2021-10-21

# IMPROVING THE PERFORMANCE OF SOFTWARE DEFINED NETWORKS IN MULTI-METRICS PERSPECTIVE

BELAYNEH, TESHOME


**BAHIR DAR UNIVERSITY**

**BAHIR DAR INSTITUTE OF TECHNOLOGY**

**SCHOOL OF GRADUATE STUDIES**

**FACULTY OF COMPUTING**

# IMPROVING THE PERFORMANCE OF SOFTWARE DEFINED NETWORKS IN MULTI-METRICS PERSPECTIVE

**BY:**

**BELAYNEH TESHOME**

**October 21, 2021**

**Bahir Dar, Ethiopia**


**BAHIR DAR UNIVERSITY**

**BAHIR DAR INSTITUTE OF TECHNOLOGY**

**SCHOOL OF GRADUATE STUDIES**

**FACULTY OF COMPUTING**

# IMPROVING THE PERFORMANCE OF SOFTWARE DEFINED NETWORKS IN MULTI-METRICS PERSPECTIVE

**BY:**

**BELAYNEH TESHOME**

**A thesis is submitted to the school of graduate studies of Bahir Dar Institute of Technology, Bahir Dar University in Partial Fulfillment of the Requirements for the Degree of Master of science in Information Technology in Faculty of Computing.**

**Advisor: Mekuanint A. (PhD)**

**October 21, 2021**

**Bahir Dar, Ethiopia**

## DECLARATION

This is to certify that the thesis entitled "**Improving the Performance of Software Defined Networks in Multi-Metrics Perspective**" submitted in partial fulfillment of the requirements for the degree of Master of Science in Information Technology under Faculty of Computing, Bahir Dar Institute of Technology, is a record of original work carried out by me and has never been submitted to this or any other institution to get any other degree or certificates. The assistance and help I received during the course of this investigation have been duly acknowledged.

| Belayneh Teshome | | 22/10/2021 |
|---|---|---|
| Name of student | Signature | Date |



i

**BAHIR DAR UNIVERSITY**

**BAHIR DAR INSTITUTE OF TECHNOLOGY**

**SCHOOL OF GRADUATE STUDIES**

**FACULTY OF COMPUTING**

## Approval of Thesis for Defence Result

I hereby confirm that the changes required by the examiners have been carried out and incorporated in the final thesis.

Name of student Belayneh Teshome Signature [signature] Date 22/10/2021

As member of the board of the examiners, we examined this thesis entitled **"IMPROVING THE PERFORMANCE OF SOFTWARE DEFINED NETWORKS IN MULTI-METTRICS PERSPECTIVE"** by Belayneh Teshome. We hereby certify that the thesis is accepted for fulfilling the requirement for the award of the degree of Masters of Science in **"Information Technology".**

**Board of Examiners**

| | | |
|---|---|---|
| Name of Advisor<br>Mekuanint A (PhD) | Signature<br>[signature] | Date<br>22/10/2021 |
| Name of External Examiner<br>Sosina Mengistu (Ph.D.) | Signature<br>[signature] | Date<br>19/10/2021 |
| Name of Internal Examiner<br>Yirga Y. (Ph.D.) | Signature<br>[signature] | Date<br>22/10/2021 |
| Name of Chairperson<br>Abinew Ali (Ass.Prof) | Signature<br>[signature] | Date<br>October 21, 2021 |
| Name of Chair Holder<br>Dereje w Lake | Signature<br>[signature] | Date<br>22/10/2021 |
| Name of Faculty Dean<br>Asegahegn E. | Signature<br>[signature] | Date<br>12-02-2014 E.C |

Faculty Stamp

## ACKNOWLEDGEMENT

First and for most, I would like to thank the almighty God for giving me the strength to do whatever it is expected to be done by me. Secondly, I am grateful to Dr. Mekuanint A. for his recommendations, readiness and open-mindedness to support me in any situations. Thirdly, I would like to thank Bahir Dar University for offering full sponsorship to learn my degree of masters. Finally, I would like to say thank you my beloved family and everyone who strengthen me in different ways and for being with me throughout the accomplishment of this thesis.

# LIST OF ABBREVIATIONS

| | |
|---|---|
| API | Application program interface |
| BPS | Bit per second |
| CBR | Constant bit rate |
| ECMC | Equal cost multipath |
| FEC | Forwarding equivalence class |
| GBITS | Gigabits |
| ID | Identifier |
| GBYTE/SEC | Gigabyte per second |
| GMPLS | Generalized multiprotocol label switching |
| GNS3 | Graphical network simulator-3 |
| GUI | Graphical user interface |
| ICMP | Internet control message protocol |
| IoT | Internet of things |
| IP | Internet protocol |
| IPFIX | IP flow information export |
| ISP | Internet service provider |
| KB/S | Kilobits per second |
| MAC | Media access control |
| MB/S | Megabits per second |
| MBYTE/SEC | Megabyte per second |
| MPLS | Multiprotocol label switching |
| MPTCP | Multipath transmission control protocol |
| MS | Millisecond |

| | |
|---|---|
| NOX | Network operating system |
| OFGW | Openflow gateway |
| ONOS | Open network operating system |
| OVS | Openvswitch |
| QoE | Quality of experience |
| QoS | Quality of services |
| RIOA | Real-time Online Interactive Applications |
| RTT | Round trip time |
| SDN | Software defined network |
| SR-MPLS | Segment routing multiprotocol label switching |
| TCAM | Ternary content accessible memory |
| TCP | Transmission control protocol |
| TTL | Time to live |
| UDP | User datagram protocol |
| UNI | User network interface |
| VLAN | Virtual local area network |
| VM | Virtual machine |
| VPN | Virtual private network |
| WAN | Wide area network |

# TABLE OF CONTENTS

# LIST OF TABLES

# TABLE OF FIGURES

# LIST OF FIGURES IN THE APPENDIX

# ABSTRACT

A software defined network (SDN) is an emerging technology that enables open networking. It is a network that is largely made up of virtual devices and is managed by software. The major advantage of SDN is that it simplifies network administration. Like traditional network, Software defined networks is characterized by technical goals. Among technical goals, performance is the hot research issue. There has been a lot of works done on the performance of SDN over the years, but still, there is a gap in considering multiple parameters and their dependencies so, this thesis takes these two things into account. We employed policy-based packets classification and multiprotocol label switching (MPLS) to enhance the overall performance of SDN. We applied Policy-based packets classification for handling high level and medium level packets on the control plane, and we used MPLS to handle low level packet flows on the data plane because it is an efficient method to reduce the loads of the controller. We used distributed controllers instead of single controller in order to distribute the loads and to avoid single point failure. With these facts, we first classify users as low level, medium level, or high level users, and then we let MPLS handle low level packet flows, while using policy-based packets classification to handle the remaining high level and medium level packet flows. We also applied bandwidth isolation so as to improve bandwidth utilization across the entire network. Our approach decreased jitter by **0.0124ms** on average, improves bandwidth utilization by **0.013%,** and our approach also reduced packet loss to **9.5**%. From these findings, we concluded that integrating MPLS with SDN is an efficient method to improve the overall performance of software defined networks compared with formal SDN.

**Key words:** Software defined network, multiprotocol label switching, policy-based routing, bandwidth isolation, packet loss, throughput, latency, jitter

# Chapter One: Introduction

## 1. 1. Background

The desire of having programmable network began in 1990s (Burno, etal., 2014); two groups actively involved in the process of making programmable network practical, such as open signaling tried to make ATM programmable in 1995 and active networking group appeared in the mid-1990s with the motive of exposing resources of network nodes via API and it enables network operators to monitor nodes by running arbitrary codes (Xenofon, et al., 2015)

SDN is a network that is primarily made up of virtual devices and managed by software. The concept of software defined networking is introduced in 2008 (Larry, 2018) with the idea of programmable data plane by decoupling the control and data plane. In traditional networks, data plane and control plane are tightly coupled, making the network management task is difficult for network administrators, but in SDN, control plane and data plane are loosely coupled, making it preferable over traditional network.

Software defined network is a network based on network function virtualization having the aim of reducing the complexity of network management and control system (Bhat, et al. 2015). Since its origin, it has come with many opportunities, like network function virtualization, internet of things (IoT), virtual private network (VPN), reduction of network hardware costs and simplifying managements (Latif, et al., 2019). SDN is also the solution for cloud computing since it has network function virtualization opportunities (Xenofon, et al., 2015). SDN has three major advantages these are: cost reduction, complexity reduction and boosting network operation. Regarding reducing the cost, SDN allows us to use virtual devices and it also provides network function virtualization so, we can reduce the cost that we waste for purchasing network devices.

When we see the complexity, the controller is aware of the entire network status through global view, and it is able to dynamically configure and reconfigure the network routers and switches even services added to the network, as a result, it will be easy for users and network administrators. When we see the advantage of boosting the network operation, the dynamic configuration of servers and other hosts or any network devices, makes the entire processes in the network to run faster and faster. Nowadays, software defined networks are becoming popular and applicable in many organizations. SDN also eliminates the need of firewalls and Intrusion Detection Systems (IDS) in the Network topology (Burno, et al., 2014).

The principle in SDN is the network managed by the software (Bhat, et al. 2015); it has three parts these are: application plane, control plane and data plane. The control plane is responsible for managing the entire network, and the data plane is responsible for flow of information and storage. To do so, the control plane and the data plane are decoupled, and the controlling and data forwarding functions run in the separate devices connected via API i.e. open flow protocol that manages the communication between the control plane and the forwarding plane.

The control plane consists of one or more controllers that run open flow protocols, and the controllers are responsible to control the entire network with global view. The data plane consists of openvswitches and hosts capable of forwarding and receiving packets. The communication paths are set by the controllers and they are found on the data plane to do actual packet transmission.

SDN has APIs that enables users and developers communicate with it; those APIs can be classified in to two i.e. south bound and north bound. South bound interface is the interface between the data plane and the control plane, and lets end users or the data plane communicate with the control plane whereas the north bound is the interface is the interface between the applications that run on the top of the controller and the controller, or it is the interface between the developers and the control plane.

SDN is now being applicable in small scale, and there is infrastructure for deploying it. Now a day, network function virtualization paves the way for SDN to be practical i.e.

there are a lot of virtualization technologies supporting open networking including NS2, NS3, mininet (Sam, 2020), etc. Like traditional networking and other networking options, SDN shares the features that describe: its capacity to grow, satisfy end users, and other related features. Network is characterized by the technical goals, such as scalability, performance, manageability, security, availability, reliability, affordability, etc.

The network is expected to be: scalable, manageable, secured highly performing, available with minimal fault tolerance by having redundant devices connected one another, highly secured having sophisticated security mechanisms including both hardware and software security ensuring mechanisms, it should be highly scalable for future expansion, and it should have hierarchical network structure, it should be reliable meaning end users should not worry about loss of data, and the network should run as it is expected to achieve certain objectives, it should also be easy to own the network, and it must be economical.

The most interesting part in this study, among the technical goals of the network is ensuring quality of service. Ensuring highly performing network service is what is expected by end users from service providers, and even from enterprise networks. So, network operators are always looking for solutions to overcome performance bottlenecks (Xenofon, et al., 2015). Every network operator should run network audits to identify problems related with quality of services and other issues concerned with the technical goals. This mechanism helps to optimize network functions in such a way that it satisfies end users requirement, and even be efficient. To overcome the problems, feedback from end users is necessary: we may not be aware of what is going on sometimes so, feedback from end users helps us to easily tackle performance bottlenecks.

We also need to hire qualified network administrator and supporting staffs capable of developing sophisticated algorithms and approaches to enhance network performance and monitoring network operations. Particularly in software defined network, there exists a performance problem due to the fact that it is based on virtual environment and the limitation of TCAM to store flow rules established by the controller while forwarding the

packet across the entire network. Not only this, there is also significant load on the controllers since packet forwarding decision is made by the controllers.

Although SDN is being applicable in many aspects and it has such the above advantages, it has a lot of problems to be addressed, and those problems need hardworking of researchers in the area; its growth is infant this implies it has many holes to be filled for example, the performance issue is the hot issue to be addressed (Lawal, et al., 2017). The quality of service (QoS) is what users need to be addressed permanently since it is a headache for users.

In this research work, we have used both policy-based packet classification and multiprotocol label switching (MPLS) to improve resource utilization and the performance of SDN due to the fact that MPLS is an efficient method of improving resource utilization by prioritizing packets in such a way that packet can be handled based on QoS requirement. Concerning with policy-based-packet classification, we have implemented policies on distributed controllers by programming the controllers using API provided by them according to the needs of the owner of the network.

## 1. 2. Statements of the Problem

The performance of software defined networks is still questioning, and it needs hardworking of researchers; it is among the hot issues to be explored. In many dimensions, there are certain issues need to be addressed for example, rather than querying the data plane for statistics periodically, one can use IP flow information export (IPFIX) protocol to export network statistics and analyze QoS, and then he/she can monitor the flow(Faisal, 2018). Many other QoS metrics also need to be explored, like switch capacity and bandwidth isolation. Exploring the relationship between QoS metrics is also among the very interesting issues for composite parameter QoS application (Alemayehu, 2019).

Many works have been done on the performance of SDN; the following are what we have taken as inputs for our study. To enhance the performance of SDN, many researchers have proposed different approaches by specifying their target areas, such as

(Tootoonchian, et al., 2012), have published a paper focusing only on the performance of controllers. The authors investigated the performance of controllers, such as beacon NOX-MT, NOX and maestro interms of throughput and response time, and they found that NOX-MT achieved better performance compared with the three others. However, they did not consider data plane features, like bandwidth isolation switch capacity, and even the impact of each performance parameter on the data plane.

(Maim, et al., 2019), have published a paper focusing only on link failure scenario i.e. they have focused on latency and throughput in case of link failure, and then the authors tried to measure the effect of using paths other than the shortest path on latency and throughput by shutting down certain links or switches connecting hosts. However, they did not consider jitter, packet loss and bandwidth utilization.

(Hossain, et al., 2018), have done on enhancing and measuring the performance of SDN with three interconnected networks connected to one central controller via one subnet, their topology looks mesh topology. The authors investigated response time and the throughput with different packet schedulers; however, their work did not consider other independent variables, like switch capacity and bandwidth isolation.

There is also a work done on ensuring Quality of Service to Selected network flows using status monitoring and QoS based path setup modules developed with floodlight controller; the author has grouped the packet flows into three groups, like critical with QoS, with no QoS, and QoS non-critical(Faisal, 2018). Nevertheless, this work lacks considering the inter-relationship among QoS metrics and other traffic flow characteristics.

(Alexandre, et al., 2019), have published a paper on Providing QoS to High Performance Distributed Applications. The authors' main idea is resource allocation by having service requester and service provider applications, and then the controller performs resource allocation in accordance with network conditions and traffic flows, but their work focuses on single metrics which is latency reduction i.e. they did not consider other QoS metrics.

(Manmeet, et al., 2018), have done on dynamic end-end link capacity estimation of SDN using open source tools, and the controller with the help of port statistics, however, this

work did not consider the issue of excessive traffic flow reduction to optimize the flows in such a way that the flow becomes manageable and the impact packet flows on offloading the controllers becomes minimal.

(Jacob, 2015), has done his Msc thesis on performance management of SDN using dynamic load balancer targeting reducing end to end packet loss and increasing the throughput by assuming that the load balancer has exact real-time knowledge of link capacity. The author presented his result in comparison with static load balancer and optimal load balancer, and then he showed that dynamic load balancer is better than static load balancer in packet loss, there are also some occasions for dynamic load balancer achieving better throughput compared with static load balancer. However, his assumption of the load balancer has real-time knowledge of link capacity is not always true and even bandwidth utilization is not efficient.

There is also a work that investigates the latencies that openvswitches encounter while setting up flow path (Khalili, et al., 2018). The authors used two approaches namely, 1) reducing the update time of every switch, but due to the fact that openvswitches do have minimal updating time, this will not incur severe latency in flow setup. 2) Reducing the number of switches updating at a time during path setup, these two techniques enable them to manage flows accordingly, however, the approaches would increase error rate throughout the entire network, and they didn't even consider other traffic flow characteristics. Previous works lack considering multiple metrics and other independent variables, such as switch capacity and bandwidth isolation. With this fact, we formulated the following research questions to be answered at the end of this thesis.

- How the interdependency of QoS metrics affect the performance of SDN?
- How much effect does bandwidth isolation has on bandwidth utilization?
- What influences do switch capacity and queue size have on the performance of SDN?
- How the integration of MPLS with policy-based packet classification reduces latency?

