Designing SDN supported Seamless MPLS by using Open Flow Switch

Student Name Elias Nigatu

A Thesis Submitted to

Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment for the Degree of Master of Science in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

June,2022

Adama, Ethiopia

Designing SDN supported Seamless MPLS by using Open Flow Switch

Student Name Elias Nigatu

Major Advisor: Dr-Ing. Ferezewed Lema

A Thesis Submitted to

Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment for the Degree of Master of Science in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

June, 2022

Adama, Ethiopia

Declaration

I hereby declare that this Master Thesis entitled “Designing SDN supported Seamless MPLS Framework by using Open Flow Switch” is my own work and has not been submitted to any university for similar purpose. All sources of materials that are used for this thesis have been duly acknowledged through citation

Elias Nigatu		
Name of Student	Signature	Date

Recommendation of Advisors

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled “Designing SDN supported Seamless MPLS by using Open Flow Switch” prepared under my guidance by Elias Nigatu submitted in partial fulfillment of the requirements for the degree of Mater’s of Science in Department of Computer Science and Engineering Therefore, I recommend the submission of revised version of the thesis to the department following the applicable procedures.

Dr-Ing. Ferezewed Lema

Major Advisor	Signature	Date
Co-Advisor	Signature	Date

Approval Page

I, the advisors of the thesis entitled "Designing SDN supported Seamless MPLS by using Open Flow Switch" By Elias Nigatu, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Dr-Ing. Ferezewed Lema

Major Advisor	Signature	Date
Co-advisor	Signature	Date

We, the undersigned, members of the Board of Examiners of the thesis by Elias Nigau have read and evaluated the thesis entitled "Designing SDN supported Seamless MPLS by using Open Flow Switch" and examined the candidate during open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in Computer Science and Engineering.

Chairperson	Signature	Date
Reviewer 1	Signature	Date
Reviewer 2	Signature	Date

Finally, approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

Department Head	Signature	Date
School Dean	Signature	Date
Office of Postgraduate Studies, Dean	Signature	Date

Acknowledgement

I give my praise to the Almighty God, who made everything possible. Next, I would like to express my appreciation and great thanks to Dr-Ing Ferezewed Lema for his endless effort in advising, checking and providing knowledgeable, constructive and valuable comments by the quality of his professional experience Finally, I would like to express my Thanks to my Families for their supports.

Table of Contents

List of tables

List of Figures

List of abbreviations/acronyms

AGG: Aggregation Gateway

API: Application program interface

ARP: Address resolution protocol

AS: Autonomous system

ATM : Asynchronous transfer mode

BGP: Boarder Gateway Protocol

CDPI: Control Data-Plane Interface

CoS: Cost of services

CSG: Cell Site Gateway

CSP: Constrained Shortest Path

DCLC: Designed Delay Constrained Least Cost

DDoS: Distributed Denial of of Service

DiffServ: Differentiated Services

DSLAM: Digital Subscriber Line Access Multiplexer

E2E: End to end

eBGB: External BGP

FEC: Forward Equivalence Class

iBGP: internal BGP

ICMP: Internet Control Message Protocol

IntServ: Integrated Services

IP: Internet protocol

IS-IS:IntermediateSystem- Intermediate System

LARAC: Lagrange Relaxation based Aggregate Cost

LER: Label Edge router

LSP: Label Switch Path

LSP: Label switch path

MCP: Multi Constrained Path E2E

MSAG: Multi-service access getway

MSAN: Multi- service access node

NBI: Northbound Interfaces

ONF: Open Network Foundation

OSPF : Open Shortest Path First

OTWG : Optical Transport Working Group

OVS: Open vSwitch

QoE: Quality of Experience

QoS : Quality of Service

SDN: Software Defined Networking

SLA: Service level agreement

S-MPLS: Seamless Multi-protocol level switching

SNMP: Simple Network Management Protocol

TE:traffic engineering

VPN: Virtual privet network

Abstract

The huge traffic demand of graphics and also video applications has led to a high demand for end-to-end QoS guarantees. Many telecom service providers, use seamless MPLS-based solutions to construct a multi-service bearer network. Seamless MPLS was developing in the need for superior flexibility in scenarios where old MPLS over confuses network management. It offers an alternative for implementing end-to end MPLS networks by integrating different domains into a single MPLS domain. However, current seamless multi-protocol label switching application is a hard configuration procedure and does not offer a central view of the network topology to telecom service operator.

Employing Seamless MPLS without SDN leads to several challenges like timewasting, complicate system segmentation, difficulty of learning to manage such a huge systems and devices and more. This research paper proposes SDNs supported seamless MPLS stream of traffic to be managed in the framework of as in one-piece network rather from unified but locally controlled devices. It focuses the network intelligence in software-based central controllers, which aims to bring better and more efficient control, advance quality of service and quality of experience in networks, permitting network engineers to have a central view and control of the network topology. In general, it providing seamless multi-protocol label switching services in a simplistic approach, will make it to achieve adaptive routing and traffic engineering.

In this study, Experimental synthesis and prototyping system development approach is used to achieve the research objective. Mininet software and Wireshark packet analyzer are emulate and using many different scenarios in order to evaluate the connectivity and performance of SDN based seamless MPLS compare to seamless MPLS without SDN. Consider the functionality of seamless MPLS network in SDN framework and also the performance of SDN based seamless MPLS is evaluated quantitatively.

Keywords: MPLS, seamless MPLS (S-MPLS), Software Defined Networking (SDN), Open flow

CHAPTER ONE

1. Introduction

1.1. Background of the Study

The growth of wireline and wireless technologies, multimedia applications, voice, video and data, in the world has led to the increase of traffic generated on the internet. The rapid growing diversity of huge traffic demand of graphics and also video applications has been making domination in access networks and generate a massive demand for speedy internet services. One method to safeguard that these applications deliver the acceptable handler experience today is the presence of high-speed links for back haul connections or even high speed links at the access layer of the network by (Tristian, R., Muntean, G.-M., & Katrinis, K, 2013).

The escalation in traffic growth the network operators look the essential to advance and setup innovative efficient and economical solutions, characterized by all-IP Ethernet back haul and large increase size in mobile backhaul then core parts. For instance, according Peter (Croy ,2011) the backhaul size requisite for Africa wireless cell site will rise gradually and near at around 75-90Mbps. As to the core switch needs for some mobile service provides evaluation more than three-times increase of their core bandwidth desires, for example from under 40Gbps to 130Gbps.

A research conducted by (García-Dorado, J. L ,2012) on multimedia bandwidth demand states that video traffic is on a steady rise. In the next few years, it is assessed that 2/3 of traffic on the internet will be video traffic. Portion of that will account for video applications and other real time heavy applications. The growth of bandwidth demand also leads to significant transformation in the back haul requirements and migration to all-IP Ethernet backhaul. Routing serves established on IP well on the Internet in the mid90s, but IP technology was able to be fruitless at forwarding packets since software must search for routes using the longest match algorithm. As an importance, the forwarding capability of IP technology can act as a bottleneck.

In contrast, Asynchronous transfer mode technology practices labels of fixed length and maintains a label table that is much smaller than a routing table. Compare to IP, ATM is more well-organized at forwarding packets. (Asynchronous transfer) ATM is a challenging protocol, however, with high deployment costs, that hold back its widespread use. Since outdated IP technology is simple and costs little to deploy, a combination of IP and ATM capabilities would be ideal. It has established the development of MPLS technology. MPLS was prepared to increase forwarding rates. Different IP routing and forwarding, MPLS examines a packet header only on the edge of the network and not at each hop. MPLS therefore decreases packet processing time.

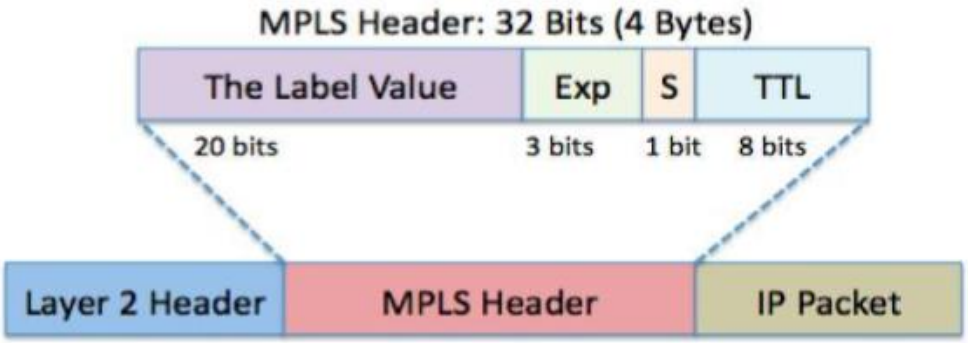


Figure 1: MPLS header

The usage of hardware-based tasks centered on application-specific unified circuits has made IP routing far more efficient, so MPLS is no longer required for its high-speed forwarding uses. Beside, seamless MPLS provisions multilayer labels, and its forwarding plane is connection-oriented. For these reasons, MPLS is largely used for virtual private network (VPN), traffic engineering (TE), and quality of service (QoS). Seamless MPLS protocol is applicable on Internet Protocol (IP) backbone networks. By linking routing technologies and switching technologies, seamless MPLS leverages the elasticity of IP routing and the easiness of switching.

Currently, the only service provider Ethiotelecom implemented three layered topologies: Backbone Layer, Core Layer and Edge Layer.

Backbone Layer: totally 10 sets of BR spread to five towns, with each town set up two sets of BR, full-meshed connection. This layer is mainly liable for service traffic forwarding and international traffic.

Core Layer: totally 32 sets of CR used in 14 cities. ER tools of this network are linked to local CR equipment.

Edge Layer:

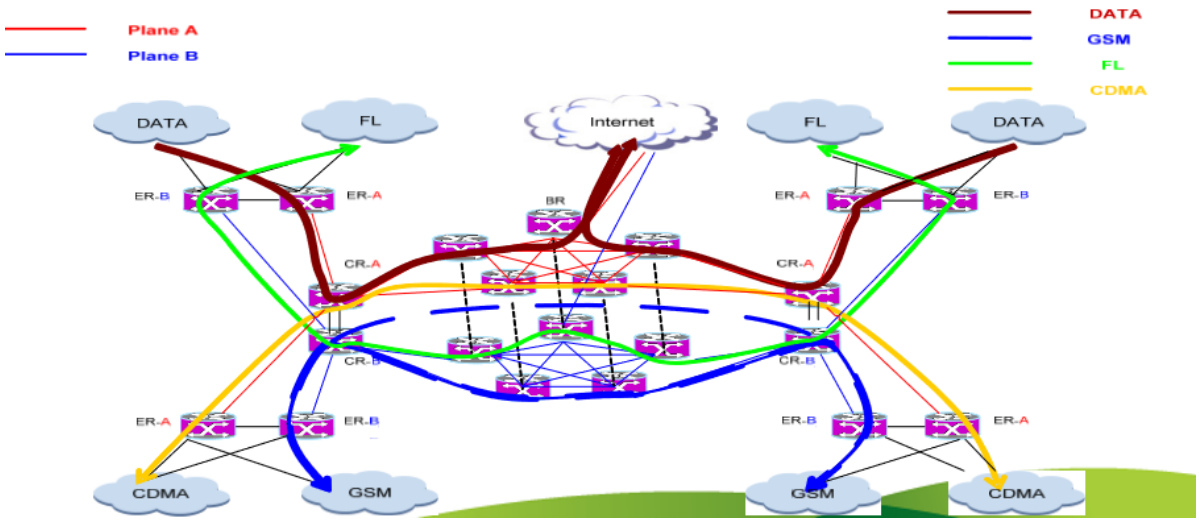


Figure 2:Logical Architecture of ethio telecom Network

For access network, numbers of access devices are required and sites are dispersed largely. Static line signaling service, media service and broadband services become into to access to many small sites in the all of Ethiopia. Those services hosts are not at the same locations with data devices of IP network. Grounded on this, it is the key to exercise aggregation switch first to aggregate the accesses devices at individually site; then combined the aggregation switch to the location where IP network devices are existing through transmit network.

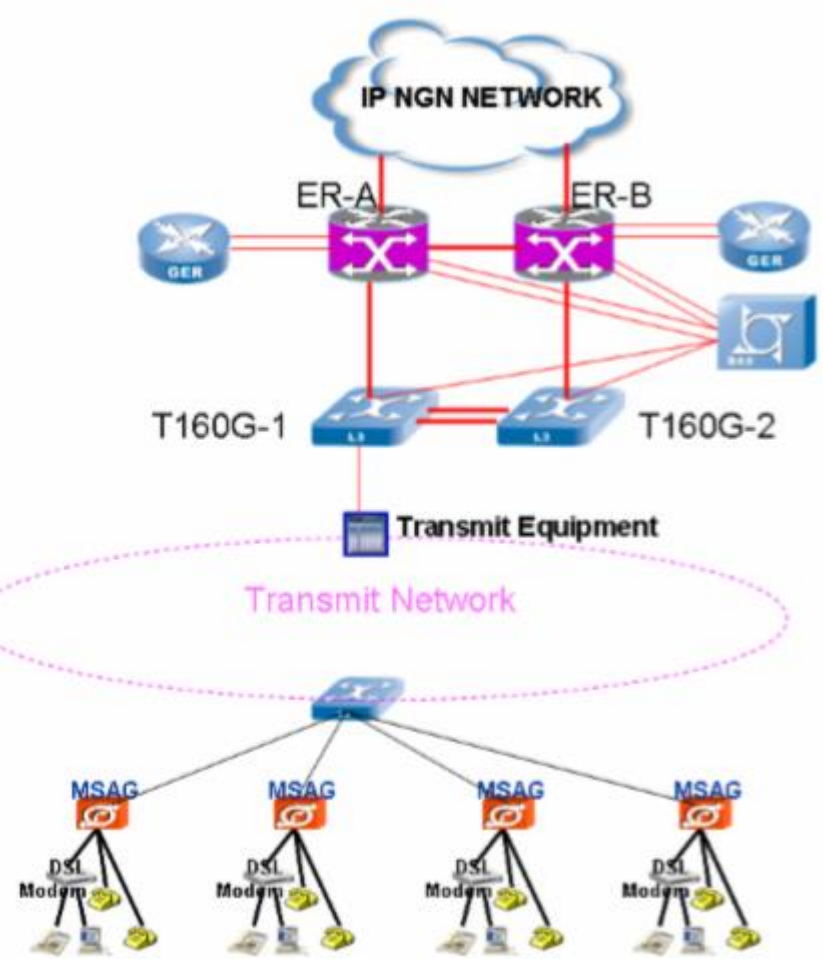


Figure 3:ethio telecom fixed network service deployment

MPLS network distributions delivers limited flexibility in provisioning, as it is strongly attached with the topological location of the network nodes and operationally so, people has to arrangement with various technologies to troubleshooting and fault recovery.

Seamless MPLS arrangements a superior alternate for applying end-to-end MPLS networks system by incorporating access, aggregation and core networks into one MPLS network. It is also aimed for optimizing and improving the efficiency of end-to-end MPLS networks. With S-MPLS architecture, the entire network uses unified IP/MPLS networking technology, with an end-to-end control plane by (Habtamu Kumela, 2018). It established a Border Gateway Protocol (BGP) LSP through the access, aggregation, core layers to practice point to point service transmission. LSP can be established between any two nodes on a seamless MPLS network to transmit service packets upstream and downstream. This network structural design

ensures high service scalability, while only needing to deploy signaling protocols at service access points.

An autonomous system(AS) is group of networks in a common routing rule which is managed by a single expert. Switching info inside an autonomous system is known as intra-domain routing. Instead, inter-domain routing emphases on the interchange of routes to permit the transmission of packets among different autonomous system. When an autonomous system(AS) is connected to multiple different autonomous system, it is stated that as a multi-homed autonomous system. Instead, autonomous system (AS) linked to a single autonomous system are recognized as single-homed autonomous system.

Border Gateway Protocol (BGP) are the Effective standard for inter-domain (inter-AS) routing protocol used to exchange reachability information throughout the Internet. When reachability info is switched between two border Gateway Protocol (BGP) routers placed in different autonomous system, the protocol is denoted as external border Gateway Protocol (eBGP). On the other hand, when reachability info is switched between BGP routers located at the same AS, the protocol is stated to as internal border Gateway Protocol (iBGP).

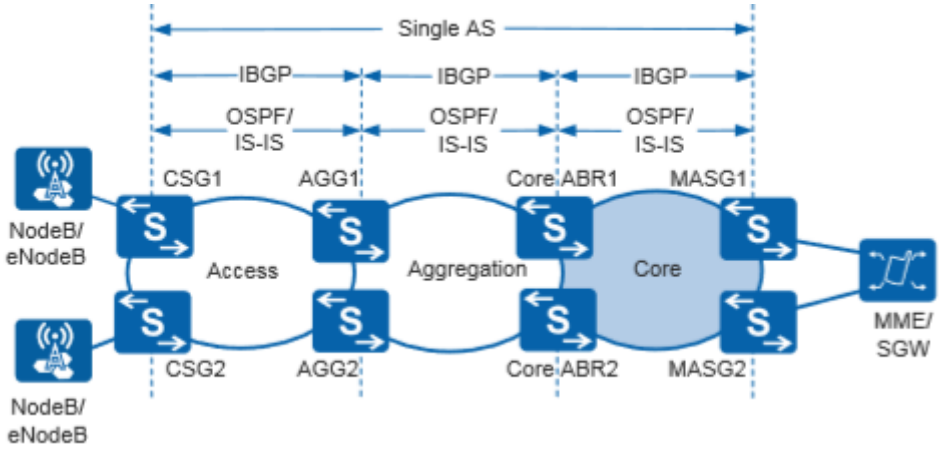


Figure 4:routing protocols for the intra-AS seamless MPLS

- Helps simplicity communication between the access, aggregation, and core layers.
- Make simpler network provisioning, operations, and maintenance because the technology encapsulates all services and transmits these services along an E2E LSP.
- Delivers high flexibility and scalability.

