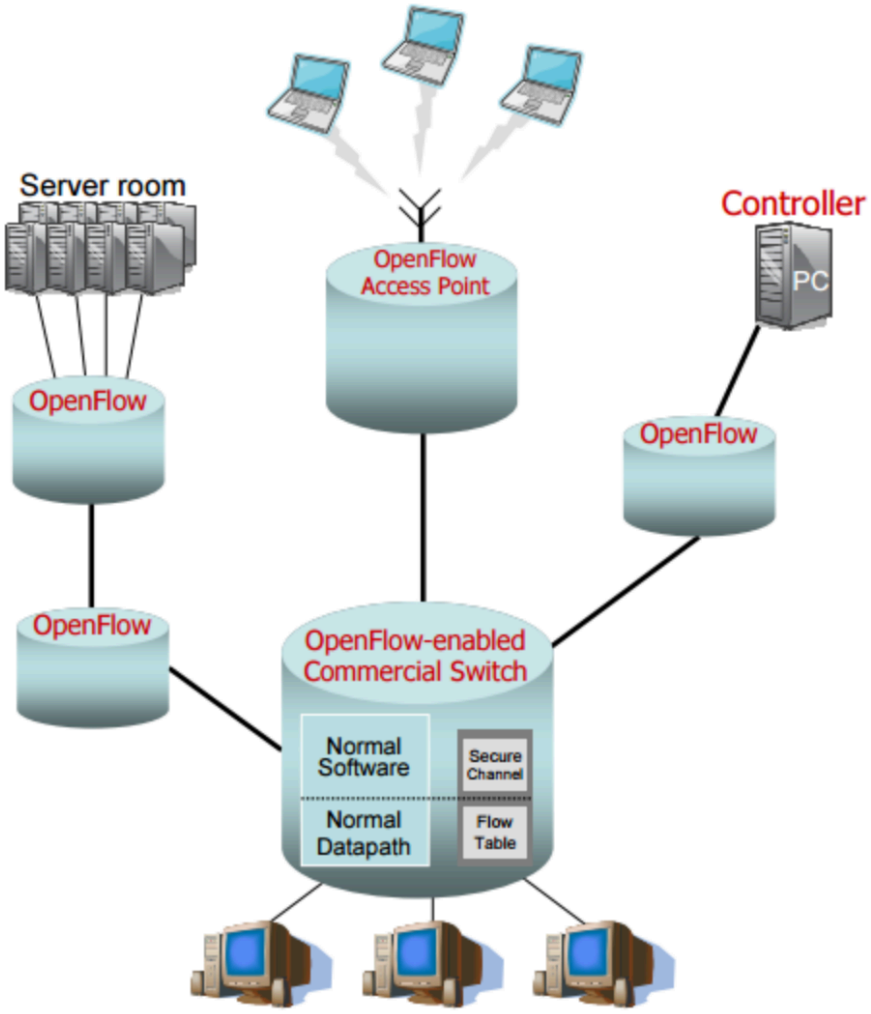
Trinity College
The University of Dublin

Telecommunications Assignment

Assignment 2 - OpenFlow

Senán d'Art - 17329580

1. Introduction
The problem description for this assignment was to create a version of OpenFlow for communication between routers and controllers of a company. It was required to design a version of the protocol as well as implement it on the router and control side.



A further optional extension was to implement either Link State Routing or Distance Vector Routing on each of the routers.

2. Theory of Topic

2.1 *OpenFlow*

OpenFlow is a new method of managing a network of routers that differs from traditional routing in several key ways. Although it is still possible, OpenFlow does not solely rely on routers to build their own routing tables through the use of Link State or Distance Vector Routing, rather they are also able to connect to a central 'Controller' and ask for help. This controller has the ability to modify and update each router's routing table. This potentially allows for a much more responsive network and one that can balance loads more effectively than a traditional network as the controller can instruct traffic to follow different paths to the same destination (see Figure 1 for example). It also allows for better allocation of resources as each individual router is not required to attempt to build its own overview of the network.

