Mobile Computing Project Report

(WS 2021/2022)

Title of Project:

Group 6 - Networking Approach:SDN//Service:Text-based Chats

Guidance

Prof. Armin Lehmann

Submitted By

Aishin Abdulla Yoosufali (1359022)

Chhavi Sareen (1360105)

Jayshri Taywade (1343384)

Work Contribution

We are pleased to show our individual work contribution in this project.

Team Members	Task Performed
Aishin Abdulla Yoosufali	1. Network implementation of SDN 2. Access Network & Transport Network 3. Open Flow Protocol Network Implementation
Chhavi Sareen	1. Access Network 2. Documentation (Project Implementation) 3. Presentation
Jayshri Taywade	1. LXC Implementation 2. Documentation

Table of Contents

Abstract

Software-defined networks (SDN) are about the ability to control the behaviour of a network dynamically and programmatically through software applications. SDN is a fascinating technology that allows for new approaches to network design and management and is part of a long history of efforts to make computer networks more programmable, even though it appears to have emerged out of nowhere. This project focused on text-based communication between end-to-end clients using SDN technology.

The concept of SDN builds upon ideas from early efforts to make networks programmable. We focus on elements like robustness, scalability, performance, security, and dependability, as well as new prospects for carrier transport networks and cloud providers while designing switches and control platforms. Last but not least, we look at SDN's role as a crucial facilitator of a software-defined world.

Keywords: SDN, Container, Open vSwitch, OpenFlow, Open Daylight Controller,vxlan

1. Introduction

Software-Defined Network is a current trend in computer network research. On computer network devices, SDN offers a control-plane and data-plane separation paradigm. The control plane of a computer network device determines how the device responds to the flow/packet/frame that it receives. The data plane, on the other hand, is responsible for forwarding flow/packet/frames [1]. SDN is a new networking paradigm that promises to overcome current network infrastructure restrictions. First, it separates the network's control logic (the control plane) from the underlying routers and switches that forward traffic, thereby breaking vertical integration (the data plane). Second, because the control and data planes are separated, network switches become simple forwarding devices, and control logic is implemented in a logically centralised controller (or network operating system1), policy enforcement, and network (re)configuration and evolution are simplified. Figure 1 depicts a simplified representation of the architecture [2].

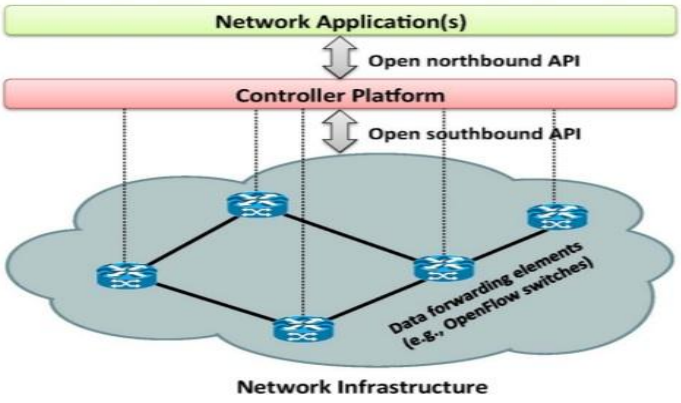

Fig 1: Simplified version of SDN architecture [2]

As networks evolve, so do their applications and the needs they serve. These objectives frequently impose a wide range of network infrastructure requirements in terms of latency, scalability, availability, dependability, security, and other factors. Fifth-generation (5G) networks are intended to

support a large number of new use cases, each with its own set of requirements. It is impossible to create a network that is one-size-fits-all and can support all use cases on a single architecture. For 5G networks, potential solutions for meeting these expectations are being developed.

While virtual networks are not a new concept, network slices have certain unique concepts that set them apart. Network slices, by definition, are end-to-end logical networks that run on top of shared network infrastructure, are segregated from one another, and are controlled and managed independently. Slicing of networks has been proposed as a key feature of next generation 5G mobile networks. As mentioned in a technical proposal by the open network foundation, the concept of network slicing is naturally supported by the SDN architecture (ONF) [3].

