Contents

[1.Introduction 1](#_Toc165979602)

[2.Initial Network Configuration 1](#_Toc624795805)

[2.1.Proof of Concept 1](#_Toc1052567072)

[2.2.OpenFlow Packet Analysis 1](#_Toc389721111)

[3.Network Functionalities 1](#_Toc1800938683)

[3.1.Firewall Implementation 1](#_Toc133093938)

[3.2.Load Balancing 1](#_Toc292624743)

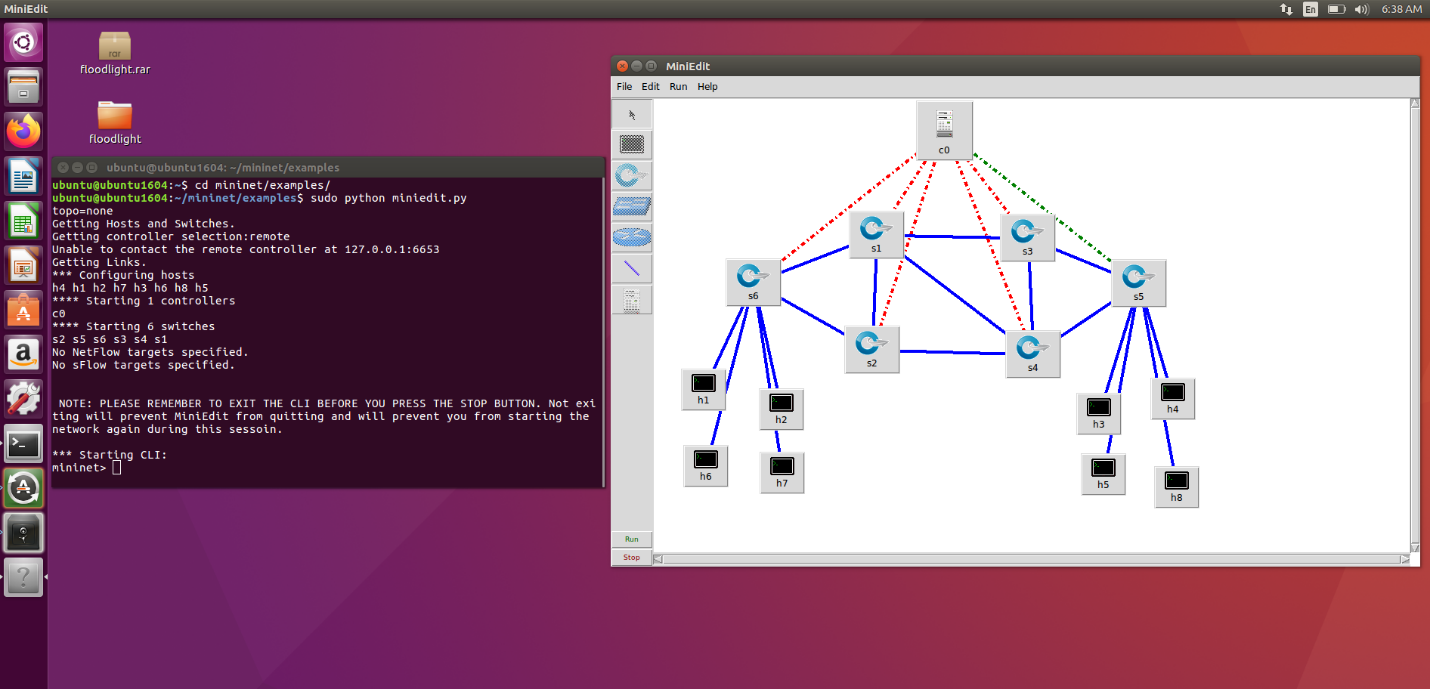
[3.3.Network Monitoring 1](#_Toc1876303415)

# 1. Introduction

The purpose of this report is to analyse and examine a network infrastructure using SDN core concepts and technologies. This will include a proof-of-concept design using an open source SDN solution known as Mininet, as well as an overview of the Floodlight Openflow controller, network ping tests, packet analysis with Wireshark, firewall implementation, load balancing and network monitoring.

# 2. Initial Network Configuration

## 2.1. Proof of Concept

The topology was created via Miniedit, accessed through a command line python script as shown in figure 1. The “c0” represents the central network floodlight controller, which is connected to 6 switches, delegating traffic between 8 different hosts machines. Additional options that were required to be changed were the controller port to 6653, the controller type to remote and the protocol to TCP.

After the topology was created, the Floodlight Openflow SDN controller was implemented and chosen. Figure 2 shows the hosts connecting to the switches IPs using the TCP 3-way handshake process

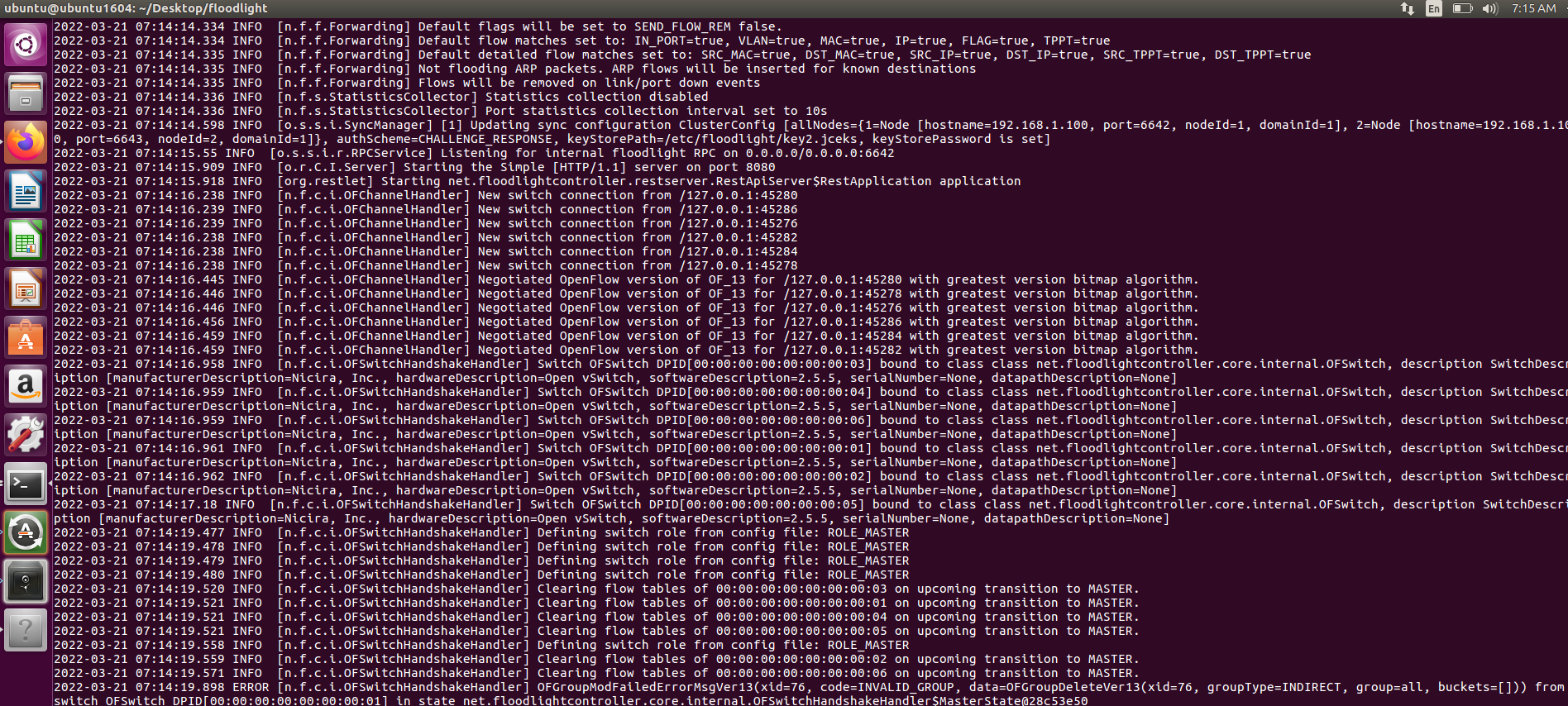
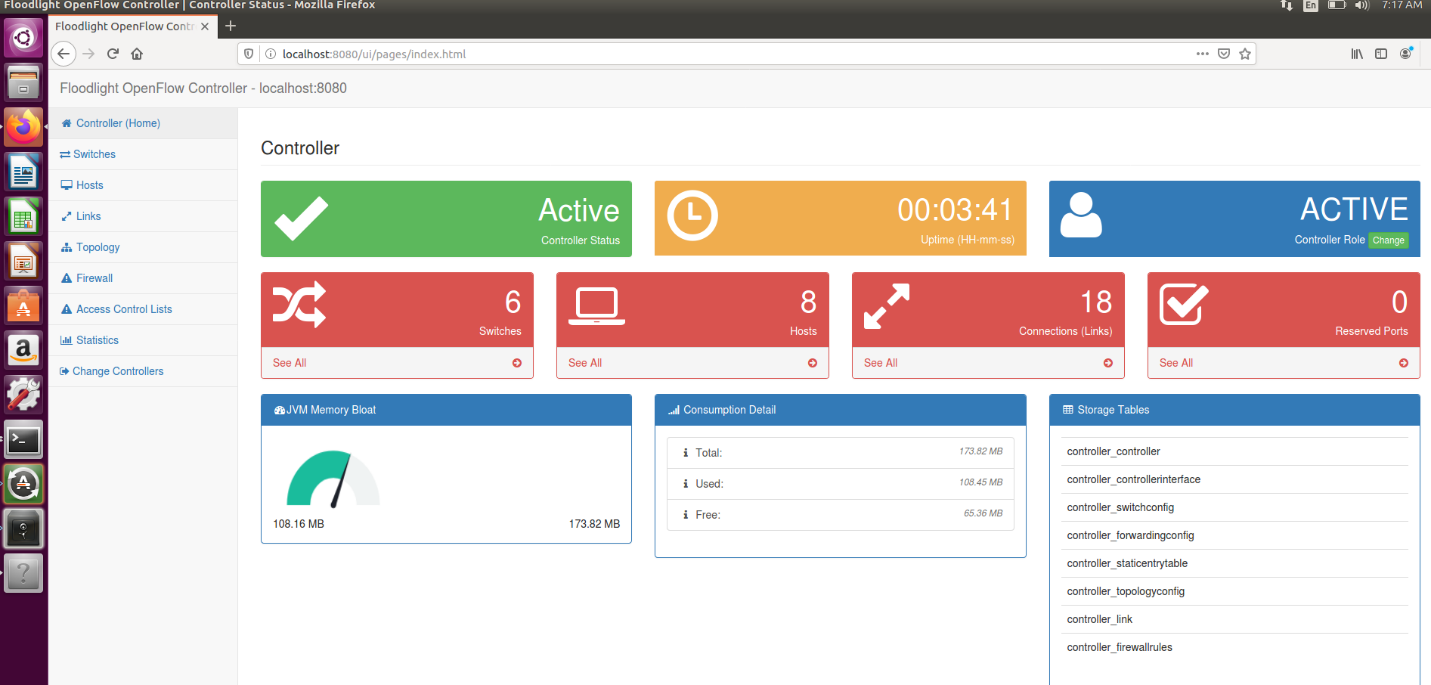
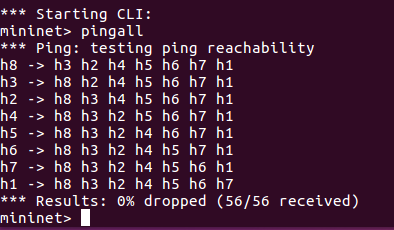


