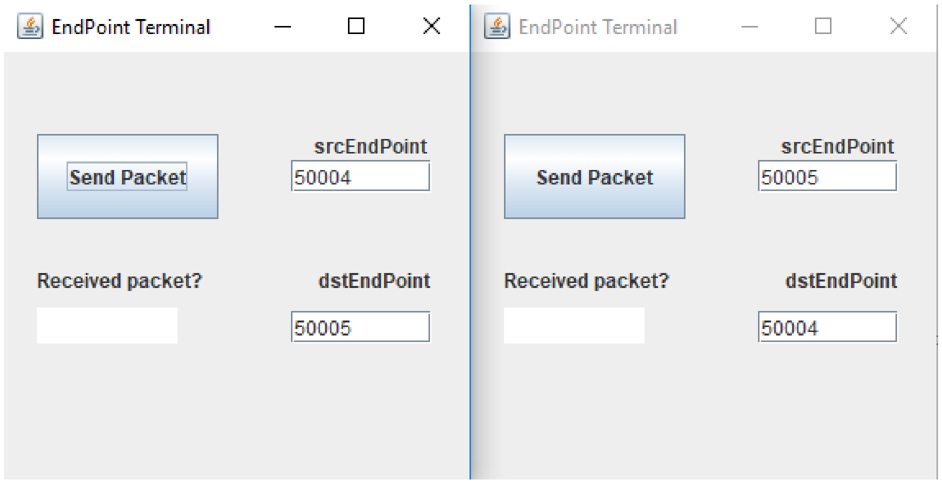


Figure 2: two EndPoint instances, which are configured to send to each other.

The figure is showing, two EndPoint class objects, based on the EndPoint terminal, they have the functionality for sending a packet to the configured pair EndPoint, in this case they are both configured to send to each other, via the network of routers inbetween them using OpenFlow.

2.2 EndPoint operation explained:

The EndPoint class, implements sendPacket() and onReceipt() methods, similar to the Node which it is based off.

The Subscriber implements three key methods to function properly

- synchronized onReceipt(Datagram packet)
  The onReceipt() method for the EndPoints are simple, they check the contents of a received packet, which was forwarded to them by a router, from the corresponding EndPoint, they check it contains the "OK" keyword, and change a textField on their terminal to signify that the tranfer is complete.
- sendPacket()
  This method is also a simple sender method, It constructs a packet containing the key message, which the corresponding endpoint will check to be sure of the validity of the packet received. It uses the standard datagram socket procedure.
- start()
  The start method for the EndPoint, calls this.wait() until interrupted by the onReceipt() method call.

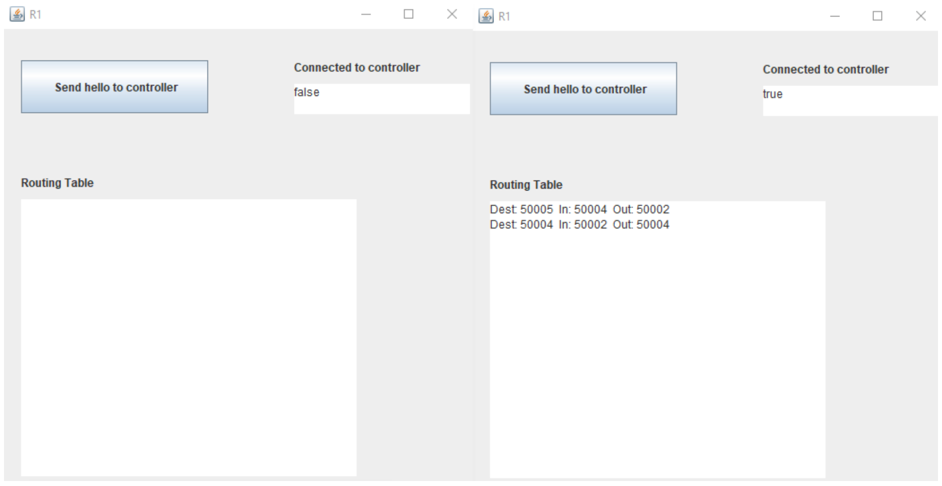
2.3 EndPoint Additional Information/ Edge Cases

- Input:
  The input is very simple, and no errors can really occur here, since the input mechanism isn't variable, it is just a button press.
- EndPoint creation:
  As many EndPoints can be created as long as they are configured correctly, this includes, setting the source and destination ports, equal, or making an invalid combination with multiple endPoints, or using a port that is already in use, by a corresponding router or controller.
- EndPoint Terminal:
  I decided to continue using this jframe implementation of the terminal instead of the standard tcd.lib one, since it was easy to modify from the previous assignment, and allowed for more possibilities in general.