## 1. 3. General and Specific Objectives

### 1. 3. 1. General Objective

The ultimate objective of the research is enhancing the performance of SDN in multi-metrics perspective using distributed controllers.

### 1. 3. 2. Specific Objectives

- ✓ To investigate the effect of distributed controllers on reducing latency
- ✓ To investigate the relationship between QoS metrics
- ✓ To analyse the effect of bandwidth isolation on bandwidth utilization
- ✓ To analyse the outcome of using IPFIX protocol on reducing excessive traffic flow
- ✓ To analyse the effect of policy-based packet classification on bandwidth utilization
- ✓ To investigate the quantitative effect of access control lists on reducing latency

## 1. 4. Significance of the Research

The output of the study can be vital in different aspects, i.e. it is a good inputs for other researchers who are interested in this area to know the state of the art of software defined networks, and it can be used to know the gap and future works to be done later; it paves the way for further investigation and improvement using different approaches.

The study indicates the techniques and protocols that have significant impacts on the performance of software defined networks; this study improves the bandwidth utilization throughout the entire network significantly, and it addresses the performance problems by enhancing the throughput, reducing end to end latency and decreasing the response time using sophisticated policy-based packet classification and MPLS.

This work also investigates the interdependency of QoS metrics that affect the entire performance of the network. The relationships between many QoS metrics have meaningful effect on the performance of SDN so, this work investigates their

dependency. Generally speaking, the promising contribution of this thesis is addressing the overall performance problem observed in software defined networks.

## 1. 5. Scope of the Research

The thesis is bounded by the area of improving the overall performance of SDN. It includes proposing a network topology with distributed controller architecture, improving the bandwidth utilization, reducing the latency and the response time, increasing the throughput by enabling packet processing on both the data plane and the control plane instead of packets being processed only on the control plane, investigating the relationship between QoS metrics, and running experiments & evaluating the performance of the entire architecture. To do so, we setup mininet emulator installed on Ubuntu enabled virtual machines that runs POX, open-daylight and ONOS openflow controllers integrated with GNS3 simulators.

### 1. 5. 1. Limitations of the Thesis

To conduct experiments, we need at least three hardware devices interconnected one another, and capable of running SDN controllers, but we couldn't do that due to the fact that we have limited resources and budget. As a result, we decided to work with virtual machines running on a single device. Still, the performance of those virtual machines is limited; as a result, we mainly focused only on the performance of software defined networks in multi-metrics perspective; other related issues are no included.

This thesis does not include the following tasks.

- Preparing test beds for the architecture,
- Evaluating scalability of the architecture
- Improving other resource utilizations except bandwidth

## 1. 6. Thesis layout

In this study, five chapters are included. The first chapter deals with the history, brief introduction of software defined network, the statements of the problem, objective of the study, scope of the thesis and the contribution of the study.

Chapter two deals about the literature review; it includes general view of the performance of the network, the performance of conventional network, the effect of the performance of controllers on the overall performance of SDN, SDN switches and flow tables, northbound and southbound interfaces and it presents related works on the performance of controllers and on the overall performance of SDN.

Chapter three is about the methodologies that we used in our investigation; it includes tools and techniques to collect network statistics, performance measuring metrics, simulation tools, protocols and algorithms, SDN controllers we used and the conceptual model we developed.

Chapter four deals about result and discussions: it includes the result of every scenario and experiment that we carried out, comparison of our model with previous works and general discussions about the overall results. We presented the result of the each experiment interms of graph, table and charts or images.

Chapter five is about the conclusions and recommendations for future works on the basis of the result presented on chapter four. This chapter presents the conclusions we made according to the result we found from every experiment, and it puts directions for future work on the performance of software defined network originating from the result and challenges that we faced throughout the accomplishment of this thesis.

# Chapter Two: Literature Review

## 2. 1. Overview of SDN Architecture

SDN has mainly two parts these are data plane and the control plane.

**The control plane:** The control plane is the core component of software defined network where packet forwarding decision is made by the controllers. It uses a set of flow rules whether to forward or drop the packet. The controllers are responsible for establishing the path between data transmitting nodes in order to ensure successful transmission of the data in the network. To establish the path, the controllers look up the flow tables and or group tables of openvswitches.

**Data plane/forwarding plane:** the data plane is a component of software defined network where actual user data transmission takes place. The data plane is also called forwarding plane, and it contains a set of devices often generalized as switches. The forwarding plane is responsible to carry out the actions, such as packet forwarding following the decision made by the controllers so, the data plane consists of a number of paths for transferring packets. Paths are set by the controller using flow tables or group tables.

**Application plane:** application plane is a repository where business applications can be deployed. Applications can be developed and deployed on the top of the controllers to manipulate network operations by interacting with the controllers through API called northbound interface. Those applications run without the concern of physical network infrastructure, and they are often user specific, multipurpose and open-sources. Applications communicate with the controllers through northbound interface of SDN to perform specific functions. The applications often access network resources to perform specific applications through northbound interfaces.

Unlike southbound bound interface, such as openflow, there is no commonly used communication protocol in the northbound interface (Xenofon, et al., 2015) so, every vendor will have its own customized protocol to establish and facilitate the communication of applications with the control plane. This could result in complexity problem and those applications may not be interoperable.

### 2. 1. 1. SDN Northbound Interface

A northbound interface is an API that allows software applications to be deployed at the top of controller what we call application plane, and it enables the software on the top of the controller to operate without concern of the individual characteristics and situation of the network devices themselves (Astuto, et al., 2014). One of the prominent advantages of this level of abstraction is that network applications can run the network service independently from the underlying physical network.

The services are allowed to host devices in such a way that those hosts are heedless that the network resources they are using are virtual and not the physical ones for which they were originally designed. There is no currently accepted standard for the interaction of the controller with applications. So, each brand of controller will have unique methods to ensure communication with applications. It is very difficult to manage applications having conflicting function even though controllers provide very low level abstraction so, there is a need of high level programming language that translates high level policies into low level ones (Xenofon, et al., 2015).

### 2. 1. 2. SDN Southbound Interface

The southbound interface facilitates the communication between the control plane and the data plane. It is the communication protocol between controllers and openvswitches/ data plane devices. There are many different brands of southbound interfaces, like OVSDB, openflow and etc., but openflow is the most common one (Mittal, 2017; Latif, et al., 2019). Through openflow protocol, the data plane devices commonly openvswitches communicate with the control plane, and the controllers take actions on behalf of flow entries stored on the flow tables of openvswitches. The control plane could

have one or more controllers capable of configuring devices, setting paths or monitoring the entire network with global view. On the top of the control plane, there are business applications running specific tasks by communicating with the controllers with the northbound interface. The following figure illustrates the components/general architecture of software defined networks.

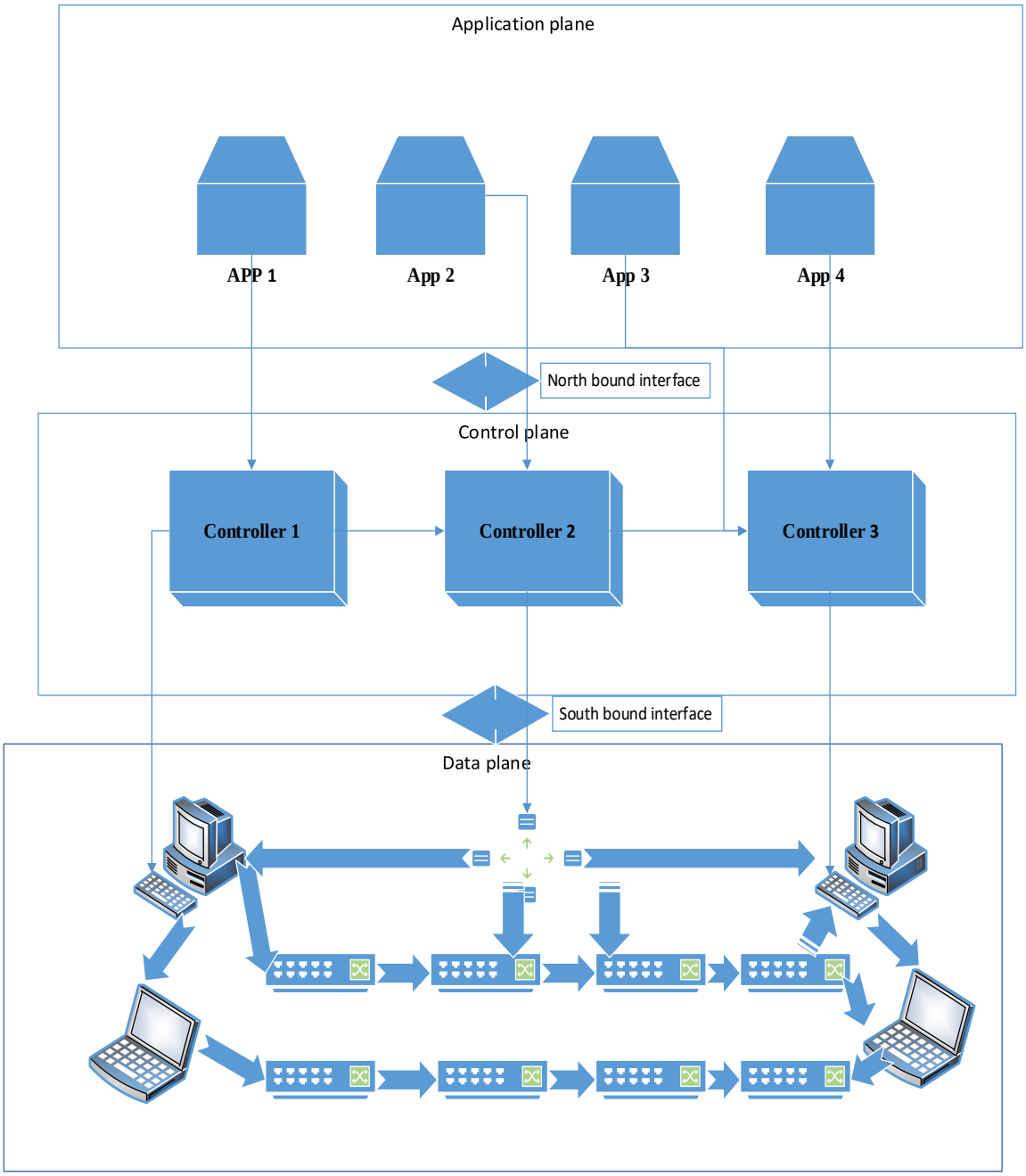


**Figure 2.1: SDN Architecture**

## 2. 2. Overview of Network Performance

The performance of the network is among the technical goals that a network should achieve. Network performance is a quality of service parameter that is used to measure whether it is satisfying users need or not, and whether the network functions are running as they are intended to be. Network performance can be expressed by many parameters. Among the parameters, throughput, bandwidth, delay, packet loss rate, jitter and retransmission rate are common ones. Users need network to run with minimal delay and jitter, enhanced throughput, reduced packet loss and small amount of time required for packet retransmission.

Both in traditional network and SDN, there exists a performance problem so, researchers are working hard to cope with the problems observed in and raised by users. SDN is highly affected by the performance of controllers and other independent parameters. Many researches have been done on the performance of SDN, and those works have remarkable results in many aspects; some researchers focus only on the performance of the controllers, others study only about flow setup delay, some researcher focused on throughput and other researchers used multiple metrics to measure and enhance the performance of software defined networks. However, when we talk about the performance of the network, it should be expressed with multiple metrics, and those technical goals associated with the performance of the network should be considered.

To investigate about the performance of the network, we need to consider all issues associated to it. For example, the hardware, software, operational aspects physical environment conditions, security issues, network management policies, number of queues, etc..

From hardware perspective, the proper functionality of every network device matters on the overall performance of the network i.e. the capacity of the device to handle the number of packets does matter on the flow of packets so, in software defined networks, switch capacity possibly have significant effect on the performance of the entire network. On the basis of the software, we need to take those applications run in the network into account; they will degrade or enhance the performance of the network accordingly. Some

applications may consume much amount of bandwidth and more processing power, whereas others may require less computational power and bandwidth.

Operational aspects of the network have also its own consequence on the performance of the network in such a way that it is related to traffic management tasks. Traffic handling and managing policies have a great impact on flow of packets across the entire network. The other issue is that the queue size. The maximum number of queues or instructions that the packet processor handles at a specific period of time could also degrade the performance of the network since it will offload the controller since it is tightly coupled with the memory capacity of the device.

### 2. 2. 1. The Performance of Controllers

SDN controller is the main component of software defined network responsible for monitoring all parts of the network with the capability of global view to configure services, switches and other end devices (Bhat, et al., 2015). All decisions to forward or drop the packet are made by the controller, and it sets routing paths to route packets to their target. It uses flow tables and group tables to perform its routing decision. Flow table is like routing table in traditional network that contains flow entries to set forwarding path.

The control plane contains one or more controllers i.e. centralized or distributed controller architecture. In centralized architecture, there is only a single controller that controls the entire network, this has advantages in case of it is simple to manage, but this is affected by single point failure, scalability problem (Paliwal, et al., 2018; Mamushaiane, 2019), and it leads the controller to be more burdened. The distributed controller architecture is the one with two or more controllers running on the control plane so, the network will no longer be affected by single point failure, and this architecture solves the problems that centralized controller architecture experiences, however, this leads to complexity.

Distributed controller architecture can be grouped into four categories such as, logically distributed controller approach, physically distributed approach, hierarchically distributed

approach and hybrid controller distribution approach (Mamushaiane, 2019). Physically distributed controller architecture is characterized by physical distribution of controllers on the control plane, and those controllers equally view the entire network. Logically distributed controller architecture is distributed SDN architecture based on the idea of centralized controller architecture where every controller is responsible to manage its own network segment unlike physically distributed controller approach.

Hierarchical distributed controller architecture is known to be the control plane incorporates more than one layer of controllers; in this architecture, lower layer controllers have different burdens with partial view, but the upper layer controller is responsible to manage the entire network with global view so, the upper layer controller is exposed to single point failure like that of centralized controller architecture. Hybrid distributed controller architecture is constructed by taking the advantages of each type of distributed controller architecture to jointly solve the weakness of each type of distributed controller approach, however, this could be very complex for large network.

The performance of controllers has significant effect on overall performance of software defined network (Lawal, et al., 2017). Controllers are being developed by having unique features to boost network operations and packet transmission in the entire network. So, the performance of controllers matter on the overall performance of software defined network. The performance of the controllers can be expressed interms of two parameters i.e. time required to process a single request and the number of input packet request it can handle in per unit time (Paliwal, et al., 2018).

Since packet forwarding decision is made by the controllers, controllers should process packets in minimal time interval, and the maximum capacity that the controller can handle and process input packets per unit of time has a great impact on the overall performance of SDN. (Tootoonchian, et al., 2012), have published a paper on the performance of controllers: they have used customized cbench tool to measure the number of flows setup per second that the controller can handle. The authors mainly focus on flow setup process, and they reason out the setup process is expected source of performance bottleneck. From their experiment, they claimed that NOX-MT achieved

better throughput compared with beacon, maestro and NOX. This demonstrates that the performance of the controller matters on the overall performance of SDN.

### 2. 2. 2. SDN switches

In SDN concepts, data plane devices, such as switches, routers and access points are generalized as SDN switches. Switches contain three basic components these are flow table, secure channel and openflow protocol. Secure channel connects the switches with the controller, whereas openflow establishes the communication with the external controller (open networking foundation, 2017).

In some SDN deployment, switches are able to process/control small packet flows for example, in devoflow, it is a modification of openflow that classifies packet flows as mice flow and elephant flow (Xenofon, et al., 2015). The switch capacity has also significant impact on the overall performance of software defined network because some switches are able to process small amount of packet flows what we call mice flow above and the flow setup latency has a great impact on response time of openvswitches. (Tootoonchian, et al., 2012), also observed that NOX-MT switch applications reduce contention by partitioning MAC address table between pool of hash tables chosen by hash of MAC address.

According to (Lawal, et al., 2017)**,** the capacity of switches to provide flow rules and the rate to setup the flow differs, and then it has significant impact on the overall performance of SDN. The authors quoted the capacity of OVS switches to provide flow rules as 200, 000 with flow setup rate of 42,000 per second and the capacity of HP ProCurve 5406zlwith K.15.10.0009 firmware to support flow rules is only 1500 flow table entries with hardware at 275 rule set up per second. They pointed out the flow rule support of openvswitches is constant.

