International Journal of Latest Engineering Research and Applications (IJLERA) ISSN: 2455-7137

Volume – 03, Issue – 04, April 2018, PP – 92-98

Implementation of Layer 2 Rules using Software Defined Networking

G Anagha[1], Deepthi G S[1], Archithaa S Rao[1], Pooja K[1], B Sudha[2], Sunita Katre[3]

UG Student[1], Assistant Professor[2], Research and Development Engineer[3]

Department of Telecommunication Engineering, Bangalore Institute of Technology, Bengaluru, India[1,2]

TATA Power Strategic Engineering Division, Bengaluru, India[3]

Abstract: Software Defined Networking (SDN)is required for integrating and converging networks and for bringing in flexibility to networks and to reach the objective of vendor independent networking. By effecting the traffic patterns, improving scalability and achieving vendor independence, a controlled network is developed. Here we work on a network consisting of virtual hosts, switches, controller and links. This calls for the need of a network emulator and we employ Mininet for the same. Mininet runs on Linux platform. This idea when realizedon a PC, we require to run more than one operating system at a time. Virtual box(VB) allows a user to run software written for one operating system(OS) on another without having to reboot. Here we make use of an open source SDN controller, OpenDayLight Controller. This provides modular open platform for customizing and automating networks of any size and scale. Postman is the http client which is employed here to push the flows. The layer 2 rules which include flow rules for packet transfer are pushed upon desired network as a part of designing.

Keywords: Software Defined Networking, Emulator, Network Topology, Layer 2 Rules, Port matching, Packet capture

I. Introduction

In this section we briefly discuss the major softwares and applications used in the implementation of the above-mentioned idea. This gives the brief note onSoftware Defined Networking(SDN), Mininet, VirtualBox(VB).

A. Software Defined Networking (SDN)

SDN is an approach to using Open protocols like OpenFlow. This makes use of software control for operation of network and control the network elements such as switches and routers. This kind of networking is a novel approach to cloud computing that facilitates network management and enables programmatically efficient network configuration in order to improve network performance and monitoring [1]. There is a provision for centralization of network intelligence where packet forwarding and packet routing are not associated to one another according to functional idea of SDN. SDN architectures decouple network control and forwarding functions enabling network control to become directly programmable and the underlying infrastructure to be abstracted from applications and network services [2].

Functional elements in SDN:

The components involved in a network controlled by the method of SDN are described below

i. SDN Application: SDN Applications are a set of suitable codes that operates directly and independently on the network elements for exchange of information regarding network requirements.
ii. SDN Controller: The SDN Controller is a programmatically controlled functional element that translates information from SDN Application layer to SDN data paths. It also gives a summary of view of network to SDN Applications.
iii. SDN Datapath:The SDN Datapath is a logical network element that displays and monitors data forwarding and data processing ability. The representation may consist of all resources or a selected group of resources.
iv. SDN Control to Data-Plane Interface (CDPI): The SDN CDPI is the linking element between SDN Controller and SDN Datapath and uses program to control data forwarding end event alerting process.
v. SDN Northbound Interfaces (NBI):A northbound interface of a component is an interface that conceptualizes the lower level details. SDN NBIs are interfaces between SDN Applications and SDN Controllers

B. Mininet

Mininet is a network emulator. It runs on a collection of end-hosts, switches, routers, and links on a single Linux kernel. In a virtual way one can programmatically make a single systemto resemble the entire communication network, that runs on a common kernel, system and user code by using Mininet. A Mininet host behaves just like a real machine and the user can secure shell (ssh) into it. The programs run on Mininet can send packets over virtual ethernet switch or router with the given amount of queueing. In other words, Mininet's network elements are created using software rather than hardware while they behave similar to discrete hardware elements. The user can create and run Mininet topologies both simple and complex by writing Python Scripts [3].