Figure 2: Topology Connections

In contrast, another popular controller is Opendaylight, which is similar to Floodlight and can provide northbound services to multiple stacks and of neutron networks. However, Floodlight provides more efficient services in terms of packet loss and latency.



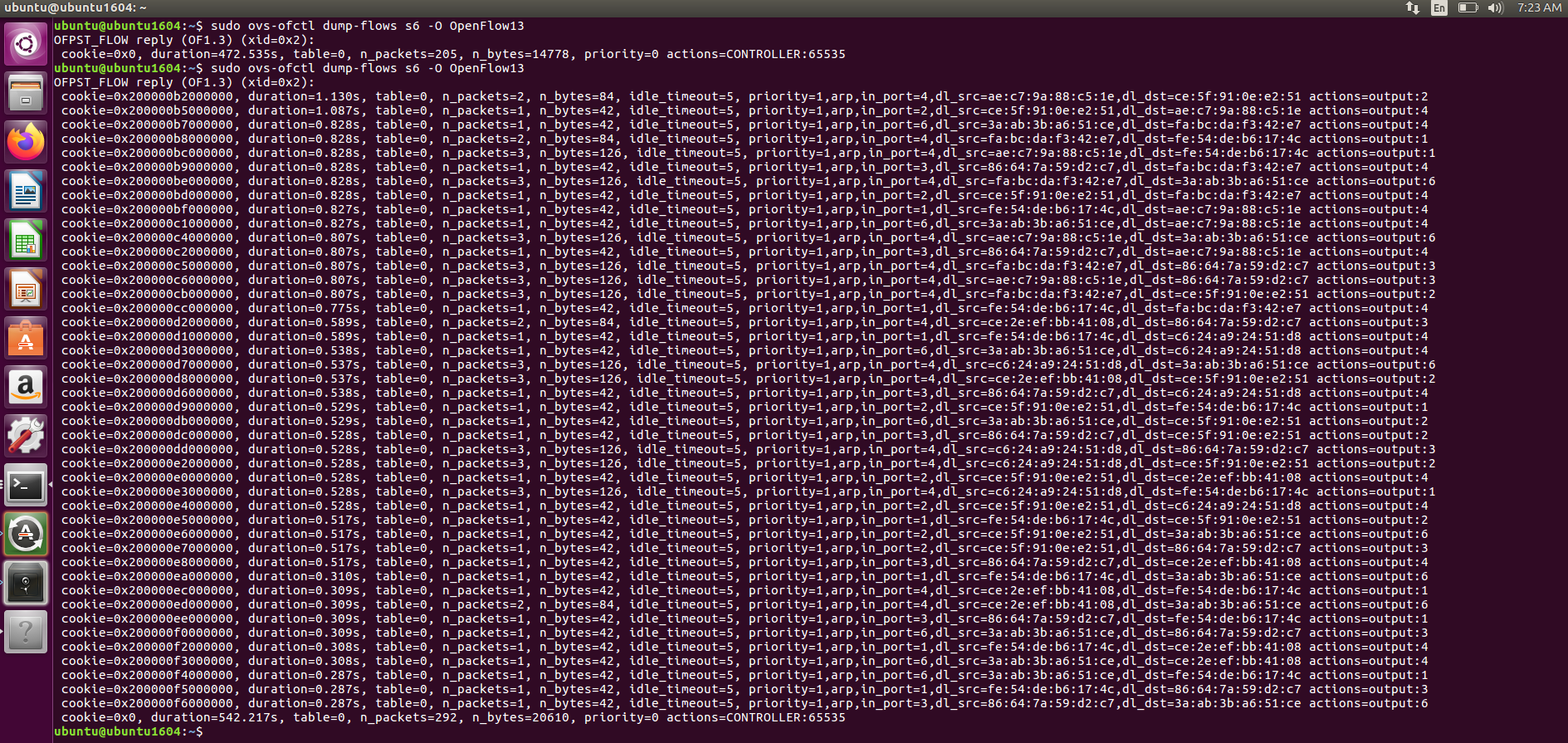
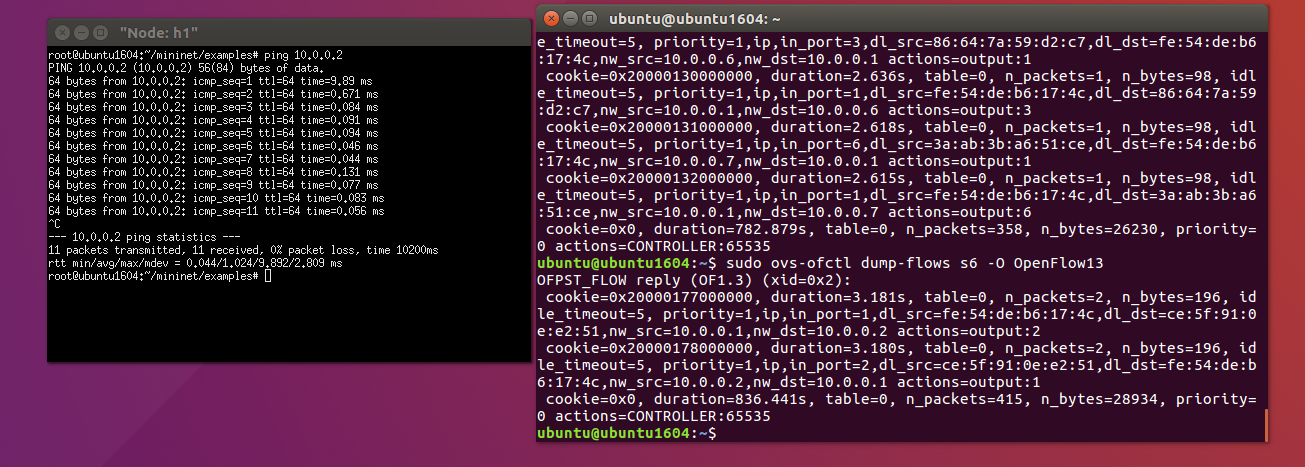