#### 2. 2. 2. 1. Flow Tables

Flow tables are the fundamental data structures in an SDN device (Astuto, et al., 2014). Flow tables contain many prioritized flow entries. Each entry in the flow table has three fields: i) match fields, ii) instructions governing how the packet should be handled and

iii) counters, this stores the information, like the number of packets and bytes of every flow and the time when a packet of the flow was lastly forwarded (Xenofon , et al., 2015; open networking foundation, 2017).

To forward the packet to the destination, there should be flow rule that matches with the packet in the flow table of openvswitch. Packet matching starts at the first flow table and it continues through other flow tables accordingly; the controller can use openflow to take actions against flow entries, like, delete and update. The action on behalf of packet forwarding depends on matching. Whenever no matching is found in the flow table, the packet may be forwarded to the controller through the Openflow channel, dropped, or the matching may continue to the next flow table.

## 2. 3. Related Works

### 2. 3. 1. The Effect of MPLS on the Performance of Traditional Network

MPLS in traditional network combines the layer two and layer three forwarding techniques to forward packet to the target destination, and it is considered to be layer 2.5 protocols. It uses both the advantages of circuit switching and packet switching techniques to enhance the performance of the network. In traditional network there is a recent work (Alemayehu, 2019), has done his MSc. thesis on improving quality of services on traditional network using segment routing multiprotocol label switching (SR-MPLS), and his approach reduces: the packet loss by 20.4% and the jitter by 16.2%, he pointed out on considering the effect of SR-MPLS on resource utilization and investigating the interrelationship between QoS metrics to be done in the future.

(Adnan, 2000), has done on the effect of MPLS on end-end latency. The author used constrained paths with traffic engineering link state database. In his topology, he took four subnets and three transit networks to evaluate the number of packet on the backbone, and implemented traffic engineering on transitNet respectively. The constraints he put are the service type and the specific applications run on subnets. The author claims that his

approach reduces latency. Since then, there are a lot of works done on the MPLS showing MPLS as a vital method for packet scheduling and resource optimization approach.

### 2. 3. 2. Overall Performance of SDN

There are also a lot of research works done on enhancing the performance of SDN, and those works have remarkable results. Many researchers over the past years have published their research work, and each of them have a significant contribution to improve the performance of software defined networks; some researchers focus on single metrics, like throughput or delay only and others use multiple metrics to measure and improve quality of services. Those researchers used different methodologies and frameworks to come up with solutions for the problems they claim. Among the works done on the performance of SDN, the following are recent ones that we used as good inputs for this study.

(Hossain, et al., 2018), have published their work known as "Enhancing and Measuring the Performance in Software Defined Networking"; they have used mininet emulator to create topology, and they have developed QoS module using python. In their architecture, there are three networks interconnected one to another and there is one central controller; they also compared the performance of the module with other modules regarding throughput and response time, and they claim that their module achieved remarkable result compared with NOX, ROIA, and other packet schedulers, however, sometimes, it achieved lower, but they didn't justify why?.

(Faisal, 2018), has done his PhD dissertation on "SDN-Based Mechanisms For Provisioning Quality Of Service To Selected Network Flows" using status monitoring and QoS based path setup modules developed with floodlight controller; he has grouped the packet flows into three groups, like critical with QoS, with no QoS, and QoS non-critical, hence, the throughput of the critical flow was recorded to be 9.5mbps, he has used two traffic metrics, and the average throughput of traffic metrics was recorded to be 49.4% for TCP. He claims that using MPTCP with ECMP increased the throughput to 64.4% with 2 subflows and 72.1% with 3 subflows respectively. But his work lacks

considering the interrelationship between QoS metrics and even the average throughput is not good enough.

(Themba, et al., 2019), have proposed solutions to improve the performance of wide area network; they break down the WAN into subnets, they have distributed controllers on subnets, and they have done network function virtualization at the edge, then they proposed overlay that serves as middleware between transport layer and application layer to support encapsulation. They have used different openflow controller, like ONOS, opendaylight, RYU and floodlight, and the authors observed that each controller had its own occasion of achieving better performance, then they concluded that compared with other controllers, ONOS reduces the average RTT to 0.2%, however, they did not consider resource utilization aspects and even the response time is not by far reduced.

(Khalili, et al., 2018), have studied about the latency that Openvswitches encounters during setting up the flow path. The authors used two alternative approaches i.e. the first one is reducing the update time of every switch, but due to the fact that openvswitches do have minimal updating time, this will not incur severe latency in flow setup. The second approach is reducing the number of switches updating at a time during path setup, this enables them to manage flows accordingly. They have tested their model with different topologies, and they claim that their approach reduces the flow setup latency in end users perspective, however, they did not consider all types of traffic flows, and the approaches that they used to reduce the number of updating switches during path setup leads increasing of error rate in the network, and then packet loss increases.

### 2. 3. 3. MPLS with SDN

SDN can be deployed in MPLS network to further improve the performance of the entire network. There are works done on improving the performance of hybrid network **(**Azodolmolky, et al., 2011), have proposed an overlay method to use openflow controller in optical packet switched network; their key idea/approach is separating the controlling function of the network in to two i.e. on the top, there is an openflow controller that is capable of controlling the entire network, and then in the middle of the network, there is GMPLS that controls the optical packet switched network.

In their architecture, the openflow controller communicates with GMPLS through OFGW user network interface (UNI). When there is a request from optical circuit switching network client to communicate with the client in packet switched network, the commutations go through GMPLS controller to reach to the open flow controller, and then the openflow controller forwards the request to the packet switched network client. This approach could potentially reduce the load of openflow controller, but still, there would be significant delay as the number of hopes increases, and the average throughput decreases as the number of switches increases.

(Kempf, et al., 2011), have proposed a method to incorporate MPLS in openflow by extending openflow tuple from ten to twelve where the two fields are used to store MPLS information, such as TTL push and pop labels. The authors developed open source router with NetFPGA which is a PCI card having virtual port creation support. They implemented forwarding plane and control plane in the same box, then the authors observed that their model increases the throughput significantly compared with traditional MPLS network. But still, there is a gap in considering latency and other metrics.

(Mohammad, et al., 2018), used both SDN and MPLS to improve resource allocation and to study the interdependency between flows. In their approach, they have identified two resource re-allocator modules namely flow level resource re-allocator and LSP level resource re-allocator. The authors use openflow switches to assign flows to the existing MPLS network LSP so, flow level resource re-allocator is used to avoid congestion by rerouting the flow whenever there is overflow in the link from predefined threshold. And the LSP level resource re-allocator is used to control the flow whenever the flow level resource re-allocator is unable to control congestion; in their model, the flow level resource re-allocator sends reassignment request to LSP level resource allocator, then LSP re-allocator re-assigns packets to the LSP to avoid congestion. Their approach improves resource utilization and throughput, but they did not consider the packet loss during rerouting, and they have considered the propagation delay of every link as the same, however, it would not actually be the same.

(Bellessa, 2015), has implemented MPLS in software defined network with the idea of enabling the controller to assign labels to the packets when the edge switches request for labels. In his approach, the author used edge switches and core switches together; the edge switches forward packets to the core switches using labels assigned by the controller. The controller is also responsible to install the path on core switches. Another approach that the author used is storing the labels on the MAC address table, and introduced label mapping concept to convert all labels that will be assigned to a particular packet into a static label.

To apply forwarding rules in the core switches, the author used both MAC address and TCAM. In the MAC table, every forwarding rule is inserted to the switch's MAC address table as a static address resolution protocol entry. The entry consists of the label value and the destination port number on the switch. In TCAM, every forwarding rule is added to TCAM as an OpenFlow entry. Then the entry is given the label value as the layer 2 destination and wildcard for other fields. The author claimed his approach achieves high hit rate to fill flow rules in the entire topology. However, this approach will causes significant load on the controller.

## 2. 4. Summary

The following table presents summary of related works.

**Table 2 . 1: Summary of Related Works**

| **Authors** | Title | **Methodology** | Result | Gap |
|---|---|---|---|---|
| Hossain, et al., 2018 | Enhancing and Measuring the Performance In SDN | Three interconnected networks using central controller | Reduces the response time and increases throughput than RIOA and NOX | Assessing the effect of switch capacity, bandwidth isolation |

| | | | | |
|---|---|---|---|---|
| Azodolmolky, et al., 2011 | Integrated Openflow GMPLS control plane: an overlay model for SD packet over optical networks | separating the controlling function of the network in to two | reduce the load of the controller | Delay as number of hopes increases |
| Bellessa, 2015 | Implementing MPLS with Label Switching in SDN | Enabling the controller to assign labels to the packets when the edge switches request for labels. | high hit rate to fill flow rules in the entire topology | Offloads the controller |
| Mohammad et al., 2018 | SDN-Based Resource Allocation in MPLS | SDN and MPLS | improves resource utilization and throughput | Considering the packet loss during rerouting |
| (Kempf, et al., 2011 | Openflow MPLS and the Open Source Label Switched Router | Extending openflow tuple from ten to twelve | increases the throughput significantly compared with traditional MPLS | Considering latency and other metrics. |
| Themba et al., 2019 | An SDN Solution for Performance Improvement in Dedicated Wide Area Networks | Network function virtualization to support encapsulation Using overlay that acts as middle ware between application and transport layers | ONOS reduces the average RTT to 0.2%, | Resource utilization aspects and even the response time is not by far reduced. |

From the literature, we grasped the method approaches and trend to improve the performance of SDN, and then the gaps to be addressed in the future. We also identified the lot of gaps that the authors did not address. Not only this, the literature also helped us to know the current state of software defined networks, and it helped us to consider other parameters for measuring the performance of software defined networks including switch capacity and bandwidth isolation, these are among independent variables having significant effect on the overall performance of SDN.

We identified a lot of gaps from the literature, but among the gaps, we focused on reducing latency, reducing packet loss, reducing jitter, increasing throughput, investigating the relationship between QoS metrics and improving bandwidth utilization i.e. Studying the flow setup latency and improving resources utilizations other than bandwidth are not included in the scope of this thesis.

# Chapter Three: Methodology

## 3. 1. Research Method

The research method we have used is simulation research method. It includes selecting parameters, setting up experimental environment, and measuring those parameters identified; we run different experiments to observe and investigate the effect of individual parameter on other parameter or the interdependency of many QoS metrics, and to examine the overall performance of the approach that we deployed.

## 3. 2. Data Collection Methods

Traffic Flow information is used to measure the performance of the proposed approach; different protocols, like TCP, UDP, ICMP and so on are analysed in packet flows. To collect network statistics, we have used IP flow information export (IPFIX), wireshark and IPREF tools.

IPFIX is general protocol to collect and analyse vital network traffic from routers, switches and other end devices that is able to support the protocol so that this information can be used to recover the network from serious slowdown quickly (Sam, 2020). It collects data in the entire network, and then the data can be organized by flow exporter. IPFIX protocol is different from netflow in: firstly, it is able to integrate information sent to syslog and SNMP, thus it avoids the need of additional services to collect information from every network devices (Marc, 2020), secondly, it allows variable length fields that is not offered by netflow; this allows us to include dynamically increasing data items. Lastly, it offers opportunities for users to customize their request and make the system able to do user specific tasks, like organizing data and analyzing them.

Wireshark helps us to demonstrate the relationship between two or more variables graphically for example, we can sketch throughput graph against time. IPERF is a service or protocol that runs on nodes and other devices, and it is used to observe the bandwidth between devices/link capacity, jitter, packet loss delay and other metrics.

With the collected data, we evaluated the performance of the network. We are able capture packets while devices are communicating one another. The type of data that we collect include: ICMP, TCP, UDP, ARP and so on. Once we collect, the network statistics in different parts of the network, we could observe the throughput, latency, packet loss rate, response time, etc.

## 3. 3. Performance Measuring Metrics

To measure the performance of the network, we can use different metrics, like bandwidth utilization, throughput, packet loss rate, response time, jitter, retransmission, and so on. In our case, we have used multiple metrics that have significant effects on the overall performance of the network. Particularly in this study, throughput, packet loss rate, response time, bandwidth utilization, jitter and retransmission are given strong focus since they are commonly used ones in the process of measuring the overall performance of the network.

## 3. 4. Algorithms, Protocols and Techniques

In this study, we follow simulation research approach. We have used different approaches and techniques i.e. to improve resource utilization, to balance the load, reduce latency and increase throughput. Our approach has distributed controllers that are possibly used as load balancing solution. We have applied bandwidth isolation, and policy-based packets classifications and/or multiprotocol label switching. Bandwidth isolation is efficient way of using dedicated bandwidth by allocating fraction of total bandwidth to every application or separate end devices accordingly; this reduces unnecessary wastage of bandwidth.

### 3. 4. 1. Multiprotocol Label Switching

Multiprotocol label switching (MPLS) is a packet forwarding technology based on packet labelling with a unique id that identifies the source and the destination of the packet. Multiprotocol label switching is a technique not a service used for traffic engineering; MPLS is also efficient in using the dedicated bandwidth in the wide area networks (WAN) or ISP networks; It can be used to prioritize packets, and it can be applied to use our resources efficiently. Instead of using packer headers and source IP address, MPLS uses labels to send and receive the packet in the entire network.

The labels are used to prioritize the packets in such a way that the performance goals can be achieved. Packets are assigned unique label (ID) called forwarding equivalence classes (FEC) as they enter to the gateway. The labels are used to identify the priority, and they can be used to determine destination of packets because each router assigns a unique label for each packet as it enters to it, and the labels will be distributed to the neighbouring routers so, this makes MPLS efficient way of traffic engineering in ISP networks.

The most important part of MPLS is its capability to handle any type of user traffic with the help of label swapping algorithm. It is also possible to assign user specific labels to identify packets having the same priority (Adnan, 2000).

#### 3. 4. 1. 1. Building blocks of MPLS

Multiprotocol label switching contains the following components.

- Ingress router
- Intermediary router and
- Egress router
- Label switching path
- Forward class equivalence
- Label
- Class of Service
- Stack Field
- Upstream and Downstream
- Label Distribution Protocol

**The ingress router** is an entry point where packets originate from their sources. This edge router adds (pushes) label and forward packets to destination. By establishing LSP and assigning labels, ingress edge router starts forwarding packet in MPLS core network.

**The intermediary router** is the label switching router (LSR) in the middle of the label switching path (LSP) where it receives the labeled packets, and performs label swapping, and forwards the new packet to the target interface. On the basis of its location in MPLS, this router performs label Pushing, swapping and Popping,

**Egress Router is the edge router** where packets reach their destination. This router removes (pops) label from the packet and forwards to destination. It disposes label from the arrived packet only when bottom-of-stack pointer recognizes if the detected label is bottom label of the stack or not.

**Label Switch Path (LSP):** is the path followed by packets from source to destination in the entire MPLS-enabled network. The path is simplex it consists of several label switching routers from ingress to egress ones. For a particular data to be transmitted from source to destination, LSP should be established that will probably contain one or more (ingress, intermediary and egress) label switching routers.

**Forward equivalence class (FEC):** the class comprises a cluster of packets of a typical application forwarded in its switched path in the same pathway in the same packet handling mechanism. Each packet of a specific class contains the same service essentials. Every type of traffic flow is assigned with a unique FEC and is done only once while they entering the MPLS environment.

**Label:** It is a short fixed length field packet identifier, used to classify a FEC of local impact. Immediately following the classification, the packet is assigned a label value that is used for packet forwarding. The labels are created based on network topology, request, and incoming traffic.

**Class of Service (COS) or (Experimental) field:** This 3-bit field determines queuing and reject algorithms applied to the packet while it is being transmitted through the network. Since the COS field has 3 bits, there are eight distinct service classes to be used.

**Stack Field:** a label stack is an ordered set of labels where each of them runs with a specific function. Whenever an ingress router pushes more than one label in a single packet, it forms a stack of labels. The bottom-of-stack indicator is created to indicate the end of the label of stack. It can be represented as follow interms of bits.

- ✓ When‘0’, indicates label stack implementation is in use
- ✓ When‘1’, indicates bottom of the label stack

**Time to Live** (TTL): is 8-bit field undergoes the same function of IP’s TTL, where the packet is rejected when TTL is ‘0’ to prevent looping in the network. If a labeled packet passes through LSR, the value is decremented by one. It can be represented with bits as follow.

- ✓ TTL is ‘0’, Packet rejected,
- ✓ TTL is ‘1’, Labeled packet navigates LSR

**Upstream and Downstream:** The terms downstream and upstream are fundamentals for understand label distribution and packet forwarding in MPLS enabled network. Both upstream and downstream are origins of packet transmission. Downstream for a target network is the destination point, whereas upstream is the amount of data sent from end users to the central packet handler i.e. updates belong to a specific prefix are always broadcasted upstream.