The proposed solution centered on SDN. Therefore, success a highly efficient network that encounters the requirements of all the users is preferred. In addition, the simplicity and manageability of this network will highly transform data transportation in networking. This study notes that the techniques stated above to offer QoS on packetized networks shaped the proposed solution and highly depended on them.

SDN can practice to changes in its environment in a method to rise the efficiency in terms of service delivery and the maximum use of the resources. This network should also be easy to configure, manage and monitor. It contributes understanding on whether the networks can meet QoS demands and Quality of Experience (QoE) demands. To achieve the paper derives a portion from the act of traditional routing in networks and tradition QoS provisioning mechanisms.

SDNs are a newly networking architecture that is proposed as a facilitating technology for network evolution and network virtualization. It has concerned important attention from both academic scholars and industry. The strategic institutions that give to the development of SDN is the Open Network Foundation (ONF) not profitable business association of network operators, service suppliers and vendors that helps the SDN architecture and motivations the standardization process, (White paper ,2012).

SDN as a technology that control plane is separate from forwarding plane and is directly programmable". It focuses the network intelligence in software-based central controllers, which aims to bring better and more efficient control, customizability and adaptability. As outcome, the networking devices are rotated into packet forwarding equipment that can be programmed via an open interface.

The main benefits that the SDN technology might offer are listed in (White paper ,2012) and summarized below: -

- Centralized unified control of network devices from different vendors
- Better automation and control, as an abstraction of the real network is created
- Easy and faster implementation of innovations, for instance the network control is centralized and there is no need every individual device to be reconfigured
- Improved network reliability and security, because of fewer configuration errors and unified policy enforcement, provided by the automated management and the centralized control
- Ability to easily adapt the network operator to changing user needs, as centralized network.

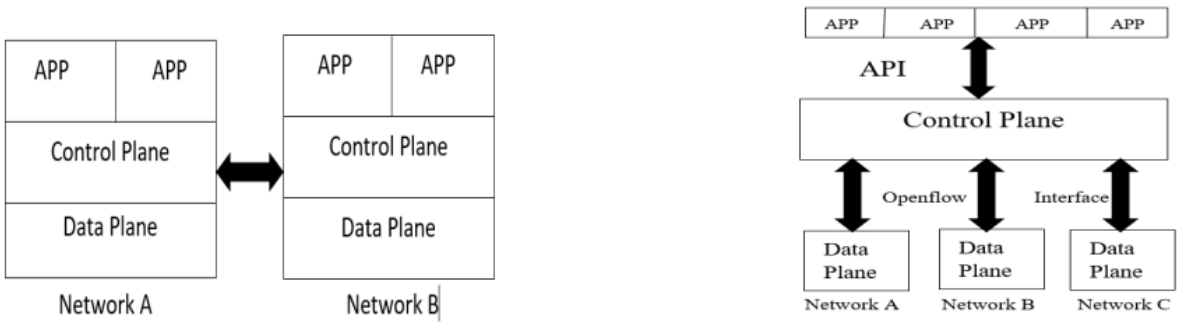


Figure 5:Traditional IP networks and Software-Defined Network

1.2. Statement of the Problem

Current seamless MPLS carrying out a complex structure technique and it does not deliver a central controlling of the network topology to network engineers. The complexity of seamless MPLS deployment comes from the non-centralized view of the network nodes by the service providers. Particularly if it is being taken in a wide area network (WAN) implementation. The networking devices does not administrator centrally means, for all x device, we have a total of x control planes need to be configured which is not enable telecom service operator to provide flexibility, programmability, and adaptability of services.

Most telecom service provider companies have been implemented seamless MPLS network without centralize control management system. Implementing seamless MPLS without controlling centrally is not only unable to provide adequate performance, less security and

cost-effectiveness for service provider but also complicate network segmentation and inconvenience and incompetent of learning to manage such a huge systems and devices rise on seamless MPLS.

Nowadays, there are many smart applications that requires much higher bandwidth to facilitate robust communication, but the existing decentralize seamless MPLS networks does not meet the requests of such applications. Hence implementing SDN supported seamless MPLS was announced to achieve the desires of such applications, unlike the old-style way of doing seamless MPLS on vendor specific devices, SDN offers the centralized view of the network because it links multiple forwarding devices to one or more control planes but, not on a one to one relationship between forwarding devices to control planes.

1.3. Research Questions

- How does seamless MPLS run in current internetworking schemes?
- How does measure and evaluate Seamless MPLS architecture with SDN?
- How SDN system can be applied in seamless MPLS?
- How does the SDN based seamless MPLS differ from not SDN based?

1.4. Objective

1.4.1. General Objective

The general objective is to design and develop SDN supported seamless MPLS network environment.

1.4.2. Specific Objectives

- To assess the impact of SDN on Seamless MPLS network.
- To measure and evaluate Seamless MPLS architecture with SDN.
- To simulate SDN based Seamless MPLS architecture.
- To compare SDN supported Seamless MPLS architecture with non SDN supported seamless MPLS architecture.

1.5. Significance of the study

Seamless MPLS is a key technology in every service providers and its optimum implementation is an important factor. More than a few service suppliers, including Ethio

telecom, have deployed mobile backhaul in addition to IP/MPLS core networks. The interconnection as well as service provisioning among these different domains should be done seamlessly without introducing additional delay and flexibility problems.

The research has also illustrated that hardware components should no longer be a problem to network administrators and engineers whenever they want to come up with very complex and custom network applications because SDN has broken the glass ceiling and leveled the playing to allow for creativity and innovation. We are being able to come up with a switching procedure, this method will push and pop level based pre-defined flow rules mechanism implemented by SDN supported seamless MPLS routing. This is a guarantee that network will be right flexible and robust, adapting and changing to suit the design of the network engineer.

1.6. Expected outcome of the study

This study is helpful in redesigning how network administrators and engineers will approach the design of transport data networks with key importance put on seamless MPLS implementation. Network administrators and engineers are able to take a global end-to-end centralized control of the network architecture, in addition, they will not be concerned whether their device of select is seamless MPLS enabled or not. The research will demonstrate how easiest it will be for network engineers to setup seamless MPLS forwarding mechanism in their network topology.

This will be influenced by the power of SDN and open Flow in providing a centralized and easy configuration mechanism of networking devices, this procedure will give paths to flows based on the fast packet delivery mechanism implemented by seamless MPLS routing, beside, the network will be very flexible and healthy, adapting and changing to suit the design of the network engineer.

1.7. Delimitation Scope

This study will focus demonstrating how a routing procedure based on seamless MPLS can be achieved in SDN. It will select internet control massage protocol (ICMP) for testing the functionality of proposed system for both packet request and packet replies from different hosts in the network and it will path and manage the flow of data that move toward from various

sources into the network. The study used an SDN based simulation tool called Mininet working with open Flow protocol because of the acceptability and wide interoperability of the two technologies as far as programmable networks are concerned.

To implement the research's objective stumbled upon challenges that affected the functionality of the system. Among those limitation understanding SDN, since it is an emerging topic in computer networking was a complex task and conflicting version of software components will be major challenge.

1.8. Thesis Layout

This paper is structured in the following manner, Chapter one deals with background the study, statement of the problem, Research Questions, objectives of the study, Significance of the study, expected outcome of the study, delimitation of the study and summery of chapter. Chapter Two provides technical background research relevant to seamless MPLS and SDN networks in current interconnecting strategies. Chapter Three describes the methodology and simulation tool that used in the research. Chapter Four presents the results and discussion of the data collected. Chapter Five describes the conclusions and future work.

1.9. Summary

In a short in this section, we have discussed the background of the research, problem of statements, raised research questions, objectives of the study, significance of the study, estimated outcome of the study and delimitation scope, in the next chapter, we discuss literature review and related works. We also identify the gaps in different papers on the related topic.

CHAPTER TWO

2.Literature Review and Related work

2.1. Introduction

The chapter deeply expressions at the previous work in the field of MPLS and seamless MPLS implementation and routing in SDN. This chapter starts by discussing Traffic Engineering, MPLS and seamless MPLS operation in current internetworking strategies. It expressions at the environment to support network customization in SDN, then proceeds to review quality of services Finally, to conclude, the chapter looks at the possible strategies of implementing a label like switching technique in SDN emulating what happens in seamless MPLS but making it compatible with open Flow.

2.2. Literature Review

2.2.1. Traffic Engineering (TE)

However, In the old IP networks, routing protocols are implemented to deal Layer 3 routing information. routing protocol, packet forwarding is established on the destination address alone. Therefore, when a packet is established by the router, it governs the next-hop address using the packet's destination IP address along with the information from its own forwarding/routing table. This procedure of determining the next hop is repetitive at each hop (router) from the source to the destination, by (Foster, N., Guha, A., 2013).

Traffic Engineering (TE) in multi-protocol level switching as captured from the discussion by (Z. Li, K. Lu, 2017) can be provided by MPLS being able to create tunnels to the exact target. These tunnels can be created based on analysis of the network traffic. The services of TE can further be better to provide load balancing on an MPLS network as lectured by (Z. Li, K. Lu, 2017).

Advantage of with labels and not the destination IP address is that packet forwarding choice can be made on the other factors such as traffic engineering and QoS requirements. The skill to traffic engineer based on real-time network attributes supports strict SLAs with guaranteed

service availability. One of the objectives of Seamless MPLS is to encompass traffic engineering capability end-to-end through the access network.

2.2.2. Traditional IP Networks

In traditional IP networks, routing protocols are implemented to give Layer 3 routing info. Nevertheless, of the routing protocol, packet forwarding is addressed on the destination address alone. Thus, once a packet is established by the router, it decides the next-hop address expending the packet's end point IP address along with the information from its own forwarding/routing table. This procedure of determining the next hop is repetitive at each hop (router) from the source to the destination *by* (Foster, N., Guha, A., 2013*).*

Traditional routing algorithms tend to consider only one metric when making forwarding decisions. To provide QoS, an adaptive routing algorithm must consider more than one metric, shortest path, as witnessed in traditional routing algorithms. This becomes very important when dealing with multimedia applications because it is not always a guarantee that shortest path will always meet delay constraints a video application requires on a network.

(Zhao, J., Hammad, E., Farraj, A & Kundur, D, 2016) Discusses, Constrained Shortest Path (CSP) and Multi Constrained Path (MCP) routing algorithms have been developed to consider more than one metric when defining paths for different flows in a network topology. (Wang, K., & Hsu, Y.-H. 2015). looked at the shortcomings of MCP and CSP and designed Delay Constrained Least Cost (DCLC) with the help of Lagrange Relaxation based Aggregate Cost (LARAC).

The goal of implementing seamless MPLS Label switching in SDN is to make sure there is very minimal complexity at the networks. The normal mode of operation of the core network switches are based on rule matching. According to (Ahn, G., & Chun, W. 2000) this de facto operation is very memory intensive and makes it costly to adapt. Moreover, the normal SDN operation couples up the host requirements that are fed to the control layer with the network core behavior propagated down to the forwarding devices. Such minor inflexibility when one wants to achieve less memory intensive and simple core layer network operations motivated the work of implement MPLS kind label switching on open flow.

MPLS is a kind of label swapping standard that uses labels that are attached to the packet header as they traverse the network. The work of these labels is to identify a corresponding Forward Equivalence Class (FEC) that encapsulates the packets and forwards it in the same direction regardless of their destination addresses. This approach assures network engineers of the flexibility of the network because they can determine which paths are being taken by traffic in the network.

2.2.3. Quality of Service (QoS)

As it was confirmed QoS is configured at the edge of the MPLS network and allows high import packets to be forwarded favorably. QoS and CoS can be providing in MPLS networks, where traffic engineering is arranged, and give the chance of SLA agreements with the customers to be recognized. Two important mechanisms can be distinguished:

- Integrated Services (IntServ) – this mechanism deals guaranteed traffic parameters end-to-end. The QoS desires are signaled across the network, using RSVP and uses hard allocation of resources.
- Differentiate Services (DiffServ) – this a fewer strict mechanism that delivers CoS by categorizing traffic into different importance level, but does not offer end-to-end guarantees of the service. The DiffServ data is encompassed in the IP packet headers and is mapped in to the label information.

QoS and TE critically progress the network performance and are between the fundamental ideas that turn IP/MPLS into the key enabling technology for provided that converged telecommunication services over a single network infrastructure.

2.2.4. MPLS operation

The old IP routing lookups, as (Ould-Brahim, H. 2012, February) spreads this discourse, were performed on each router. Which means that to each router will make an independent decision when forwarding the packets. MPLS comes into play to reduce the number of routing lookups and eliminate the need to run a particular routing protocol and all devices involved in the forwarding of MPLS data. It was aimed to resolve the problems that were being experienced in IP routing. (Black, U. D. 2002), discusses that routing protocols in traditional IP networks were designed to be invoked on all the devices that were to be used for forwarding

functionality. Besides that, regardless of the routing protocol that is used, these routers could only forward traffic based on the destination addresses only.

MPLS is protocol independent. MPLS uses labels, that are numbers, to identify the packets and enable forwarding decisions to be made based on those numbers. These labels are 32 bit long and are placed between the Layer 2 (L2) and Layer 3 (L3) headers. The labels as explained by (Le Faucheur, F. (1998) usually correspond to layer 3 destination addresses or other QoS parameters.

Network that goes MPLS usually encompasses of two categories of devices. Label Edge Routers (LERs) are devices placed at the entry points and the exit points of the MPLS network. LERs are devices that state user traffic to an MPLS network and then forwards the same user data to non MPLS networks. In other terminologies, these devices are referred to as the Ingress and the Egress routers respectively.

Explains by (Black, U. D. 2002) that the edge routers are frequently responsible for the route look up activity and allocating of labels to packets. The other group devices in a network that runs MPLS is the LSR that are responsible for swapping different labels and forwarding the packets based on the received labels. The path that the data will use at the entry point of an MPLS network to the exit point is called the Label Switch Path (LSP) (Le Faucheur, F, 1998).

The MPLS architecture comprises of two main components, the control plane and the data plane. The control plane is responsible for the exchange of L3 routing information and the labels and creating the end to end path for the traffic to follow. The data plane on the other hand is responsible of forwarding data. It simply acts as a forwarding engine. Originated on the functionality of MPLS, other services can be derived from its operation.

MPLS- Virtual Private Network (VPN) is another application that has rode on the capabilities of label switching. This MPLS-VPN as (explained Lee, H., Hwang, J., Kang, B., & Jun, K. 2000) adds an additional label to determine the VPN and the destination network. VPN routing information and the labels are then propagated to the MPLS domain with the assistance of Boarder Gateway Protocol (BGP). The two labels that are used to achieve MPLS-VPN are the

top label that points to the egress router and a second label, which points at the exit interface on the egress router.

As presented *in* (White paper, 2000452-001-EN, Jan. 2012) in old IP/MPLS network design and deployments, there is a fitted coupling between the network nodes and the services delivered over it and the services are provisioned in multiple segments.

The study helps for cover seamless MPLS outside the core to the aggregation and metro area network to gain some of the same benefits that MPLS offers in the core. Take MPLS to the access, and creating MPLS the packet forwarding platform end to end through the network, needs new functionality and features and a systematic architecture that can measure multiple of nodes.

2.2.5. Seamless MPLS operation

The essential for one united packet network to deliver all fixed and mobile services regardless of the access technology, advances from time to time. The achievement of MPLS in core networks and the paybacks, it brings have enabled the way for the technology to be employed in aggregation and access networks as a substitute to ATM or legacy Ethernet-based aggregation. Now the mobile backhaul service has been installed widely, the necessity of the integration of mobile backhaul networks and core networks has been suggested (Z. Li, K. Lu, "Inter-SDN 2017).

Deploying a service from one MPLS region to another desires provisioning at several in-between points in the end-to-end network, performing trouble shooting and fault recovery more difficult. A favorite approach would be to arrange single end-to-end service and transport network architecture by (White paper, SR1610000967EN, Oct. 2016).

Seamless MPLS offers the framework for attractive MPLS end-to-end in a scalable manner, spreading the paybacks of traffic engineering and guaranteed service-level agreements (SLAs) with deterministic network resiliency. Unlike from MPLS, seamless MPLS all forwarding packets encircled by a network is based on MPLS labels, from the time a packet enters until it leaves the network, it is to mean that label adding and removing will began in the access layer of the system and it will continue adding labels until the packet arrives its destination.

The inspiration of Seamless MPLS is to deliver an architecture which supports a wide variety of different services on an on its own MPLS platform entirely assimilating access, aggregation and core networks by the adding of extra features with traditional MPLS and it gives more scalability, safekeeping, easiness, manageability and flexible end-to-end service delivery. In order to find a highly scalable architecture seamless MPLS takes into account distinctive access devices such as Digital Subscriber Line Access Multiplexer (DSLAM) and Multi-service access node (MSAN) are present some advanced MPLS features, and may have more scalability restrictions (N. Leymann,2014).

Seamless MPLS does not a new protocol earlier it was necessary to offer connectivity between the different domains and the core on per service level and not centered on MPLS (e.g. by arranging native IP Routing or Ethernet based technologies between aggregation and core). At most cases service specific configurations on the border nodes between core and aggregation were prerequisite. New services managed to additional configurations and deviations in the provisioning tools.

With Seamless MPLS there are no technology borders and no topology borders for the services. Network (or region) borders are for scaling and manageability, and don't disturb the service layer, since the transport pseudo wire (layer 2 VPN) that transports packets from the access node to the service node doesn't care whether it takes two hops or twenty.