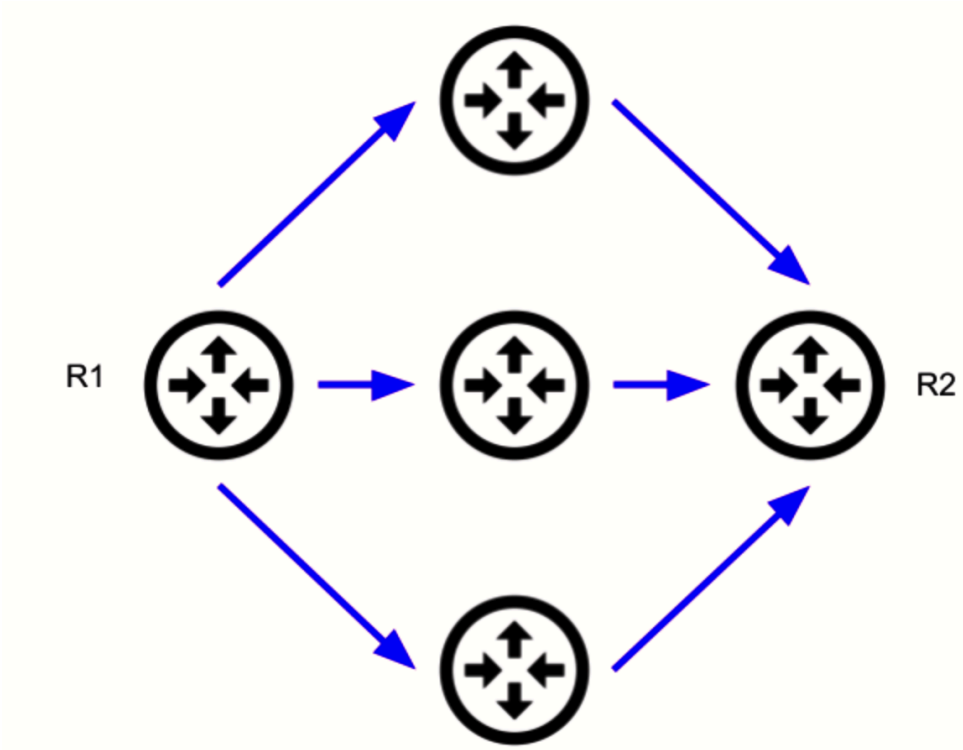


Figure 1. Traffic (in blue) routed from R1 to R2 along different paths. This can be dynamically assigned by the controller. This allows for better use of bandwidth.

2.1.1 Controller

The OpenFlow controller is responsible for managing the network. It has a direct connection to each of the routers. It will usually have a pre-configured routing data that allows it to instruct routers on how they should route packets. Another option is for the controller to build an overview of the network and allow it to create paths for packets dynamically.

2.1.2 Router

OpenFlow routers are different to traditional routers. Where traditional routers rely solely on their own ability to communicate with other routers to build a routing table (a process that can be slow and very time-consuming), OpenFlow-enabled routers also have the ability to be controlled by a centralised controller. This controller can modify their routing table, greatly improving recovery time if another router fails and allowing for reduced traffic on the network as routers are no longer required to communicate with others in order to build a routing table.

2.2 Link State Routing

Link State Routing is a method of building up a routing table used by routers. When a router uses Link State Routing to build a routing table it first sends requests to each of the routers it can reach for a list of routers they are connected to. It then adds these to a table and uses an algorithm, most commonly Dijkstra's Shorted-Path Algorithm, to determine the shortest route to each node on the network.

2.3 Distance Vector Routing

Distance Vector Routing is a method used by routers to build a routing table. Distance Vector Routing allows routers to build up a routing table through the propagation of routing tables on the network ie. routers share their routing tables on the network and use tables shared by other routers to build up their own routing table.

3. Implementation

3.1 User Interface

The user interface was not a priority in this assignment as the primary goal was to develop an implementation of OpenFlow for use on a network. As such the user interface relies on a simple Command Line Interface (CLI). This interface only runs on the Endpoints of the network and required the user input a destination and message separated by a tilde ("~"). This method was used in order to prevent unnecessary overhead in message input (see Figure 2 for example).

```
Please enter a destination and message, separated by '~':E2~Hello World
```

Figure 2. The CLI used by the Endpoints of the system and a sample input. The destination shown is E2 (Endpoint 2) and the message being sent is "Hello World".

3.2 Communication between Endpoints

Communication between endpoints occurred using a Stop-And-Wait ARQ protocol.