C. OpenDaylight Controller (ODL)

Communication providers can adapt their networks to be more flexible to their needs. At the same time, they required driving network automation to improve operational efficiency. ODL is the largest Open source SDN controller used for this purpose. ODL is the platform that can be used to automate networks with all possible topologies. This focuses on programming ability of a network. ODL Controller controls the network and is a pure software hosted by Linux foundation. It is a pure Java virtual machine (JVM) and can be run on any OS as long as it supports Java [4].

D. Virtual Box (VB)

VB allows a user to run software written for one OS on another without having to reboot. It may be installed on a number of host OS including: Linux, Mac, Windows Solaris and Open Solaris. Here we run Mininet and ODL on VB.

II. Creation of Topology Through Mininet

This section describes the methodology to create a topology involving various network elements in mininet following the process of installation of Mininet onto VB.

Import *.ovf* file of Mininet into VB. The Mininet runs when the VB is set to run in the Host-only network configuration mode [5]. Check the connectivity between root and Mininet with ping command as shown in Figure2.1. Consider the exemplar topology in the Figure2.2 whose elements are described in Table2.1.

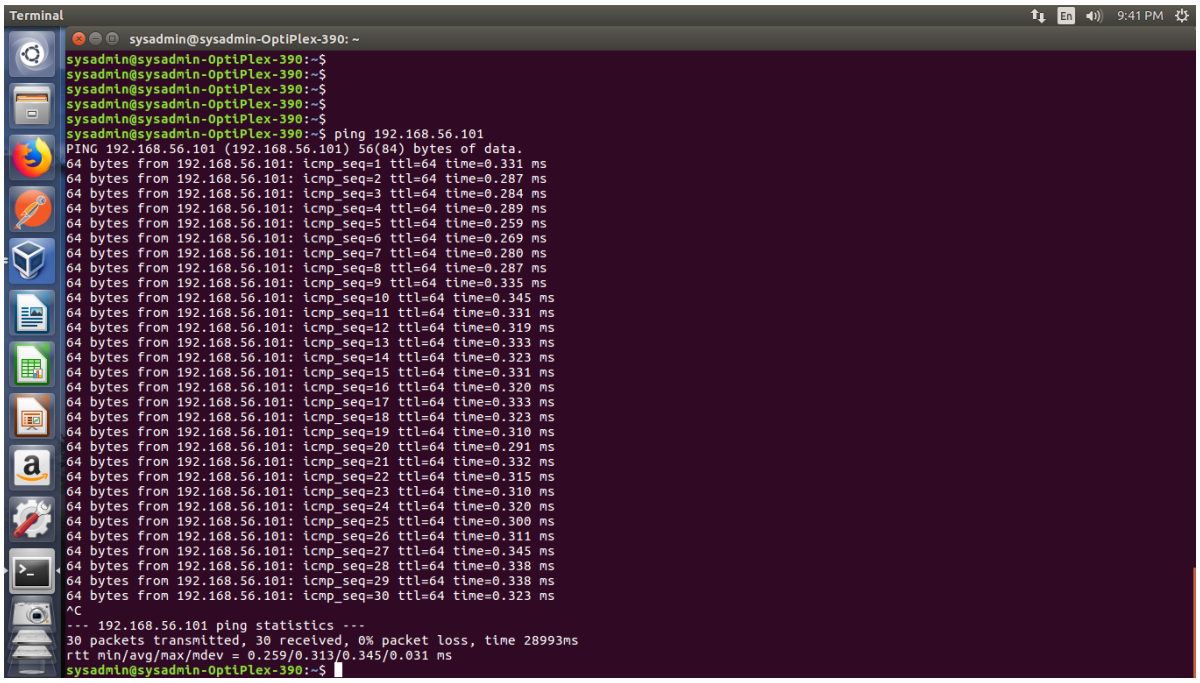