The network design is about network scaling, network flexibility and network manageability; the service design is about optimal distribution: service scaling, service flexibility (via replicated service nodes) and service manageability. The two are decoupled: each can be managed separately and changed independently. In the following subclasses key characteristics offered by Seamless MPLS are: -

- Deterministic End-to-end Service Restoration
- Decoupled Network and Service Architectures
- Service Flexibility with Simplified Provisioning and Operations
- Traffic Engineering
- Building Scalable Networks

2.2.6. Software Defined Networking.

SDNs are a new networking architecture that is suggested as a simplifying technology for network evolution and network virtualization. It has concerned important attention from both academic researchers and industry. Organizations that contribute to the growth of SDN is the Open Network Foundation(ONF) that is a non-profitable industry association of network operators, service providers and vendors that sponsors the SDN design and initiatives the standardization procedure of its major elements.

SDN as a technology where "network controller is decoupled from forwarding and is directly programmable". It focusses the network intelligence in software-based central controllers, which goals to bring improved and more efficient control, customizability and adaptability. As outcome, the networking devices are rotated into packet forwarding tools that able to be programmed via an open interface. The main paybacks that the SDN technology might offer are listed in summarized below:

- Unified control of network devices from varies operator
- Well computerization and controller, as an abstraction of the real network is created
- Easy and faster implementation of innovations, as the network control is centralized and there is no need every distinct device to be reconfigured
- Better network reliability and security, because of less configuration errors and shared policy enforcement, providing by the automated management and the centralized control
- Facility to easily familiarize the network operation to changing user needs, as centralized network state information is available and can be exploited SDN has been designed to simplify network configuration and facilitate innovation.

2.2.7. SDN Architecture

The architecture of a Software-Defined network be able to allocated into three planes, Data Plane, Control Plane and Application Plane. Below Figure shows the elements of a Software-Defined Network.

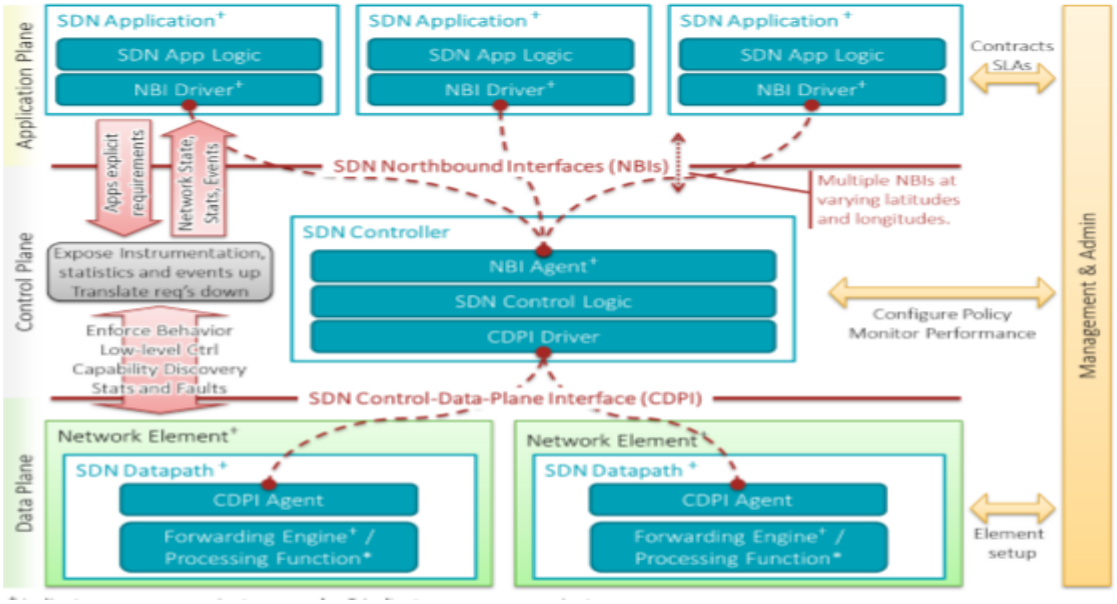


Figure 6:SDN architecture(Fernandez Sánchez, 2015)

- **SDN Applications:** The SDN architecture is demanded to greatly simplify network management and deliver a huge number of new services through the programmable software modules. A summary of the application state that will benefit from using the open Flow architecture as briefly summarized by (Mendonca, 2014).
- **Enterprise networks:** The centralized control function of SDN be able to mainly useful for enterprise networks in different methods. For example, network complication can be reduced by eliminating middle boxes and configuring their functionality inside the network control. Different network tasks employed via SDN include NAT, firewalls, load balancers and network access control.
- **Data centers:** Power intake management is a major issue in data centers, as they frequently control below capacity in order to be capable to meet peak demands. A network power manager is defined that turns off a subset of switches in a way to minimize power intake while confirming the required traffic conditions. An actual life example of SDN application in the context of data centers is presented. They describe SDN-based network concerning Google data centers worldwide. The arrangement was motivated by the essential of customized routing and traffic engineering, as well as scalability, fault tolerance and control that might not be accomplished with traditional WAN networks.

- **Infrastructure-based wireless access networks:** SDN result for enterprise wireless LAN networks is proposed in (Suresh, L.; Schulz-Zander 2012), The solution forms an abstraction of the access point infrastructure that divides the association state from the physical access point. The determination is to ensure proactive mobility controlling and load balancing.
- **Optical Networks:** Consulting to the Optical Transport Working Group (OTWG), amount of the Open Network Foundation (ONF), SDN solutions for optical transport networks can deliver multiple advantages, such as refining optical transport network control and management flexibility, allowing deployment of third-party management and control systems and arranging new services by leveraging virtualization and SDN. (*OTWG 2009).*

➢ **SDN Controller** The controller is the central of an SDN network. It lies amongst network devices at one end and applications at the other end. Any communications among applications and devices must pass through the controller by (Tristian, R., Muntean, G.-M., & Katrinis, K, 2013).

SDN controllers are established on protocols, such as Open Flow to configure network devices and select the optimum network path for application traffic and to permit servers to connect switches where to direct packets as presented in Figure 7.

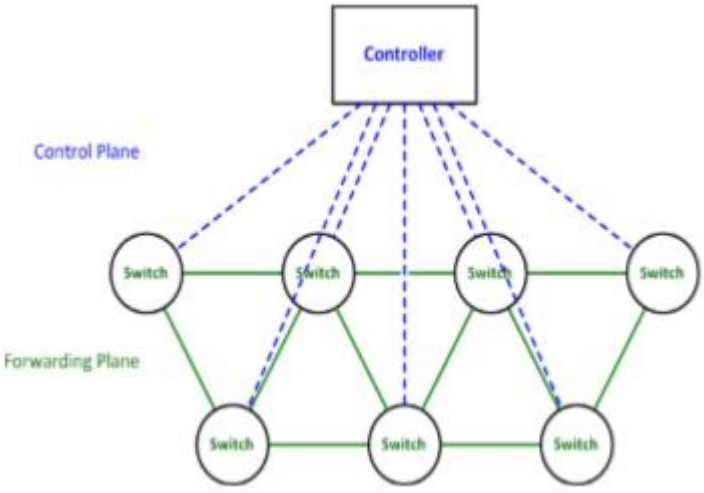


Figure 7:SDN controller

- **SDN Data Path:** The SDN Data path is a logical network scheme that contacts visibility and open control above its promoted forwarding and data handling capabilities by (Foundation, O. N2012*).*
- **SDN Control to Data-Plane Interface (CDPI):** The SDN CDPI is the interface express between Controller and data path, that transports at least
    - Programmatic control of fully forwarding operations
    - Capabilities advertisement
    - Statistics reporting
    - Event notification.
- **SDN Northbound Interfaces (NBI): SDN** NBIs are lines between Applications and Controllers and characteristically offer abstract network views and allow direct expression of network performance and requirements by (Foundation, O. N2012).
- **SDN Southbound Interfaces (SBI)** In the design of software-defined network, Southbound APIs (application program interface) that is used for communicate among the controller and the SDN network switches and routers.

2.2.8. Open Flow protocol

Open Flows are a uniform protocol that defines the communication between the control and the data forwarding plane in the SDN architecture. It exchanges the control out of the networking devices (routers, switches, etc.) into the centralized controller. The protocol customs the idea of flows that practice match rules to govern how the packets will be handled. The protocol is configured on both sides – the device and the control. The forwarding device in an Open Flow situation is an open Flow switches which covers single or extra flow tables and an abstraction layer that communicates with the SDN controller. The flow tables are occupied with flow entries which describe how the packet will be forwarded, dependent on the particular flow they are part of. (Mendonca ,2014) The flow entries have the following fields:

- *Match fields* – might encompass information from the packet headers, ingress port or
- Metadata and competitions the packets to a certain flow
- *Counters* – gather statistic about the particular flow

- *Actions* – describe how the incoming packets to be handled

An Open Flow switches essentially receives data packets, removes the packet header and matches the value to the entries in the flow table. If the value is establishing the packet is forwarded according to the instructions in the *actions* fields. In case the value does not equal any of the entries, the packet is handled according to the instructions demarcated in the *table-miss* entry. The packet be able to either dropped, forwarded to the next flow table or send to the Open Flow controller via the control channel. Another option, active in switches that have both open flow and non-Open flow ports, is to forward the packet spending standard IP-forwarding schemes. The open flow switch links with the controller over a safe channel. The controller increases, eliminates or updates the entries in the flow table (White paper 2013).

SDN rides on its ability to separate of abstract the control plane from the data plane, the defacto standard that has allowed the sharing of messages between the controller and the forwarding devices is open flow. This open standard for communication explains operates by allowing the controller to communicate with the multiple switches from a central point.

According to (B Pfaff 2015), Open flow allows SDN to achieve the abstraction of the underlying layer without worrying about the hardware. This means that application developers are be able to concern commands, via and API, to the forwarding devices in a standard format without worrying the kind of hardware that is underneath to them.

Open flow is responsible for issuing these flow entries, without changing the configurations of a forwarding device, in a flow table which the forwarding devices call the Forwarding information base. It allows the feeding of information to the OVS using their existing flow tables. It is imperative to note that each of the OVS has separate flow tables and open flow can interact with each of them by either adding, updating or deleting flow entries. It allows symmetric message exchanges to occur between the switch and the controller. These control messages include; Hello message, which is an introductory message that is first sent when a switch and controller connect for the first time or they are trying to keep-alive. Echo message is sent to check for the accessibility of the switch or the controller. Experimenter message is the default message that open flow switches use to offer any other value added services to the open flow message type space.

2.2.9. Open vSwitch overview and architecture

Open vSwitch (OVS) is a free Source Apace 2 licensed multilayer virtual switch that has opened the forwarding function to programming and extension control. OVS uses the virtual bridges or interfaces defined by the network topology designed in Mininet to add flow rules and forward packets accordingly, the behavior of the OVS is like a physical switch but only emulated in a virtual machine with virtual interfaces and virtual links discussed by (Pfaff, B. Pettit, J., Koponen, T., Jackson, E. J., Zhou, A., Rajahalme, J. 2015). OVS once invoked can run in two modes. The de facto mode is the standalone mode that allows OVS act as a learning switch. However, from the findings of this thesis, the topology of the network can make it harder for OVS to perform self-learning capabilities. Therefore, this pushes OVS to run in secure mode that relies on the controller to receive flow table.

2.2.10. Analysis of Mininet environment

The Mininet setting briefly is a simulation tool that permits creating an SDN based network by (Open Flow Tutorial on Mininet, 2018). It holds up to the groundbreaking advancements that SDN has accomplished by breaking up the control plane and data plane in communication forwarding devices.

A number of controllers frequently implements the control plane on Mininet. NOX a free source development platform that is based on C++ to control software defined networks and POX a variant of NOX, but purely runs on Python development platform for SDN networks. Floodlight controller is another SDN controller that is based on Java development platform (Gupta, M., Sommers, J., & Barford, P, 2013). This thesis chose to settle on POX controller for the implementation.

2.3. Related Work

This section reviews the different kinds of literature that are recent and more related to the objective of the study. In the following table 2.4 section, we see recent and more related past papers to our work.

Table 1:Summary of related works

Title	Author	Year of Publication	Strength of the Study	Limitation of the Study
Performance Analysis of Traditional Networks with Software Defined Networks (SDN) over Various Topologies Using Mininet	Asad Ali Memon and Nafeesa Bohra	2020	The idea is to analyze the performance of traditional and SDN networks over different topologies and integrate the SDN network	There is nothing discussion about seamless MPLS and SDN supported seamless MPLS network.
Multi-Protocol Label Switching Assisted Routing Procedure in Software Defined Networking (SDN)	Otieno, Humphrey Owuor	2018	demonstrating how a routing procedure based on MPLS can be achieved in SDN.	It discussed about MPLS supported by SDN, but not discussed about Seamless MPLS supported by SDN since seamless MPLS differ from MPLS. Beside, has gaps not measures the result of simulation in quantifiable.

Analyzing Impact of Seamless MPLS on QoS	Habtamu Kumera	2018	analyze the impact of an end-to-end MPLS architecture for enhancing service providers' IP network scalability and flexibility in service delivery using QoS parameters in comparison with the classical MPLS architecture.	It is briefly explain the advantage of seamless MPLS over MPLS but, not discuss about centralized controlling system (SDN)
Performance Evaluation of Software Defined Networking compare to Traditional Networks	Mohammed Khalil Abdalla Elmedani	2017	Software Defined Networking (SDN) is rapidly becoming the new buzzword in the networking business. Expectations are that this emerging technology will play an important role in overcoming the limitations associated with traditional networking.	Not discus about MPLS and seamless MPLS, it compares only the performance of Traditional and SDN network

Survey on Software-Defined Networking	Wenfeng Xia and Yonggang Wen, Senior Member, IEEE, publish the paper	2015	The paper defines the idea of SDN and emphasized benefits of SDN in contribution greater configuration, better performance, and refreshed innovation.	Only discuss about SDN network, nothing about MPLS and seamless MPLS
Mininet for Emulation and Prototyping Software-Defined Networks	De Oliveira et al	2014	for evaluating SDN and most research has been spent on the Mininet SDN network simulator. go through simple use and test the scalability of Mininet is a modest but dominant tool for simulating a SDN network.	Discuss only about only how to use software simulating, but not talk about traffic engineering part of SDN system

Seamless MPLS Architecture	N. Leymann	2014	Seamless MPLS provides end to end service independent transport. Therefore it removes the need for service specific configurations in network transport nodes	Define well about Seamless design but not discuss about SDN supported seamless MPLS and also does not discussed about seamless MPLS performance in detail.

2.4. Summary

In summary, although there have been proposed SDN supported MPLS in recent years, the above methods still suffer from certain shortcomings as mentioned above. It is discussed about MPLS supported by SDN but, not discussed about seamless MPLS supported by SDN system. Since MPLS and seamless MPLS are unlike, we have proposed SDN supported seamless MPLS based on open flow rules. It focuses on checking seamless MPLS either functional or not when integrating SDN on it.

We also identify the gaps that the previous work is not measure the result quantitatively. Therefore, based on the above-stated gaps of the related works, we will have discussed the proposed system in detail. We also discuss material and methodology used for the study. In this study, we have included multiple metrics such as average packet per second, and average byte per seconds and the time span are being considered.

CHAPTER THREE

3. Research Methodology

3.1. Introduction

The chapter carefully examines the approach that the study needed. The methodology to ensure the study yielded a reputable and very reliant SDN network based on a routing that is based on S-MPLS. the chapter begins by looking at the research design this study used, that design is be supported target population the research will be working with. The chapter then travels forwards to see how that data collected was analyzed to suit the needs and answer the questions presented by the study. Finally, the chapter justifies the quality of the research and looks at the ethical considerations that were made when conducting the research.

3.2. Research Design

A mixed approach will be used in this Thesis. This thesis sought to use both experimental synthesis and prototyping approach to achieve the objectives. This approach will allow the researcher to be flexible in their data collection method and analysis.

3.3. Sampling Method

The study will adopt probability-sampling method. To be precise, convenience sampling was favorable where the selected samples were easily available and convenient. According to (Singh, Y. K. 2006) this kind of sample techniques makes information-gathering process to be quick. This method was practical in selecting the SDN emulator, Mininet with the associate components like Putty, Xming X server and Wireshark.

3.4. Data Collection

The study used both primary and secondary data collection methods. Primary data are the data collected by a researcher specifically for a research assignment. In other word no one has collected and published the data in a forum accessible to the public. For this research, the primary data that forms primary information to report all simulation result observation

collected from Wireshark packet analyzer in implementing a routing procedure in SDN supported seamless MPLS environment.

Primary data was supported by secondary data that equally will play a major role in this study. Secondary data such as, articles, journals, research and other topics related to seamless MPLS and SDN. Try to find to explain adaptive routing systems could be accomplished in SDN. The secondary data also observed at the technologies required to confirm the researcher developing an application aware network. A full learning of SDN and Open Flow ideas and MPLS system was dynamic to approve the study succeeded.

3.5. Data analysis and presentation

Data from the different Mininet SDN consoles and Wireshark packet capture screens will gather and analyze. Inferences and deductions models will apply to draw conclusions from the experiments. The results will represent images with accompanied analysis. We have used both quantitative and qualitative research approaches in which we can answer how the proposed system is expected to support solutions to the problems. Quantitative research frequently deals with experiments, whereas qualitative research can practice procedures. We have used qualitative research to deal with the functionality and understanding of the situation behind the SDN supported seamless MPLS, as well as quantitative research for performance metrics, numerical analysis of result.