**Label Distribution Protocol (LDP):** is a protocol that distributes labels in a MPLS network on the basis of the routing information from IGP. Forwarding Information Base (FIB) of routers determines a hop count based path in the entire network. LDP is used only for signaling the best effort LSPs unlike paths that have been set as traffic engineering paths based on various constraints. LDP enables a Label Switching Router to determine the capacity of its peers.

We used MPLS to handle and manage network traffic and improve resource utilization on the data plane; it reduces the loads of the controllers when there is simultaneous transmission of packets among the three groups of users in the entire network. It is enabled to communicate with the data plane through openflow gateway, and we can easily incorporate into our topology. With these realities, MPLS facilitates the communication with the controllers, and it can handle low level traffic flows. By the help of RESTCONF API provided by the controllers, we are able to integrate MPLS with openflow, and the openflow controllers establish MPLS path using path computation element protocol (PCEP). The kind of abstraction that enables the controller to use PCEP to instruct routers traffic engineered path helps us to tackle the problem of worrying how the communication between the controller and MPLS routers takes place, as a result, the communication between them is hidden for users.

### 3. 4. 2. Policy-Based Packets Classification

Policy-based packets classification is a packets classification scheme based on set of rules and standards followed in the organization. It is a way of improving the performance of software defined network based on certain business constraints i.e. the owner of the network can define set of policies to prioritize services through their level of sensitivity or precedence according to business processes held in the organization.

Network administrator could be able to implement those logics on the top of the controller, and he/she can use those policies in the entire network since the controllers monitor the whole network parts with global view. The policies are written in such a way that they can be dynamically propagated throughout the entire network, and they are aware of failure happened in different parts of the network, it should also be applied to intelligently recover from failure and it must notify about the new rules set to every connected device. The policies are applied on the top of the controller and since the controllers are capable of monitoring the entire network, it is cool to apply policies.

Policy-based packets classification is a packet handling mechanism based on access lists to be used by the routing devices before forwarding packets. The access list identifies the

particular packets that are forwarded to the concerning device sequentially. Network administrator can enable policy-based packets handling whenever there is a condition that certain packet to be handled differently from the common shortest path first approach. The main aspects for consideration of policy-based packets classification are to ensure equal access, protocol sensitive routing, source sensitive routing, routing centered on interactive versus batch traffic, and routing according to dedicated links capacity. Policy-based packets classification is a more flexible mechanism for routing packets than common routing technique i.e. shortest path first (Cisco, 2019).

To apply policy-based packets classification throughout the network, network administrator manipulates the flow table to use for policy-based packets classes and create flow rules necessarily, and then those rules specify the matching criteria and the subsequent actions incase all of the matches' rules are addressed. To apply policies on specific interface, it needs indication of which output interface the device should use by using the option output interface when we setup flow entries. Packets arriving on the given interface can be treated according to their level of priority and actions found on that flow entry that the packet follows. The flow entry associated with the particular interface is used to handle all packets arriving on that interface accordingly.

To describe the action used by policy-based packets classification, we can use permit and deny key words according to the VLAN priority number assigned by the network administrator. The one thing in policy-based routing is setting access control lists (ALAXALA Networks Corporation, 2012). Access control lists help us which path the packet should follow and which path the packets should not follow. This approach reduces unnecessary flows that offload the controller and even the path. So, packets are processed and transferred via pre-configured path. To identify the criteria for which packets are going to be forwarded we used the VLAN tagging; packets are encapsulated whenever they are transferred from source to destination.

Access list number and access list name are used while we were setting the path that the packet goes through it. Address resolution protocol and link discovery protocols are also used to mitigate complexity and identify active links and/or failure respectively.

Once the network administrator is aware of the network infrastructure and those business processes run in the organization, he/she can apply those logics to manage network traffic in the entire network infrastructure. Prioritizing packets is done according to the level of users to make the network operates with expected quality of service in such a way that network management task becomes easy.

Policy-based packets classification can be used together with MPLS to apply business rule based routing of packets (Chen, et al., 2008), they argued that "Diff-Serv-aware Traffic Engineering extends MPLS traffic engineering to enable you to perform

constraint-based routing of guaranteed traffic, which satisfies a more restrictive bandwidth constraint than that of satisfied by CBR for regular traffic". We used policy-based packets classification to manipulate flow tables of openvswitches, and to manage flows by grouping users into three classes of users. To manage flows, we used virtual patch panel that acts as private VLANs. Having done this, we are able to assign priorities for flows, and we are able to apply static bandwidth isolation in the entire network.

## 3. 5. Bandwidth Isolation

Resource utilization is used to measure the efficiency of a certain products or items. In networking concepts, resource utilization is tightly coupled with the performance of the network. Among network resources, bandwidth is the major one. Bandwidth is the measure of the capacity of a link through which packets can be transmitted per second. Among the resources of the network infrastructure, bandwidth is the major one so, efficient use of bandwidth is necessary in order to enhance the overall performance of the network.

So as to improve bandwidth utilization throughout the network, bandwidth isolation is effective way. Bandwidth isolation is part of policy-based packets classification that helps us to use the overall bandwidth efficiently and effectively; implementing bandwidth isolation involves modifying the flow table of all fragments of the network as well as setting the VLAN priority field by FlowVisor (Coker, et al., 2017). Traffic from every

fragment is then assigned to one of the different priority groups; network administrator is capable of prioritizing the bandwidth per fragments.

Different applications require different portion of overall allocated bandwidth so, allocating different bandwidth for different link can improve overall bandwidth utilization in the entire network infrastructure.

Bandwidth isolation can be done according to the need of every application and organizational desire to ensure effective use of subscribed bandwidth. For example, organizations may need the links that connect administrative staffs to be allocated much amount of the portion of bandwidth, and they would prefer to deliver less amount of bandwidth to supportive workers and guests. This implies that there should be a certain policy to fairly allocate the bandwidth as it is intended to be effective, therefore, analysis of applications that every user could run and use is important.

Some organizations do have a certain mechanism to monitor users on how they are using organizational resources including bandwidth; to install a certain application, users need to be authorized by the concerning body since it consumes the resources of the organization.

In this thesis, we have applied bandwidth isolation to improve resource utilization across the entire network. Since bandwidth has a significant effect on throughput and other performance measuring metrics, we have also used it as a way of improving throughput; efficient utilization of bandwidth enables us to enhance throughput of the network.

To accomplish this task, first, we consider users classification accordingly like administrative staffs, technical workers and supportive workers; we divided the type of users into three different groups according to the type of network applications they use, and based on their priority as top level users, medium level users and low level users. With this fact, top level users have been given more fraction of dedicated bandwidth, medium level users have been given fraction of dedicated bandwidth less than that of high level users and greater than that of low level users, then we take the different network edges i.e. enterprise edge, e-commerce edge, WAN edge, core, and distribution

areas in to account. Having done these measures, we could allocate the dedicated bandwidth in the entire network efficiently.

## 3. 6. The Capacity of Switches

The capacity of switches could also affect the overall performance of the network; the difference on flow setup capacity from switch to switch as it is described in the literature often differs so, it hinders the overall performance of software defined networks. To study this issue, we used different brands of switches that have different flow setup capacity, and we simulate the topology with different switches having different capacity.

Switch capacity can be expressed interms of the type of controller they support, and the type of programming language that it is developed by; indigo switch is designed to be used by the floodlight controller, and it is developed with java programming language, ONOS switch is developed by ONOS group to be used by ONOS controller, and it is based on java programming language. So, the operational aspect of switches directly goes with the performance of controllers. In this study, we used OVSbr, OVS and user space switches. We configure openvswitches having management interfaces as standalone devices in order to enable them they can handle low level traffics without the involvement of the controllers.

## 3. 7. SDN Controller

The controller is the brain of the SDN (Bhat, et al., 2015) so, the performance of controllers has significant effect on the performance of the network itself since routing decision is made by the controllers, and the controllers are responsible to manage the entire traffic flow throughout the network. When we set up the controllers, we used physically distributed controllers approach. This enables each controller node to have equal view of the entire network, and when there is a failure in one controller node, the whole network functions as it is with no interruption because the other controllers look after the network infrastructure with global view, and there will be no interruption in the network operation process. Another advantage of this approach is it is no more necessary

to worry about complexity; this implies the controllers can be placed on different parts of the network on a flat controller's distribution approach; as a result, it becomes easy to manage them.

The placement of controllers often varies because the length of the path has significant effect on packet transmissions. When nodes are far apart from the controllers, the time it takes to reach to the controllers often increases; as a result, there exists significant delay in the transmission. Therefore, we need to take care when we place the controller nodes; when the hosts have different distance from different controllers, the hosts necessarily scarify certain delay in case of single point failure. To overcome this problem, network administrator must consider campus infrastructure, and he/she had better consider path length.

Since the beginning of SDN, many controllers are being developed, and there is a dramatic change in the development process. Now a day, highly performing open source controllers are available for fast prototyping and observing the effect on actual traffic flows. Criteria to compare controllers are: programmability of the network, openflow support and efficiency are most preferable for our case. According to recently published papers, the best performing controllers for distributed controller architecture are ONOS, open daylight and POX (Salman, et al., 2016)**.** So, in this thesis, we were working with them to enhance the overall performance of SDN.

### 3. 7. 1. Open-daylight

Open-daylight is a java based SDN controller designed and developed for distributed controller approach, and it is highly modular, well-documented, consistent, it supports multithreading (Zhu, et al., 2019), and it supports openflow flow manager application that is suitable to manage flows. The opendaylight flow manager used to manipulate flows, and it helps us to manage flows in such a way that we are able to write custom flows. We used this flow manager application to handle flows through RESTCONF API, and it lets us to let MPLS handle low level flows.

### 3. 7. 2. Open Network Operating System (ONOS)

ONOS is an open source full featured controller written in java programming language, and it is still on progress of development (Zhu, et al., 2019). It supports distributed controller architecture, it is documented properly, it supports multithreading, it is highly modularized, and it is consistent enough. To run ONOS and open-daylight, we need the following softwares.

**Apache karaf application container**: it is a lightweight flexible application container that provides dynamic configuration, advanced logging system, remote access methods, role based access control to ensure security, etc.

**JDK 11 and above**: since ONOS and opendaylight are developed with java programming language, we need JDK, and the recommended one is to use JDK11 and above.

**Docker image**: it is a read only application containers that enables users to create ONOS clusters, to import Docker container guest devices and it acts as a house of many files bundled together.

**Maven**: it is a project management tool used to manage java based projects; it reports the documentations and project build success or failure.

### 3. 7. 3. POX

Pox controller is an openflow controller based on python programming language, and it emanates from network operating system (NOX) (Bhat, et al., 2015). We prefer pox because it helps us to use modules and applications included in mininet module; both mininet and pox are programmed in python so, it is possible to import applications running in mininet into pox, and we can easily customize to our particular demand. Pox has also a great advantage on load balancing; therefore, it is possible to incorporate that application to improve the overall performance of SDN. This can be done according to the fact that the distribution of load matters on the performance of network; network traffic can be fairly distributed into distinct traffic handler and controllers mainly.

Formerly, POX was designed for centralized controller SDN architecture, but now, it is possible to use for distributed ones that contains multiple controllers running on the control plane via the same port number, and having the same function, as a result, single point of failure will not be exhibited in the network, and then it is possible to boost the performance of the entire network architecture.

### 3. 7.4. The Communication between Controllers

The communication between controllers takes place through west-east interface that they provide. In this study, we used physically distributed controller approach where all controllers have equal burden with global view so, the communication among controller nodes about reachability update, route update and information related with flow setup takes place through west-east interfaces. All controllers should reach into consensus about all these information to resolve conflict among them.

## 3. 8. Conceptual Model

Having used these tools and techniques, we proposed the model to enhance the performance of SDN in multiple metrics, and to investigate the relationship among many QoS metrics. Not only this, in this thesis, other independent variables are also taken in to account for example, switch capacity, the impact of number of queue and bandwidth isolation. The general architecture of the model contains the following components:

1. The control plane with distributed controller nodes: contains multiple SDN controllers having the same function and responsibility and those applications that run on the top of the controllers (what we call application plane).
2. North bound interface: it is an API between developers and the control plane, and helps the controllers to interact with applications that run on the top of the control plane.
3. Data plane: it is also called forwarding plane. This is where actual user data transmission takes place i.e. network traffic flow from hosts to switches and vice versa.

4. South bound interface: it is an interface between the data plane and the control plane, and it helps the control plane to interact with the forwarding plane. The most common south bound interface is openflow that facilitates the communication between the control plane and the data plane.
5. Openflow: it is a communication protocol between the control plane and the data plane it is also called south bound interface.
6. Integration of MPLS with openflow.
7. Bandwidth isolation module: this where bandwidth isolation scheme is applied
8. The module that investigate the relationship between QoS metrics.
9. Other independent variables (switch capacity, bandwidth isolation)

Here, some components are what SDN generally has, and some of them are what we add to the general structure of SDN, such as integration of MPLS with openvswitches, bandwidth isolation scheme and the module used for investigating the relationship between QoS metrics are components that SDN will not have necessarily.

The model is represented by two figures; figure 3.1 shows the compact model that includes packet capturing process while devices are communicating and it also indicates other independent variables like bandwidth isolation, the impact of the number of queues and even switches capacity. It illustrates actual packet transmission on both the data plane and the control plane, and then the generated data is used for analysis of the result and interpretation. Figure 3.2 shows the detailed flow chart for handling the packet flows on both the data plane and the control plane. It depicts how packets flow from source to destination according to their priority labels. High level and medium level Packets can be handled on the control plane in such a way that they can be easily managed through policies implemented on the controllers. If the packets match with the policy, they can be forwarded to the destination unless they have to be dropped. On the data plane, low level packets can be handled according to their level of priority assigned by the MPLS. Packet could be assigned a unique label called forwarding equivalence class, and then the low level packets transmission can be handled on the data plane with the help of switches that act as standalone devices. The following figures show the compact model and the detailed flow chart respectively.