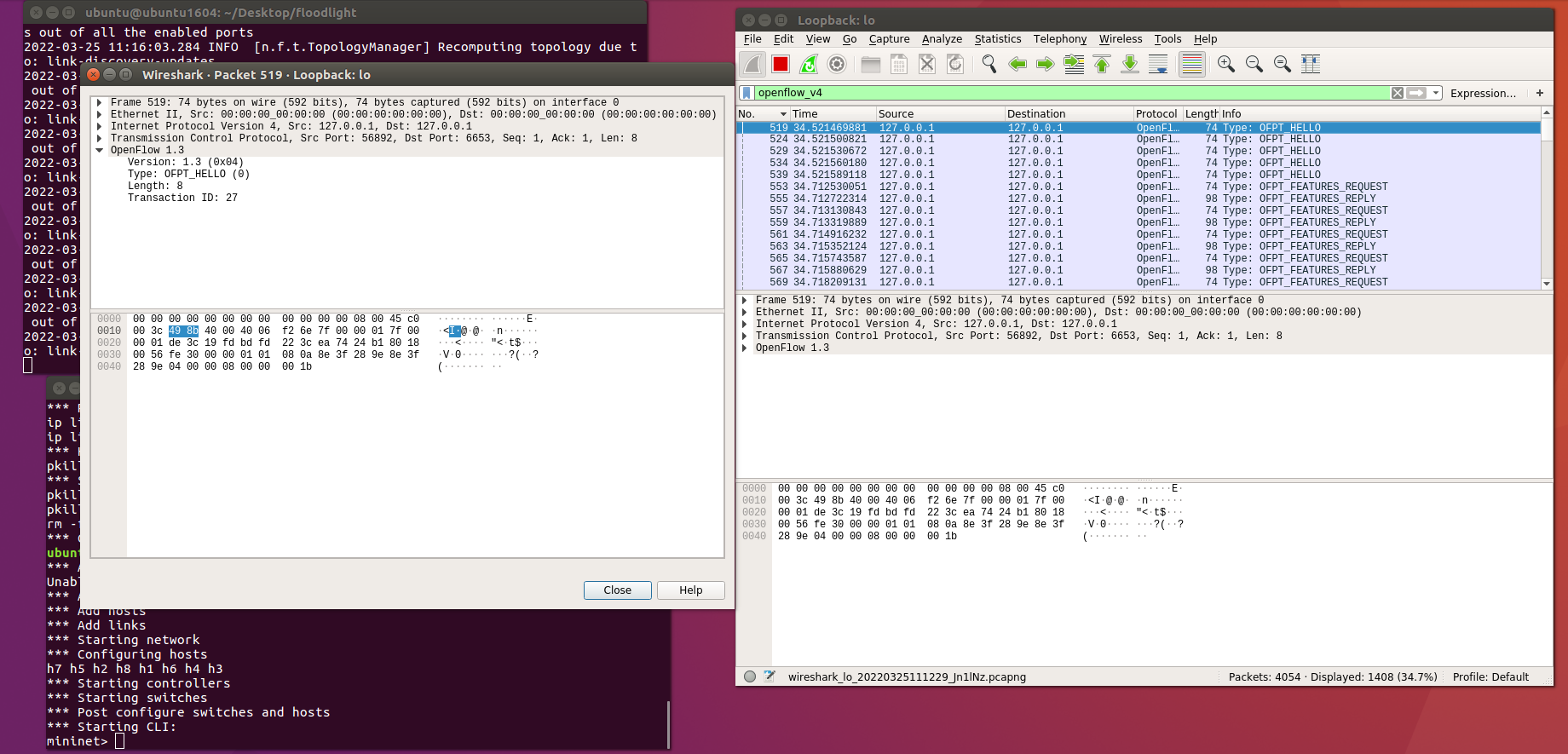
Figure 3 shows the Floodlight GUI which has a visual representation of all the hosts and switches, and to test the network works, a ping all test was done in Mininet which resulted in 0 packet loss, showing the topology working correctly (Figure 4).

Figure 5 shows the topology flow table, meaning any packet information is displayed here and this further shows the network working correctly without any errors or drops.

Figure 6 represents the test done to achieve the flow tables, and that each individual node can ping each other, concluding the network is setup accurately with no packet loss.

## 2.2. OpenFlow Packet Analysis

The Openflow packet analysis was achieved using Wireshark, where the Floodlight controller was started, and traffic was filtered to Loopback via openflow\_v4. This is because the network uses the Openflow framework and therefore this filter achieves relevant traffic (figure 7).



OFPT\_Hello establishes the handshake when an Openflow connection is confirmed and contains elements to help the connection setup. The Floodlight controller and switch uses OFPT\_Hello to also establish a familiar version number in order for messages to be communicated across the network.

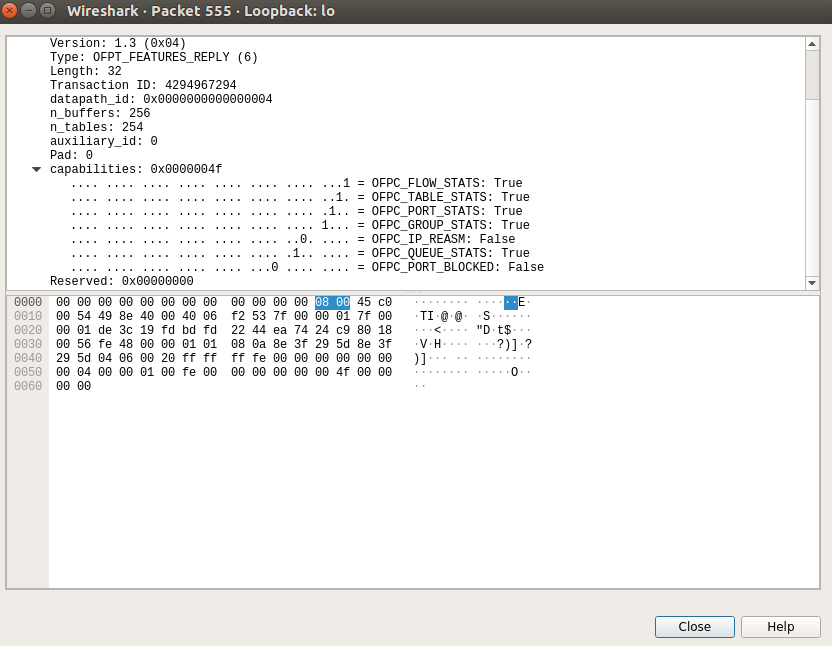
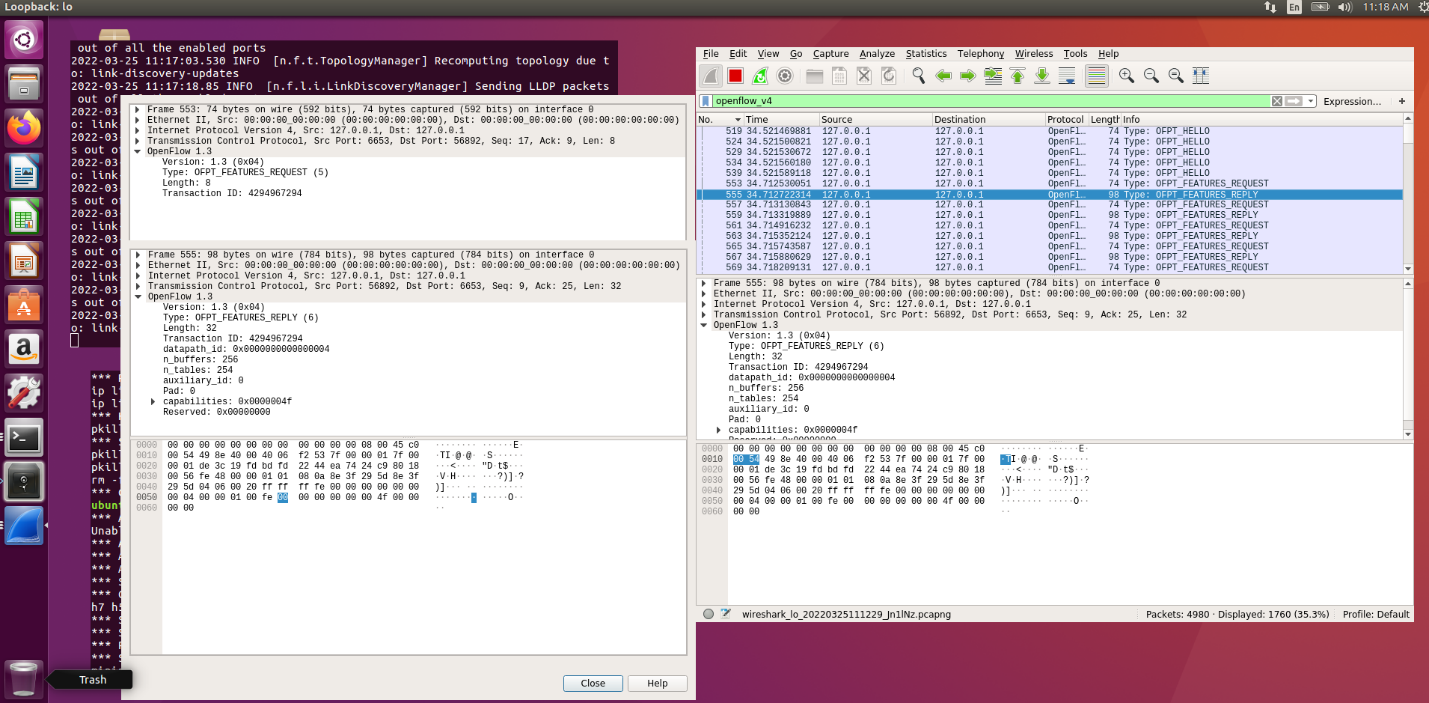


Figure 8: Wireshark OFPT\_Features\_Reply

Figure 9: Wireshark OFPT\_Features\_Request

Figures 8 and 9 exhibit the Openflow features messages, which are both used by the controller. The features request identifies the switches on a network and reads its basic attributes, consequently leading to the session being created simultaneously. The features reply establishes and confirms that the switch is connected to the controller and includes STP if the switch supports it.