Figure 2.1. Connecting PC and Mininet

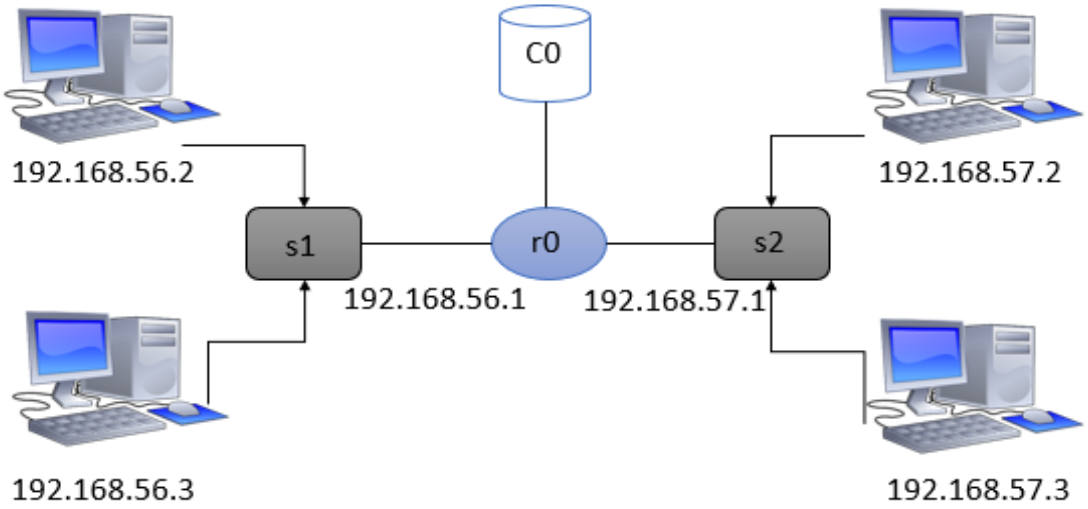

Figure 2.2. Exemplar Topology

Table 2.1. Table describing the Network elements in the exemplar topology

Network elements	**Number**
Host	4
Switch	2
Router	1
Controller	1

We write a Python script based on the following flowchart shown in Figure 2.3.

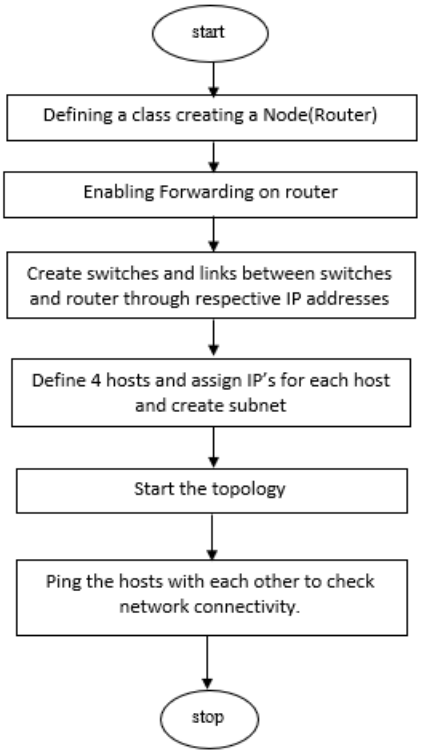

Figure. 2.3. Flowchart to realize the exemplar Topology

III. ODL: Linking the Remote Controller with Mininet

ODL can be accessed from web to keep the account of packet transfer, it is used primarily as Graphical User Interface(GUI) to view the topology created. It shows nodes, IP addresses of hosts and several possible arrangements of network elements. Port 8181 is used for web interface, and it is accessed through browser by entering the following Universal Resource Locator(URL)

http://<ODL_IP>:8181/index.html

The web page to be loaded asks for user name and password to Login where both are pre-defined to be "admin".

The Topology section visualizes network topology. It creates visual graph of the network, showing all the managed switches and routers and how they are connected together. Figure 3.1. is the view of the considered exemplar topology in ODL web. The figure shows two switches and four hosts. The Router acts as a node defined to serve both the subnets. Hence, it is depicted as a part of both the subnets.