One of the most widely used software switches is Open vSwitch. From its inception, OpenvSwitch has been an OpenFlow-enabled switch, designed to be versatile and programmable. The most essential components of Open vSwitch are ovs-switchd, a user-space daemon, and a data path kernel module, both of which have operating-system-specific performance implementations. The data path kernel module receives, forwards, and alters packets according to ovsvswitchd commands. It's been ported to a variety of virtualization platforms and operating systems, and it's been used in a variety of commercial products and large-scale production scenarios. The ovsdb-server, a database server that stores the configuration state of the Open vSwitch instance, is used to setup Open vSwitch. The Open vSwitch Database (OVSDB) protocol can be used to connect to the ovsdb-server [3].

2. SDN Architecture

SDN architecture specifies how a networking and computing system can be developed utilizing a mix of open, software-based technologies and commodity networking hardware. The SDN architecture includes tools for slicing networks. Flow processing is performed utilizing the SDN architecture and OpenFlow via the IMS-based signaling platform in this architecture model. It not only fully utilizes the remote controller to dynamically manage network resources, but it also allows network slicing via the SDN architecture, which meets the diverse service requirements of mobile networks [4].

The application layer, northbound interface, control layer, southbound interface, and infrastructure layer are all part of the SDN architecture, as shown in Figure.

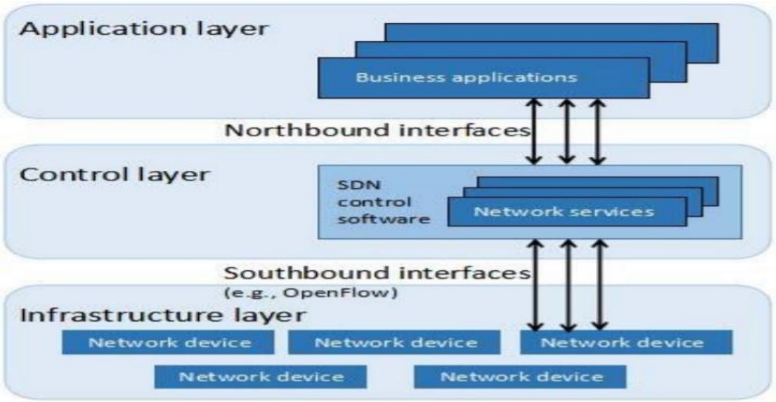

Fig 2: SDN Architecture Layer [4]

The network function applications are implemented by the network application layer. The northbound interface is used by these programs to configure, manage, and operate infrastructure layer devices. The northbound interface is a two-way communication channel between the controller and application layers.

The SDN Controller is the heart of the design, and it's responsible for giving upper-layer network applications programmable capabilities as well as unified infrastructure layer setup, management, and control. The controller layer's and infrastructure layer's interfaces are connected by the southbound interface. A high-speed data forwarding capability exists in the infrastructure layer [4].

3. Protocol

OpenFlow is a framework that was created at Stanford University and is now being developed as a standard [OpenFlow] by the Open Networking Foundation (ONF). OpenFlow is a protocol for controlling an OpenFlow switch from a logically centralized controller. OpenFlow is the most widely used SDN protocol. One or more flow tables are maintained by each OpenFlow-compliant switch and are used to perform packet lookups. Regarding packet lookup and forwarding, separate measures must be conducted. The switch standard also includes a group table and an OpenFlow route to external controllers [5].

3.1 OpenFlow Switch

Flow-tables and group-tables are used by Openflow Switch to match and forward packets. There are many flow entries in a flow table. Through an Openflow channel, an Openflow switch is connected to an Openflow controller. Using the Openflow protocol, the controller will manage the switch. The controller could update, add, and delete flow entries in the flow-table using the OpenFlow protocol [6].