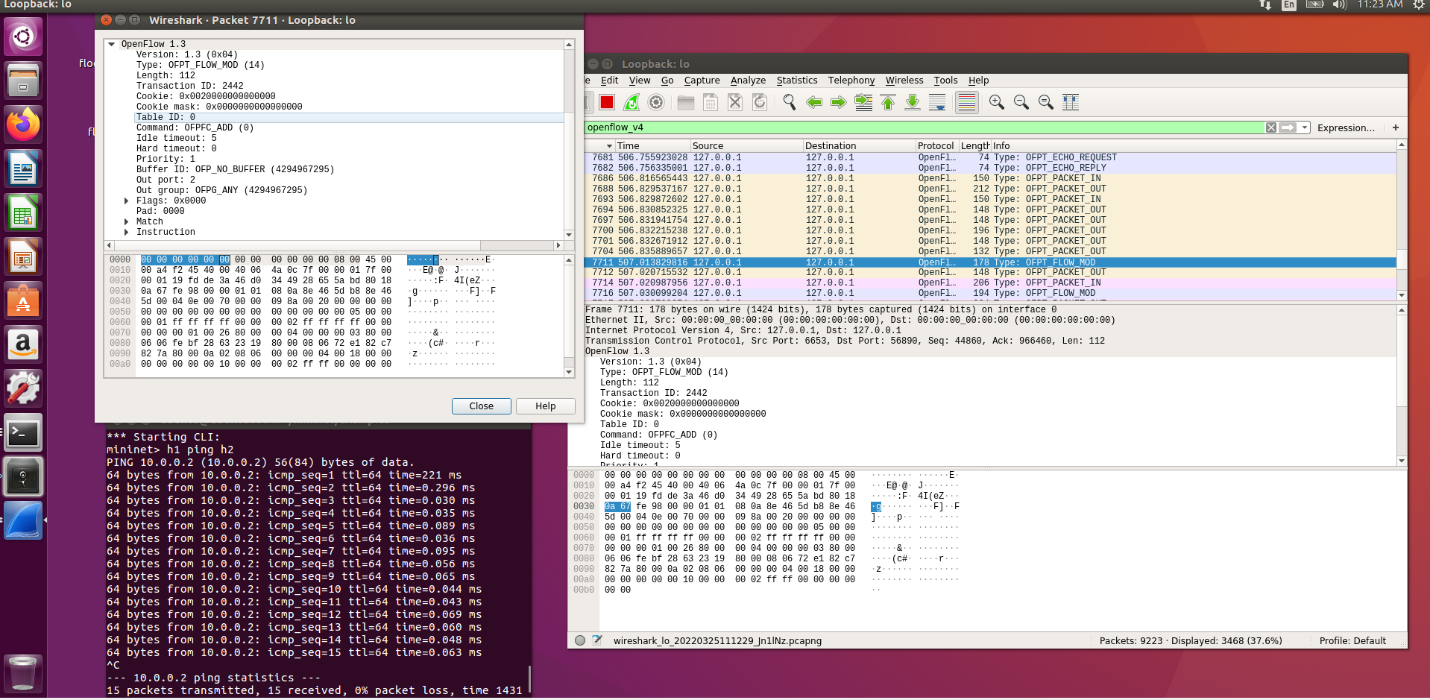
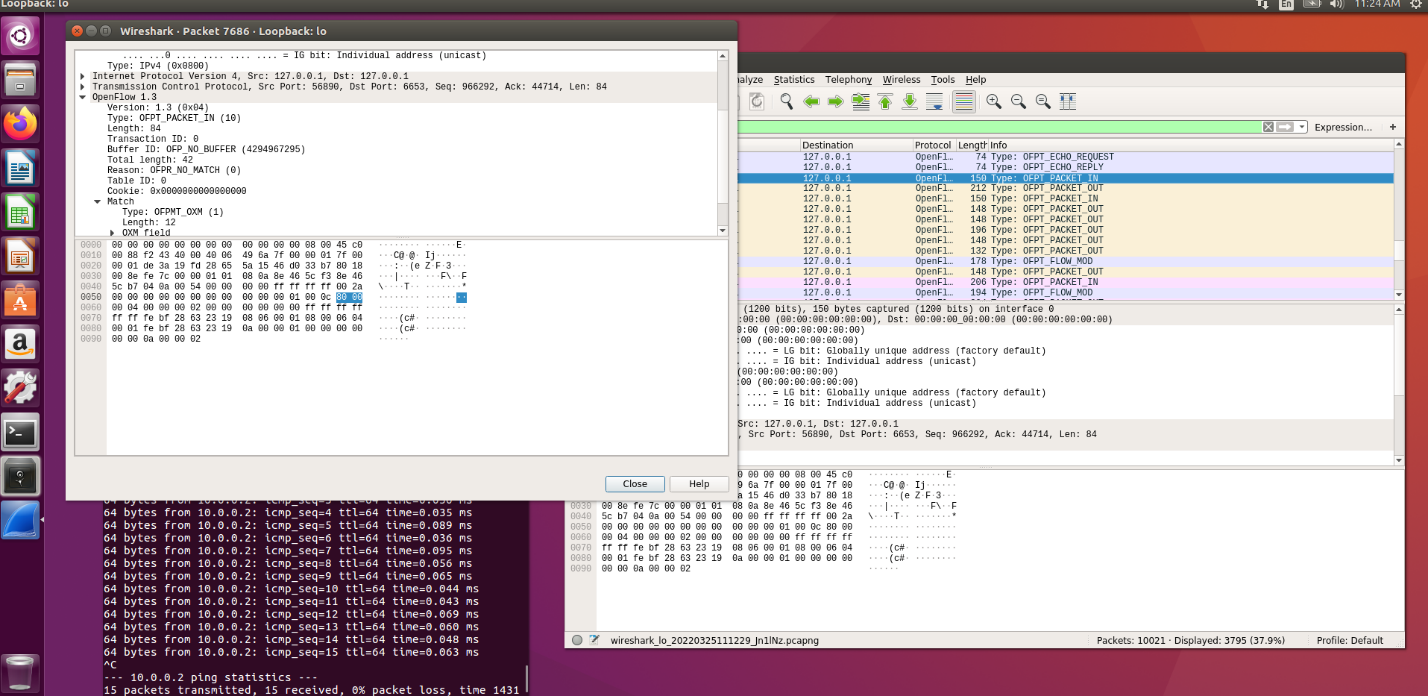
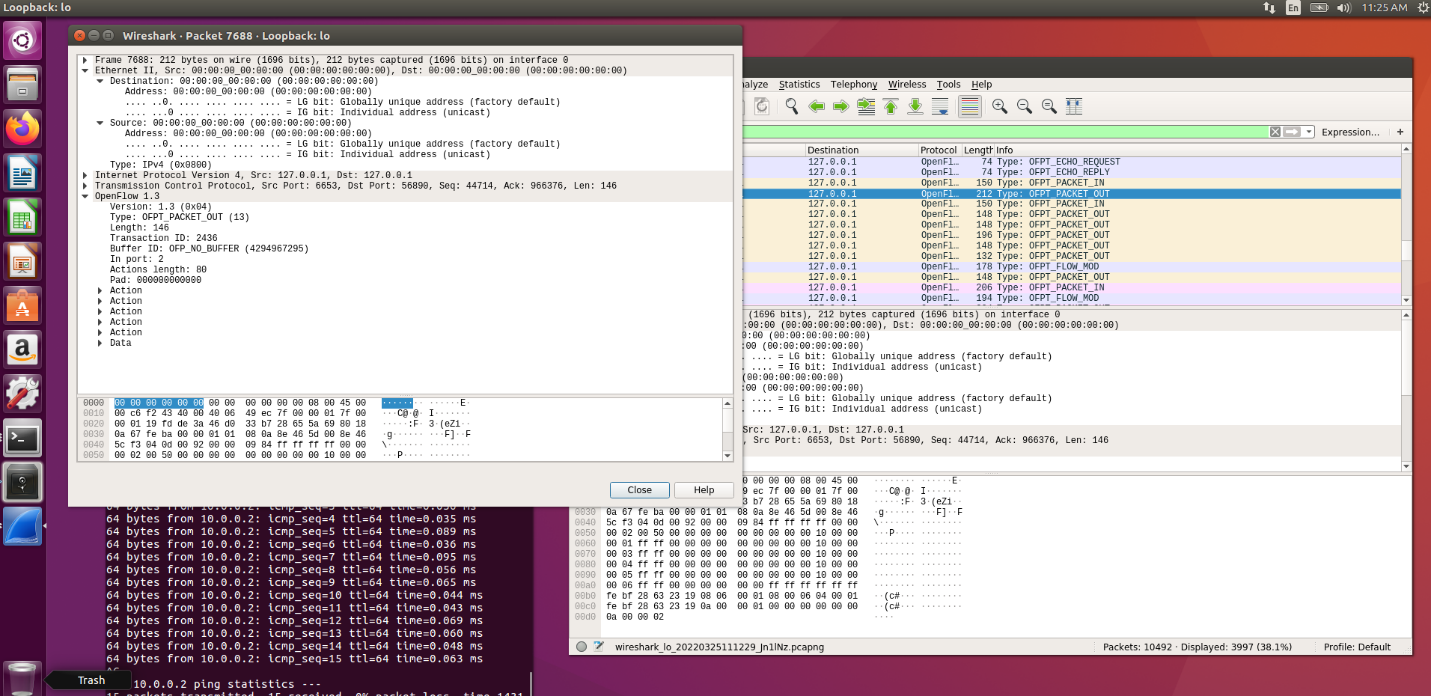


Figure 10: Wireshark OFPT\_Flow\_Mod

Flow table modifications and configuration entries can be adjusted through the FLOW\_MOD traffic request. This is carried out from the controller, and therefore any new entries to the specified flow table are through this request (Figure 10).