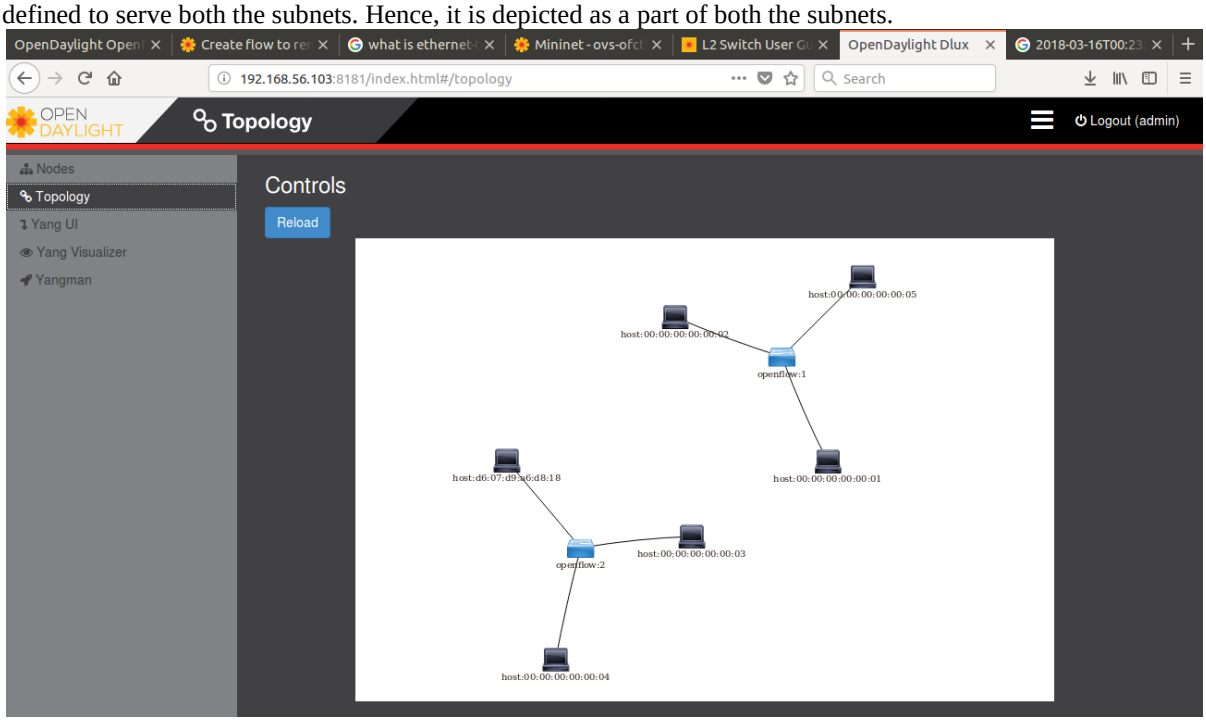

Figure 3.1. View of the Topology in ODL web

IV. Working with Postman

Postman is a Rest client and HTTP request composer. It provides web services and supports multiple requests. The implementation considered in this paper deals with Layer 2 rules associated to Matching of the hardware ethernet ports of mininet and ODL. The following are the steps to push the flow:

- Enter required headers in the "headers" space.

Content-Type: application/xml
Accept: application/xml
Authentication

- Type the xml code for matching of ethernet ports and required hosts in the "body" space. This requires entering the hardware ethernet addresses of Mininet and ODL in the respective slots of the XML code with ODL as destination and Mininet as source.
- Select "put" if the user wishes to push the flow on the topology.
- If you wish to verify with the pushed flow, you can view it by selecting "get" option.
- Enter the url and click on "send".
  http://<controller_ip>:8181/restconf/config/opendaylight/inventory:nodes/node/openflow:1/table/<table_number>/flow/<flow_number>
- With an error free code, the flow push success is indicated by *"200 OK"* or *"201 created"*.