3.6. System Development Methodology

The development and simulation methodology that this thesis will apply prototyping. Prototyping required the creation of an invention that goes through several iterations that span building, testing and reworking until an acceptable prototype that meets the research's objective is arrived at. I chose this approach because it works best for scientific research that does emulation experiments.

3.7. SDN Evaluation

To assessment SDN in exercise is not conventional forward as the technology is still very young. There are real physical devices available but not broadly and nor cheaply. The key

element of a SDN network, the controller (software), instead has there are many options readily available for download for free.

3.8. The SDN Controllers Considered for the study.

The controller on the control planes are the major element used for entirely operations of data plane management. Hence, the performance and competences of the controller itself are really important. Furthermore, the tools used to standard their performance essential be perfect and real in measuring different evaluation parameters. This thesis chose to settle on pox controller for the implementation. Helium is route as a Karaf delivery and any extra parts can be installed within the running distribution.

3.9. Software used for the Experiment.

We use Mininet simulation, that runs a group of hosts, switches, routers, and links into a distinct Linux kernel. It carries out serious virtualization to style an only method look like a whole network, running the alike kernel, system, and user code. Wireshark is an open-source packet identifier that used for network troubleshooting, investigation software and communications protocol development, originally named Ethereal. It lets you capture and interactively browse the traffic running on a computer network. It is the defacto standard through several businesses and learning institutions.

3.10. Build SDN Networks

We will practice Mini Edit, the Mininet graphical user interface, to established a rivalled network equipped by (Open Flow switches and Linux hosts Juniper Networks, 2012). Here the Mininet window will look on your computer's desktop. At that time, we Construct the network containing of a diagram to switches with a dominant core switch linked to two extra switches that are linked to two hosts, each. Attach a controller to entirely the switches.

3.11. The Operation of the Experiment.

Mininet emulation software originates as a pre-built virtual machine (VM) image, The VM was allocated two 2 GHz Intel Core 7 processors and 4 gigabytes of RAM (Random Access Memory). The POX controllers were then installed on the Ubuntu 16.04. Setting up POX Controller is straightforward. Download the package, extract the package after we install Ubuntu 16.04 operating, we install Mininet & Pox controller using Virtual Machine, and then practice

xterm to access to mininet machine. Finally, we Live capture and offline analysis the traffic running on a computer network by Wireshark., run it. To start install mininet in VMware we install Ubuntu 16.04 operating system.

3.12. Topologies applied in the Testing

In Seamless MPLS Network, the access and aggregation layers are inside a single autonomy system(AS), and the core layer belongs to another autonomy system(AS). Then the three domains are integrated end-to-end into one MPLS domain using Seamless MPLS architecture**.**

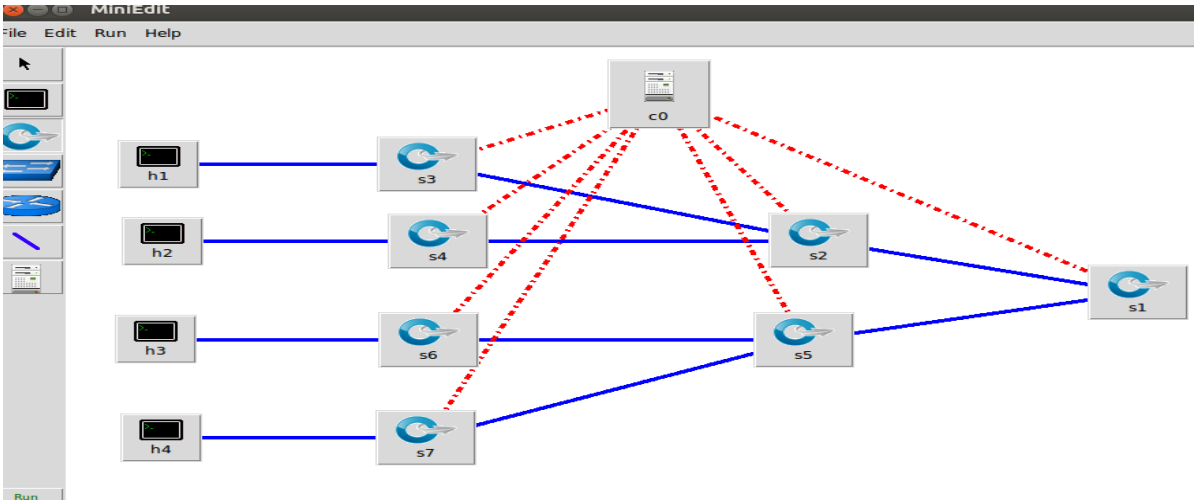


Figure 8:Topology of proposed Architecture

Tree Topology consisting of seven nodes are going to build to simulate basic functions initiate in the existing network and to make practice about SDN specific functions and, to see how mininet works in Connecting the nodes together. Three MPLS domains are used where access, aggregation, core regions are in one seamless MPLS domain. These topologies are representative of today's seamless MPLS architecture. In this study, S3, S4, S6, S7 are consider as access nodes because it is connected from hosts and S2, S5 are consider as intermediate node or aggregation layers because different access nodes are going to collected to it. S1 is consider as core layer because of all access nodes are collecting to it.

Due to memory restrictions of personal computers and the process needs serious of the simulation tools used, it makes necessary to use only seven switches and four hosts. These topologies make our simulation environment look like a real network. The decision to select

the hosts and switches was based on my interest to be better demonstrate the concept about SDN supported seamless MPLS.

3.13. Description of proposed system.

The figure above contains four likely users that will cooperate with the system. The Open Flow controller will be accountable for serving the Open Flow messages that will follow an MPLS coupled routing in SDN system. The user in the system will be accountable of starting the ICMP requests that will target H4. H1 will be liable in packetizing that request in an ICMP request message and supplement the essential data to permit the switches in the SDN arrangement to doing on it consequently. Now the system will get this information, check the packet either it is an ICMP packet or not grounded on the Open Flow rules set to it by the controller. Later, the process will separate the ICMP demand packets and forward them to the switches predefined by the POX controller to hold MPLS request in the network. This process will confirm the ICMP demand packet is transported to H4.

H4 is liable of beginning an ICMP reply and creating an ICMP reply packet, accumulation the suitable information and forwarding the packet to the system for management. The system will replication the procedure of reviewing the packet either it is an ICMP reply or does not. Entirely ICMP responses will be give in to the flow pass distinct by the system to the fitting switches that will then distribute to H1.

3.13.1. sequence diagram

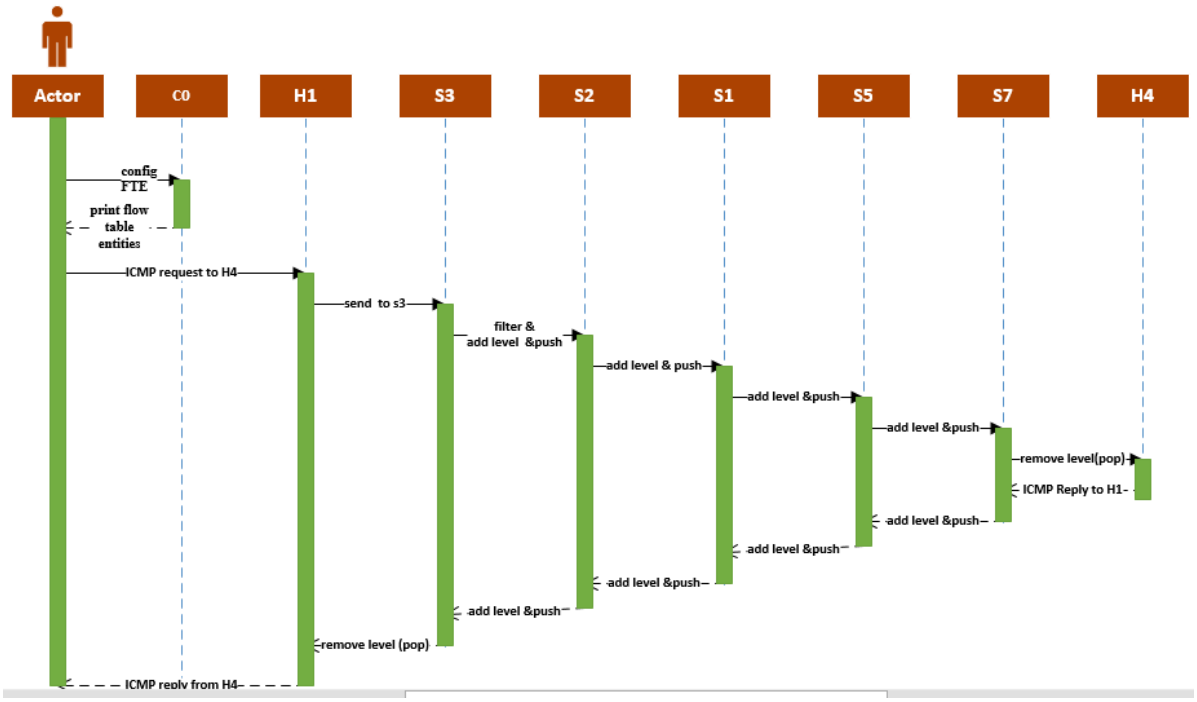


Figure 9:sequence diagram

The sequences diagram showed above frameworks the chain of events the actors and the system will be involved in to bring seamless MPLS routing procedure in SDN to accomplishment. The user who consider as the network engineer will send configuration messages via OpenFlow protocol to the OpenFlow pox controller that will configure the script to entirely switches and the flow table records will be printed back to the user. The actor will start an ICMP request message to H1. S3 which are consider as an access node will recognize the ICMP packets received. This switch will form these packets across its interface and attempt to compare the flows to the flow table entries. Effective comparison will permit S3 to push a seamless MPLS label and send to its output port that connects it toS2. S2 and S1 and S5 will receive these packets, tagged to be forwarded to them through their respective input ports, and push their own seamless MPLS labels to the ICMP packet and forward them out through their output ports sequentially and S5 will forward directly to S7. S7 which is

consider as an access node will be responsible in popping the MPLS labels attached to the ICMP packets and forwarding this packets through its output port that connects it to H4.

H4 will take ICMP request, and build an ICMP reply send to S7 again. S7 in the instance will act as an access node for the ICMP reply coming back from H4 will identify the ICMP reply packets sent to it and will check these packets across its interface and attempt to match the flows from the flow table. Effective comparison will let the switch to push an MPLS label and forward to its output port that links to S5. S5, S1 and S2 will receive these packets tagged its own level and push their own seamless MPLS labels to output ports in sequence. S2 will send directly to S1. S1 will acts as an access node do the popping of seamless MPLS labels attached to the ICMP reply and send this packet to its output port that link to H1. H1 will understand the packet and mark the close of the activities.

3.13.2. Open Flow Rules.

Using the next script, we be able to mark the OpenFlow Rules to the nodes, thus the network doings as seamless MPLS network. The script configured rules for ICMP traffic for seamless MPLS Ethernet packets and other packet will be left. Two flow entries table have for S1 and S7, because S1 and S7 are consider as access nodes as proposed topology. This study is a good initial point in order to realize the mechanics of the seamless MPLS protocol and its linking with OpenFlow and Open vSwitch.

```
#! /bin/bash
echo "Recognized Switches to exertion with Open Flow 1.3"
for j in s3 s2 s1 s5 s7; do
sudo ovs-vsctl set bridge $j protocols=OpenFlow13
done
# Clear flow tables
echo "Clearing Flow tables."
for j in s3 s2 s1 s5 s7; do

sudo ovs-ofctl -O OpenFlow13 del-flows $j
done
```

Rules at S3

```
echo "Setting rules at s3"
## TABLE 0 ##
sudo ovs-ofctl -O OpenFlow13 add-flow s3
"table=0, in port=$S3_ETH1,eth_type=0x800,actions=goto_table:1"
```

```
sudo ovs-ofctl -O OpenFlow13 add-flow $S3
"table=0, in_port=$S3_ETH2,eth_type=0x8847,actions=goto_table:1"
## TABLE 1 ##
# If it came in through port 1 & it's an IPv4 packet:
# push by adding label 24 and send it through port 2
sudo ovs-ofctl -O OpenFlow13 add-flow $S3
"table=1,in_port=$S3_ETH1,eth_type=0x800,
actions=push_mpls:0x8847,set_field:24->mpls_label,output:$S3_ETH2"
# remove label and send it through port 1
sudo ovs-ofctl -O OpenFlow13 add-flow $S3
"table=1,in_port=$S3_ETH2,eth_type=0x8847,mpls_bos=1,
actions=pop_mpls:0x800,output:$S3_ETH1"
```

```
###### Rules at s2 ######
echo "Setting up rules at s2"
# add new label push new label
sudo ovs-ofctl -O OpenFlow13 add-flow $S2
"table=0,in_port=$S2_ETH1,eth_type=0x8847,
actions=push_mpls:0x8847,set_field:120->mpls_label,output:$S2_ETH2"
sudo ovs-ofctl -O OpenFlow13 add-flow $S2
"table=0,in_port=$S2_ETH2,eth_type=0x8847,
actions=push_mpls:0x8847,set_field:115->mpls_label,output=$S2_ETH1"

###### Rules at s1 ######
echo "Setting up rules at s1"
# add new label push new label
sudo ovs-ofctl -O OpenFlow13 add-flow $S1
"table=0,in_port=$S1_ETH1,eth_type=0x8847,
actions=push_mpls:0x8847,set_field:100->mpls_label,output:$S1_ETH2"
```

```
###### Rules at S7 ######
echo "Setting up rules at s7"
## TABLE 0 ##
# If interface type is IPv4 (800) or MPLS output (8847), go to table 1
sudo ovs-ofctl -O OpenFlow13 add-flow $S7
"table=0,in_port=$S7_ETH2,eth_type=0x8847,actions=goto_table:1"
sudo ovs-ofctl -O OpenFlow13 add-flow s7
"table=0,in_port=2,eth_type=0x8847,actions=goto_table:1"

## TABLE 1 ##
# If it originated in through port 1 & it's an IPv4 packet:
# push label 99 and send it through port 2
sudo ovs-ofctl -O OpenFlow13 add-flow $S7
"table=1,in_port=$S7_ETH1,eth_type=0x800,
actions=push_mpls:0x8847,set_field:99->mpls_label,output:$S7_ETH2"
# remove label and send it through port 1
sudo ovs-ofctl -O OpenFlow13 add-flow $S7
"table=1,in_port=$S7_ETH2,eth_type=0x8847,mpls_bos=1,
actions=pop_mpls:0x800,output:$S7_ETH1"
```

3.14. Summary

In this chapter, we have discussed the research design, data sources, data collection approaches, solution design approach and tools used the proposed method implementation, evaluation methods, and the research methods. In the next chapter, we observe the simulation, and, evaluation and discussion of the results.

CHAPTER FOUR

4.Results and Discussions

4.1. Introduction

This chapter focuses on how the proposed system design in the previous chapter was achieved. And we are going to trying to test the topologies that we suggested it in the above chapter and studying it to see if we reach the goal of our thesis that we bring and test the SDN based seamless MPLS Basic function compare to the seamless MPLS not supported by SDN. The chapter continues to report on the examination outcomes that the agent collected using several terminal window and Wireshark capture outcomes in print screen.

The system results are going to implement by matching ICMP traffic and embedding MPLS labels on ICMP packets and check link discovery to confirm connectivity between the hosts then check the MPLS network either push or pop the incoming packet. Moreover, it implementing SDN firewall to check up the system are going to filter traffic between hosts according to the rules and let it pass through or not to protect my network from un wanted packet and, finally carrying out analysis of performance.

4.2. Simulation

4.2.1. Connectivity among Hosts

Mininet is run in Ubuntu 16.04 operating system and it needs to be run as the root user. And POX controller will be up after POX controller has been up, this command will run in mininet console "**sudo mn --topo –coustm,4 –mac --switch ovsk --controller=remote"** then connectivity results that were obtained by the researcher by using ping all command are seen below.

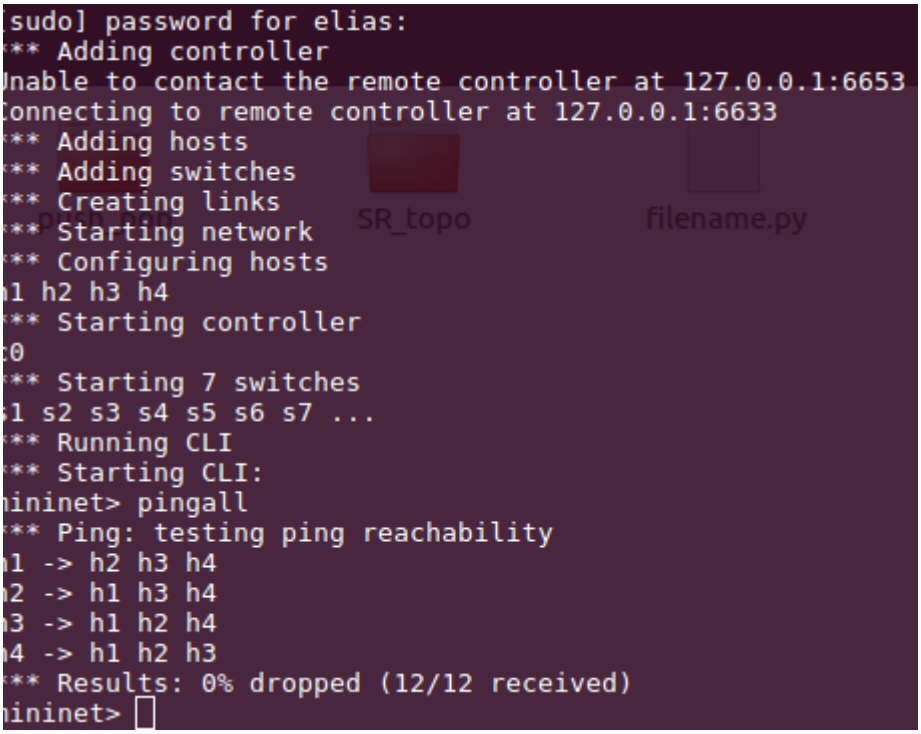


Figure 10:Connectivity result after POX controller up

4.2.2. Implanting labels on ICMP packets.

Seven switches are going to feed rules of their tasks, we are going select S3, S2, S1, S5 and S7 to make the LSP that seamless MPLS labeled ICMP packet would pass through it. S3 and S7 is considered to do both MPLS Popping and Label pushing since which were the access nodes(AN)switches. Those switch are designed to routine pair of flow table. The first table is for ID of ICMP traffic and pushing Address Resolution Protocol (ARP) needs and, the other flow table was for pushing and popping seamless MPLS labels. The tables are recognized as table 0 and table 1. Table 0 is submitting traffic flow to table 1 for MPLS labelling. S2, S1 and s5 are designed to perform label pushing in two directions. The flow tables are recognized by their defaulting value that is 0. The underlay figures demonstrate the different flow table from S1 to S7.