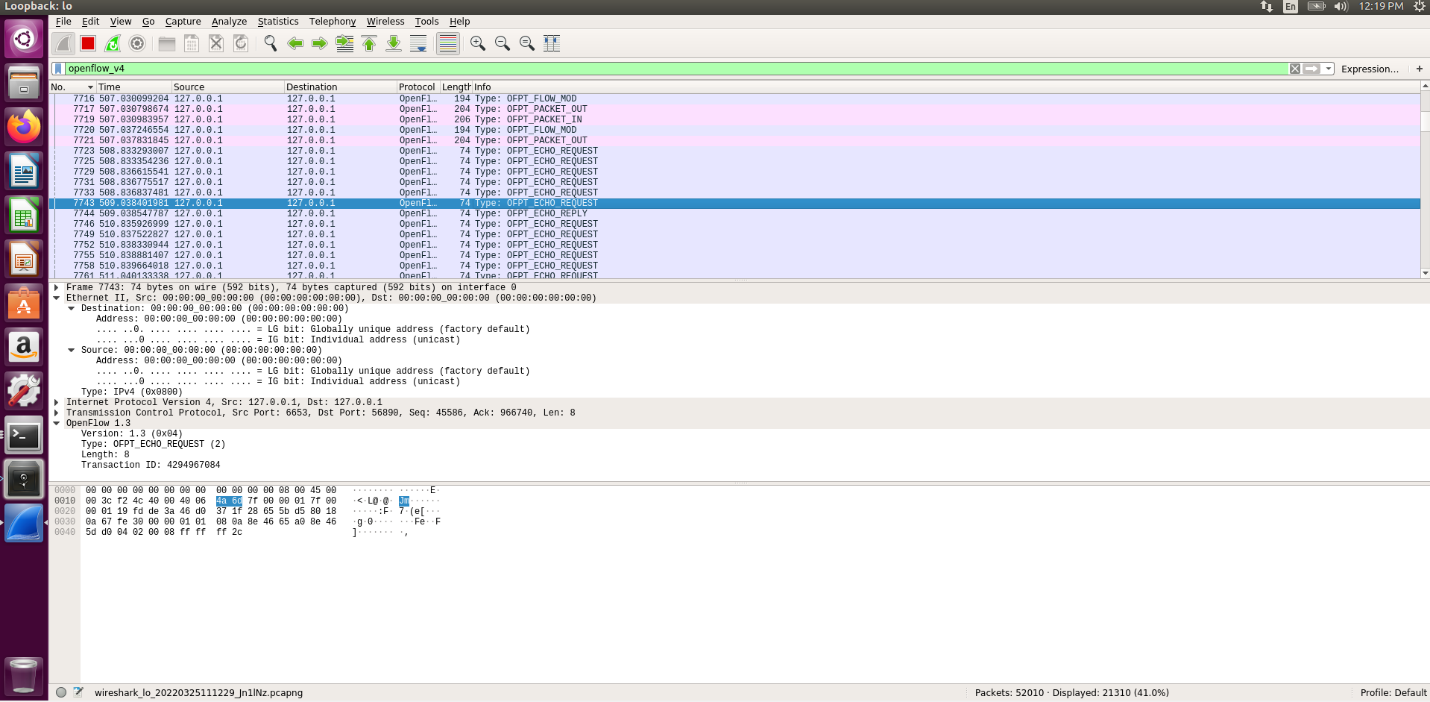
Figures 11 and 12 show the Openflow packet in and out messages. The ”packet in” messages use the data path to show packets received, which are then sent to the controller. The ”packet out” message is used when the controller sends a packet back through the data path.

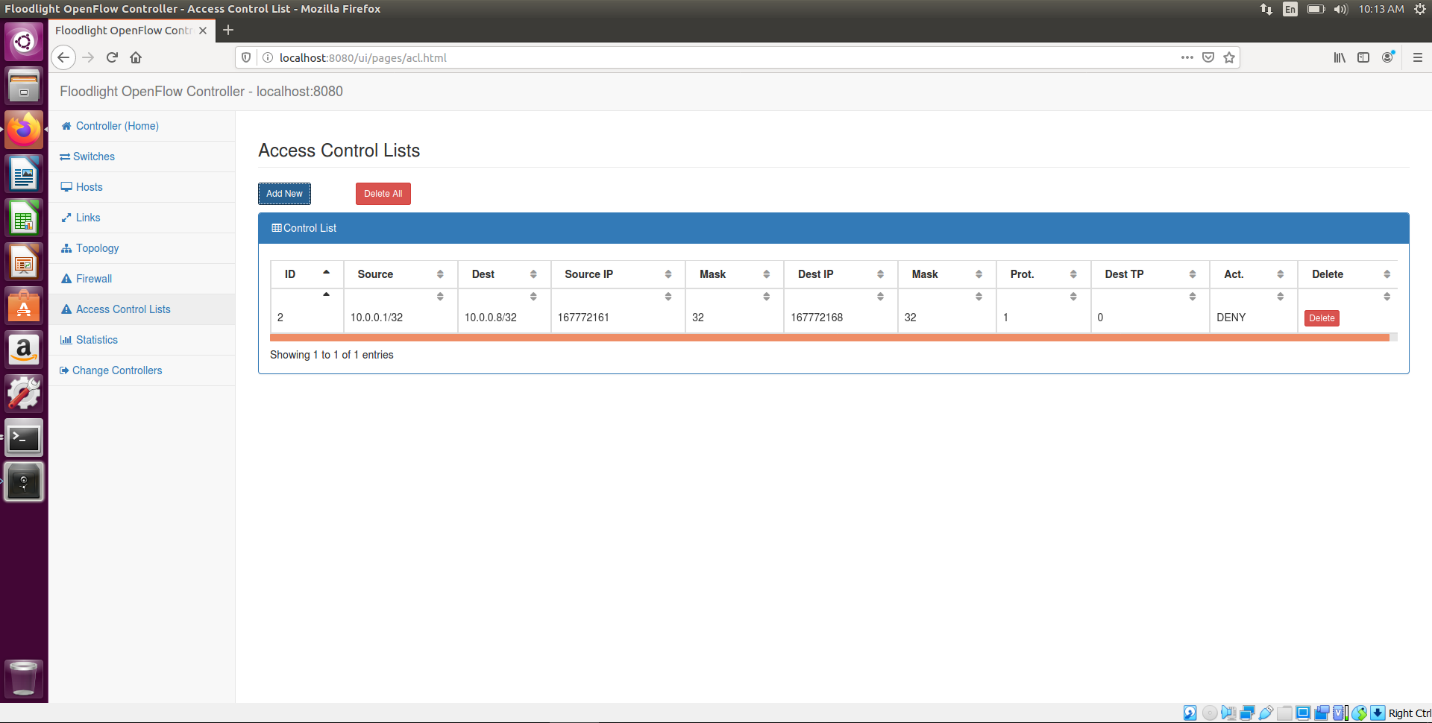
Figure 13: Wireshark OFPT\_Echo\_Request

Figure 13 shows the different types of echo-reply and sent messages across Wireshark. This means that the switch and controller can confirm an integrity between the connectivity of Openflow, to ensure connection is officially established. Each of these will help with connectivity troubleshooting and testing.

# 3. Network Functionalities

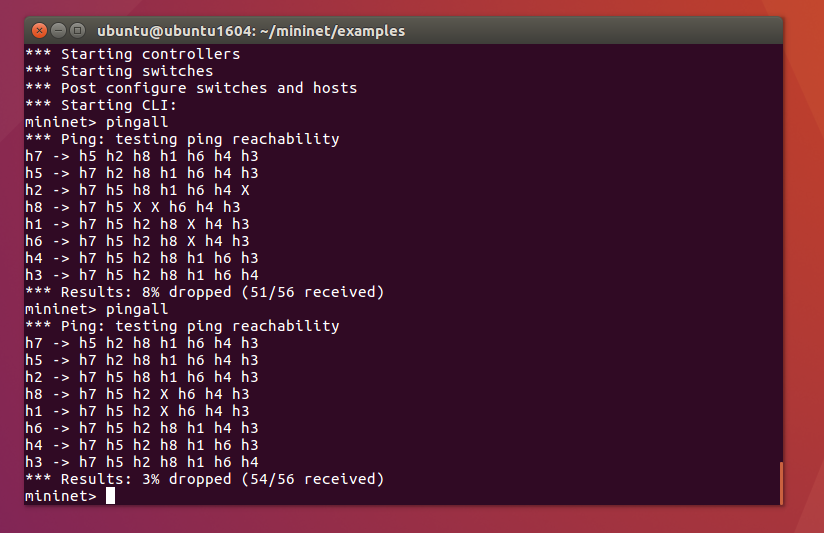
## 3.1. Firewall Implementation

The firewall security policy that will be applied will be done through the access control list (ACL) as shown in Figure 14. Within the network topology (figure 1), the hosts are split between different switches. These hosts could represent different departments and it’s possible to block specific traffic between these departments. Therefore, the ACL rule was created was to block ICMP traffic from the source, host 1, to the destination, host 8.



This is significant because without ACL firewall policies, it would be very easy to one department to access other departments data, and that the ports within the network would be open to the whole company, creating a security risk based on the principle of least privilege. Additionally, if one host was to become hijacked, that potential attacker could access network traffic within the whole network.