3.2.1 Packet Structure

Packets were created by serialising an object. This is different to the approach of the previous assignment where byte arrays were manually created. The reason for this change is due to the necessity of having metadata stored within the packet. The object used was called Content and contained a single byte designating the type, an array containing metadata on the intended destination, and source as well as the packet ID/sequence number. Finally it contained a string object that would contain the data being sent (see Figure 3 for example). This approach was not as efficient as the approach used in the previous assignment as there is a lot of excess data being sent. However it is much easier to parse the data when it arrives.

The order of items in the array are as follows: senderID, targetID, senderPort, targetPort, packetID.

```
0000  02 00 00 00 45 00 00 cc  1f 9d 00 00 40 11 00 00   ····E··· ····@···
0010  7f 00 00 01 7f 00 00 01  c7 38 c3 50 00 b8 fe cb   ········ ·8·P····
0020  ac ed 00 05 73 72 00 07  43 6f 6e 74 65 6e 74 00   ····sr·· Content·
0030  00 00 00 00 00 00 01 02  00 03 42 00 04 74 79 70   ········ ··B··typ
0040  65 4c 00 04 64 61 74 61  74 00 12 4c 6a 61 76 61   eL··data t··Ljava
0050  2f 6c 61 6e 67 2f 53 74  72 69 6e 67 3b 5b 00 07   /lang/St ring;[··
0060  74 67 74 49 6e 66 6f 74  00 13 5b 4c 6a 61 76 61   tgtInfot ··[Ljava
0070  2f 6c 61 6e 67 2f 53 74  72 69 6e 67 3b 78 70 04   /lang/St ring;xp·
0080  74 00 0b 48 65 6c 6c 6f  20 57 6f 72 6c 64 75 72   t··Hello  Worldur
0090  00 13 5b 4c 6a 61 76 61  2e 6c 61 6e 67 2e 53 74   ··[Ljava .lang.St
00a0  72 69 6e 67 3b ad d2 56  e7 e9 1d 7b 47 02 00 00   ring;··V ···{G···
00b0  78 70 00 00 00 05 74 00  02 45 31 74 00 02 45 32   xp····t· ·E1t··E2
00c0  74 00 01 30 74 00 05 35  31 30 30 30 74 00 01 31   t··0t··5 1000t··1
```

Figure 3. The contents of a Data packet sent by an Endpoint. The word "Content" is clearly visible, this refers to the name of the class. The string "Hello World" is visible, this is the message being sent. Also visible is the source: E1 and the destination E2 near the end of the file. The destination port number 51000 can also be seen.

3.2.2 Communication Protocol

The communication protocol used between Endpoints functioned as follows:

1) The transmitting node sends a Hello message.
2) The receiving node sends a Hello-Acknowledgement response.
3) The transmitting node sends a Data message.
4) The receiving node sends an ACK for this message.
5) The transmitting node sends an ACK with no timeout.
6) Transmission complete.

3.2.3 Stop-and-Wait ARQ

Stop-and-Wait ARQ is a very simple method of flow control. Every message sent from the transmitting node is responded to with an Acknowledgement (ACK) from the receiving node. If an ACK is not received, the packet is automatically resent. When an ACK is received, the next packet is sent (see Figure 4 for an example).

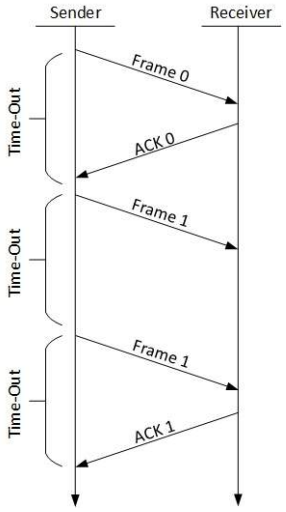


Figure 4. Stop-and-Wait ARQ in action. Frame 0 arrives successfully and is ACK'd. Frame 1 is sent when ACK 0 arrives but is lost in transit. Frame 1 is then resent when the timeout expires. When Frame 1 arrives, it is acknowledged with ACK 1.

3.3 Communication between Router and Controller

Communication between a router and the controller was similar to communication between endpoints. The same objects and methods were used to create the packets ie. the same Content object was used and serialised to create the byte array for the Datagram packet.

3.3.1 Packet Structure

The packet structure used in communication between the controller and routers was similar to that used by the endpoints. However the data String remained empty and only the array containing metadata was used.

3.3.1.1 Hello Packet

A Hello packet is sent from a router to the controller upon startup. The packet contains the ID of the router and the packet type. There is no other data contained within the packet. This is because there is no other content required since the purpose of this packet is to inform the controller that the router is online (see Figure 5 for an example).