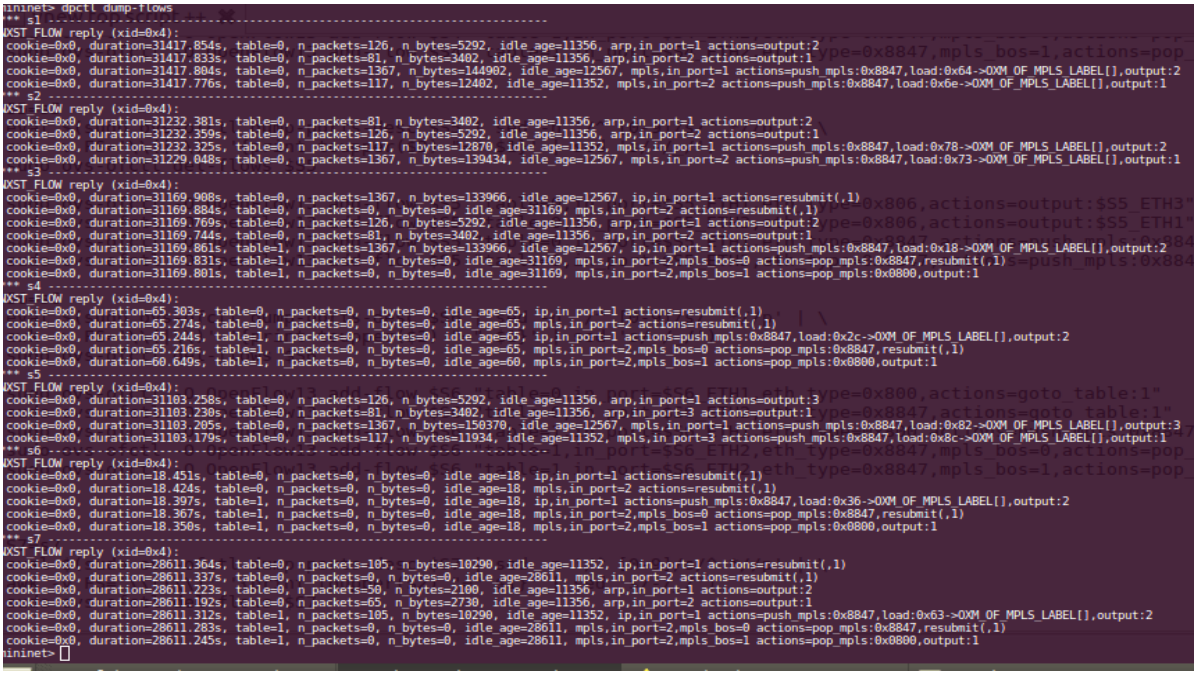


Figure 11:system flow table entities

The above flow table outcomes which obtained by administration dptcl-dump flows command on the VM terminal. From the flow table outcomes overhead, it can observe that S3, S4, 66 and S7 are designed having two flow tables, table=0 and table=1. The 1st table is named 0 is configure to separate IP traffic that means ICMP traffic and accelerative it to table1 for the accumulation MPLS labels. Table= 0 at the similar time was recycled to forward ARP demands. The other switches in the LSP, S1, S2 and, S5, have one flow table0 which applied to push MPLS labels.

The goal of implementing level is to test whether access switches are able to send ICMP traffic and at the same time pop and push MPLS labels. Furthermore, the rest intermediate switches in the LSP is going to be analysis either it can be push seamless MPLS labels in two directions of traffic flow. Xterm terminal windows showed the results from the coupled with Wireshark packet capture results.

From Figure shows below, S3 on Eth2 was able to add label 24 to the traffic leaving it as it goes to h4 which is identified by IP address 10.0.0.4. The figure also displays that the switch sends ICMP packet.

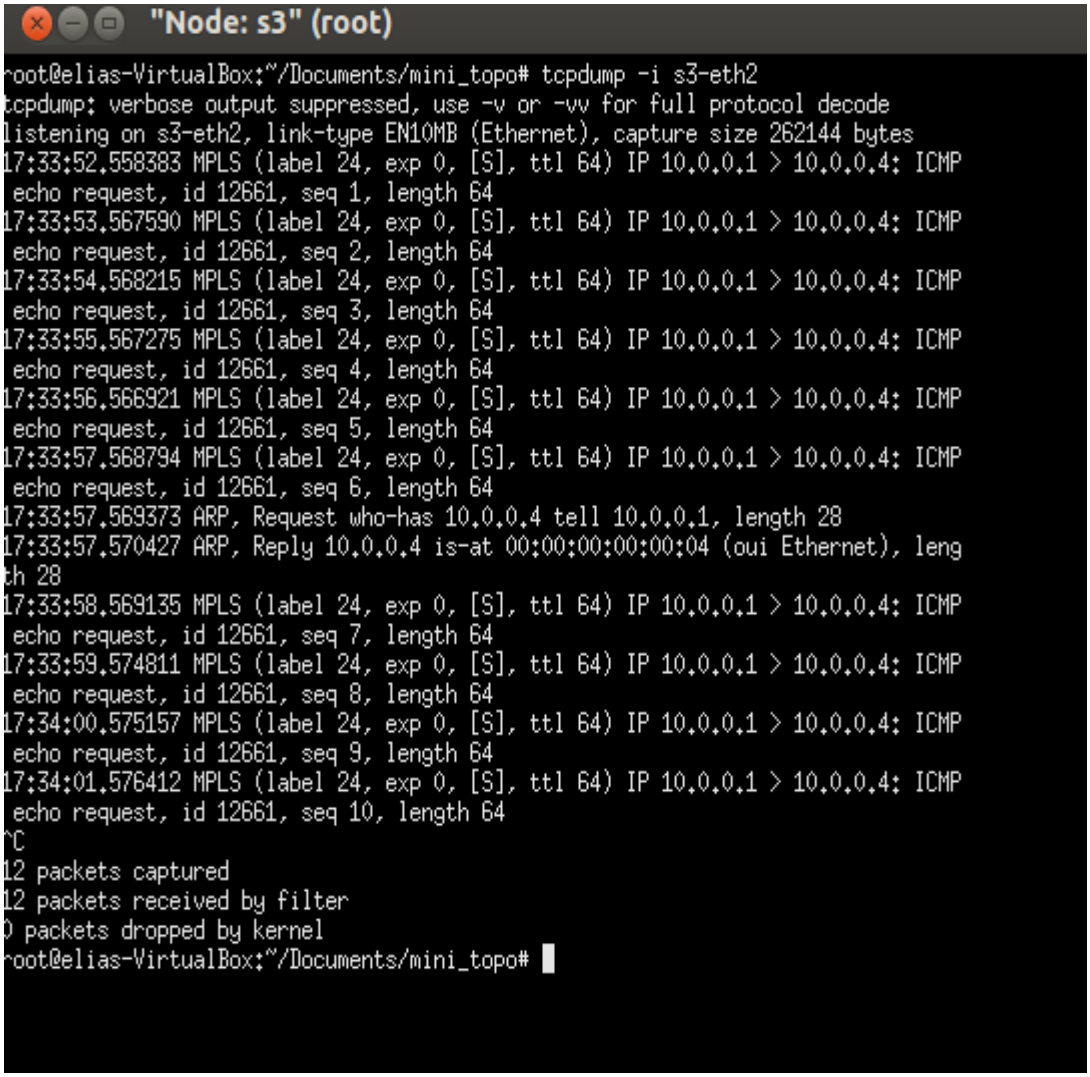


Figure 12:Xterm Results for S3

Below Figure showed how S2 on interface 1 is able to receive traffic coming from H1 which has passed through S3 was added label 24 and, S2 adds label 115 to the traffic passing through it as it goes to H4 which is identified by IP address 10.0.0.4.

On the other hand, S2 on interface 1 is able to receive traffic coming from H4 which has passed through S1 was added label 140 and, S2 adds label 110 to the traffic passing through it as it goes to H1 which is identified by IP address 10.0.0.1.

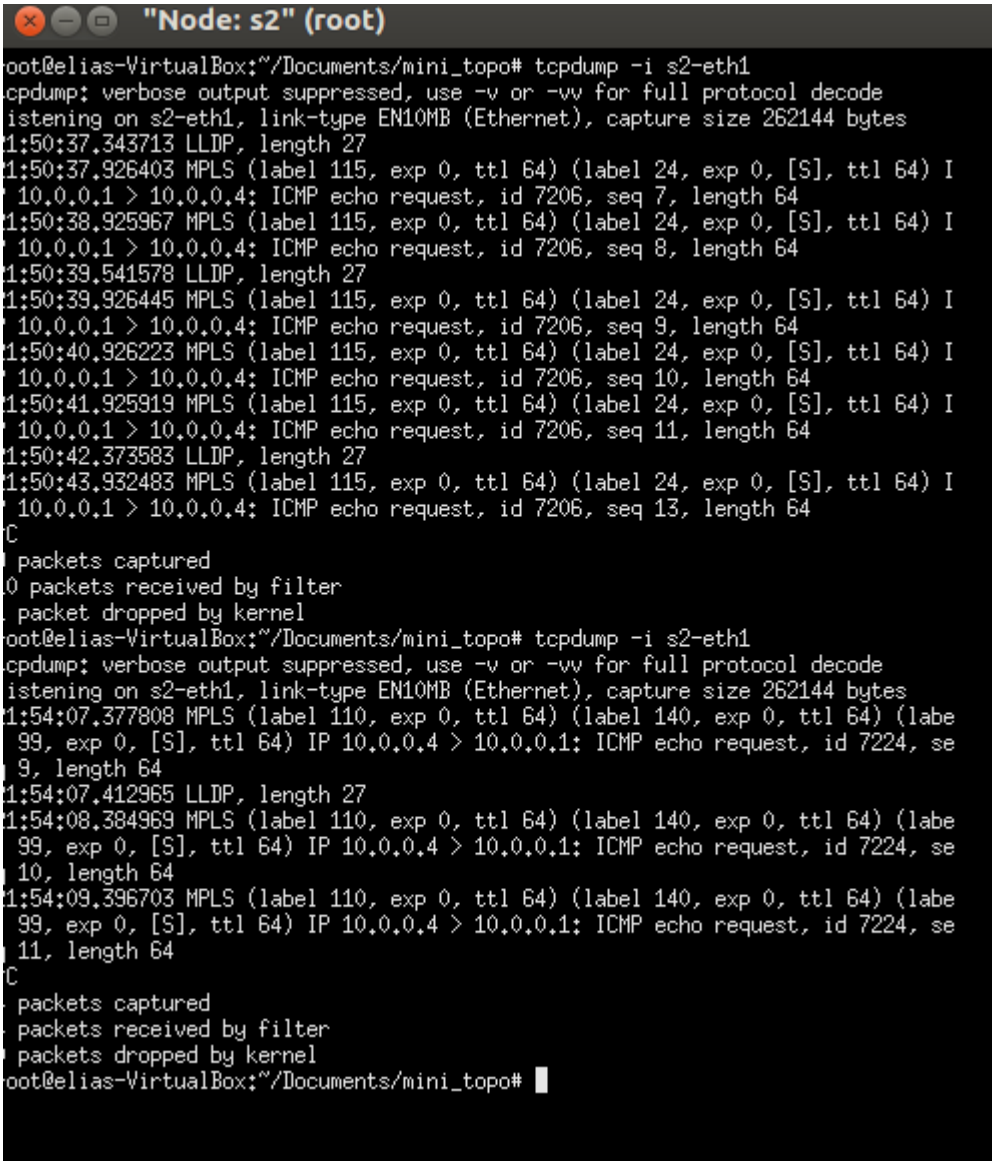


Figure 13:Xterm Results for S2

Below Figure showed how S1 on interface 2 is able to receive traffic coming from H1 which has passed through S2 and label 115 was added to S2 and, S1 adds label 100 to the traffic passing through it as it goes to H4 which is identified by IP address 10.0.0.4.

On the other hand, S1 on interface 1 is able to receive traffic coming from H4 which has passed through S5 and label 140 was added to S2 and, S1 adds label 110 to the traffic passing through it as it goes to H1 which is identified by IP address 10.0.0.1.

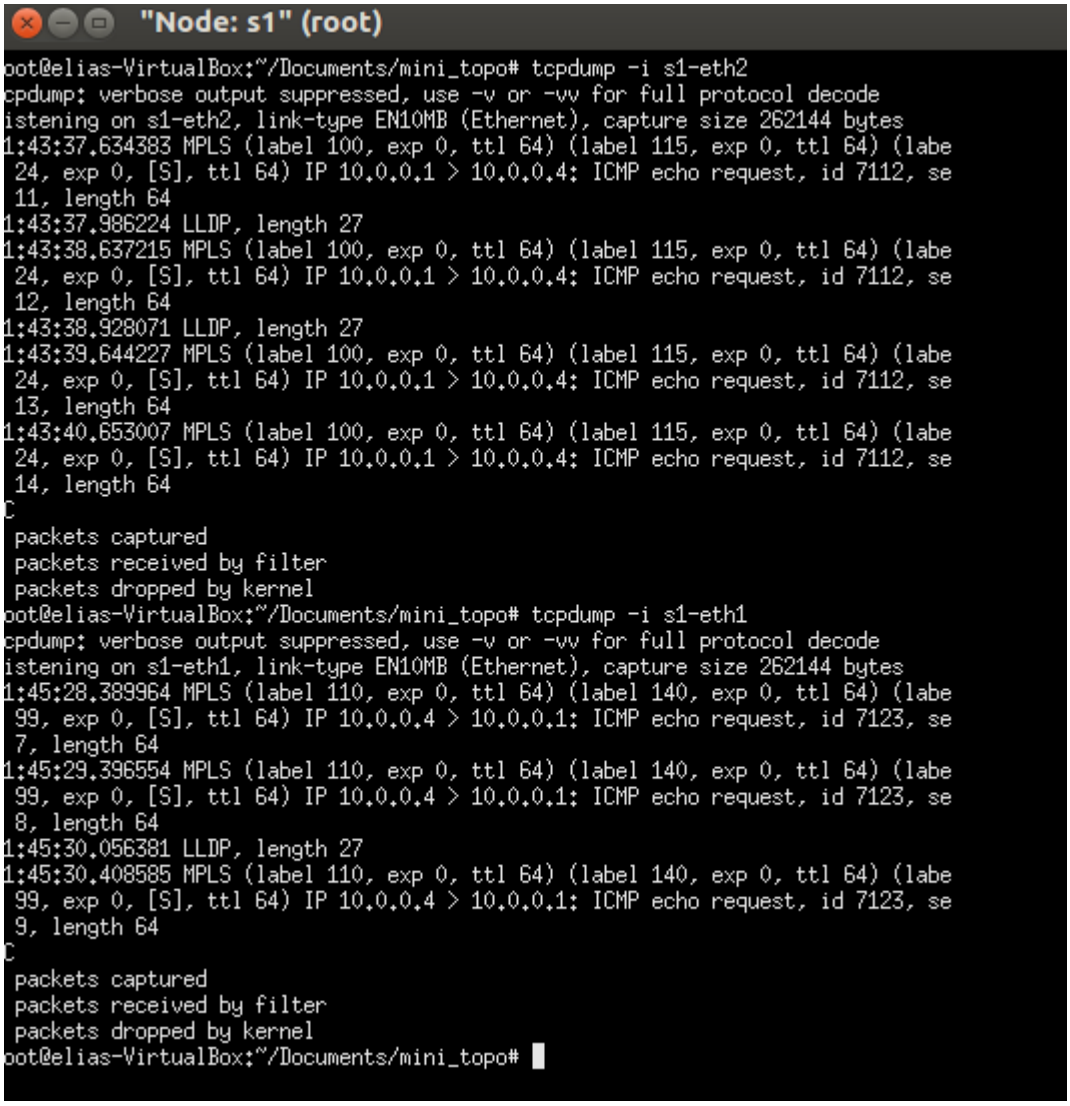


Figure 14:Xterm Results for S1

Below Figure showed how S5 on interface 1 is able to receive traffic coming from H1 which has passed through S7 and label 99 was added to S7 and, S5 adds label 140 to the traffic passing through it as it goes to H4 which is identified by IP address 10.0.0.4. On the other hand, S5 on interface 3 was able to received labels 99 packet from H7 of eth2.