The Figure 4.1. is the XML code for matching the ethernet ports and establishing the connection and transferring the packets between the desired hosts.

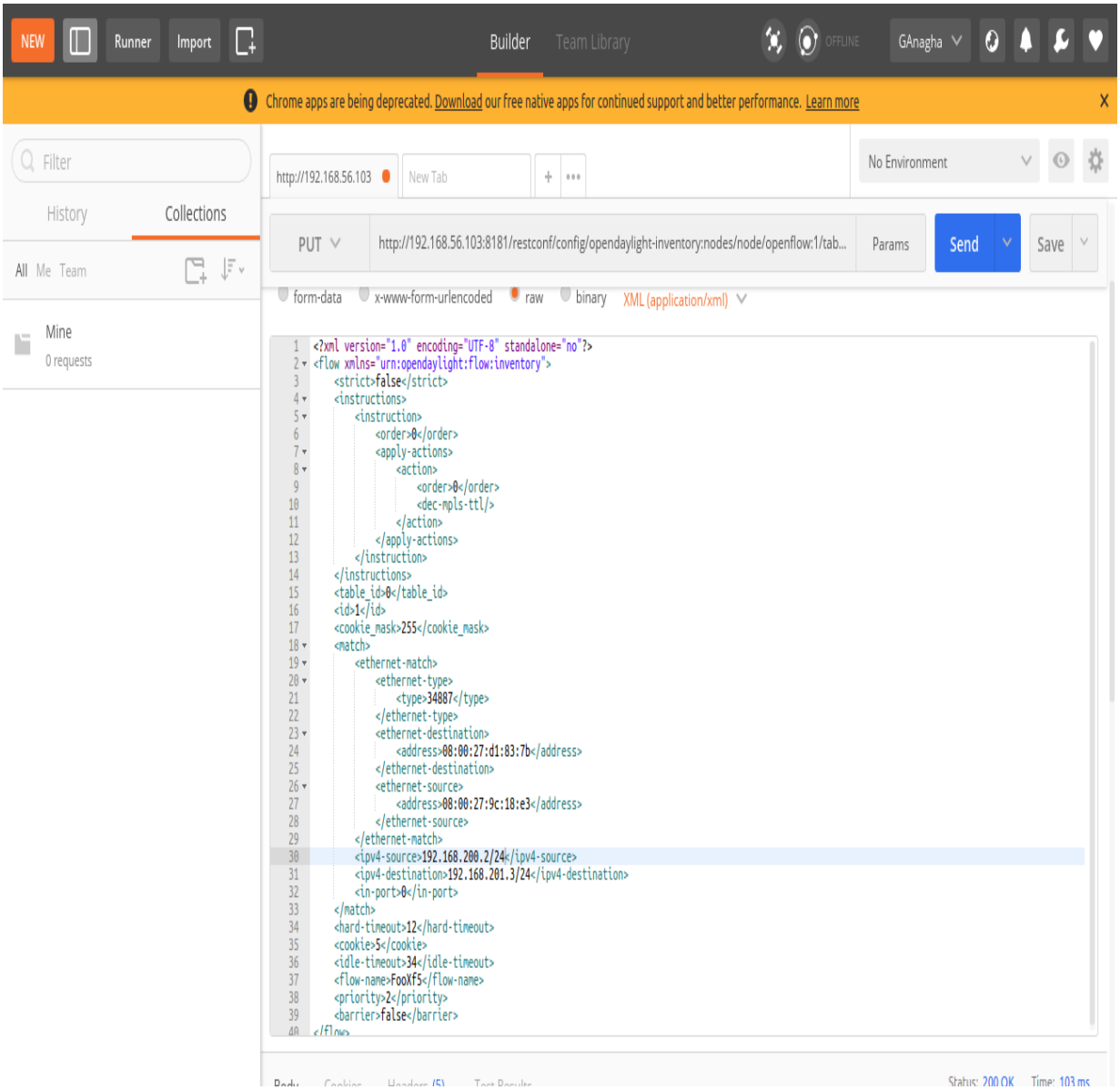

Figure 4.1. Executed XML Code for Matching the Ethernet Ports

V. Simulation Results

In this section, we analyse the success of the flow rule pushed and view the packet capture using Wireshark.