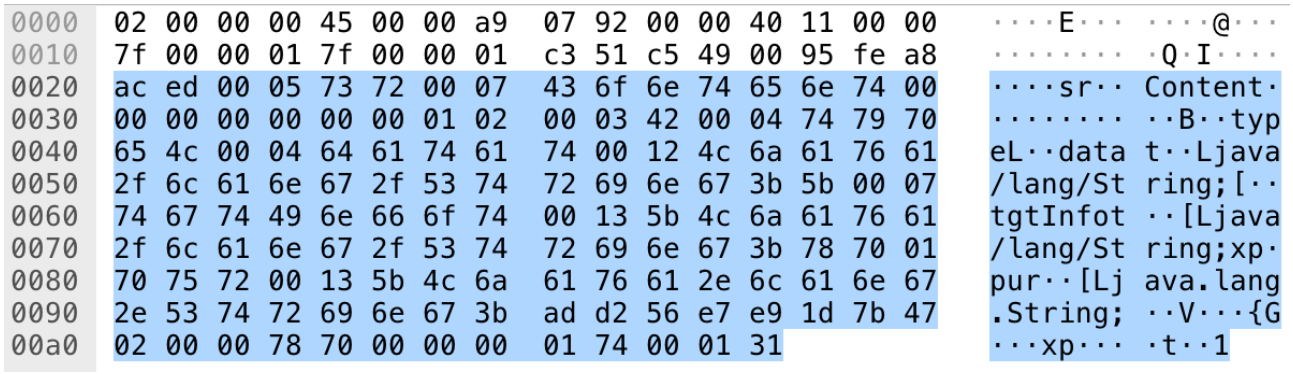


Figure 5. A Hello packet sent from a router to the controller. The router ID can be seen at the end, in this case it is number 1.

3.3.1.2 Help Packet

A Help/Help Request packet is sent from a router to the controller when it receives a packet and does not know where to forward it to. The array in the packet sent to the controller contains: the destination the router is trying to reach and the source that sent the packet (see Figure 6 for an example).

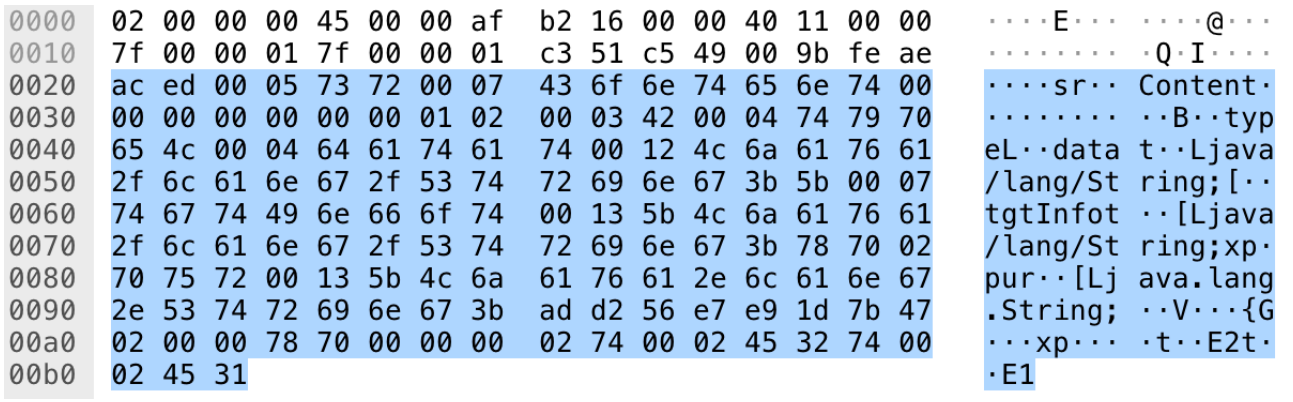


Figure 6. A Help packet sent from a router. The router is asking for the next hop in the path from E1 to E2. These can be seen near the end of the packet.

3.3.1.3 Update Packet

An Update packet is sent from the controller to all routers along the path between two endpoints. This packet contains routing information. The packet contains the two endpoints that are trying to communicate and the next hop along the route that this router should forward traffic to (see Figure 7 for an example).