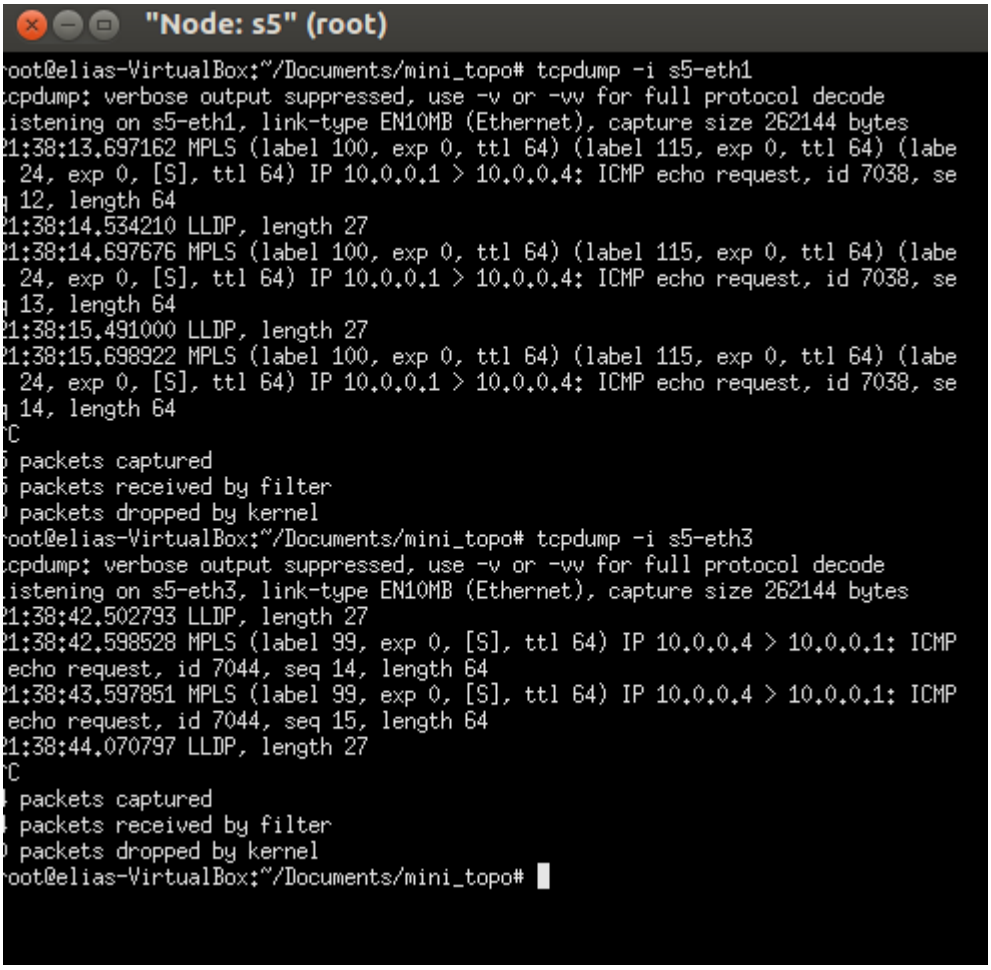


Figure 15:Xterm Results for s5

From Figure shows below in what way s7 on Eth2 was able to add label 99 to the traffic passing through it as it goes to H1 which is identified by IP address 10.0.0.1.

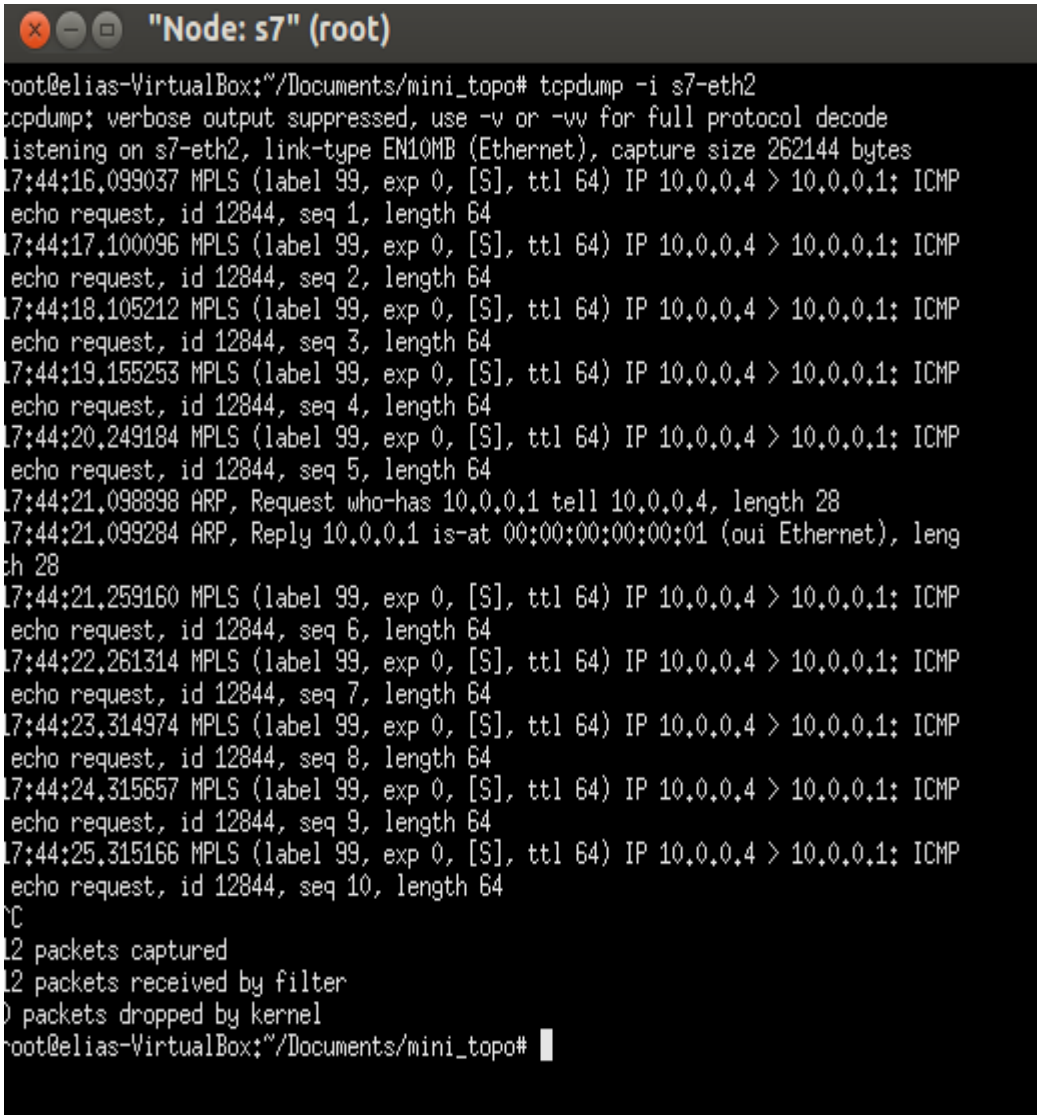


Figure 16:Xterm Results for S7

4.2.3. Implementing SDN firewall

Software-Defined Networking is an increasing expertise that will determination the networks of the next technology. Network managers are given the independence to introduce their networks, but alike it brings with it new security problems.

Firewalls are a major element of network security systems with the ability to block network data traffic flows according to pre-defined rules. Firewall's key function is to bounds uninvited traffic. It will path and manage the flow of data that move toward from various sources into the network and functions on the standard of preconfigured rules. Firewalls are the important elements of the network infrastructure.

For an SDN founded protection, we are successful to use the SDN control Open Flow controller to select traffic among hosts centered on about rules and therefore let it permit through or not. The method we do this is by expenses the POX controller to introduction our prerequisite *rules*. After making decision by rules to implement through our firewall. It should also be well-known that it will be a two directional block, and if a rule exists, the controller will confirm that switches neither let incoming or outgoing traffic from either host arrive to other host. For this cases we can have the next two firewall rules.

- h1 and h2 are mutually blocked
- h3and h4 are mutually blocked

If the Firewall implementation really did what it was supposed to, that just needs to begin Mininet topology simulation and the POX controller, need to introduce firewall algorism to POX controller to implement Firewall. Now, we go back to the Mininet terminal and type in pingall to realize if our firewall really works.

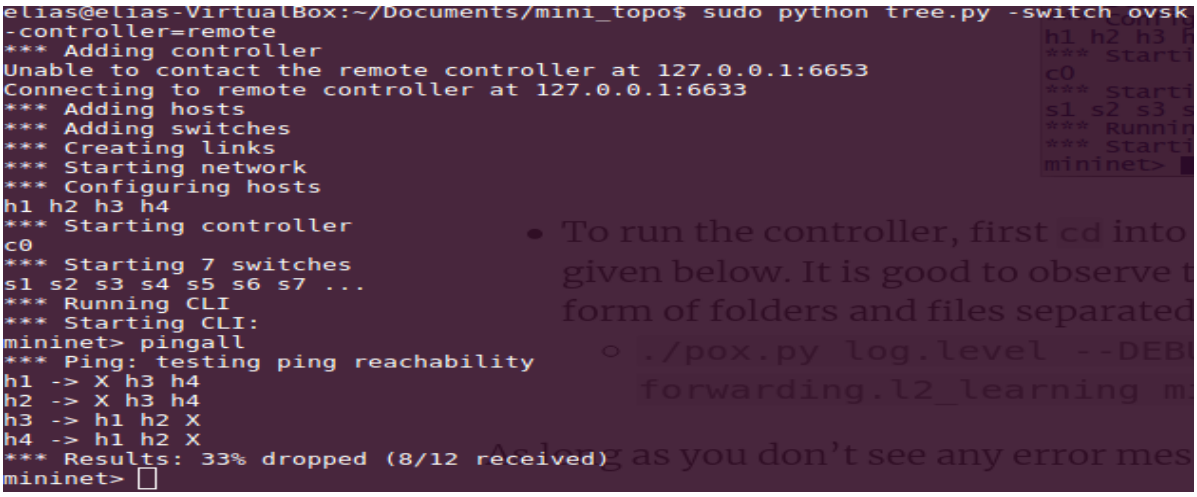


Figure 17:host connectivity on SDN Firewall

4.3. Analysis of simulation of Results

This section describes the environment used to perform the experiment of the proposed SDN based seamless MPLS over the existing one. In addition, we will explain the performance measurement metrics and elaborate on the results, explanations, and evaluations of the proposed method. Simulation results are displayed using different datasets. Finally, parameters such as network parameter and traffic parameter are taken into account to measure the proposed system.

It refers to analyzing compatibility and interoperability between software used for simulation and virtualized switches for seamless MPLS network and software define network(SDN). It checks software to be compatible with different Operating Systems like Windows, Unix, Mac OS etc. besides that, it refers to the integrality of software standards being demonstrated by the system implementation. In this parameter, it broadly discusses Network Topology Parameters, refers to evaluating which switches are connected directly to each other and build the global topology using Link Layer Discovery Protocol.

It was installed Mininet & Pox controller using Virtual Machine, and then It was also important to make sure there is connectivity between the hosts in the topology before implemented the proposed seamless MPLS network. This was achieved by looking for a component on OVS that does Link Layer Discovery Protocol (LLDP) and allows devices with

in a network to advertise their uniqueness. The author joined this vendor neutral link layer protocol with Open Flow link layer discovery tool to ensure the nodes could connect to one another. Ping results that were obtained by the researcher after running the ping all command from the Mininet console all hosts were reachable. They are connected each other as we can see from below Wireshark capture so that now we were confirmed that all installed software was going to working well.

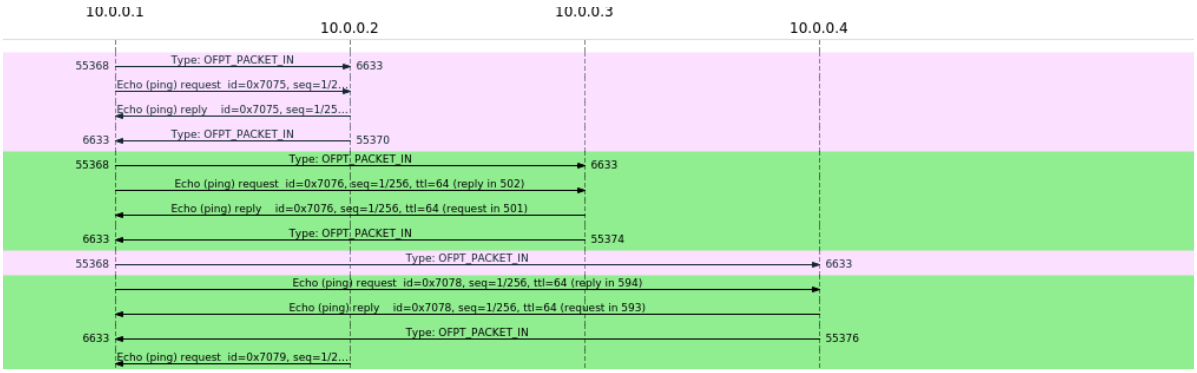


Figure 18:Hosts connectivity Flow graph

After testing the connectivity and be certain that the functionality is working fine, then we testing simulation results that we discussed in the above unit and analyzed proposed network system under traffic management measurement metrics and also a simple solution is to split traffic from the package level, in this parameter functions or capabilities associated with SDN is identified among vast use of SDN, we are going to discuss how implemented SDN controller supervise each labeled packet and being popped and pushed by label switching routers.

Moreover, in this metric we prove either SDN controller handled routing procedure in the entire network and detect ICMP traffic packet from hosts in the network and forward them to appropriate switches, On the other hand, Recently, high-speed broadband facilities, social networking interacting, and cloud situations have improved internet penetration significantly. Due to this, there is huge amount of user data available in seamless MPLS network including personal data, enterprise data and financial data. This leads to serious dangers from malicious users. In this Parameter, we have seen how Python-based SDN controller permitted or rejected hosts.

Tree Topology consisting of seven nodes (S1, S2, S3, S4, S5, S6 and S7) four hosts (H1, H2, H3 and H4) and one controller (c0) are going to build to simulate basic functions initiate in the existing network and to make practice about SDN specific functions and, to see how SDN works in Connecting the nodes together. For this simulation analysis only H1 and H4 were randomly chose to participate in the forwarding of ICMP on SDN based seamless MPLS packets. However, it is important to note that any device could be chosen and can as well forward traffic in the network, additional, all seven switches are configured as for Seamless MPLS operation, but primary selection to participating in this valuation was S3, S2, S1, S5 and S7. S3 and S7 were designed as Access nodes (AN) that level assigning is going to started and the other switches S2, S1 and S5 were designed as intermediate nodes in which previous level is swapped by other level. One thing we noted that all switches used the concept of Label stacking, which is the encapsulation of an MPLS packet inside another MPLS packet that is adding an MPLS header "on top of" an existing MPLS header.

From the outcomes capture by the different Xterm screens, Mininet console and Wireshark, the study has demonstrated that S3, S2, S1, S5 and S7 were configured to do seamless MPLS based on the flow table entries shown in the chapter above. These flow passes were very specific to each switch in the instance that there were predefined labels that each of the switches were supposed push to the next nodes

When it comes to label pushing, and popping labels from H1 to H4, the xterm results showed, the ICMP request enters the switch S3-eth1 without any label, the packet leave S3 with the label 24 inserted, and send to S2 on interface 2. S2 on port1 is able to receive traffic which leveled 24 and added 115 to the packet coming to it then thought eth1 port send traffic to S1. S1 is able to receive traffic on eth1 from S2 and add level 100 to transmit to S5 thought eth2 port then S5 add level 130 and send to S7. Finally, s7 on eth3 popping packet and remove level from packet then send to H4.

By now let us make analysis packet transmitted from H1 is going H4 by passing S3, S2, S1, S5 and S7 and, arrived to its destination. The Wireshark captures seen in figure 20 showed how the ICMP packets on the filter bar goes from hosts to MPLS depending on the link it's traveling, and how the labels are being switched.