A. Test the success of flow pushed:

We use the command: ***$ shovs-ofctl –O OpenFlow10 dump-flows s1***Figure 5.1 shows the output describing the successof the flow rule pushed.


```
*** Adding switches:
s1 s2
*** Adding links:
(h1, s1) (h2, s1) (h3, s2) (h4, s2) (s1, r0) (s2, r0)
*** Configuring hosts
h1 h2 h3 h4 r0
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 r0
h2 -> h1 h3 h4 r0
h3 -> h1 h2 h4 r0
h4 -> h1 h2 h3 r0
r0 -> h1 h2 h3 h4
*** Results: 0% dropped (20/20 received)
mininet> sh ovs-ofctl -O OpenFlow10 dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x2b00000000000009, duration=10.157s, table=0, n_packets=10, n_bytes=868, idle_age=8, priority=2,in_port=3 actions=output:1,output:2,C
ONTROLLER:65535
 cookie=0x2b0000000000000a, duration=10.156s, table=0, n_packets=25, n_bytes=2282, idle_age=8, priority=2,in_port=1 actions=output:3,output:2,
CONTROLLER:65535
 cookie=0x2b0000000000000b, duration=10.154s, table=0, n_packets=11, n_bytes=910, idle_age=8, priority=2,in_port=2 actions=output:3,output:1,C
ONTROLLER:65535
 cookie=0x2b00000000000002, duration=14.149s, table=0, n_packets=0, n_bytes=0, idle_age=14, priority=100,dl_type=0x88cc actions=CONTROLLER:655
35
 cookie=0x2a000000000000e, duration=8.927s, table=0, n_packets=8, n_bytes=728, idle_timeout=600, hard_timeout=300, idle_age=3, priority=10,dl
_src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:02 actions=output:3
 cookie=0x2a00000000000012, duration=8.839s, table=0, n_packets=1, n_bytes=42, idle_timeout=600, hard_timeout=300, idle_age=3, priority=10,dl_
src=00:00:00:00:00:01,dl_dst=00:00:00:00:00:05 actions=output:1
 cookie=0x2a000000000000f, duration=8.927s, table=0, n_packets=8, n_bytes=672, idle_timeout=600, hard_timeout=300, idle_age=3, priority=10,dl
_src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:01 actions=output:2
 cookie=0x2a00000000000017, duration=8.828s, table=0, n_packets=1, n_bytes=42, idle_timeout=600, hard_timeout=300, idle_age=3, priority=10,dl_
src=00:00:00:00:00:05,dl_dst=00:00:00:00:00:02 actions=output:3
 cookie=0x2a00000000000016, duration=8.829s, table=0, n_packets=1, n_bytes=42, idle_timeout=600, hard_timeout=300, idle_age=3, priority=10,dl_
src=00:00:00:00:00:02,dl_dst=00:00:00:00:00:05 actions=output:1
 cookie=0x2a00000000000013, duration=8.838s, table=0, n_packets=1, n_bytes=42, idle_timeout=600, hard_timeout=300, idle_age=3, priority=10,dl_
src=00:00:00:00:00:05,dl_dst=00:00:00:00:00:01 actions=output:2
 cookie=0x2b00000000000002, duration=14.129s, table=0, n_packets=0, n_bytes=0, idle_age=14, priority=0 actions=drop
mininet>
```

Figure 5.1.Verifying the Pushed Flow

B. Verifying the packet capture:

While pinging two different hosts of the exemplar network, we can verify the connection establishment by using Wireshark which performs packet capture. This is illustrated in the Figure 5.2.