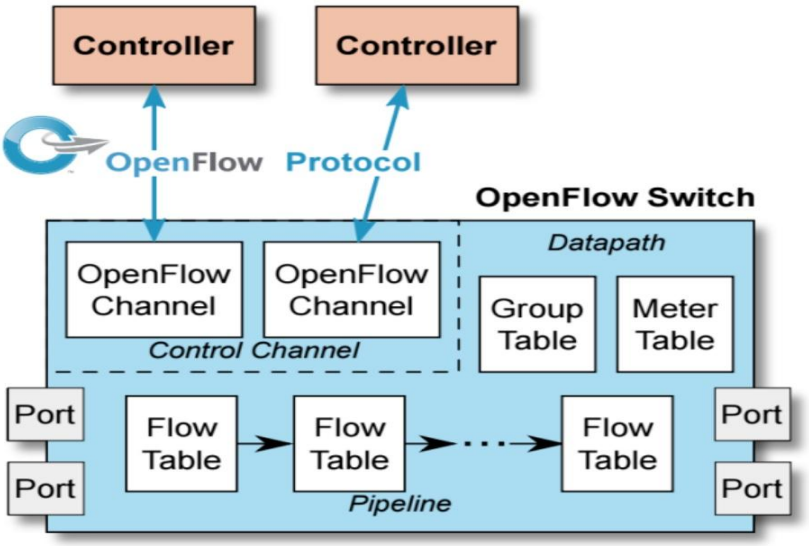

Fig 3: Main components of an OpenFlow Switch [7]

When the controller dynamically learns where devices are in the topology and needs to update flow tables on those devices to develop an end-to-end connection, reactive flow entries are formed. Because switches in a pure OpenFlow system are merely traffic forwarders, the controller must first dictate and encode all reasonable logic. If a host on switch A needs to communicate with a host on switch B, messages will be forwarded to the controller to determine how to reach this host. The controller will program the logic into each switch's flow tables after learning the switches' host MAC address tables and how they link. This is a flow entry that is reacting to something.

Before traffic comes, Proactive Flow Entries are programmed. The controller can program these flow entries on the OpenFlow endpoints ahead of time if it's previously known that two devices should or shouldn't connect [7].

The network interfaces for passing packets between OpenFlow processing and the rest of the network are called OpenFlow ports. OpenFlow switches communicate with one another logically through their OpenFlow ports. Physical ports, logical ports, and reserved ports are the three types of ports that must be supported by an OpenFlow switch [7].

4. Network slicing

Network slicing is a key technology for the impending 5G system, as it allows operators to serve numerous services with varying requirements across a shared infrastructure. Network slicing, in particular, is an essential element for 5G system deployments because it allows operators to organize network resources flexibly while providing a variety of services to subscribers and third-party clients.

SDN introduced the concept of network programmability, focusing on packet-based metro-core networks, by offering both a standard protocol for programming network devices (OpenFlow) and a common vision of network devices. These ideas make network slicing deployment a lot easier by bringing flexibility and dynamicity to the data plane architecture [8].

Control and forwarding planes are decoupled in SDN, and control is moved to a centralized controller. This controller implements the network operating system and provides a library of APIs that can be utilized to make networking application development easier. Two distinct network slicing methodologies might be considered while using SDN. In the first approach, each slice provides a traditional network to end-users (e.g., imposing certain SLAs), whereas in the second approach, each slice provides an SDN network to end-users, i.e., a network that can be configured via an SDN controller [8].

4.1 Network Virtualization

Network virtualization (NV) uses containers to divide functions while sharing hardware resources, resulting in lower costs, faster network service launch, and increased network efficiency. This virtual network is independent of real hardware and can be used for actions such as creation and deletion. The control plane and data plane of the switches are separated by Software-Defined Networking (SDN). SDN uses protocols like OpenFlow (OF) to control data-plane SDN switches from afar [9].

It's also designed to construct numerous virtual networks on top of a shared physical network, each of which may be built and controlled separately. Network virtualization is thought to be the answer to the network rigidity problem for the next-generation Internet, allowing for network variety and dynamicity. Network virtualization, which may transport numerous separate virtual networks above the same physical network, has shown to be a remarkable method to designing the architecture of the future Internet [10].

4.2 Tunneling mechanism used: vxlan

Virtual eXtensible Local Area Network (Vxlan), which is a critical tunneling protocol in the data center, is the most visible concern for Open vSwitch [11].