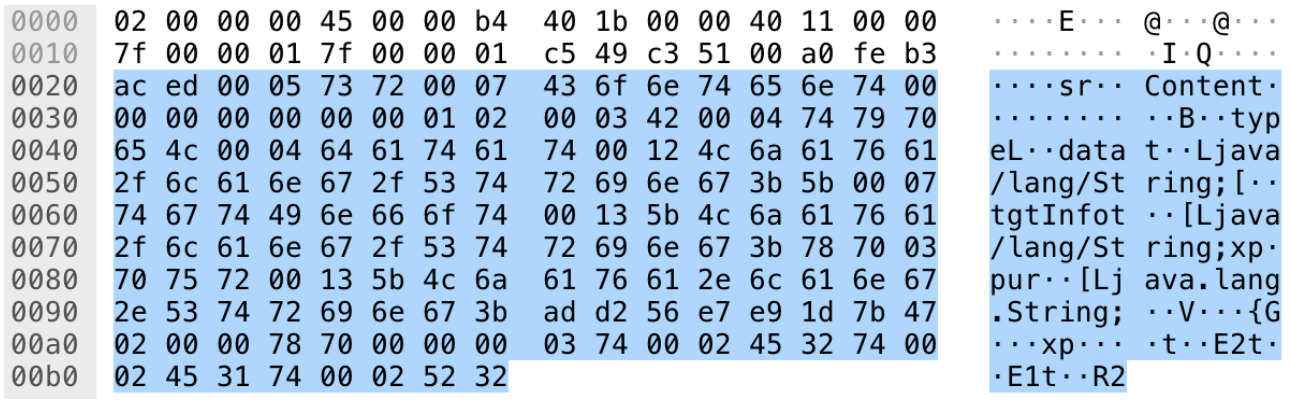


Figure 7. An example of an update packet. This packet was sent to the router R1. It details the next hop in the series in the path from E1 to E2. In this case the next hop is R2.

3.3.2 Communication Protocol

The communication protocol between a router and the controller was quite simple. There were 3 different types of communication: Hello - a conversation held on startup, Help - when a router asks the controller for information regarding the next hop along a path, Update - sent by the router to all controllers along a path informing them of a new route.

3.3.2.1 Hello

The Hello conversation between a router and the controller occurred as follows:
1) Router sends Hello packet with ID.
2) Controller marks router as active and responds with a Hello packet of its own. This packet does not have a timeout.
3) Router cancels packet timeout.
4) Transmission Complete.

3.3.2.2 Help Request

A help request is sent from a router to the controller when it receives a packet and does not know where to forward it to. The transmission goes as follows:
1) Router sends Help packet to controller.
2) Controller responds with update sequence (see next paragraph).
If no response is received, the timeout expires and the packet is resent. If a response is received, the timeout timer is cancelled.

3.3.2.3 Update

The Update conversation is held between the controller and every router along the path that a packet will follow. The reason this was separated from the previous paragraph is because this can occur without a router sending a Help Request as usually only the first router in the sequence will not know where to send the packet. The sequence occurs thus:
1) Controller sends Update packet to router.
2) Router updates routing table and sends an ACK with no timeout.
3) Upon receiving the ACK, the controller cancels the timeout of the Update packet.
4) Transmission complete.

3.4 Packet Forwarding

When a router receives a packet and does not know where to forward it to, it stores it in an array in a hash-map. Any subsequent packets with the same origin and destination are stored in the same array. When an update is received from a router specifying the next hop in the sequence that the packet should be sent to, the router removes the array from the map and sends all waiting packets.

4. Summary

My solution for this assignment uses a simple user interface with a Stop-and-Wait ARQ. This was done to reduce the complexity of the solution. A more complicated interface was not required as that was not the goal of the assignment.
I made extensive use of efficient data structures such as hash-maps throughout the assignment. This allows for a very efficient and scaleable solution.
The controller contains preconfigured data and uses this to send updated routing data to any router that requires it, as well as any router along the path.
When a router sends a Help request, all routers along the specified path are updated, this allows for greater efficiency as usually only one router in the sequence is required to cache the packet while waiting for the next hop.

5. Reflection
I am quite happy with my solution to this assignment. I feel like I learned a lot while working on the first assignment and this greatly improved my work on this assignment. This solution is far more slim and efficient from a design standpoint than the first assignment. I focussed heavily on not creating unnecessary code and overhead in the program.
Excluding the report I spent roughly 30 hours on this assignment.