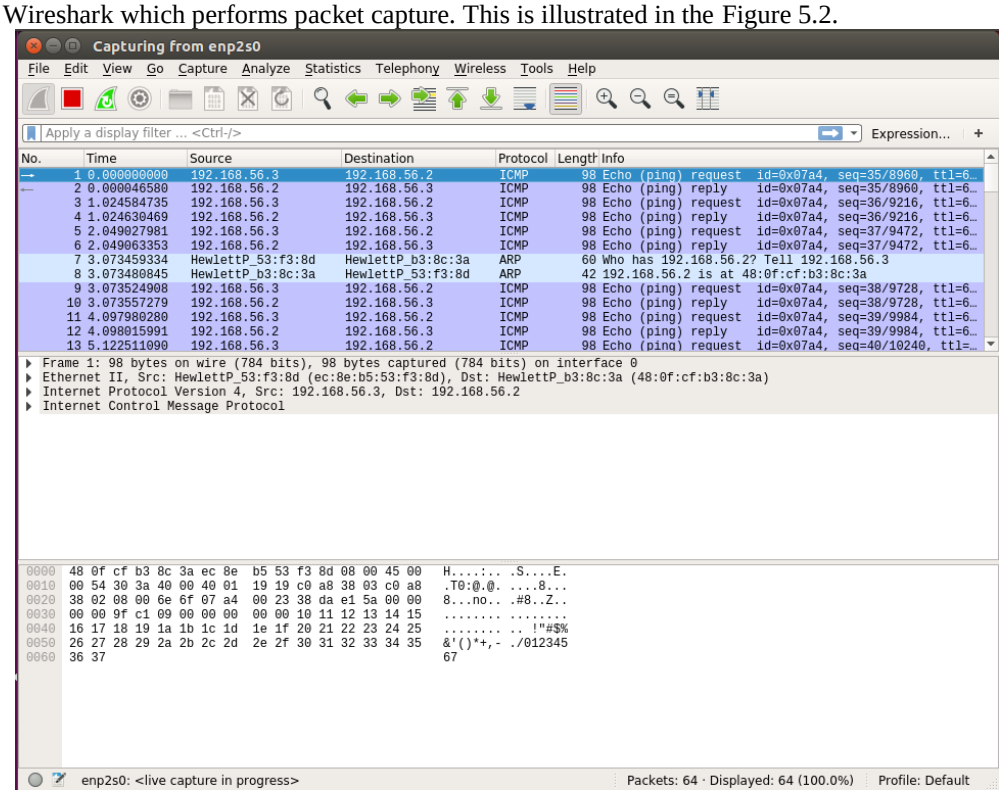

Figure 5.2 Packet Capture in Wireshark

VI. Conclusion and Future Work

The paper talks about exploiting the advantages offered by software defined networking. It makes use of network emulator mininet and a remote controller ODL to control entire network and manage topologies of various size and scale. Further, the aforementioned applications are matched using postman. The success of matching and the packet capture are verified with the aid of Wireshark.

This paper provides provision for the future work. There is enough scope for hardware realization. SDN employs control console to enable network manager to operate and modify working of network elements by allowing unconditional control of rues fed into the elements. SDN allows for entire control of network to allow quick response to change in network or business needs. It is possible to create a mininet network that resembles hardware network, or a hardware network that resembles a mininet network and to run the same binary code and application on either platform, thus enabling the realization of a programmable network and easy handling of network operation in real time [3].

References

[1]. Benzekkikamal et al.Software-defined networking (SDN): a survey.Security and Communication Networks 9, no. 18 (2016)

[2]. "Software-defined networking (SDN) Definition", Opennetworking.org. Retrieved 26, October 2014

[3]. Mininet, "Mininet version 2.2.1. http://mininet.org/overview/," 2.2.1 ed. 2015

[4]. OpenDaylight, Nitrogen release, "https://github.com/opendaylight/controller/releases"

[5]. VirtualBox specific Instructions "https://github.com/mininet/openflow-tutorial/wiki/VirtualBox-specific-Instructions".