Fig 4:Vxlan protocol [11]

Vxlan is a three-layer protocol encapsulation method that combines two-layer packets. The Vxlan protocol is extensively utilized in cloud computing data centers since it is more in line with the multi-tenant data center business. One of the most important technologies for implementing overlay in a data center is the tunnel protocol. Vxlan is a tunneling protocol that operates at the layer 3 level. Vxlan is more versatile and compatible with contemporary networks than other tunnel protocols, such as GRE. Vxlan is a one-to-many tunneling protocol, whereas GRE is a one-to-one tunneling protocol. Vxlan is a type of vlan that is similar to vlan [11] .

5. Project Description

Objective:- The SDN approach used to network virtualization and containerization helps to optimise network resources and adapt networks quickly to changing applications and traffic. It creates a software-programmable infrastructure by separating the network's control and data planes.The OpenFlow Switch environment built on the foundation of SDN must be involved. It supports a wide range of protocol plugins as well as a wide range of services and applications.The implemented network must include multiple hosts and data centres that run on lxc containers, as well as physical network devices that support the OpenFlow protocol. Host machines must function as text-based chat clients, and the entire data centre must serve as a text-based chat server. Using the tunnelling mechanism with vxLan, each slice must communicate independently via chatting.

5.1 Approach Followed

To implement this project, the approach has been described as follows. Firstly, an OpenFlow Switches has been developed to communicate within SDN in a northbound interface which then enables it to communicate with Clients using OpenFlow version protocol. Then the tunnelling mechanism using vxLan has been used. In this approach Four Open vSwitch (OVS) is used to build the core network and two other switches used for providing access to the data from one Client to another. The given figure: 5 Topology for implementing this approach.