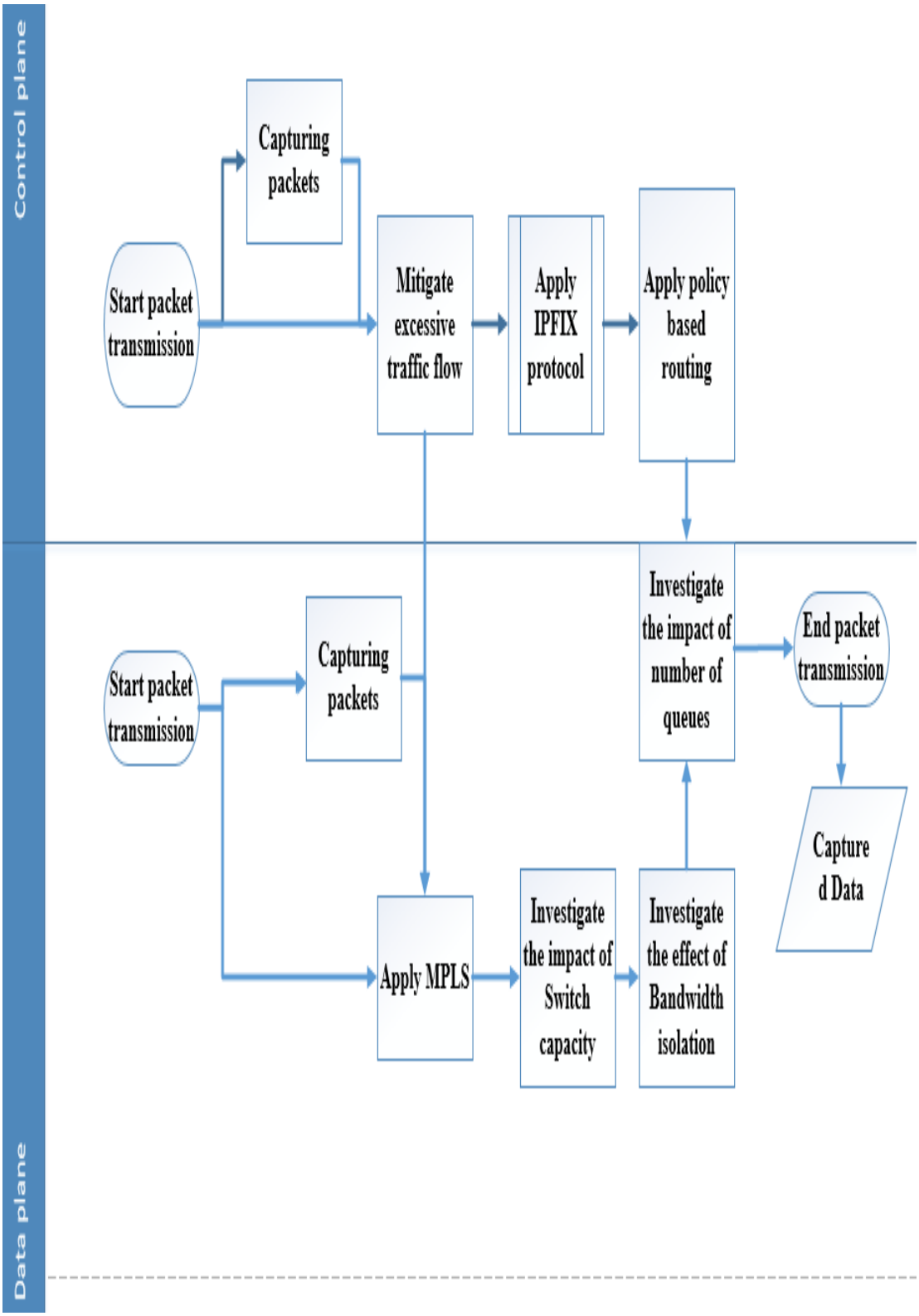


**Figure 3.1: Compact Model**

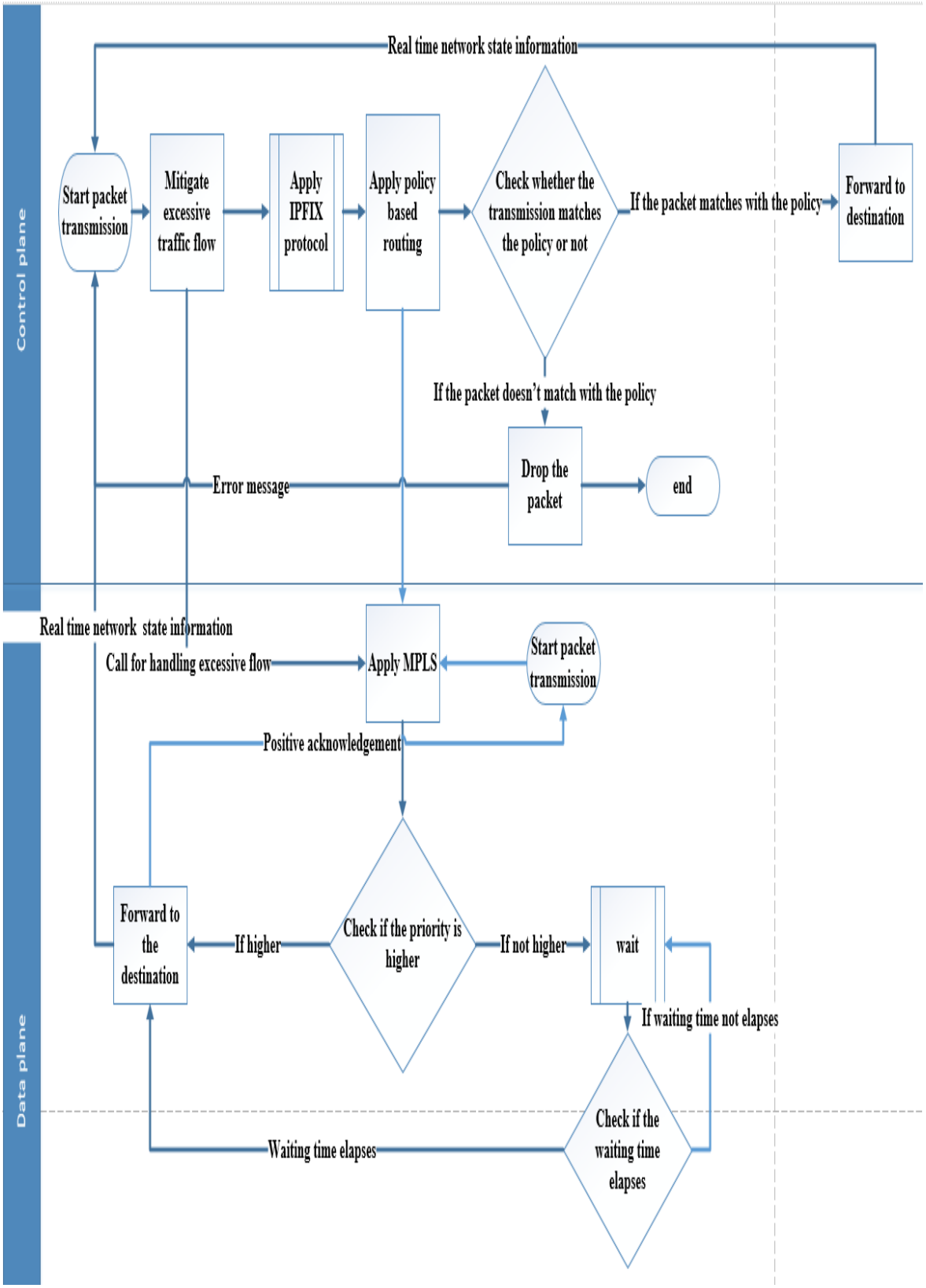


**Figure 3.2: Detailed Flowchart**

The above model is represented by the following Pseudo code.

```
Initiate packet transmission
    Add flow entries to flow tables
    Apply IPFIX
    Apply packet encapsulation/VLAN tagging
            If (Vlan_Id==1&& Vlan_Id==2)
                Controllers set path
                Prioritize packets with VLAN ID
                    If (Vlan_Id==1)
                            Forward to the destination
                            End
                    Else
                        Wait
                            If no waiting
                                Forward to the destination
                                End
        Else if (Vlan_Id==3)
            MPLS sets the path
                Prioritize packets with labels
                  If (labe_l>label_2)
                        Forward label_1 to the destination
                       Add label_2 to waiting
                            If  no waiting
```

Forward label_2 to the destination

End

Else If (labe_l==label_2)

Add custom label

Forward the packet having high priority to the destination

End

End packet transmission

### 3. 8.1. Model Explanation

The control plane filters out unnecessary flow of packets using IPFIX protocol. IPFIX protocol is used to filter out packets in such a way that it enables network administrator can control the flow by applying his own rules based on specific demand. All packet processing policies are implemented on the top of the controllers, and they can be deployed as applications. Having done this, the applications can communicate with the controllers and the rest of the network with northbound interface.

The late versions of openflow protocols including openflow1.1 support MPLS (Kempf, et al., 2011) so, MPLS networks can easily communicate with openflow enabled network where MPLS manages low level traffics on the data plane and the controllers control the high level traffics.

Data plane is able to process and control small amount of packet flows by the help of MPLS. As we tried to mention in the literature review part of this work, switches can handle small flows what we call mice flows; this results in the reduction of loads of the controllers. With the help of MPLS, it becomes possible to handle packets on the data plane since MPLS operates between layer 2 and layer 3 of OSI reference model, and it is commonly considered as layer 2.5 protocols. With the help of data plane programmability feature, low level packet flows can easily be processed on the data plane without interfere of the controller; this reduces the loads incurred on the controller nodes. The controllers

add flow entries to the flow tables of openvswitches, and manage the communication between MPLS and openvswitches through the two interfaces of openvswitch connected with MPLS.

Access control lists (ACLs) are parts of the policies applied. With ACL, network administrator can control packet transmissions, and he/she can control unnecessary flows from offloading the controllers. Network administrator can classify packets accordingly i.e. he/she can classify packet flow hierarchically starting from the top, and he/she goes down to guest users.

Users can be classified into three groups, like high level users, medium level users and low level users. Having this information, network administrator can create VLANs with different priority. The new version of openflow supports to assign VLAN priority number with three bits (3-bits) ID that identifies the priority. With this assumption, we could classify packets as: high priority, medium priority and low priority packets. Using the VLAN priority, low level users are assigned to VLAN having low level priority, and then Low level users' packet flow is processed on the data plane with the help of MPLS.

When there is a need to communicate with the control plane, or when there is a need to have the communication between the low level users with high level users, the communication happens through MPLS i.e. MPLS calls the controllers to establish the path dynamically.

Packet flow from top/high level users to low level users takes place and processed by the controllers automatically, and the controllers take over every action associated with the packets. But if the flow is from low level users to top/high level users, the controllers will not take over the controlling task automatically; there should be a request from MPLS network to approve the flow. The same is true when there is packet transmissions from medium level users to low level users, the controllers take over every task associated with the transmission, but when the transmission is from low level users to medium level users, the request should be approved by the controllers.

With the above conditions, packet scheduling could be handled both on the data plane and the control plane; on the data plane, packet scheduling is done by MPLS

routers/switches. Whereas on the control plane, packets scheduling is done by the controllers according to their priority assigned by their class of service, means, like low level service medium level service and high level service associated with VLAN tags or packet encapsulation mechanisms.

## 3. 9. Simulation Tools and Techniques

The conceptual framework that we use in our experiment is mininet emulator installed on Ubuntu enabled VMware workstation pro integrated with GNS3. The VMware workstation is installed on HP-desktop computer having: 8GB RAM, 1TB hard disk and corei7 processor with speed of 3.19GHz. Mininet is open source network emulator software that is capable of emulating SDN controllers, open source switches and end devices.

Mininet enables creating topologies of very large size with thousands of nodes and conducting test on them very easily (Karamjeet, et al., 2014; Bhat, et al., 2015). It has very simple command line tools and API. Mininet lets users to create custom topologies, and it enables them to export source codes; with the help of miniedit (GUI component) of mininet, we can easily add and drop controller, switch and host nodes, and we can easily connect via the links. Then we can export source code for further customization.

Mininet lets the users to easily create, customize, and test SDN networks using different programming languages, like C, C++, java, python and others. In our case, we have used python and java because both are user-friendly, and they offer GUI supports, not only this, they also support highly performing open source controller i.e. POX and ONOS respectively. We used python for POX controller, whereas ONOS and Opendaylight are based on java. Java is more preferable because it is platform independent, multi-threading support and modular competencies (Lawal, et al., 2017).

GNS3 is an open source software graphical network simulator that we can use to configure, test and troubleshoot virtual and real networks. It helps us to run small topology connected with few devices, but able to host as many devices virtually on multiple servers or on the cloud environment. GNS3 contains two components these are:

- The GNS3-all-in-one software (GUI) and
- GNS3 VM

GNS3 GUI is the client part of GNS3 and is graphical user interface (GUI). That we install on our local PC; it supports Windows, MAC, Linux operating systems and we created our topologies using this software.

GNS3 VM is a local server that acts as a container for many applications so, we can import applications and virtual devices running in this machine to be used by the GNS3 GUI.

### 3. 9. 1. Topology

We created a network topology containing MPLS core network integrated with SDN; the core network contains five MPLS enabled legacy router having the capability to process the packets on the data plane without the involvement of the controller application. In this scenario, the controllers are responsible for top to bottom flows, whereas low level flows, such as data transmission among low level users is handled by MPLS core.

The controllers are installed on virtual machines such ODL VM and ONOS VM, and we integrated them with GNS3 to operate together. Mininet is used to create topologies integrated with GNS3, then we added the interfaces of routers with MPLS capability to openvswitches interfaces so as to establish the communication between the core MPLS and mininet topology.

The interfaces of the routers connected to openvswitches are managed by the openflow controller while they are sending and receiving the packet. To establish the communication between the MPLS domain and openvswitches, NAT is enabled on both MPLS core and openvswitch that interconnects Docker containers. NAT is used to ensure global communication across virtual devices and physical network. So, we established the communication between virtual machines containing SDN controllers and cisco routers in MPLS domain through NAT. We have two kinds of topologies these are 1) the topology inside GNS3 environment and 2) the topology inside mininet. Inside mininet topology, h1-h16 represent hosts, c0-c4 represent controllers and s1-s4 represent

openvswitches. R1-R5 inside GNS3 topology represent routers, and the topology shows us the physical integration of MPLS core with openvswitches, whereas the topology inside mininet does not show physical integration of MPLS with openflow networks, it is abstract. Actually, we can use Docker container openvswitches and Docker container Ubuntu guests to replace mininet openvswitches and hosts respectively. The topologies are presented as follow.

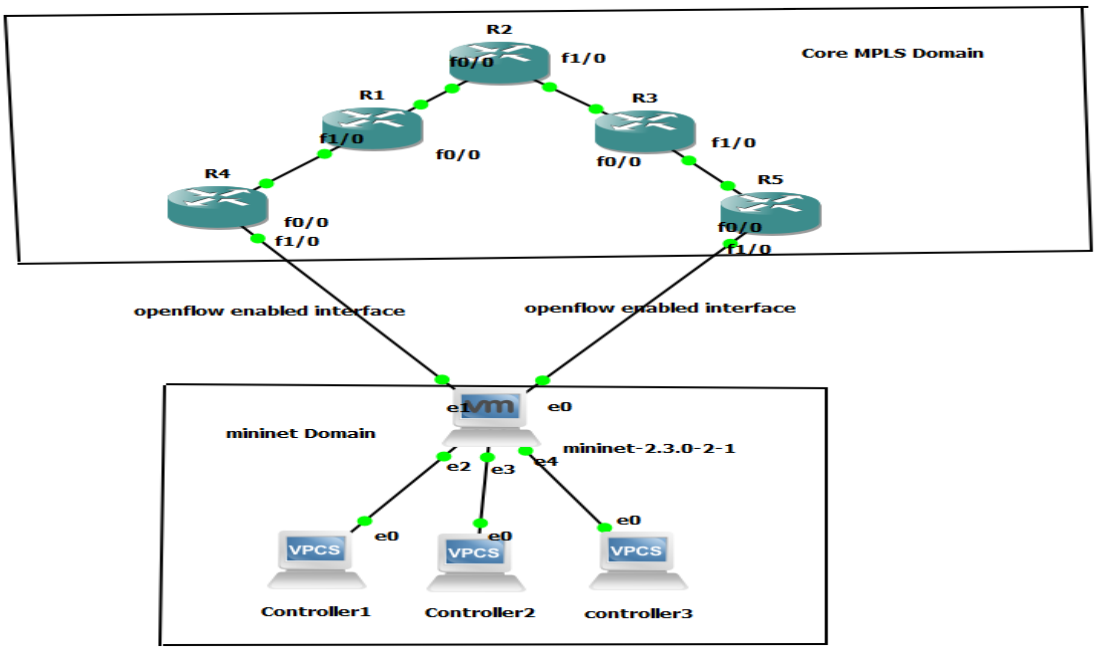


**Figure 3.3: GNS3 Topology**

We created the following mininet topology containing four controller nodes using miniedit GUI.

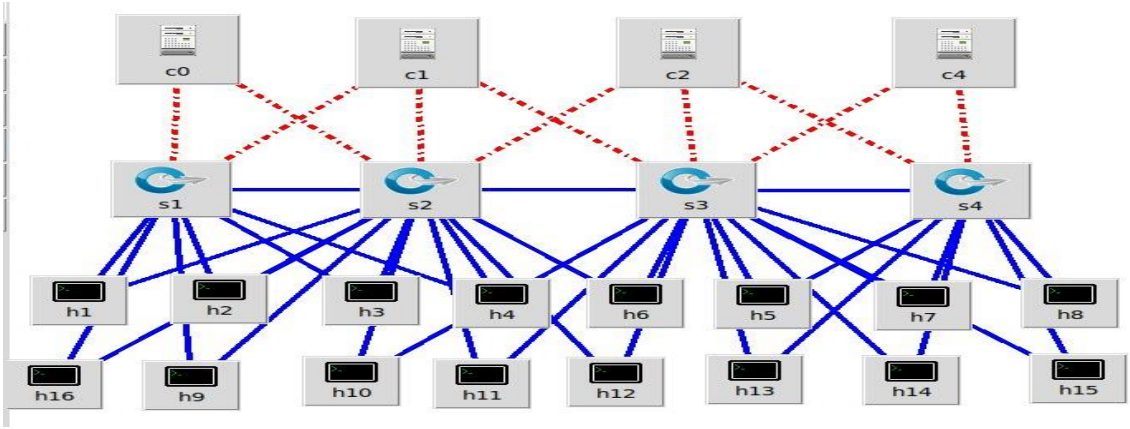


**Figure 3.4: Mininet Topology**

# Chapter Four: Result and Discussion

## 4. 1. Parameters Explanation

The result of the simulation needs to be expressed via many measuring parameters. In our case, performance measuring metrics are the key parameters to measure the overall performance, and evaluate the model accordingly. As we stated in the methodology section of the study, the following are key metrics measured.