3 Router Analysis

3.1 Router components:

- Router terminal class:
  The router terminal class is very similar in concept to some of the previous custom terminals mentioned. It contains a mutable routing table display, a input section to send a request to connect to a controller, before any communications are established, and a confirmation textArea for knowing that this current Router is connected to it's corresponding controller.
- Router class:
  The router class implements the methods it needs to function, these will be outlined and explained in the operation section next.
- RouterRoutingTable class:
  The main functionality of the router falls into the routingTable, this had to be a separate class from the controller routingTable, This object contains an ArrayList of RouterRoutingEntry objects.
- RouterRoutingEntry class:
  This object is used inside the routers table, it a 3-tuple of data, it contains the destination, inPort and outPort, as seen in figure 1, on the first page, near each one of the routers.



```
false
```

```
true
Dest: 50005  In: 50004  Out: 50002
Dest: 50004  In: 50002  Out: 50004
```

Figure 3: Publisher implementation using RouterTerminal, the router is shown before on the left and after on the right, after connecting to the controller

3.2 Router operation process:

The router is an extension of the Node class, which contains the threads methods

- The router should be able to, send and receive a hello message to and from the controller, to establish a connection and receive it's initial routing table when requested, When a packet arrives at the router object, It needs to lookup in it's own routing table, the next hop to send the packet to.

  It also needs to have a method to request an updated routing table object from the controller.

The Router implements the following methods:

- synchronized void onReceipt()
  The onReceipt() method is for receiving packets from the controller, other routers and maybe endpoints. It contains three cases, 1: when the router recieves a 'hello' reply from the controller, this activates the connection, and updates the router terminal to show the connected status. 2: If the router received back a routingRequest, from the controller, this is a packet that updates the routingTable of the router, based off of the controllers routing information. 3: A case where, the router has received an 'OK' packet which it needs to forward to the receiver endPoint based on it's routingTable.
- synchronized void start()
  The start method is identical to the EndPoint's method, it starts a loop, to this.wait() the thread, it is only interrupted, when the thread is notified at the start of onReceipt().
- send(DatagramPacket p)
  This is a simple method used to forward the packet received to the next hop.
- sendHello()
  This sender method is run on the press of the sendHello button in the terminal, it establishes a connection with the corresponding controller.
- sendTableRequest()
  This method is another sender method, it sends a request to the controller to update, the routing table, or this current router, it is called first when the router doesn't know where to redirect the packet, and can be called again, in the case of a changed routing configuration in the controller.

- int getNextHop(ArrayList¡routerRoutingEntry¿ table,int currentPort)
  This method operates on the routingTable, and the currentPort, it looks-up in the routing table the next router / endpoint to send the packet to. It is used in conjunction with the simple send() method, to complete the routers key functionality.

3.3 Router Additional info / Edge Cases

- Router creation
  The router class can be run in as many instances as you like, but, the ports need to not be contradictory to each other, and must be compatible with the rest of the system.