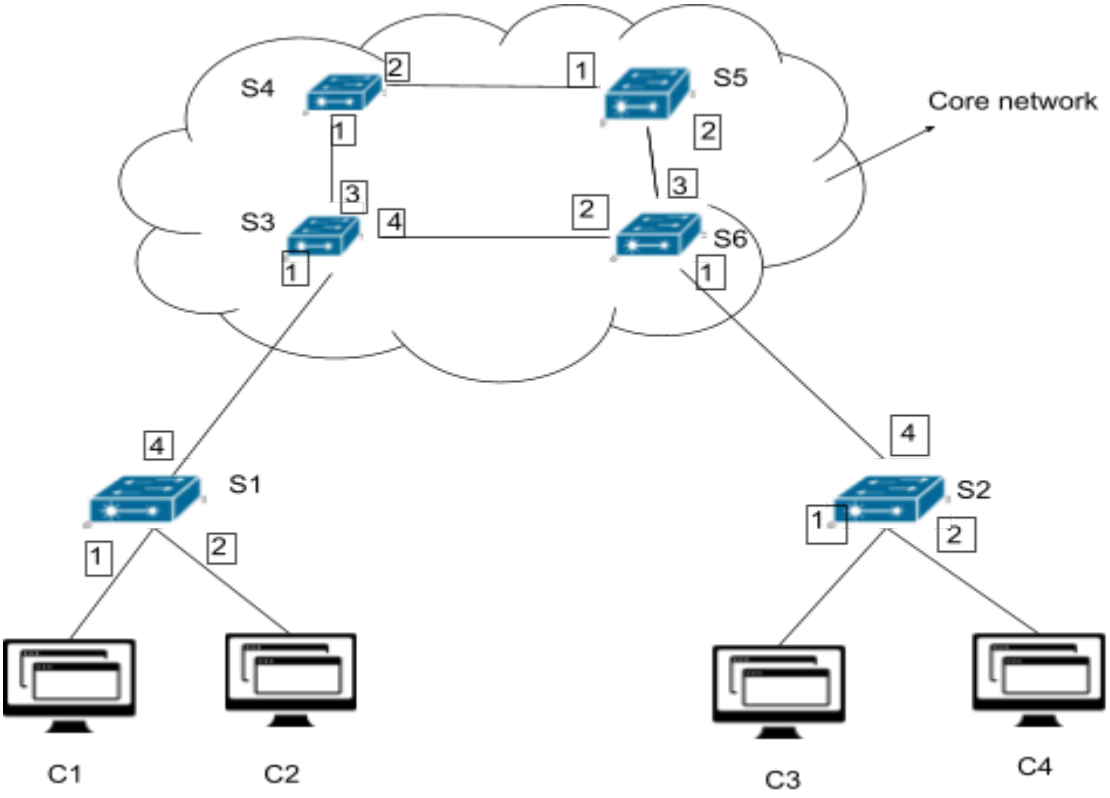

Fig 5:Topology for Implementing Approach

6. Project Implementation

To implement this project first a Containernet hosted system machine that is Linux Ubuntu version 18.04 LTS is considered. In a VmWare workstation Ubuntu iso file has been installed.Then the Lxd has been installed using the command in it.After this the initialization of containers has been done.Ovs switch installed in virtual machine and connected to two clients(c1,c2)with switch1 and the same has been done for the other virtual machine where two clients (c3,c4)are connected to switch2. This access network is connected to the core network.

Transport network permits and constraints the movement or flow.It allows the maintenance of operations & ensure the delivery of needed services.We have created the network using 4 openvswitch that has OpenFlow protocols enabled and the open flow controller can control the switches by looking into flow tables in each switch.This is also a concept of software defined networking where we are giving static flow entries to each of the switches in the core network.

To connect these switches we have used vxlan tunnelling mechanism.As you can see every node of Ovs is connected via vxLan tunnels.So far if we want to send a packet from Client1 as Client 3 as destination,the packet will reach to switch1and from switch1 it will forward the ICMP packet to transport network.And in switch3 it will check the flow table entry to where to send this packet next.This is what open flow controller is doing here.

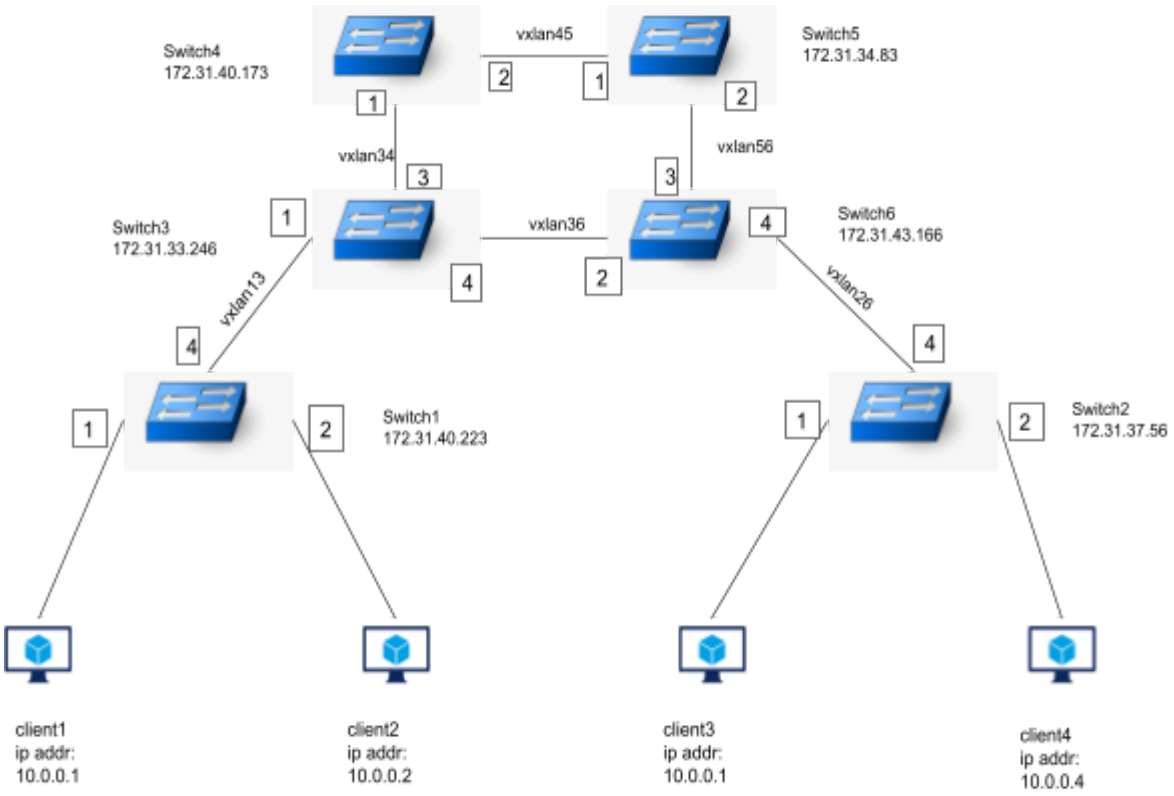

Fig 6: SDN Architecture using Open vSwitches, OpenFlow Protocol & Tunneling

Access Network A:

Client 1 is connected via eth1 port and client2 is connected via eth1 port to switch1

```
root@ip-172-31-40-223:/home/ubuntu# lxc ls
+---------+---------+----------------------+-----------------------------------------------+------------+-----------+
|  NAME   |  STATE  |         IPV4         |                     IPV6                      |    TYPE    | SNAPSHOTS |
+---------+---------+----------------------+-----------------------------------------------+------------+-----------+
| client1 | RUNNING | 10.252.160.180 (eth0) | fd42:c884:8182:7019:216:3eff:fe53:2503 (eth0) | PERSISTENT | 0         |
|         |         | 10.0.0.1 (eth1)      |                                               |            |           |
+---------+---------+----------------------+-----------------------------------------------+------------+-----------+
| client2 | RUNNING | 10.252.160.205 (eth0) | fd42:c884:8182:7019:216:3eff:fe6a:e4d6 (eth0) | PERSISTENT | 0         |
|         |         | 10.0.0.2 (eth1)      |                                               |            |           |
+---------+---------+----------------------+-----------------------------------------------+------------+-----------+
```

Access Network B:

Client 3 is connected via eth1 & client 4 is connected via eth1 port to switch ports

```
root@ip-172-31-37-56:/home/ubuntu# lxc ls
+---------+---------+----------------------+-----------------------------------------------+------------+-----------+
|  NAME   |  STATE  |         IPV4         |                     IPV6                      |    TYPE    | SNAPSHOTS |
+---------+---------+----------------------+-----------------------------------------------+------------+-----------+
| client3 | RUNNING | 10.126.162.188 (eth0) | fd42:e842:19e0:9bb4:216:3eff:feb4:add8 (eth0) | PERSISTENT | 0         |
|         |         | 10.0.0.3 (eth1)      |                                               |            |           |
+---------+---------+----------------------+-----------------------------------------------+------------+-----------+
| client4 | RUNNING | 10.126.162.67 (eth0)  | fd42:e842:19e0:9bb4:216:3eff:fe21:4a87 (eth0) | PERSISTENT | 0         |
|         |         | 10.0.0.4 (eth1)      |                                               |            |           |
+---------+---------+----------------------+-----------------------------------------------+------------+-----------+
```

Four Open vSwitches are used to create the transport network, and in all switches OpenFlow is enabled.Every switches have a flow rules that will be implemented when we do a packet transmission from client1 to client3 or we can say network slicing.We have set the rule from switch1 and switch2 ports to drop the transfer of packet to client and client4 ports. For the interconnection of switches, we have used VxLan tunnelling mechanism.