Alternatively, a company could implement a firewall to improve security posture of the network. This is because firewalls perform stateful inspection, resulting in each packet being inspected per flow, whereas ACL are limited to stateless which is per packet. Furthermore, ACL do not include intrusion detection and generally only include layer 4, however, ACL is very easy to setup to create a blocked connection between 2 hosts.



Consequently, as a result of the ACL rule implementation between hosts 1 and 8, a Pingall test was carried out, as shown in Figure 15. This is to ensure that the ACL filter rule works correctly within the network. An incorrect test would mean that the ACL rule is working incorrectly, and therefore could be a security issue.

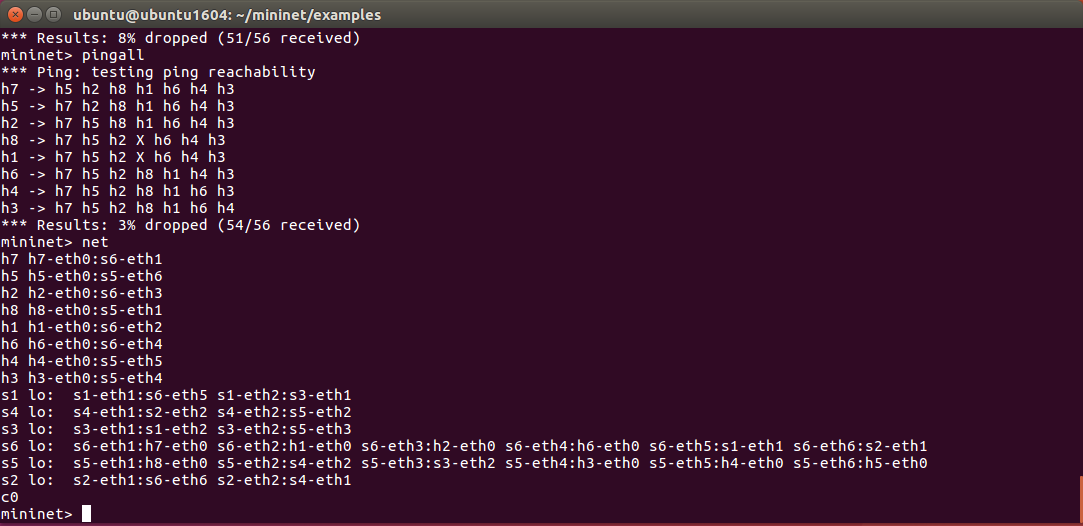
Using the command line within Ubuntu, the Pingall test within the Mininet interface resulted in only 2 packets dropped (figure 15), and those packets dropped are represented by the ‘X’, shown on host 1 to host 8, and on host 8 to host one, resulting in a successful ACL implementation.

## 3.2. Load Balancing

In order to balance network loads across distributed paths, load balancing needs to be introduced to the network. A company with 250 employees simultaneously using a small network topology would need the most efficient distributed data paths, due to high traffic areas being managed and capacity utilisation being upheld to ensure one server is not overworked.

Additionally, load balancing performs different functions such as only sending traffic to online servers to ensure availability and reliability of requests, which can occur from both client and network side of things. For example, request happen either end of the network, from the controller, switches, or hosts. Not all 250 employees will be logged on simultaneously, therefore, if all the servers were in use, capacity utilisation will be wasted, incurring higher costs.

However, load balancing provides a flexible service where servers can be added or subtracted anytime depending on demand levels, meaning expenses and capital of running the servers will decrease. (Creating manually – installing flows in flow table of switch)



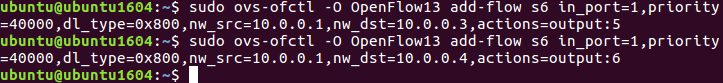


Figure 16 shows a connection peered Mininet network, which is an overview from the topology as shown in figure 1. From this, connections can be identified in order to make load balancing changes to the network. For example, all hosts only have 1 port, and they are all connected to port 0. Host 7 is connected to port 0 through switch 6 on port 1. The below switches show different ports since they have access to multiple.

Within the network topology (Figure 1), the load balancing will take place at switch 6, port 1, on host 3 port 5, and on host 4 port 6 as shown in Figure 17. From figure 16, you can see where switch 6 is connected to port 1, and where each of the different ports are connected from the above commands. For example, host 3 is connected to port 0, through switch 5 on port 4. This load balancing is taking place through switch 1, where switch 6 uses port 5, and on switch 2, uses port 6.

Installing the flow entries means we can see the load balancing take place through the number of packets going through each switch and each port.

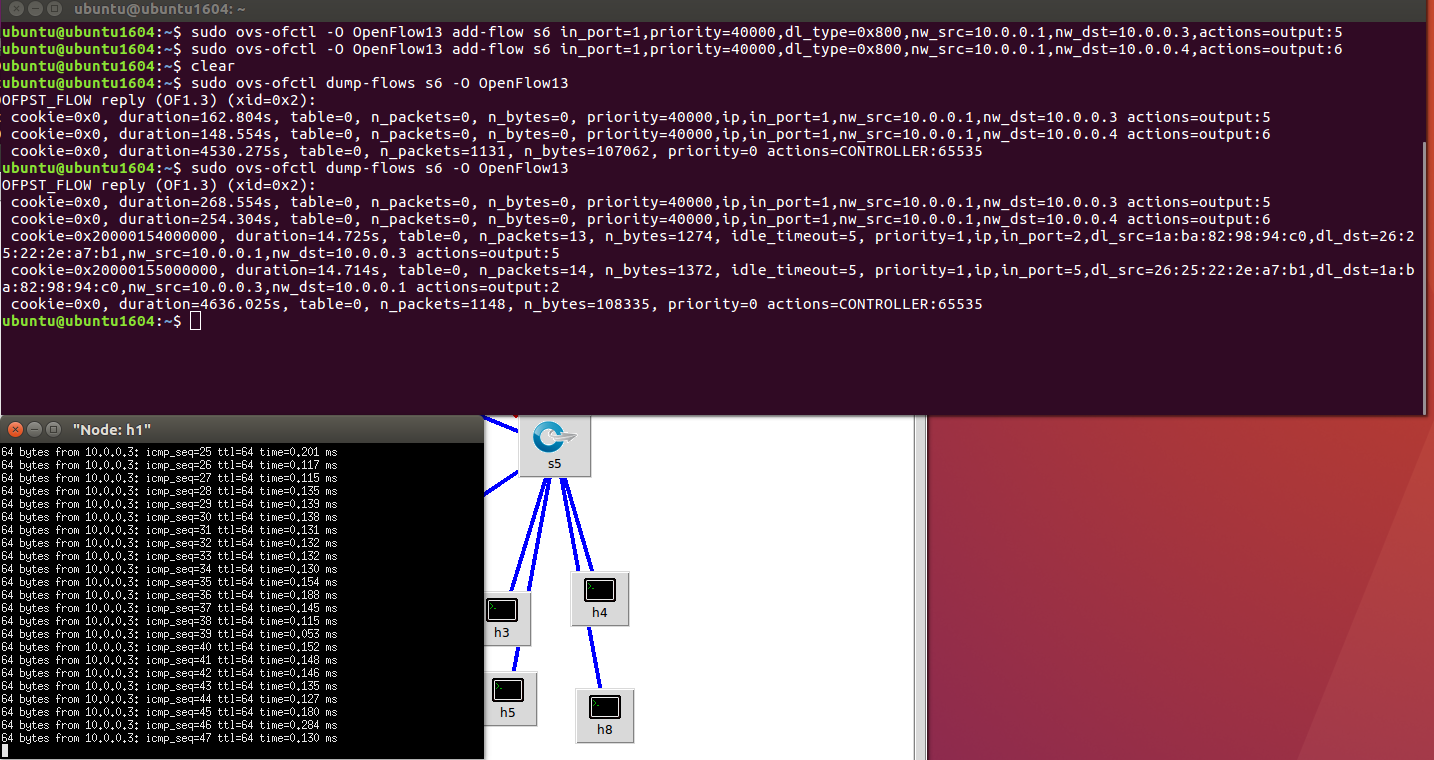
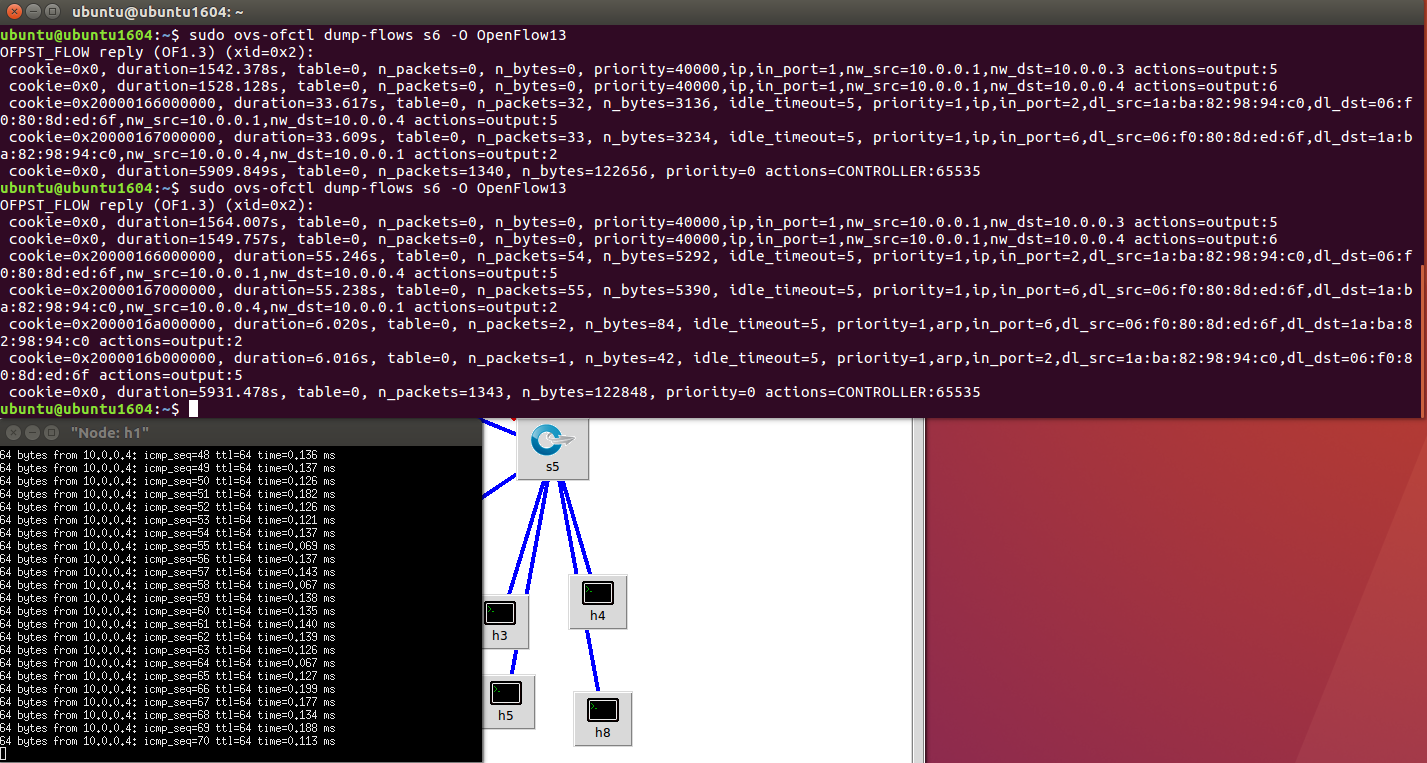