4 Controller Analysis

4.1 Controller components:

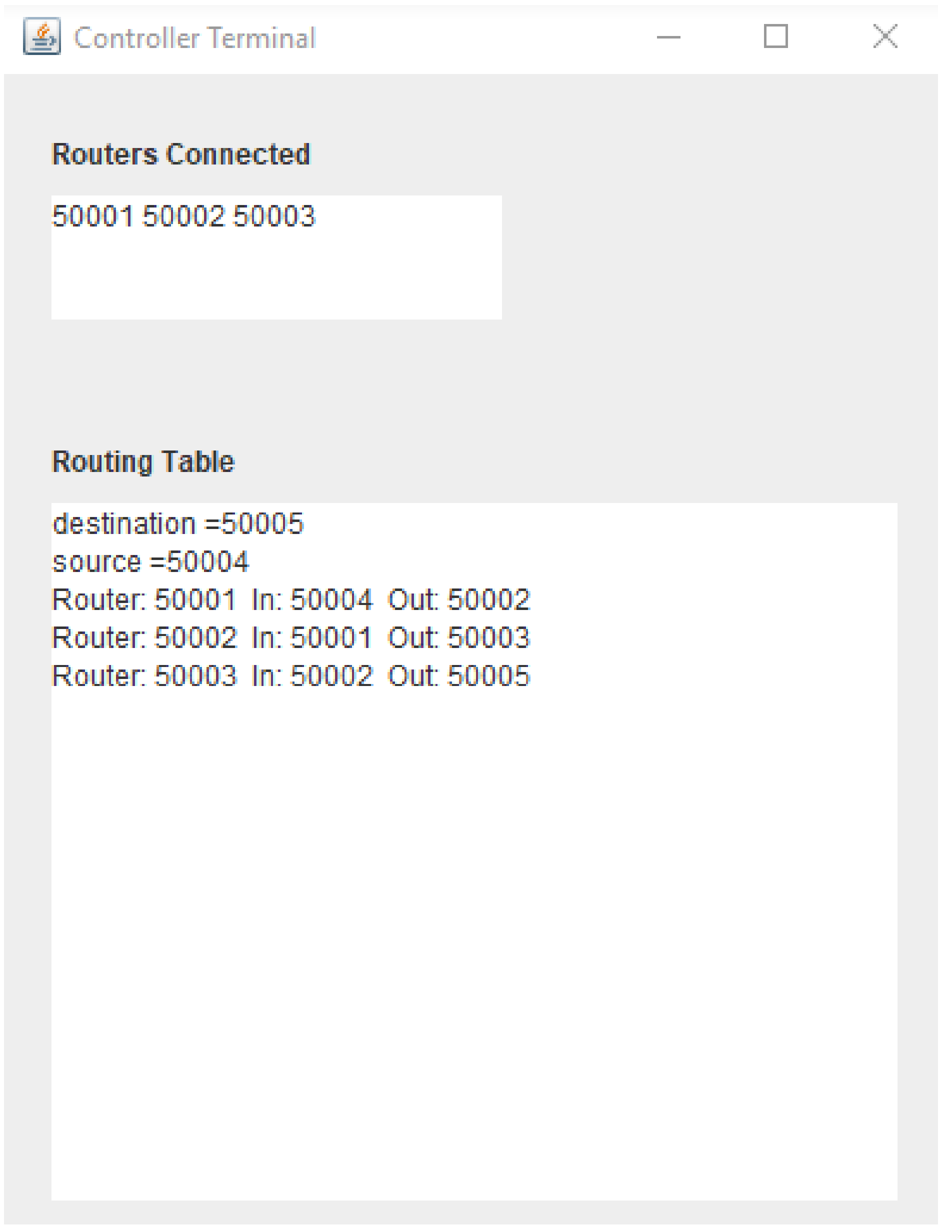
- ControllerTerminal class
  The controller terminal is a jframe application, that contains a list of connected, routers which changes dynamically as more routers send requests to connect. It also contains the routing table which is preconfigured into the configuration or generated using a link state routing/ distance vector routing approach.
- Controller class
  The controller class is similarly to the router and endPoint an extension of the Node class, it inherits some of the thread methods, and implements methods sendHelloBack() sendRoutingBack() and onReceipt().

4.2 Controller operation process:

The controller implements the following methods, mentioned above:

- synchronized void onReceipt() The controller, has only two cases for it's onReceipt() method: 1: When the controller receives a hello from a router, it needs to reply back with a hello message, In this process the controller also adds the specific router to it's routers list, and send the reply packet. 2: if the message received is a routingRequest, the controller finds the port from which the packet was sent, and queries the routingTable to find entries that are useful to this router in particular.

  The other methods sendHelloBack() and sendRoutingBack() are called based on the case.
- synchronized void start() Identical to the others' start methods, wait until notified by a onReceipt() call.
- sendHelloBack() and sendRoutingBack() The controller sends a helloMessage back as described above, and sendRoutingBack sends an updated version of the needed routing entries to the router in question.



```
50001 50002 50003
```

```
destination =50005
source =50004
Router: 50001  In: 50004  Out: 50002
Router: 50002  In: 50001  Out: 50003
Router: 50003  In: 50002  Out: 50005
```

Figure 4: The controller class, after three routers have connected to it, and the preconfigured routing table is loaded

5 General Operation

5.1 Operations Sequence

- Call Sequence
  The controller will be launched, and the routers who desire it will connect to the controller, using hello messages Once the endPoint user presses the sendButton, to send out a hello packet. This packet will go to the configured entry router, This router not knowing what to do with the packet, will ask for a routingRequest from the controller. The packet will make it's way through the network using the routingTables, of the routers updated by the controller, Once the packet arrives at the corresponding EndPoint, the "Received" message will become active in the receiver endPoint.

6 Assignment Reflection

6.1 Implementation Advantages

Extendable implementation in terms of amounts of endPoints Controllers and Routers.

Easy to change user interfaces and input methods.

6.2 Implementation disadvantages

A existing protocol was not followed closely in the design of the data packets and the ack/nack packets, In this field there are many improvements to be made in terms of saved memory/space and efficiency

There is no system implemented for generating a routing table for the controller, the controller has to be pre-configured to operate.

6.3 Time Spent

If I was starting again, I would try to implement the routing table generation through a system like link state routing

I spend less time on this assignment than on Publish Subscribe due to the skills learned during the first assignment in terms of using the libraries and techniques, time spent, 10-12 hours on code, 3-4 on the report.