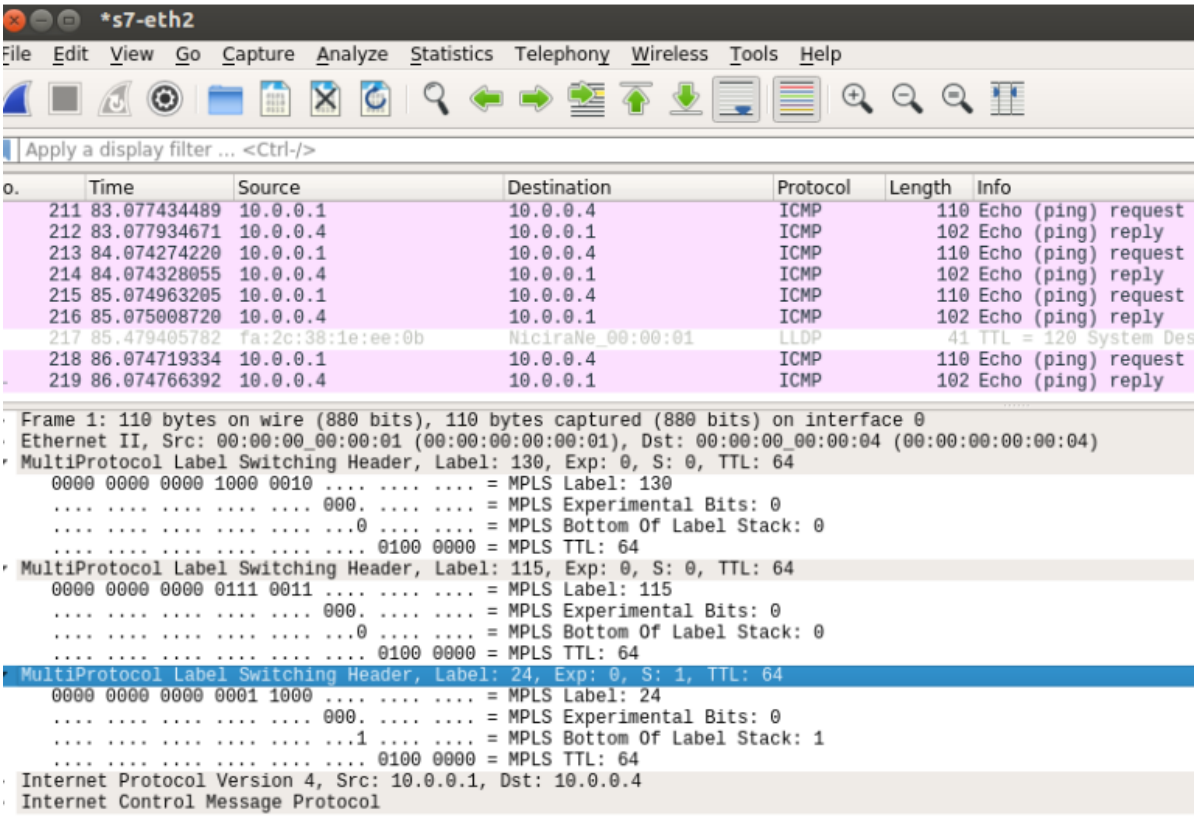


Figure 19:Wireshark capture results from H1 to H4

Further, from figure 20 below verified that from H4 to H1, the ICMP request enters the switch S7-eth1 without any label then the packet leaves S7 with the label 99 inserted, pushing to S5 on interface 2. then S5 on port3 is able to receive traffic which already leveled 99 and added new level 140 to the packet coming to it, then thought eth1 port send traffic to S1. S1 is able to receive traffic on eth2 from S2, and add new level 110 to transmit to S2 thought eth1 port then S2 add level 120 and send to S3. Finally, S3 on eth3 popping packet and remove levels from packet then send to H1.

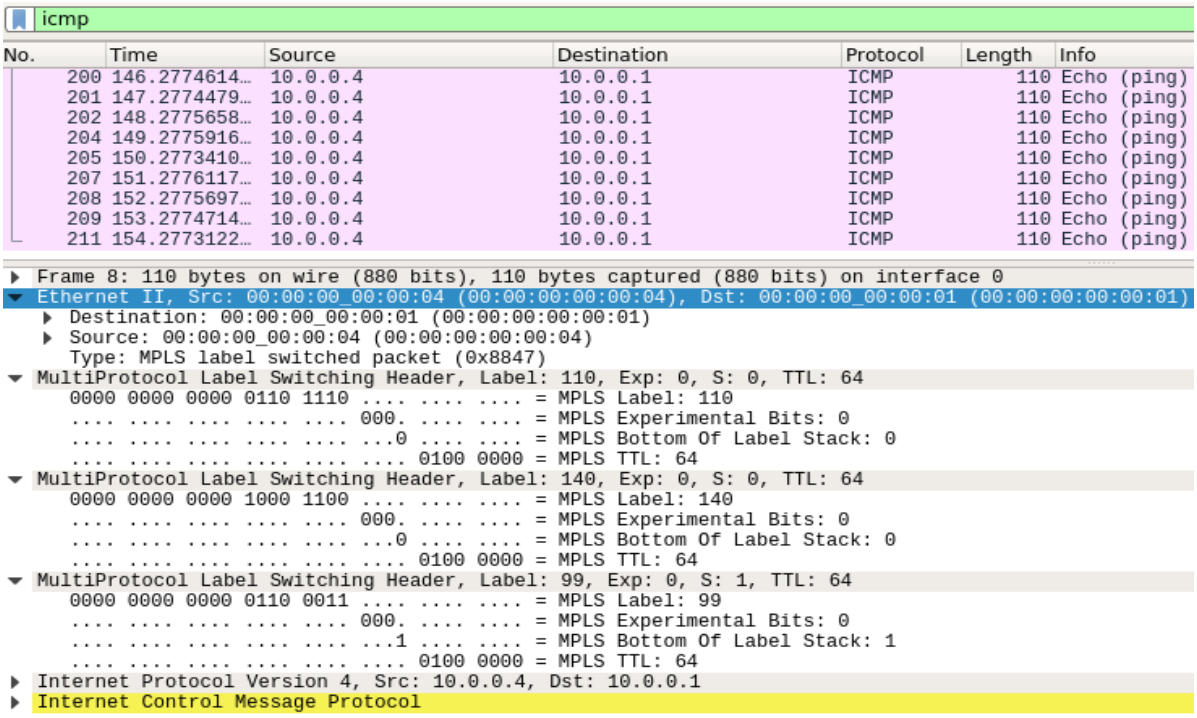


Figure 20:Wireshark capture results from H4 to H1

Finally, we're going made analysis how to implemented SDN Firewall by means of the POX Controller and Mininet. Initially, I decided on the *rules in which H1 and H2, H3 and H4 are bidirectional blocked.* Then we typed pingall to the Mininet CLI to see if our Firewall actually works. According to below flow graph (H1 and H2) and (H3 from H4) are not ever can connect bidirectional, other hosts except the host blocked by the controller is connect each other, We got that there is a packet loss of 33%, that is absolutely correct. Hence, we were effective in our attempt of implementing an SDN Firewall. below are the results capture from Wireshark.

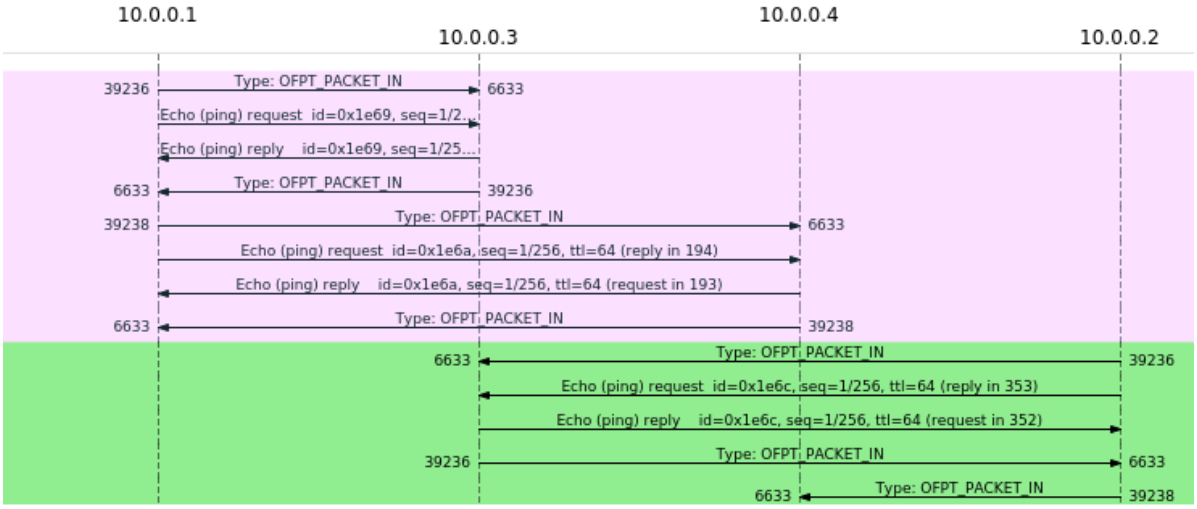


Figure 21:flow graph for pass and blocked hosts

4.4. Performance Results

4.4.1. Qualitative performance

Table 2:summery of result Table

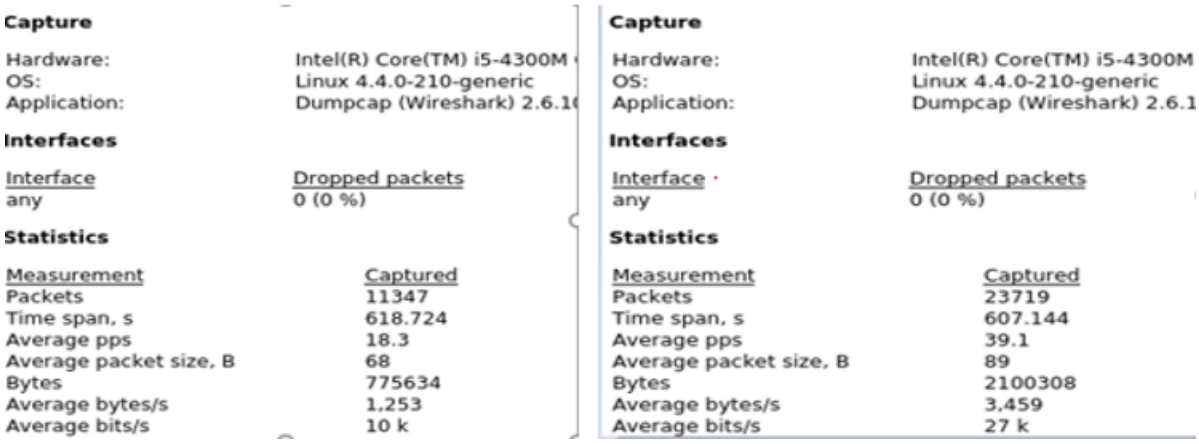
Test Parameter	Test Result	Description
integrality and compatibility to software standards being demonstrated by the system implementation	It is functional	tools ensure the nodes could connect to one another.
output being captured on Wireshark indeed prove seamless MPLS was applied on the specified packets	It is functional	Leveled packet transmitted from h1 is going h4 by passing s3, s2, s1, s5 and s7 and, arrived to its destination
ICMP traffic from nodes in the network and forward them to appropriate switches	It is functional	ICMP packet transmitted from h1 is going h4 by passing s3, s2, s1, s5 and s7 and, arrived to its destination

labels being popped and pushed	It is functional	S1, s2 and s5 pushing level, s3, and s7 popping packet (remove levels from packet) then send to h1 and h4 respectively
output being captured on TCP dump indeed prove level stacking was applied	It is functional	From figure 20 encapsulation of an MPLS packet inside another MPLS packet – that is, adding an MPLS header "on top of" an existing MPLS header.
output being captured on Wireshark flow graph indeed prove SDN firewall was applied	It is functional	Firewall actually works. according to in figure 22 flow graph, (h1 and h2) and (h3 from h4) are not ever can connect bidirectional

4.4.2. Quantitative Performance

The Wireshark showed below, maximum Packets per second (throughput), maximum bytes per second and the time span to transmit the captured packet in the network of not SDN supported seamless MPLS and SDN supported seamless MPLS network.

Seamless MPLS without SDN **Seamless MPLS with SDN**



Capture

Hardware: Intel(R) Core(TM) i5-4300M
OS: Linux 4.4.0-210-generic
Application: Dumpcap (Wireshark) 2.6.1

Interfaces

Interface	Dropped packets
any	0 (0 %)

Statistics

Measurement	Captured
Packets	11347
Time span, s	618.724
Average pps	18.3
Average packet size, B	68
Bytes	775634
Average bytes/s	1,253
Average bits/s	10 k

Capture

Hardware: Intel(R) Core(TM) i5-4300M
OS: Linux 4.4.0-210-generic
Application: Dumpcap (Wireshark) 2.6.1

Interfaces

Interface	Dropped packets
any	0 (0 %)

Statistics

Measurement	Captured
Packets	23719
Time span, s	607.144
Average pps	39.1
Average packet size, B	89
Bytes	2100308
Average bytes/s	3,459
Average bits/s	27 k

Figure 22 : Wireshark capture statistics

Packet per second (pps) was analyzed in topologies of SDN supported seamless MPLS and without SDN supported seamless MPLS networks. This parameter measure how much data can be transmitted from source to destination within a specific period of time. In this case, 11347 packets were transmitted within 618.7 seconds with no SDN support seamless MPLS and 23719 packets with 607.1 second in SDN supported seamless MPLS.

The results of the experiments are shown in Table 3 below The average time span in SDN supported seamless MPLS network is lower than not SDN supported seamless MPLS network, but average packet per second transmitted in SDN supported seamless MPLS is higher than with no SDN supported seamless MPLS. And also average bytes/s for SDN supported seamless MPLS is higher than from no SDN supported seamless MPLS.

Table 3:summery Performance measurement.

measurement	**Without SDN support**	**With SDN support**
Total Packets	11347	23719
Average bytes/s	1253	3459
Time span(s)	618.724	607.144
Average Packets per second(pps)	18.3	39.1

The graphs are stated in Figure26 and figure 27 are going to assured that transmitted Packet per second(pps) in SDN supported seamless MPLS is higher than not SDN supported seamless MPLS and also SDN supported seamless MPLS networks provides better performance in transmitting more packet with less time than not SDN supported seamless MPLS. So, we concluded that SDN can achieve the requirements of today's seamless MPLS network that requires high bandwidth to operate.

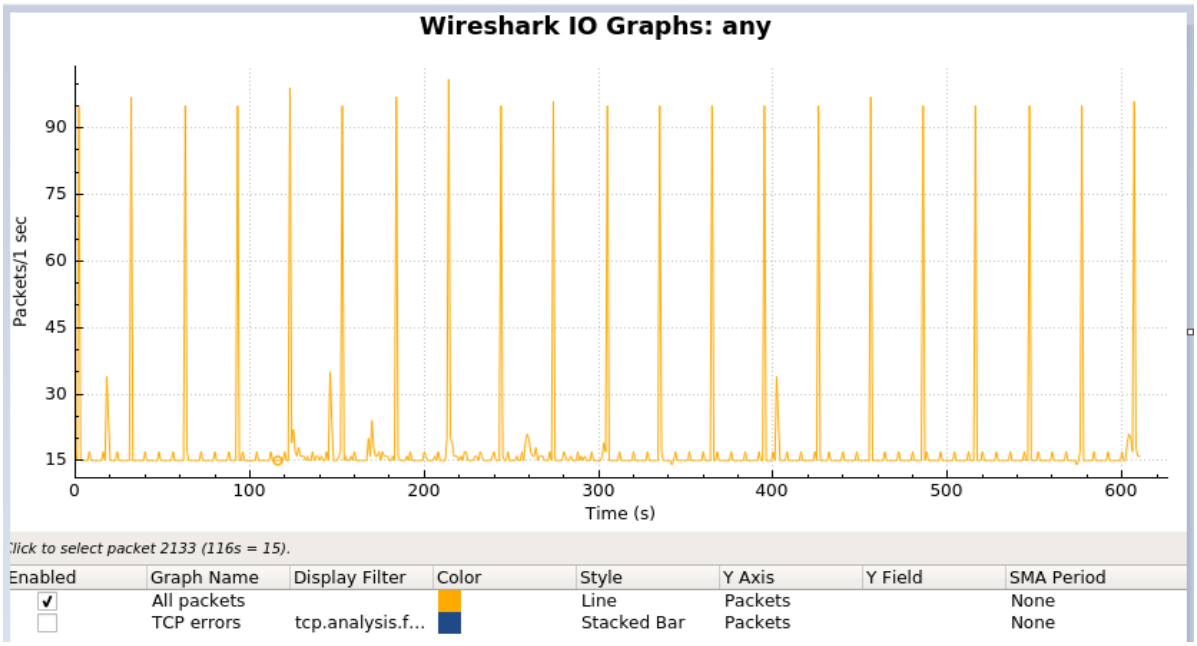


Figure 23: relation of Packet per second and Time without SDN

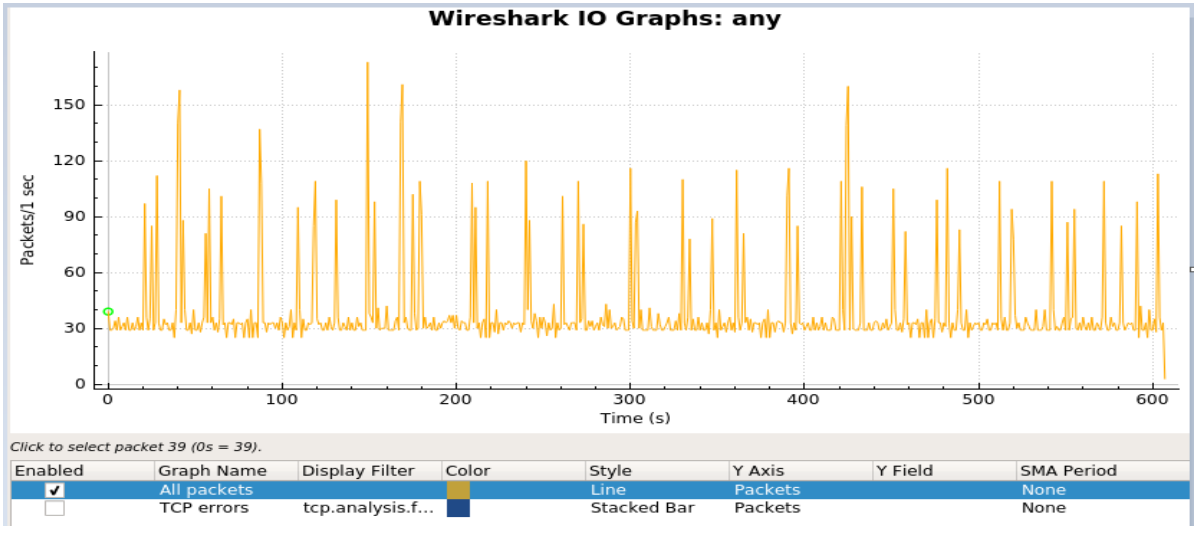


Figure 24: relation of Packet per second and time for SDN based network

4.5. Discussion

The discussion is used to highlight the importance of the study and describe the limitations of the study and implications for future research. The speech in this chapter expressions at how Mininet was able to applied in seamless MPLS supported on SDN with the help of OVS and Open Flow.