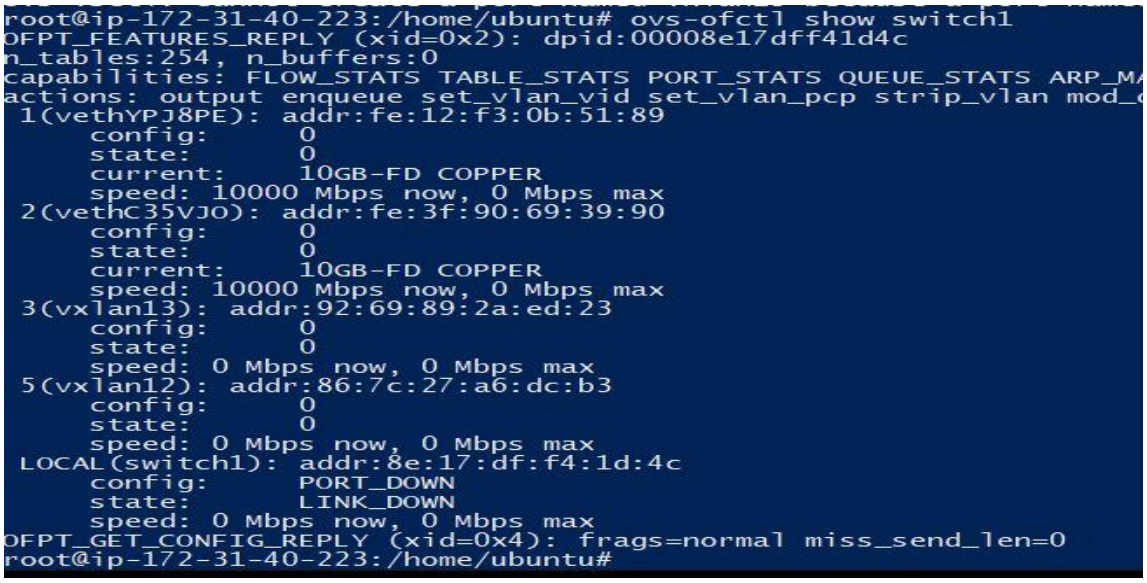

Fig 7: Configuration of OpenFlow in Switch1

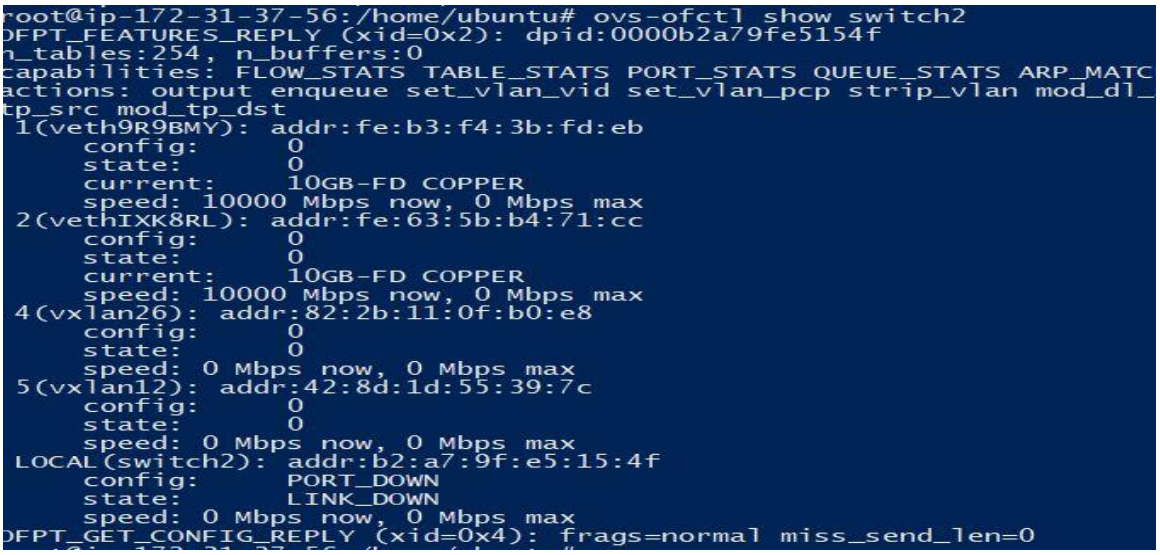

Fig 8: Configuration of OpenFlow in Switch2

```
root@ip-172-31-33-246:/home/ubuntu# ovs-ofctl show switch3
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000eaa052152a42
n_tables:254, n_buffers:0
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: output enqueue set_vlan_vid set_vlan_pcp strip_vlan mod_dl_src m
tp_src mod_tp_dst
 1(vxlan13): addr:96:82:c2:0f:dc:86
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 2(vethfa593145): addr:b6:c7:cd:38:d5:59
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 3(vxlan34): addr:22:92:32:d0:5a:90
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 4(vxlan36): addr:62:aa:93:7f:ef:4e
     config:     0
     state:      0
     speed: 0 Mbps now, 0 Mbps max
 LOCAL(switch3): addr:ea:a0:52:15:2a:42
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

Fig 9: Configuration of OpenFlow in Switch3