Figure 18: Testing Packet Count

Now the flow entries are installed, the packet count can be traced through the switches and hosts to ensure load balancing is occurring. Using the command line node, host 1 is opened to ping host 3 continually. The Openflow dump commands as shown in Figure 18 shows the current packet count while the ping is being carried out, and this packet count increases to 13 on host 3 and 14 on host 1.

Continuing with the packet testing, Figure 19 shows the same commands being used from host 1 to host 3, except packet count is increasing without the 2nd link being utilized. As a result, as long as only one packet counter is increasing, load balancing is shown to be taking place.



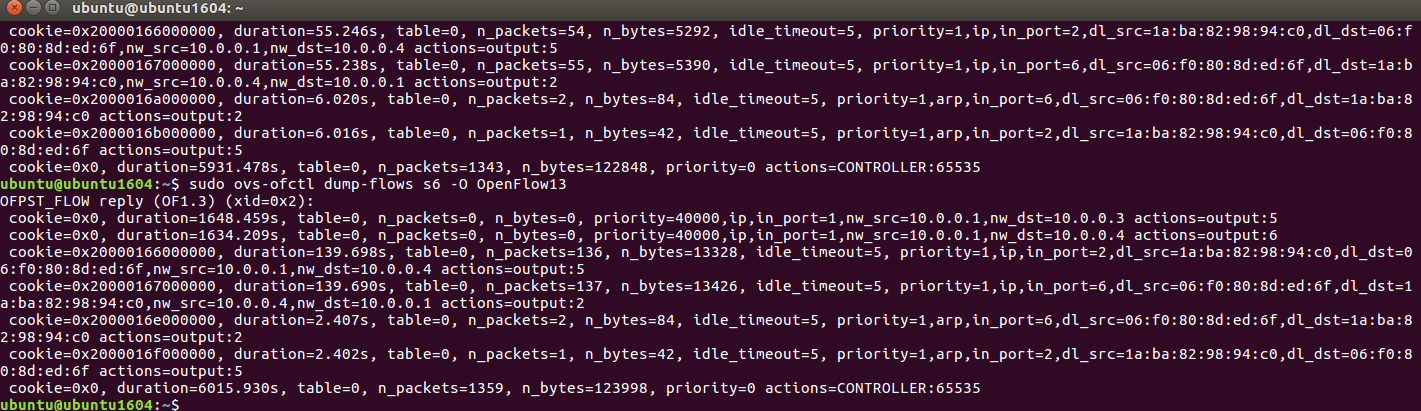
To ensure it works correctly across all hosts, host 4 was also tested as shown in Figure 20. From the dump flow tables, packets were increasing across the switches specified for load balancing, which shows the load balancing works correctly.

Figure 21 shows load balancing working with the packet count increasing. The controller is bidirectional, meaning doing the load balancing manually on one side results in the controller automatically load balancing the other.

## 3.3. Network Monitoring

The last step of the network setup is to monitor the network to ensure all the controllers, switches and Openflow devices work correctly. Additionally, any statistics that a company want to find out can be viewed via the Floodlight rest API.

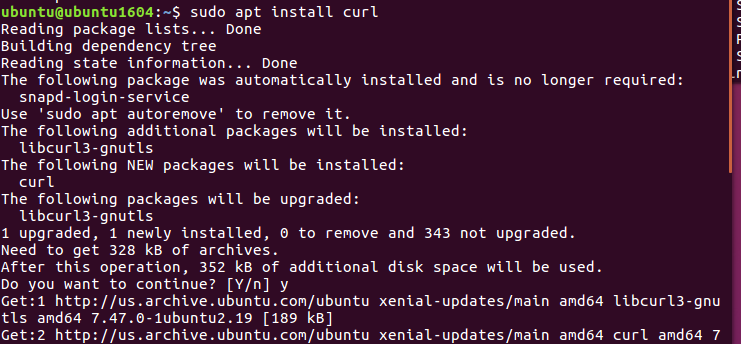


Figure 22: Curl Installation

In order to access the rest API, curl needed to be installed (Figure 22), which is used to transfer data from different servers. Alternatively, python could be used, however, using the command line interface provide ease, rather than creating a new script.

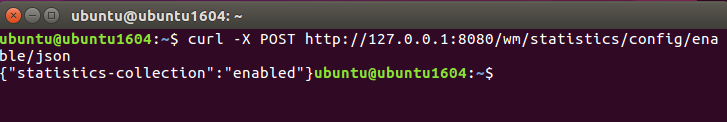


Figure 23: Enable Statistics Collection

To collect statistics, the controller has to be enabled through command in Figure 23. 127.0.0.1 is the default loopback address for the controller, and the port 8080 is the default API on the controller.

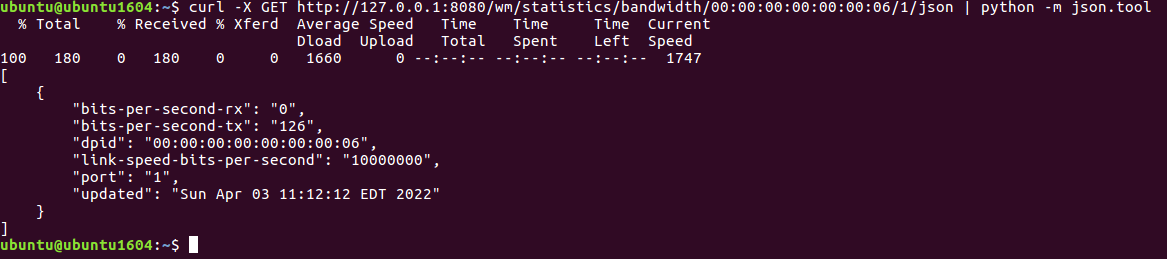
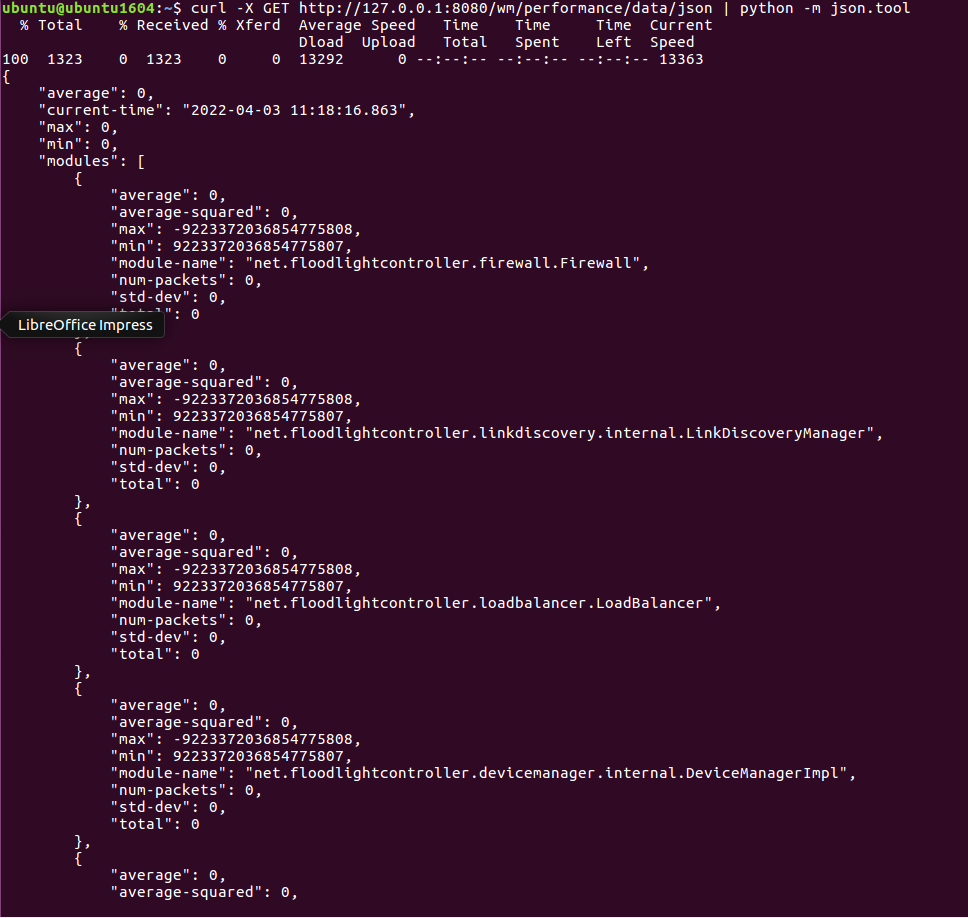