### A. Throughput

Throughput can be defined in many ways and it is highly affected by link capacity. It is the measure of the number of error free packets transferred per second, or it is the measure of the number of packets successfully transmitted to target destination compared with total packets transferred. (Alemayehu, 2019), expressed throughput as a measure of the amount of units of information a system can process in a given amount of time. To measure throughput there are certain standards that should be taken into account. Throughput can be expressed by the following formula.

$$\text{Throughput} = \frac{\sum \text{number of transferred bits}}{time(s)}\ bps$$----------Equation 1

**Table 4.1: Throughput Standards** (Sugeng, et al., 2015)

| Standards | Numerical Value (%) |
|---|---|
| **Excellent** | **100** |
| **Good** | **75** |
| **Medium** | **50** |
| **Poor** | **<25** |

### B. Response Time

Response time is a measure of the time it takes to transfer packets from source to destination and the time required until the acknowledgement sent back to the origin of packets. It also includes the time required for retransmission, and it can be expressed interms of round trip time. Round trip time is the time required for two nodes until they exchange acknowledgement; when we talk about acknowledgement, we mean that the receiver sends a message that ensures successful delivery of packet. The general formula to calculate response time is expressed as follow

$$\text{Response time} = 2 * \text{propagation delay} + \text{queuing delay} + \text{processing delay} \quad \text{----------Equation 2}$$

**Table 4.2: Standards to Measure Latency** (Sugeng, et al, 2015)

| Standard Group | Numerical Value (ms) |
|---|---|
| Good | 0-150 |
| Medium | 150-400 |
| Poor | >400 |

### C. Jitter

Jitter is a measure of variation in latency. Or it is a measure of the difference in latency between successive packets. Latency often varies due to the fact that there are many influencing factors associated with it. The one factor is the impact of the number of queues in a given link this implies link capacity is also tighten with delay variation. Jitter affects successful delivery of packets in such a way that it leads a packet to be dropped or rejected by the routing decision due to elapsing of time to live (TTL). The general formula to calculate jitter is presented as follow.

$$\text{Jitter} = \frac{\sum delay\ varuation}{\text{packet received}}\ second \quad \text{----------Equation 3}$$

**Table 4.3: Standards to Measure Delay Variation** (Sugeng, et al., 2015)

| Numerical Value (ms) | Standard Group |
|---|---|
| 0-20 | Good |
| 20-50 | Medium |
| >50 | Poor |

### D. Packet Loss

Packet loss is a measure of errors experienced in packet transmission or it is a measure of packets that are not successfully transferred, or it is a measure of packets dropped or rejected compared with number of packets transferred. Packet loss will happen due to link congestion, packet processing capacity of devices, routing policy mismatch, and other internal errors. Packet loss can be expressed with the following formula.

$$\text{Packet loss} = \frac{packet\ sent - packet\ received}{packet\ sent} \times 100$$ ----------Equation 4

**Table 4.4: Packet loss measuring Standards** (Sugeng, et al., 2015)

| Number of Packets lost (%) | Standard Group |
|---|---|
| 0 | Excellent |
| 3 | Good |
| 15 | Medium |
| 25 | Poor |

## 4. 2. Result Obtained

### 4. 2. 1. Response Time

Associated with response time, there are different factors to be taken in to account for example, link delay, propagation delay, queue size, inter-device and intra device latency, etc. In this thesis, we examined the effect of link delay, queue size and the type of switches used.

The ability of handling packets on both the data plane and the control plane allowed network operations to be quicker and quicker because the controllers are no more be overloaded, and low level packets are handled by the switches by the help of MPLS. MPLS shares the load of the controller when there is simultaneous packet transmission among the three groups of users; not only this, having more than one controller on the control plane also share the load of packet handling of a single controller, and the

controller are no longer be in a busy tone, as a result, there is no more queuing delay. Using all these facts, we measured the latency for ten consecutive experiments, and we have taken the maximum value since it has high discriminant power. In every experiment, the latency is decreased compared with previous works. In these two former works presented below, the latency is caused by queuing delay when there is excess traffic flow, but in our method, low level packets are handled on the data plane separately from medium level and high level packets, this avoids queuing delay. The following figure illustrates how much the latency is reduced compared with two former works.

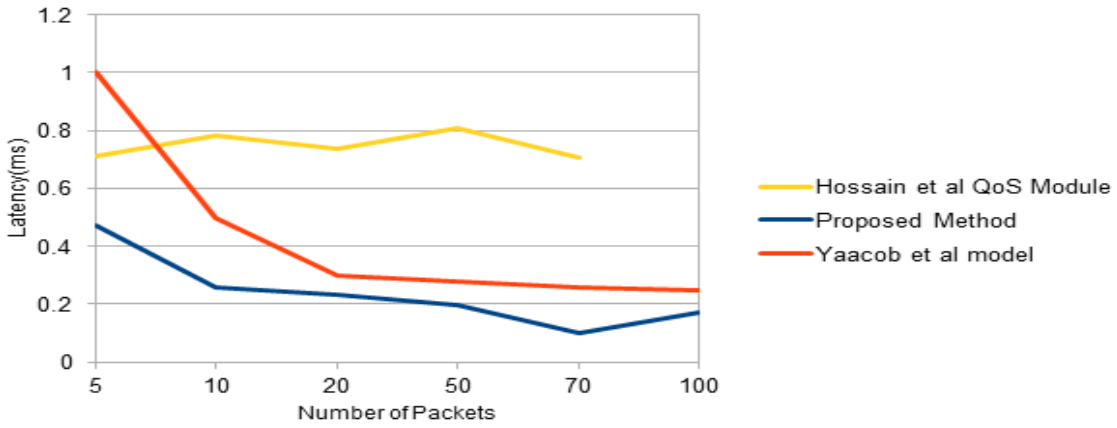


**Figure 4.1: Latency of Proposed Method**

The above figure shows that the proposed method is better than those two methods in RTT; in all cases, the RTT of the proposed method is below **0.5ms**. We compared our method with those methods using packet transmission in the same hop counts and the same number of packets. The figure shows that the latency decreases as the number of packets increases, this is due to the fact that there is no need of waiting for path establishment or path computation once the path is set for the first group of packets.

### 4. 2. 1. 1.Effect of Operational Aspects of Switches on Response Time

As we stated in chapter three, the type of switch matters on the overall performance of software defined network. In our experiment, we used three different types of switches namely, OVSbr, OVS and user space switches, and we found that there is slight difference interms of minimum, maximum and average response time between them. It is known that OVSbr acts as a bridge, and it is expected to be better in all cases, however, it

is not as we expect it to be. Even user space switch is better than OVS in maximum RTT. The effect of the type of switches on latency is presented in the following table.

**Table 4.5: The Impact of the Type of Switch on Latency**

| Number of Packets | Type of Switch | Response time Interms of Round Trip Time in(ms) | | |
|---|---|---|---|---|
| | | Minimum | Maximum | Average |
| 12 | OVS | 0.021 | 3.737 | 0.380 |
| | OVSbr | 0.025 | 0.147 | 0.039 |
| | User space switch | 0.039 | 1.611 | 0.230 |
| 20 | OVS | 0.015 | 2.286 | 0.205 |
| | OVSbr | 0.025 | 0.136 | 0.063 |
| | User space switch | 0.032 | 1.670 | 0.317 |
| 30 | OVS | 0.013 | 3.045 | 0.151 |
| | OVSbr | 0.025 | 0.283 | 0.067 |
| | User space switch | 0.037 | 1.384 | 0.291 |
| 50 | OVS | 0.020 | 7.847 | 0.223 |
| | OVSbr | 0.023 | 0.269 | 0.056 |
| | User space switch | 0.048 | 1.541 | 0.477 |
| 100 | OVS | 0.016 | 18.152 | 0.246 |
| | OVSbr | 0.022 | 0.303 | 0.055 |
| | User space switch | 0.333 | 2.688 | 0.338 |

The above table illustrates that OVS is better than OVSbr and user space switches interms of minimum RTT, OVSbr is better than OVS and user space switch interms of maximum and average RTT and User space switch is better than OVS interms of maximum RTT. This indicates that further investigation should be done on why OVSbr is

not better in minimum RTT although it is considered as standalone devices and even it acts as bridge. Even OVS is known to be better than user space switches, however, according to our findings, user space switch is much better than OVS interms of maximum RTT.

#### 4. 2. 1. 2. Comparison of Controllers interms of Response Time

In our experiments, we used three controllers, and then we found that there is a little difference interms of round trip time. Among the three controllers, opendaylight is better than ONOS and pox in RTT. We observed a little difference between pox and ONOS; the RTT of ONOS and pox kept constant to **0.14ms** starting from **100** packets, but with opendaylight, the RTT is below **0.14ms** for all number of packets. However, the slight difference in latency among controllers has no visible effect on other parameters, such as throughput. The result obtained is presented in the following figure.

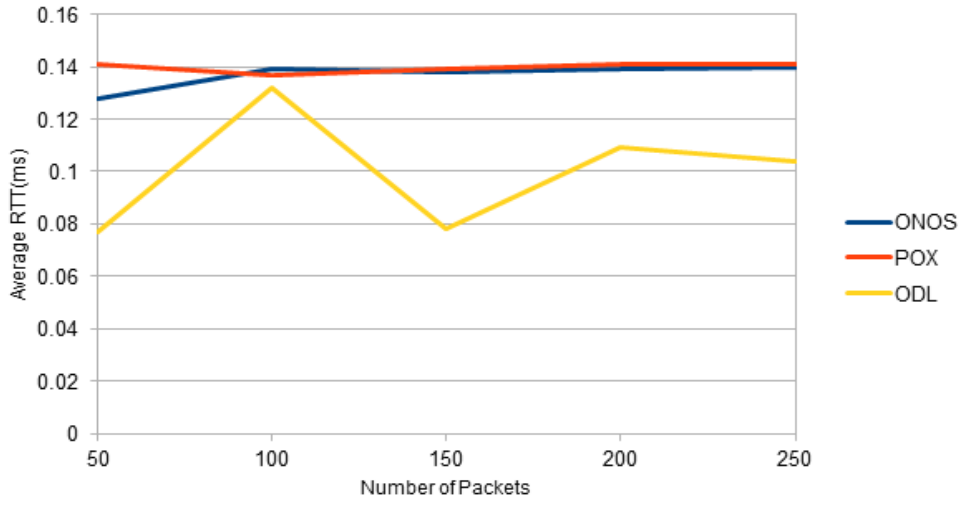


**Figure 4.2: Comparison of Controllers in RTT**

### 4. 2. 2 Throughput

The integration of MPLS with openflow improved the throughput of the network. In this thesis, we enabled packet processing and monitoring on both the control plane and the data plane. We enabled MPLS to operate with SDN controllers through path computation element protocol (PCEP) so that it can handle low level traffic flows by the help of standalone openvswitches, and the low level flows are directed to MPLS pipe lines

instead of being sent to SDN controllers pipe lines. So, this approach reduced the load of the controllers during simultaneous transmission of packets among all classes of users, and then packet transmission becomes faster and faster. Not only this, it also reduced the packet loss occurred due to offloading the controllers and queuing delay, and then it enhances the throughput. We applied static bandwidth isolation in the entire network so as to improve bandwidth utilization; this scheme enabled us to use the dedicated bandwidth effectively and efficiently. With this fact, the throughput is improved compared with previous works. We presented our findings in comparison with other works in different scenarios for example, in link failure scenario, size of data transferred and number of hops. The result is illustrated in the following figures.

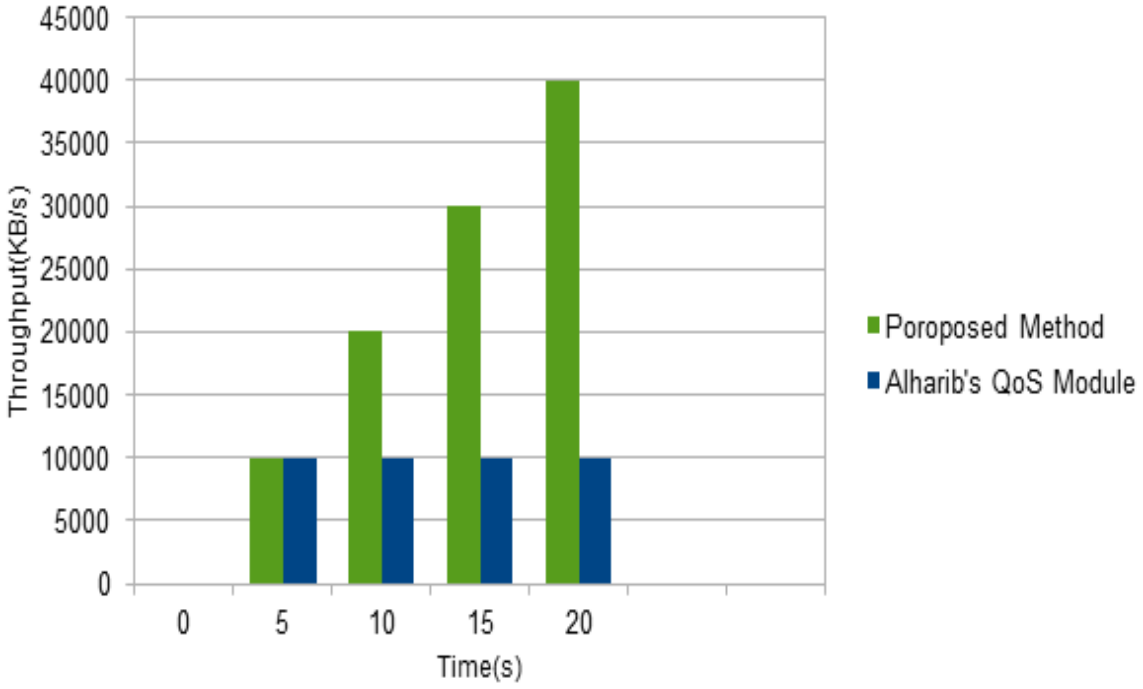


**Figure 4.3: Throughput of Proposed Method Compared with Alharib's QoS Module**

The above graph illustrates that the proposed method is better than Alharib's QoS module interms of average throughput when packet transmission took place in the same number of hops and in the same time interval. The average throughput of Alharib's QoS module kept constant to **9.5MBPS**, but as it is shown in the graph above, the throughput of the proposed method often increases as time goes increasingly. The maximum throughput that the proposed system achieved is **40MBPS** which is three fold better than Alharib's QoS module. The subsequent graph presents the throughput of proposed method in link

failure scenario compared with Yaacob, et al.'s model in link failure scenario. We ran link failure scenario by shutting down the one link among many links that connect hosts with openvswitches, and letting hosts to communicate one another using other than the shortest path. In link failure scenario, our proposed method is much better than Yaacob, et al.'s model i.e. the proposed model achieved a constant throughput of **44MBPS** within **20MS** time interval, whereas Yaacoob, et al.'s model is often varying, and the maximum throughput of their model is below **41MBPS.**