```
root@ip-172-31-40-223:/home/ubuntu# ovs-ofctl dump-flows switch1
 cookie=0x0, duration=52.154s, table=0, n_packets=0, n_bytes=0, in_port=vethYPJ8PE actions=output:vxlan13
 cookie=0x0, duration=5.375s, table=0, n_packets=0, n_bytes=0, priority=500,in_port=vethYPJ8PE actions=output:4
 cookie=0x0, duration=19964.255s, table=0, n_packets=106195408, n_bytes=8388905872, priority=0 actions=NORMAL
```

Fig 10:Flow Table os Switch1

```
root@ip-172-31-37-56:/home/ubuntu# ovs-ofctl dump-flows switch2
 cookie=0x0, duration=80.200s, table=0, n_packets=81347, n_bytes=4662830, priority=500,in_port=veth9R9BMY actions=output:vethIXK8RL
 cookie=0x0, duration=48.212s, table=0, n_packets=129835, n_bytes=9817422, priority=500,in_port=vxlan26 actions=output:veth9R9BMY
 cookie=0x0, duration=10172.432s, table=0, n_packets=59103053, n_bytes=4604590474, priority=0 actions=NORMAL
```

Fig 11:Flow Table os Switch2

References

[1] F. Baskoro, R. Hidayat and S. B. Wibowo, "Comparing LACP Implementation between Ryu and Opendaylight SDN Controller," 2019 11th International Conference on Information Technology and Electrical Engineering (ICITEE), Pattaya, Thailand, 2019, pp. 1-4, doi: 10.1109/ICITEED.2019.8929986.

[2] Diego Kreutz et al. “Software-Defined Networking: A Comprehensive Survey”. In: Proceedings of the IEEE 103.1 (2015), pp. 14–76. issn: 0018-9219. doi: 10.1109/JPROC.2014.2371999.

[3]Thomas Langenskiöld, “Network Slicing using Switch Virtualization”, Master’s thesis, School of Electrical Engineering, 19th November 2017, Available at: https://aaltodoc.aalto.fi/bitstream/handle/123456789/29159/master_Langenski%C3%B6ld_Thomas_2017.pdf ?isAllowed=y&sequence=1

[4] "A SDN Controller Enabled Architecture for the IMS", Zeqi Liu;Qichao Wang;Jae-Oh Lee 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS) Year: 2019 | Conference Paper | Publisher: IEEE

[5] "Software-Defined Networking (SDN): Layers and Architecture Terminology", Internet Research Task Force (IRTF) E. Haleplidis, Ed. Request for Comments: 7426 University of Patras Category: Informational K. Pentikousis, Ed. ISSN: 2070-1721 EICT January 2015

[6] F. Baskoro, R. Hidayat and S. B. Wibowo, "LACP Experiment using Multiple Flow Table in Ryu SDN Controller," 2019 2nd International Conference on Applied Information Technology and Innovation (ICAITI), Denpasar, Indonesia, 2019, pp. 51-55, doi: 10.1109/ICAITI48442.2019.8982149

[7] https://overlaid.net/2017/02/15/openflow-basic-concepts-and-theory/

[8] "Network Slicing in SDN", Networks D. Scano*, L. Valcarenghi*, K. Kondepu*, P. Castoldi*, and A. Giorgetti Scuola Superiore Sant’Anna, Via G. Moruzzi 1, 56124 Pisa, Italy Tel: +39 050 88 2168 e-mail: alessio.giorgetti@santannapisa.it

[9] N. M. M. K. Chowdhury, R. Boutaba. “A survey of network virtualization,” Computer Networks, vol. 54, issue 5, pp. 862-876, 2010.

[10] "Network virtualization by using software-defined networking controller based Docker", Liu Xingtao, Guo Yantao, Wu Wei Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory Shijiazhuang China lxtmay@163.com

[11] "Open vSwitch Vxlan Performance Acceleration in Cloud Computing", Data Center Published in: 2016 5th International Conference on Computer Science and Network Technology (ICCSNT) Date of Conference: 10-11 Dec. 2016 Date Added to IEEE Xplore: 19 October 2017 ISBN Information: INSPEC Accession Number: 17262569 DOI: 10.1109/ICCSNT.2016.8070222 Publisher: IEEE Conference Location: Changchun, China