4.6. Discussion of results

The evaluation of the result simulated on the Wireshark simulation tool that the proposed SDN coupling seamless MPLS is the best traffic engineering method to create suited to the dynamic computing and storage needs of today's data centers, campuses, and carrier environments. This study has verified what happens if we send an ICMP request from H1 to H4. In this case, When the first packet travels from one host to another, the host sends an ICMP request to discover the Access node then The Access node forwards the packet to the controller, the controller examines the label, looks up the destination address that is associated to that label and chooses a new label, stores the relation between the label and the source and destination addresses.

Moreover, the simulation analysis addressed SDN Firewalls which are a major element of network security systems with the ability to block network data traffic flows according to pre-defined rules. Here, proved that our SDN system control and manage the flow of data that comes from various sources into the network and functions on the principle of preconfigured rules.

Finally, the simulation analysis addressed Packet per second in SDN supported seamless MPLS is higher than not SDN supported seamless MPLS, and also SDN supported seamless MPLS networks provides better performance in transmitting more packet with less time than not SDN supported seamless MPLS.

4.7. Challenges of implementation

Understanding SDN, in computer networking was a difficult task. In addition, considerate the act and use of the different components like OVS, Open Flow message structure and POX controller operation was a challenged, as well Due to memory restrictions of personal computers, it is not possible to power on more than seven nodes in simulation surroundings

and extra routers for redundancy purposes are used individual in the core network domain. But it should be recognized that growing number of routers for testing and investigation will not change the overall result.

4.8. Summary

In this chapter, we discussed in detail about mininet simulation tool and how it works to resolve existing problem by simulated different simulations. It includes checking connectivity b/n hosts, simulation of implementing ICMP packet, simulation of implementing firewall. Finally, we have discussed improvement of the proposed algorithm, and how researcher questions are answered. In the next chapter, we discuss conclusion and future work.

5. Conclusion and Recommendations

5.1. Conclusion

To summarize, we draw the next conclusions from our SDN based seamless MPLS network simulation. We accomplished the preliminary goal of confirming our architectural ideas of familiarizing the SDN based seamless MPLS networks. We presented that we can implement the basic tasks of generating and maintaining LSP in an Open Flow enabled seamless MPLS, we can also provide higher-level services such as SDN firewall by writing applications in the control-plane.

We confirmed the simplicity of our method when related to more traditional approaches seen today. Our work of the traffic-engineering application together with all the features described in above Section, these study proposed and applied a routing that is based on seamless MPLS to send traffic on an SDN background. As verified by the research, the complex duty of configuring seamless MPLS has been accomplished in the less cost, the study has emphasized and underlined from the carrying out that permitting central connectivity, manageability and control of the network creates design and troubleshooting of the network in the possible task.

Finally, we concluded that Any service provider in our country including Ethiopians telecommunication, can implement Seamless MPLS to incorporate the separated network domains such as mobile backhaul and IP core networks into only one MPLS domain with minimum cost and simplified management to network engineer and providing easy and flexible service provision and hence increases customer satisfaction.

5.2. Future Work

Even if this thesis has been done all the objectives set in Section one, there are some concerns to be covered in the future. These issues are:

The researcher concentrated on only ICMP packets for the demonstration. However, future work should study and implement a state where extra packet type is isolated and given altered routes at the same time seamless MPLS being attached to these packet.

In this study the researcher, simulations have been run in static scenarios with one network topology designs. It would be exciting to study the performance of the same parameters by varying the networks topology. The interoperability SDN constituents must be secure by figuring out the variations between different developers.

Study and spread SDN based Seamless MPLS solution to ethio telecom networks. Primary it is better to categorize kinds of router used in end-to-end IP network. The access devices have to be tested if they can provision MPLS and Seamless MPLS. Then the numbers of distinct MPLS domains used in the company will be known.

References

Black, U. D. (2002). MPLS and label switching networks. Prentice Hall PTR.

García-Dorado. (2012) J. L., Finamore, A., Mellia, M., Meo, M., & Munafo, M. Characterization of isp traffic: Trends, user habits, and access technology impact. IEEE Transactions on Network and Service Management, 9(2), 142–155.

Gupta, M., Sommers, J., & Barford, P. (2013). Fast, accurate simulation for SDN prototyping. In Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (pp. 31–36). ACM.

Habtamu Kumera. (*Dec* 21, 2018) *impact* on *QoS* parameters is analyzed by using two scenarios; MPLS with multiple domains and *Seamless MPLS* integrating three domains

Le Faucheur, F. (1998). IETF multiprotocol label switching (MPLS) architecture. In ATM, 1998. ICATM-98., 1998 1st IEEE International Conference on (pp. 6–15). IEEE.

Lee, H., Hwang, J., Kang, B., & Jun, K. (2000). End-to-end QoS architecture for VPNs: MPLS VPN deployment in a backbone network. In Parallel Processing, 2000. Proceedings. 2000 International Workshops on (pp. 479–483). IEEE.

(Original work published 2013). Mininet. Retrieved from

Nunes, B.; Mendonca, M.; Nguyen, X.; Obraczka, K.; Turletti, T., *A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks*, Communications Surveys & Tutorials, IEEE, vol. PP, no.99, pp.1,18

Ould-Brahim, H. (2012, February). Border gateway protocol procedures for multi-protocol label switching and layer-2 virtual private networks using Ethernet-based tunnels. Peter Croy, *LTE Backhaul Requirements: A Reality Check*, Aviat Networks, 2011

Davie, B. S., & Rekhter, Y. (2000). MPLS: technology and applications. Morgan Kaufmann Publishers Inc

Trestian, R., Muntean, G.-M., & Katrinis, K. (2013). MiceTrap: Scalable traffic engineering of datacenter mice flows using Open Flow. In Integrated Network Management (IM 2013), 2013

White paper, *Software-Defined Networking: The New Norm for Networks*, Open Network Foundation, April 2012

Quoitin, S. Uhlig, C. Pelsser, L. Swinnen, and O. Bonaventure, "Inter-domain traffic engineering with BGP," IEEE Communications Magazine Internet Technology Series, vol. 41, no. 5, May 2003.

Singh, Y. K. (2006). Fundamental of research methodology and statistics. New Age International

Juniper Networks, Inc., "Building Multi-Generation Scalable Networks with End-to-End MPLS," CA 94089 USA, White paper, 2000452-001-EN, Jan. 2012.

Openflow Tutorial on Mininet. (2018). Mininet. Retrieved from https://github.com/mininet/openflow-tutorial (Original work published 2013)

Open vSwitch Documentation — Open vSwitch 2.9.90 documentation. (n.d.). Retrieved June 6, 2018, from http://docs.openvswitch.org/en/latest/

Pfaff, B., Pettit, J., Koponen, T., Jackson, E. J., Zhou, A., Rajahalme, J., … Shelar, P. (2015). The Design and Implementation of Open vSwitch. In NSDI (pp. 117–130).

Shalimov, A., Zuikov, D., Zimarina, D., Pashkov, V., & Smeliansky, R. (2013). Advanced study of SDN/OpenFlow controllers. In Proceedings of the 9th central & eastern european software engineering conference in russia (p. 1). ACM.

Zhao, J., Hammad, E., Farraj, A., & Kundur, D. (2016). Network-aware QoS routing for smart grids using software defined networks. In Smart City 360° (pp. 384–394). Springer.

Yu, T.-F., Wang, K., & Hsu, Y.-H. (2015). Adaptive routing for video streaming with QoS support over SDN networks. In Information Networking (ICOIN), 2015 International Conference on (pp. 318–323). IEEE.

Fernandez Sánchez, (2015). Software-Defined Networking and OpenFlow

Ahn, G., & Chun, W. (2000). Design and implementation of MPLS network simulator supporting LDP and CR-LDP. In Networks, 2000.(ICON 2000). Proceedings. IEEE International Conference on (pp. 441–446). IEEE.

Chengchen Hu; Chunming Wu; Wei Xiong; Binqiang Wang; Jiangxing Wu; Ming Jiang, (June 2011) *On the design of green reconfigurable router toward energy efficient internet*, Communications Magazine, *IEEE* , vol.49, no.6, pp.83,87,

Z. Li, K. Lu, (March 2017) "Inter-SDN (SDNi) in Seamless MPLS for Mobile Backhaul Network," draft li-rtgwg-sdni-seamless-mpls-mbh-00 (work in progress).

Nokia, (Oct. 2016) "Evolving to end-to-end MPLS architectures," Finland, White paper, SR1610000967EN.

N. Leymann,(June 2014) "Seamless MPLS Architecture," draft-leymann-mpls-seamless-mpls-07(work in progress).

M. Boucadair and P. Morand,(October 2005) "A solution for providing inter-as mpls-based qos tunnels," Draft-boucadair-pce-interas-01.txt (work in progress).

R. Atkinson, E. S.(August 2004) Floyd, and Internet Architecture Board, "IAB Concerns and Recommendations Regarding Internet Research and Evolution," RFC 3869, DOI 10.17487/RFC3869,

Nunes, B.; Mendonca, M.; Nguyen, X.; Obraczka, K.; Turletti, T., *A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks*, Communications Surveys & Tutorials, IEEE , vol.PP, no.99, pp.1,18

Handigol, N.; Seetharaman, S.; Flajslik, M.; McKeown, N.; Johari, R., (, 2009) *Plug-n-serve: Load-balancing web traffic using OpenFlow*, ACM SIGCOMM Demo

Wang, R.; Butnariu, D.; Rexford, J.,(2011) *Openflow-based server load balancing gone wild*, In Workshop of HotICE, volume 11,

Nayak, A.K.; Reimers, A.; Feamster, N.; Clark, R.,(2009) *Resonance: Dynamic access control for enterprise networks*, In Proceedings of the 1st ACM workshop on Research on enterprise networking, pages 11–18. ACM,

Reitblatt, M.; Foster, N.; Rexford, J.; Schlesinger, C.; Walker, D., (2012) *Abstractions for network update*, In Proceedings of the ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication, SIGCOMM

Inter-datacenter WAN with centralized TE using SDN and OpenFlow, In Open Networking Summit, April 2012

Jain, S.; Kumar, A.; Mandal, S.; Ong, J.; Poutievski, L.; Singh, A.; Venkata, S.; Wanderer, J.; Zhou, J.; Zhu, M., *et al. B4:(* 2013) *Experience with a globally-deployed software-defined WAN*, In Proceedings of the ACM SIGCOMM conference on SIGCOMM, pages 3–14. ACM,

Suresh, L.; Schulz-Zander, J.; Merz, R.; Feldmann, A.; Vazao, T., (, 2012) *Towards programmable enterprise WLANs with Odin*, In Proceedings of the first workshop on Hot topics in software-defined networks, HotSDN '12, pages 115–120, New York, NY, USA, ACM

Optical transport working group OTWG, In Open Networking Foundation ONF, 2013

In May 2006https://en.wikipedia.org/wiki/Wireshark

Foundation, O.N.: Openflow switch specification version 1.3.1. Tech. rep., Open Networking Foundation (September 2012)

APPENDIXES

Appendix A: Topology design Script

```
from mininet.log import setLogLevel, info

def treeTopo():
    net = Mininet( controller=RemoteController )

    info( '*** Adding controller\n' )
    net.addController('c0')

    info( '*** Adding hosts\n' )
    h1 = net.addHost( 'h1', ip='10.0.0.1', mac='00:00:00:00:00:01' )
    h2 = net.addHost( 'h2', ip='10.0.0.2', mac='00:00:00:00:00:02' )
    h3 = net.addHost( 'h3', ip='10.0.0.3', mac='00:00:00:00:00:03' )
    h4 = net.addHost( 'h4', ip='10.0.0.4', mac='00:00:00:00:00:04' )

    info( '*** Adding switches\n' )
    s1 = net.addSwitch( 's1' )
    s2 = net.addSwitch( 's2' )
    s3 = net.addSwitch( 's3' )
    s4 = net.addSwitch( 's4' )
    s5 = net.addSwitch( 's5' )
    s6 = net.addSwitch( 's6' )
    s7 = net.addSwitch( 's7' )

    info( '*** Creating links\n' )
    net.addLink( h1, s3 )
    net.addLink( h2, s4 )
    net.addLink( h3, s6 )
    net.addLink( h4, s7 )

    root = s1
    layer1 = [s2,s5]
    layer2 = [s3,s4,s6,s7]
```

```
    for idx,ll in enumerate(layer1):
        net.addLink( root,ll )
        net.addLink( ll, layer2[2*idx] )
        net.addLink( ll, layer2[2*idx + 1] )

    info( '*** Starting network\n')
    net.start()

    info( '*** Running CLI\n' )
    CLI( net )

    info( '*** Stopping network' )
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    treeTopo()
```

Appendix B: POX Application Code

```
from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.revent import *
from pox.lib.recoco import Timer
from collections import defaultdict
from pox.openflow.discovery import Discovery
from pox.lib.util import dpid_to_str
import time
from pox.lib.addresses import EthAddr



log = core.getLogger()
switches = {}
mac_map = {}
path_map = defaultdict(lambda:defaultdict(lambda:(None,None)))
waiting_paths = {}

# Waiting path.  (dpid,xid)->WaitingPath
waiting_paths = {}

# Time to not flood in seconds
FLOOD_HOLDDOWN = 8

# Flow timeouts
FLOW_IDLE_TIMEOUT = 15
FLOW_HARD_TIMEOUT = 36

# How long is allowable to set up a path?
PATH_SETUP_TIME = 9

rules = [['00:00:00:00:00:01','00:00:00:00:00:02'],['00:00:00:

def _calc_paths ():
```

```
"""
Essentially Floyd-Warshall algorithm
"""

def dump ():
  for x in sws:
    for y in sws:
      b = path_map[x][y][0]
      #b = adjacency[x][y]
      if b is None: b = "*"
      print(b, end=' ')
    print()

sws = switches.values()
path_map.clear()
for z in sws:
  for y,port in adjacency[z].items():
    if port is None: continue
    path_map[z][y] = (1,None)
  path_map[z][z] = (0,None) # distance, intermediate

#dump()

for z in sws:
  for x in sws:
    for y in sws:
      if path_map[x][z][0] is not None:
        if path_map[z][y][0] is not None:
          # x -> z -> y exists
          xzy_dist = path_map[x][z][0]+path_map[z][y][0]
          if path_map[x][y][0] is None or xzy_dist < path_map[x][y][0]:
            # x -> z -> y is better than existing
            path_map[x][] = (xzy_dist, z)
```

```
  #print "--------------------"
  #dump()


def _get_raw_path (src, dst):
  """
  Get a raw path (just a list of nodes to traverse)
  """
  if len(path_map) == 0: _calc_paths()
  if src is dst:
    # We're here!
    return []
  if path_map[src][dst][0] is None:
    return None
  intermediate = path_map[src][dst][1]
  if intermediate is None:
    # Directly connected
    return []
  return _get_raw_path(src, intermediate) + [intermediate] + \
        _get_raw_path(intermediate, dst)


def _check_path (p):
  """
  Make sure that a path is actually a string of nodes with connected ports

  returns True if path is valid
  """
  for a,b in zip(p[:-1],p[1:]):
    if adjacency[a[0]][b[0]] != a[2]:
      return False
    if adjacency[b[0]][a[0]] != b[1]:
      return False
  return True
```

```
def _get_path (src, dst, first_port, final_port):
  """
  Gets a cooked path -- a list of (node,in_port,out_port)
  """
  # Start with a raw path...
  if src == dst:
    path = [src]
  else:
    path = _get_raw_path(src, dst)
    if path is None: return None
    path = [src] + path + [dst]

  # Now add the ports
  k = []
  in_port = first_port
  for s1,s2 in zip(path[:-1],path[1:]):
    out_port = adjacency[s1][s2]
    k.append((s1,in_port,out_port))
    in_port = adjacency[s2][s1]
  k.append((dst,in_port,final_port))

  assert _check_path(r), "Illegal path!"

  return k


class WaitingPath (object):
  """
  A path which is waiting for its path to be established
  """
  def __init__ (self, path, packet):
    """
    xids is a sequence of (dpid,xid)
    first_switch is the DPID where the packet came from
    packet is something that can be sent in a packet_out
    """
```

```
    self.expires_at = time.time() + PATH_SETUP_TIME
    self.path = path
    self.first_switch = path[0][0].dpid
    self.xids = set()
    self.packet = packet

    if len(waiting_paths) > 1001:
      WaitingPath.expire_waiting_paths()

  def add_xid (self, dpid, xid):
    self.xids.add((dpid,xid))
    waiting_paths[(dpid,xid)] = self

  @property
  def is_expired (self):
    return time.time() >= self.expires_at

  def notify (self, event):
    """
    Called when a barrier has been received
    """
    self.xids.discard((event.dpid,event.xid))
    if len(self.xids) == 0:
      # Done!
      if self.packet:
        log.debug("Sending delayed packet out %s"
                  % (dpid_to_str(self.first_switch),))
        msg = of.ofp_packet_out(data=self.packet,
            action=of.ofp_action_output(port=of.OFPP_TABLE))
        core.openflow.sendToDPID(self.first_switch, msg)

      core.l2_multi.raiseEvent(PathInstalled(self.path))

class SDNFirewall (EventMixin):

    def __init__ (self):
        self.listenTo(core.openflow)

    def _handle_ConnectionUp (self, event):
        for rule in rules:
            block = of.ofp_match()
            block.dl_src = EthAddr(rule[0])
            block.dl_dst = EthAddr(rule[1])
            flow_mod = of.ofp_flow_mod()
            flow_mod.match = block
            event.connection.send(flow_mod)

def launch ():
    core.registerNew(SDNFirewall)
```