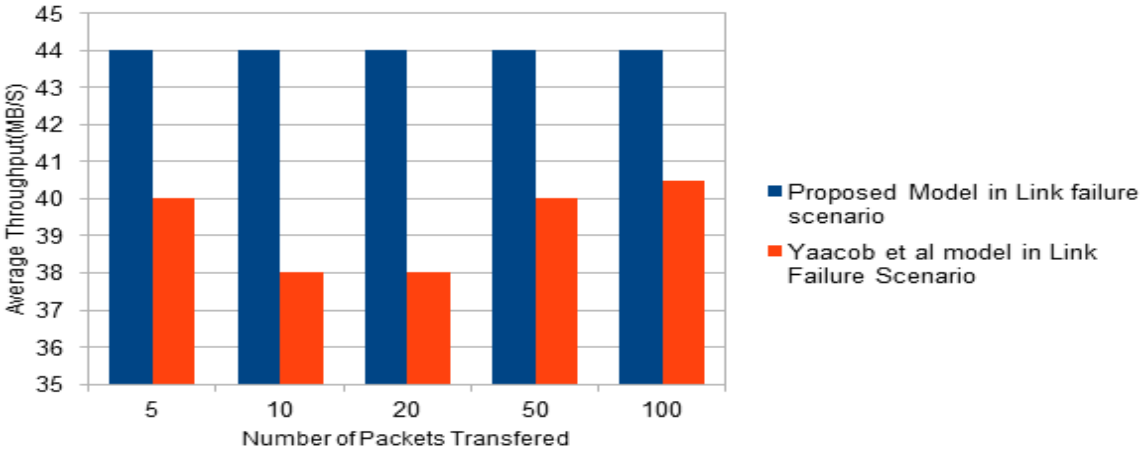


**Figure 4.4: Throughput of Proposed Method in Link Failure Scenario**

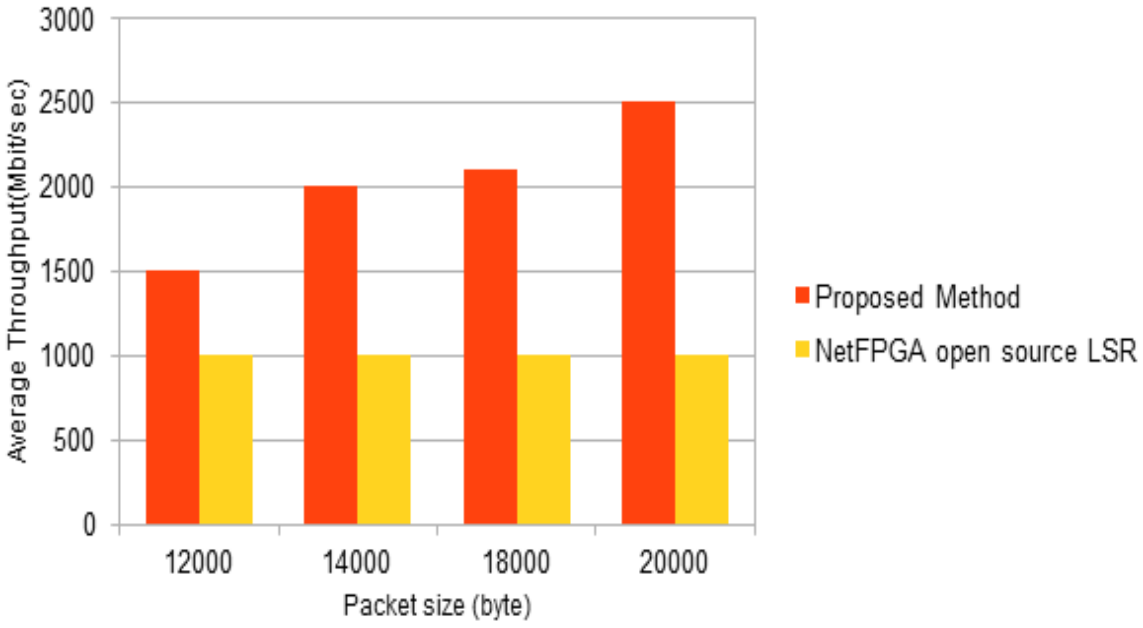


**Figure 4.5: Throughput of Proposed Method over NetFPGA Open Source LSR**

From the above figure, we can conclude that our proposed method is much better than NetFPGA open source label switching router since the throughput of the proposed method increases as the number of byte increases, and the average throughput reaches **2500MBPS**. But in case of NetFPGA, the throughput stays constant although the number of bytes increases.

The main reason for scoring this much amount of throughput is due to the fact that bandwidth isolation enabled us to use the dedicate bandwidth that mininet provides efficiently and even more. Mininet provides up to one thousand (1000) Megabits per second bandwidth to be used by every link, as a result, every link could have a maximum capacity of one thousand Megabits per second. The bandwidth isolation scheme helped us to use the allocated bandwidth efficiently in the entire network, and then it improved the throughput of the entire network. Not only this, the ability of handling packets on both the data plane and the control plane also led the entire packet transmission process to be taken place faster and faster; as a result, the number of packets transferred per second increased.

**Impact of Number of Queues on Throughput**

In this thesis, we investigated the impact of the number of queues on throughput. To observe its impact, we assigned different number of queues for different occasions. In our experiments, we enabled the sender to send as many packets as the receiver can receive at a time. To do so, we used the different window size of the receiver, and we often raised the queues size to observe its effect. With this fact, we found that number of queues has a great effect on throughput; the following two graphs clearly demonstrate the result obtained from two scenarios.

The following two graphs clearly illustrate that as the number of queues increases, throughput decreases. Figure 4.6 shows throughput obtained when the queues size is 6.97kilobyte, and figure 4.7 shows the throughput obtained when the queues size is 8.73kilobyte. From the second graph, we can observe that the throughput decreased by nearly by 25%.

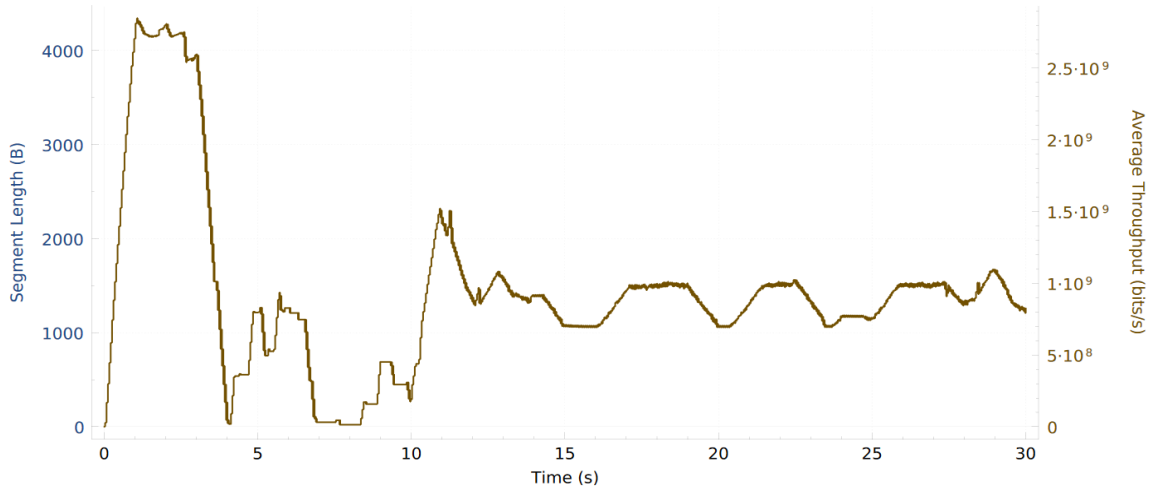


Figure 4.6: **Throughput of 6.97kilobyte queue size**

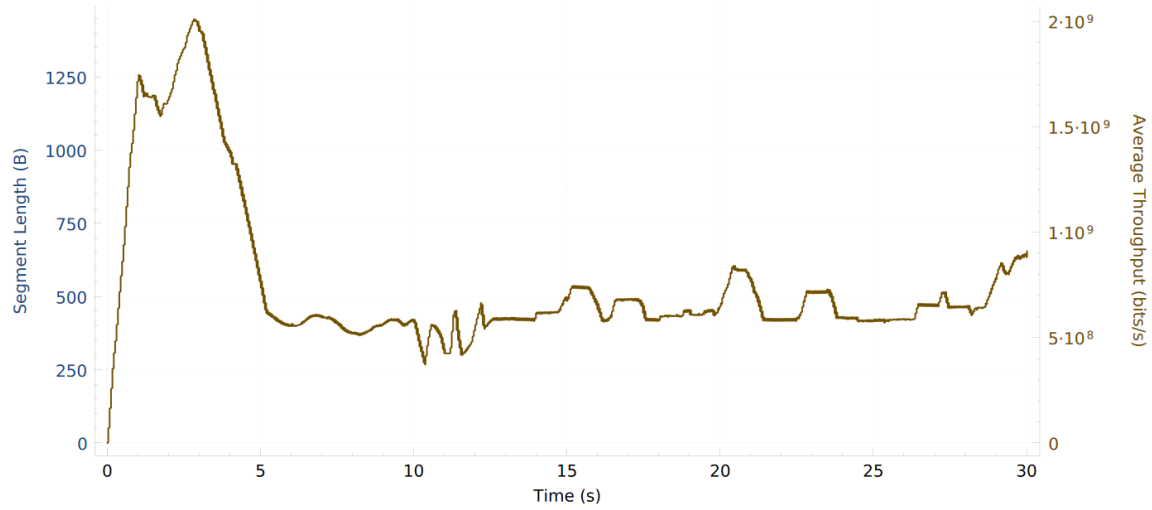


**Figure 4.7: Throughput of 8.73kilobyte queue size**

### 4. 2. 2. 1. Effect of Link Delay on Throughput

Here, we investigated the effect of link delay on throughput; we have experimented only on link delay implies, the effect of propagation delay, queuing delay, inter-device and intra-device delay are not included. To observe its impact, we ran experiments in to two cases namely, 1) when the link latency is zero and 2) by assigning different link delay for every link, this ensures observing the effect of link delay over queuing delay, propagation delay and intra device delay.

To measure the effect of link delay on throughput, we often increased and decreased the link delay for ten iterative experiments, then we have got that slight change in link latency doesn't affect the throughput; the result is recorded when the performance of the hardware was good. We ran experiments starting from link delay **1ms** to **1000000ms**. Within this range, we observed that slight increment in link delay doesn't have significant effect on throughput; link delay ranging from **1ms-10000ms** doesn't have visible effect on the throughput of the entire network.

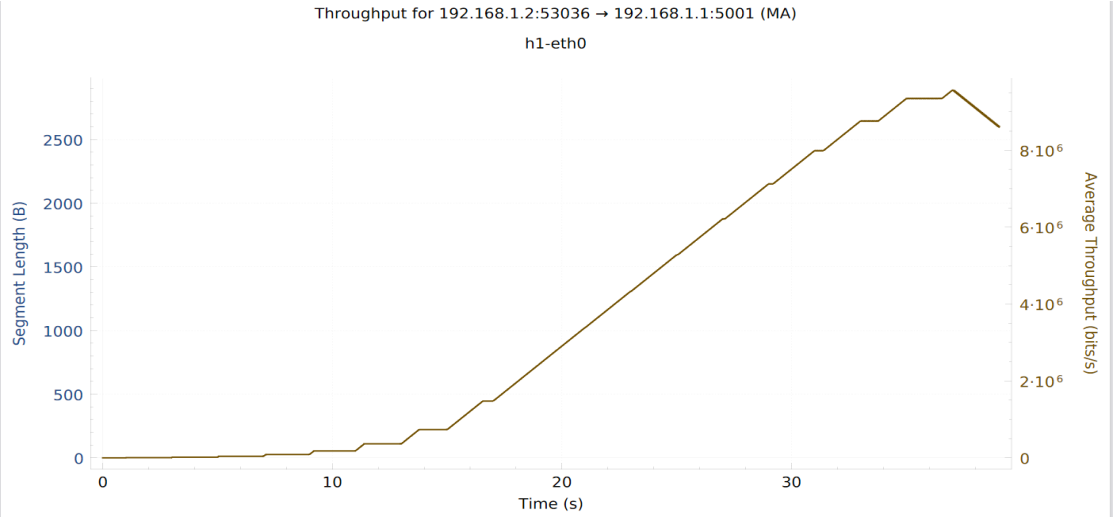


**Figure 4.8: Effect of Link Delay 500000ms on Throughput**

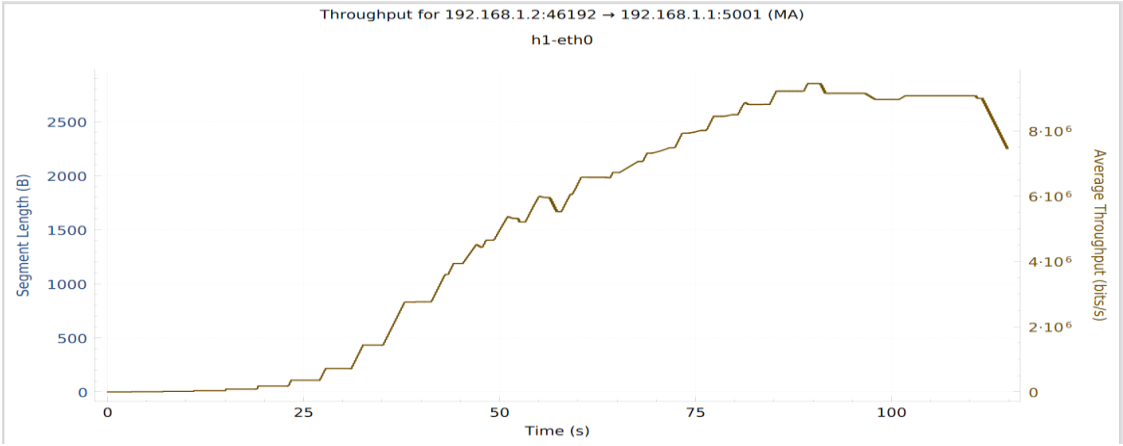


**Figure 4.9: Effect of Link Delay 1000000ms on Throughput**

The above two graphs, demonstrate that link delay has a great impact on throughput. We ran our experiments with link delay of **1m-10000m**, **500000ms** and **1000000ms**, and we found that as link delay increases, throughput decreases. As shown in the graphs above, the throughput of a network with a link delay of **1000000ms** drops by roughly **40%.**

### 4. 2. 3. Delay Variation

Delay variation/jitter has significant effect on packet transmission i.e. it would affect the throughput and other parameters, and even delay variation could be affected by many factors, such as queuing delay, propagation delay, inter-device and intra-device delay and link capacity. To investigate about jitter, we took jitter as an independent parameter separately from the effect of other parameters affecting delay variation, and we measured jitter on two occasions i.e. 1) when there is no need of sacrifices for certain link delay and 2) when there is a need of sacrifices for certain link delay.

For the first occasion, we assigned different link delay for links between communicating devices, then we measured the outcome for ten consecutive experiments. We measured the jitter within ten iterative experiments for each occasion, and we have taken the maximum value measured since it has high discriminant power. Within every experiment, we observed significant reduction of jitter compared with formal method, however, the result often varied due to internal and external factors associated with the hardware.

In the second circumstance, we examined every link to reduce delay variation. We tried to reduce the latency of every link and to let them be quicker while transferring packets by assigning high speed and high data rates. We made the latency and jitter of every link zero so as to reduce overall delay variation. Even handling packets on two places (data plane and control plane) reduced the delay variation significantly when there is simultaneous transmission of packets among all group of users. This is why the queuing delay becomes minimal when there is too many number of packet transmission.

The following figures illustrate the maximum delay variation measured in ten iterative experiments using this approach compared with formal methodologies. The first figure illustrates that MPLS with policy-based routing has reduced the delay variation compared with the formal method whenever there is the need of sacrifices for possible link latency, and the second figure demonstrates MPLS with policy-based routing is a better way of reducing delay variation across the entire network when there is no need of sacrifices for link latency.

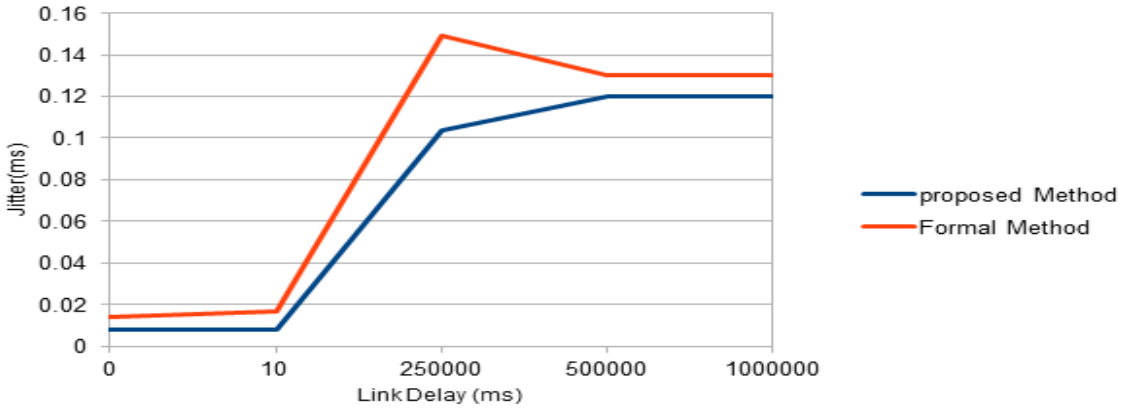


**Figure 4.10: Jitter of Proposed Method**

From this graph, we observe that the proposed method is better than formal method in terms of delay variation. Our approach reduced the delay variation on average by **0.0124ms;** the maximum reduction of jitter achieved by our MPLS with policy-based routing is **0.05ms**, and the minimum is **0.001ms**. The subsequent figure shows the jitter of proposed method in comparison with formal method when there is no need of sacrifices for link delay.