Figure 24: Bandwidth Statistics



The first statistic is bandwidth, which is important to understand the speeds of a network. The command was a part of the Floodlight rest API as seen above and the format is json because data results are more organised. As seen from Figure 24, the statistics display bits sent/received, DPID (switch ID), link speed and the port being queried.

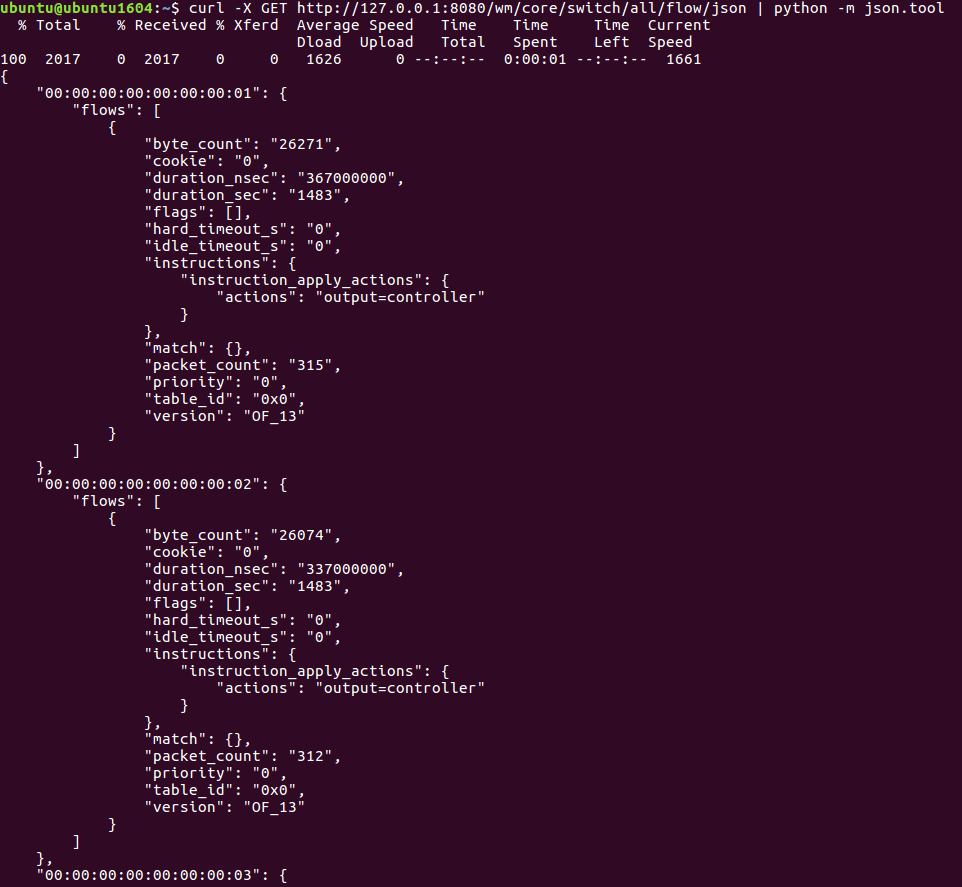


Figure 26: Switch Flow Statistics

Figure 25 shows the performance of the controller in terms of average packet in processing times, which can be significant for a company when analysing performance of the controller which controls the entire network.

Figure 26 portrays the status of all switches, including packet counts, flows and table IDs. Using this can help a company establish switch performance if any issues arise.

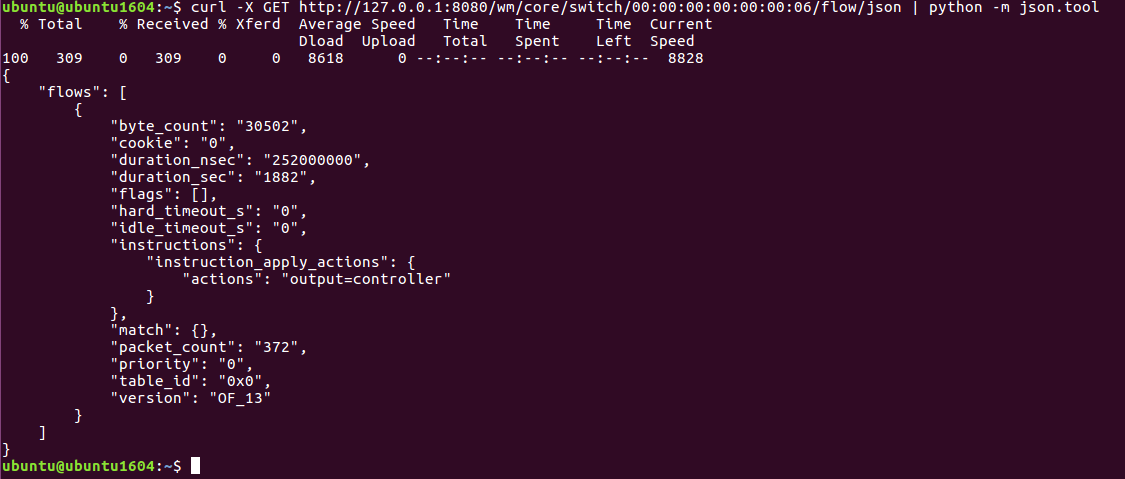


Figure 27: Individual Switch Flow Statistics

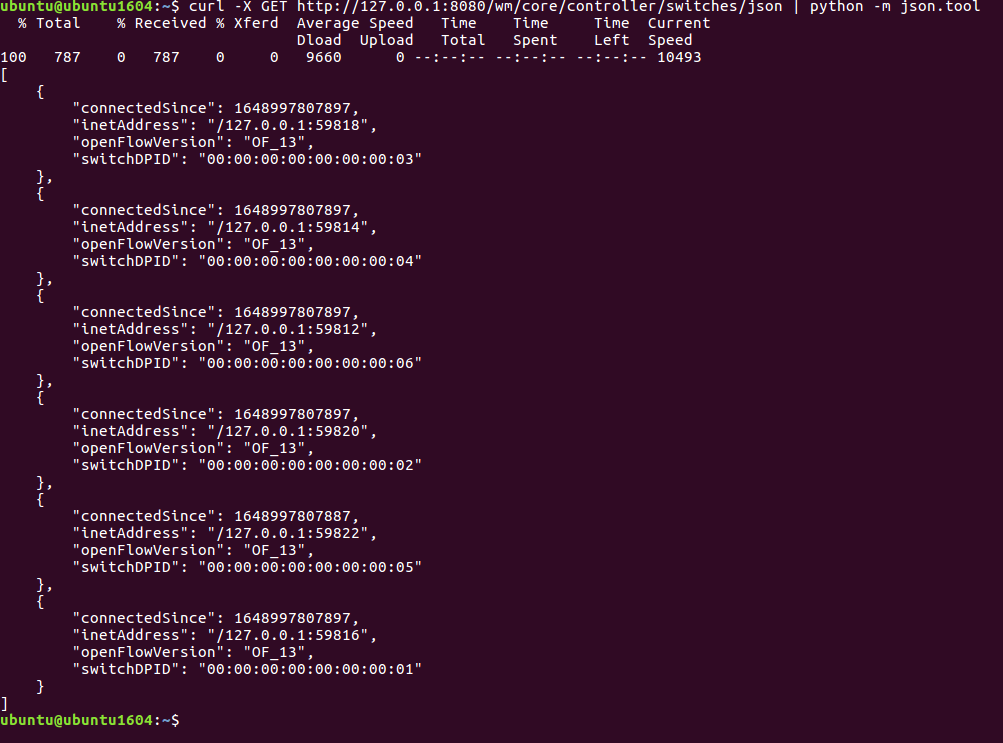


Figure 28: Controller Switches

Figure 27 shows a similar command switch, however, if a company wanted individual detailed statistics on a specific switch, for example, switch 6, then the above command finds those statistics.

Since switches are an important part of the network, the statistic as shown in Figure 28 portrays all of the switch DPIDs which are connected to the controller.



Figure 29: Controller Link Summary

The final controller API statistic shows an overall network overview as shown in Figure 29, where all switches/hosts can be seen as an overview. This means that any new added network devices can be seen through one statistic command.