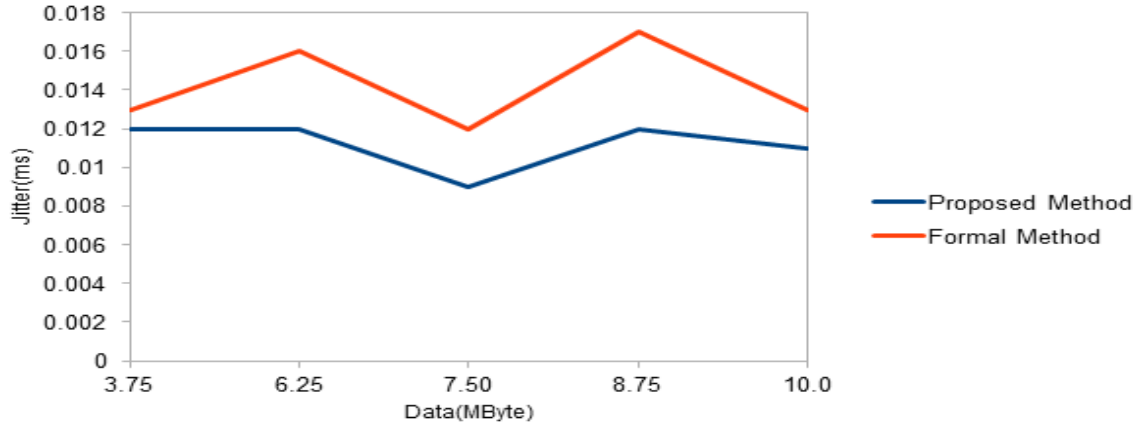


**Figure 4.11: Jitter in Comparison with Formal Method According to Data Size**

In the above graph, it is clear that MPLS with policy-based routing reduces the delay variation of SDN considerably. On average, MPLS with policy-based routing reduces the delay variation by **0.003ms** when there is no need of sacrifices for possible link latency.

#### 4. 2. 3. 1. Effect of Link Delay on Jitter

Here, we investigated the effect of link delay on jitter over other factors including propagation delay, queuing delay, intra-device delay and inter-device delay. To observe the effect of link delay on jitter clearly, we ran experiments into two scenarios namely, when the link delay is zero and when there is variation in link delay other than the threshold. We inspected every link in the entire network to let them have different link latency. And we have ensured link delay has a significant effect on overall delay variation. The result presented here is the maximum jitter measured within ten iterative experiments; the result often varies due to the fact that the performance of the hardware matters: issues related with the hardware, amount of battery, humidity and other internal and external factors matter on. With this approach, we have got the following result.

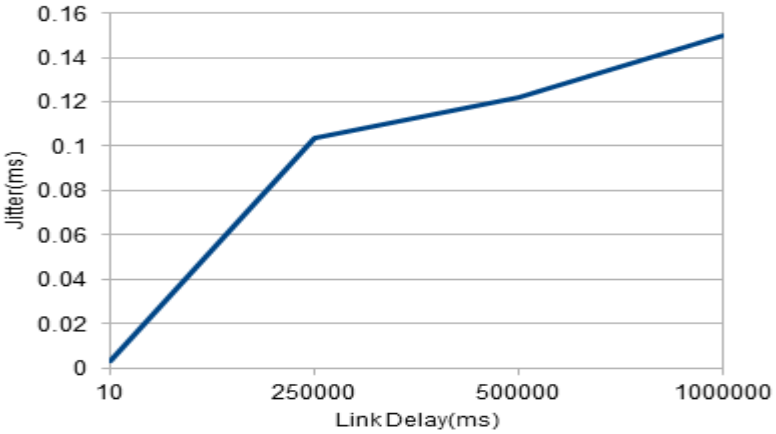


**Figure 4.12: Effect of Link Delay on Jitter**

The above graph clearly illustrates as link delay increases, the delay variation increases. Here, we only considered link delay not the whole latency caused by queuing delay, propagation delay, and inter-device and intra-device delay occurred during packet transmission. When we investigate the effect of link delay on delay variation, we kept other metrics and independent variables constant in order to ensure the change is caused only by link delay.

### 4. 2 .4. Bandwidth Isolation

Different links could have different capacity, but in SDN, every link has equal capacity by default, and this has significant effect on bandwidth utilization. To solve this problem, we divided the type of users into three different groups according to the type of network applications they use, and based on their priority as top level users, medium level users and low level users. With this fact, top level users have been given more fraction of dedicated bandwidth, medium level users have been given fraction of dedicated bandwidth less than that of high level users and greater than that of low level users. Having done this, we could effectively allocate fraction of bandwidth in the network effectively and efficiently.

We conducted experiments on bandwidth utilization in to two cases. First, we measured bandwidth utilization without bandwidth isolation, and in the second case, we applied static bandwidth isolation for every link in the entire network. Having done this, we observed that bandwidth isolation improves bandwidth utilization in the entire network. The result is summarized in the following graph.

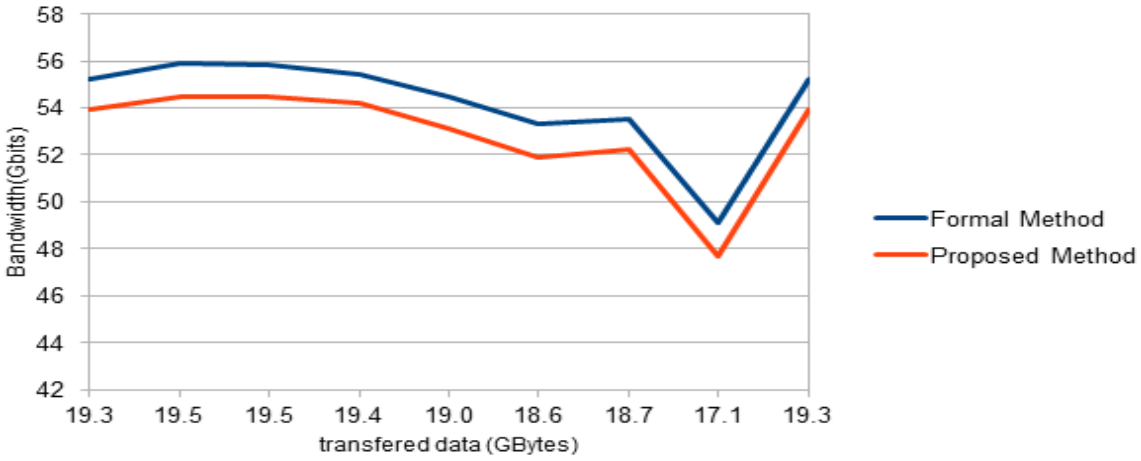


**Figure 4.13: Effect of Bandwidth Isolation on Bandwidth Utilization**

The above graph demonstrates that bandwidth isolation improved the bandwidth utilization. Our proposed method improved the bandwidth utilization on average by **0.0133%**. The maximum improvement of this technique is by **0.014%** and the minimum improvement is **0.0122%.**

### 4. 2. 4. 1. The Relationship between Link Delay and Bandwidth Utilization

We ran fifteen iterative experiments to observe the effect of link delay on bandwidth utilization, and we ensured that link delay has a great impact on bandwidth utilization throughout the entire network. To investigate the impact of link delay on bandwidth utilization over other independent variables, we kept other independent variables including number of queues, jitter and switch capacity constant, then we often varied the link latency ranging from **1ms-100000ms**; within this range, we observed that link latency ranging from **1ms-10000ms** has no visible effect on bandwidth utilization, whereas link delay ranging from **20000ms 100000ms** has a great impact on bandwidth utilization. Although we applied static bandwidth isolation, we couldn't use the fraction of bandwidth fully on that link having link latency greater than or equal to **20000ms**, link latency potentially decreases the bandwidth of the link. The result is summarized in the following graph.

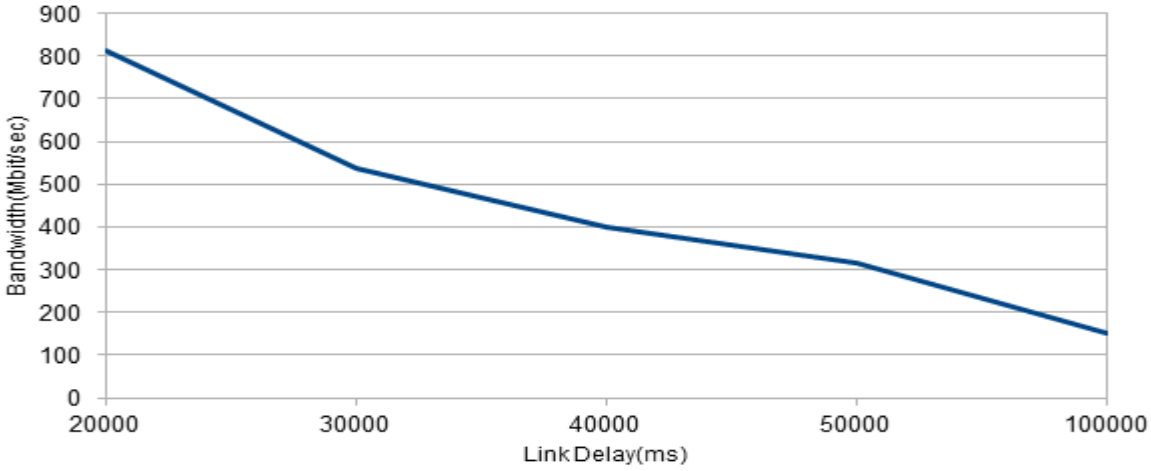


**Figure 4.14: Impact of Link Delay on Bandwidth Utilization**

## 4. 2. 5. Packet Loss

Using MPLS and policy-based routing cooperatively leads for ensuring successful delivery of packets in the entire network. Following successful transmission of packets, the packet loss is reduced significantly; when there is simultaneous transmission of packets among all group of users, the controller does not encounter severe overloads implies MPLS takes over low level packet processing on the data plane. With these

advantages, the maximum packet loss experienced is measured to be **9.5%** which is the maximum TCP packet loss recorded, and it is good according to the standards presented in the table 4 above. The following graph presents the packet loss measured for different size of data.

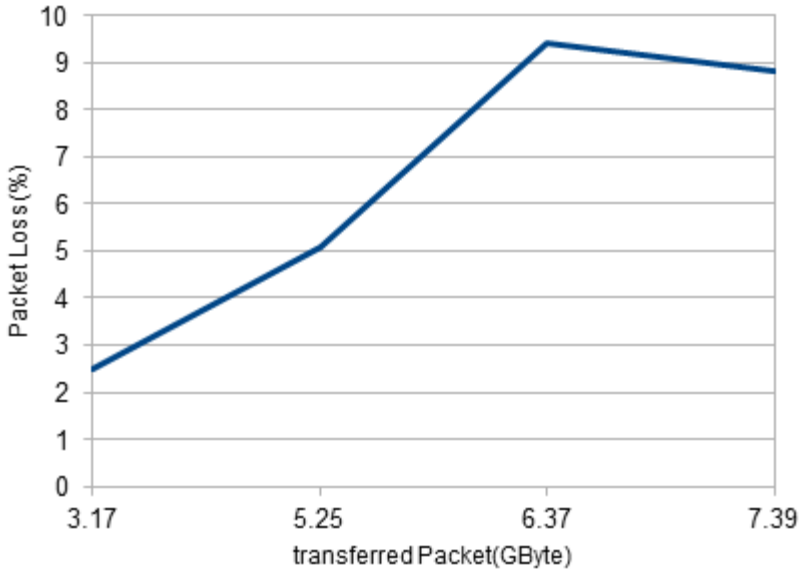


**Figure 4.15: TCP Packet Loss Rate of Proposed Method**

The reason why this much packet losses occurred is that due to TCP window size full advertisement sent by the recipient even when the network capacity is high. This is caused by the limitation of the hardware since the virtual machines share its resources, and it needs further investigation why zero TCP windows size is advertised although there is free window to receive packets.

#### 4.2.5.1 Impact of Link Delay on Packet Loss

We tried to measure and observe the effect of link delay on packet loss by letting the links to have different link latencies. We ran many iterative experiments with link delay ranging from **1ms-1000000ms**; in this range, we observed no effect on packet loss while it increases the retransmission throughout the entire network.

# Chapter Five: Conclusion and Recommendations

## 5. 1. Conclusion

In this thesis, we tried to improve the performance of software defined network. In our study, we have used MPLS and policy-based routing together, and we investigated the effect of number of queues, switch capacity, the performance of controllers and the interdependency among QoS metrics. We measured parameters, such as latency, throughput, jitter, bandwidth utilization, and packet loss.

We ran different experiments for different scenarios to analyze the effect of one parameter on other parameters and in order to observe the outcome. We classified users as low level, medium level and high level users, and we tried to handle packet processing on both the data plane and the control plane; we enabled MPLS to handle low level traffic flows on the data plane, and the remaining medium and high level traffic flows were handled on the control plane.

We investigated the relationship between QoS metrics, such as the interdependency between link latency and throughput, the relationship between jitter and link latency, the effect of link latency on bandwidth utilization and the impact of link latency on packet loss by taking them separately from being affected by other independent variables.

We investigated the effect of bandwidth isolation on improving overall bandwidth utilization throughout the entire network, and we found that bandwidth isolation improved bandwidth utilization and even it enhanced the throughput. Associated with bandwidth, we even investigated the effect of link latency on bandwidth utilization starting from link latency **1ms-1000000ms.** Within this range, we found that slight difference in link latency has no measurable effect on bandwidth utilization.

We also investigated the impact of the type of switches on network performance, and we found that there is slight difference interms of minimum, maximum and average RTT among OVS, OVSbr and user space switches. When we were running experiments, we used IPFIX protocol to collect network statistics, and we ensured that IPFIX tool is a very vital tool to export network statistics. From the above results, we can draw the following conclusions.

- Each type of switch (User space switch, OVS and OVSbr) has its own strong sides, but OVSbr is better than in interms of average RTT.
- The proposed method is better than Yaacob, et al.'s and Hossain, et al.'s model model by average RTT of **0.19ms** and **0.5ms** respectively**.**
- The throughput drops roughly by **40%** when link delay increased from **500000ms** to **1000000ms**.
- Link delay ranging from 1ms-10000ms have no significant effect on throughput.
- Bandwidth isolation improved the bandwidth isolation by **0.013%.**
- The proposed method reduced the delay variation by **0.0124ms**.
- Link delay has significant effect on delay variation independent of propagation delay, queuing delay, inter-device and intra-device delay.
- Link delay has no effect on packet loss.
- Link delay ranging from **1ms-10000ms** has no visible effect on bandwidth utilization.
- Policy-based routing is an efficient way of reducing the latency on the control plane, and MPLS achieves incredible result in reducing end to end latency on the data plane.
- Link latencies ranging from 20000ms to 1000000ms has measurable effect on bandwidth utilization, whereas link delays ranging from 1m-10000ms has no visible effect on bandwidth utilization.

Generally speaking, the integration of MPLS as packet processing scheme on the data plane is a valid method to enhance the overall performance of SDN. MPLS reduces the loads of the controller in such a way that it handles low level packet flows by the help of openvswitches that acts as standalone devices. Our proposed method has improved

bandwidth utilization in the entire network. It reduced the latency and jitter, and it improved throughput compared with works in this area done recently. This work also investigates the relationship between QoS metrics, and it finds out the interdependency between QoS metrics has significant effect on the entire performance of SDN.

## 5. 2. Future Works

In this thesis, we investigated a lot of issues including the impact of the type of switches, impact of queue size, the effect of bandwidth isolation, the interdependency between QoS metrics, and other parameters, such as latency, jitter, throughput and packet loss. From our findings, we would like to recommend the following future works.

- Why different type of switches has different occasions of achieving good performance should be investigated further.
- We used MPLS to handle packets on the data plane through GNS3 routers, but it would be better to deploy MPLS as an application on the top of the controllers.
- We applied static bandwidth isolation to improve bandwidth utilization in the entire network, but we found it hard to be applied for large networks so, on demand bandwidth isolation should be used in such a way that it doesn't incur significant load on the controllers, and it becomes feasible enough to manage the bandwidth of the entire network according to the desire of the owner of the network.
- We apply policies only to process packets on both the data plane and the control plane so, it is important to have policy-map to enable full policy-based routing.
- We worked on virtual machines, as a result, there are no enough resources to run the applications, and it had a certain effect on our experimental result so, we recommend working on real machines.
- Certain packet scheduling scheme should be applied in order to minimize the maximum RTT that some packets experienced across the entire network. Especially, during path computation and route advertisement.
- We worked on simple linear and custom topologies so, it is important to test our approach on more complex topologies.

# REFERENCES


Sam , K. (2020, July 28). *Gigamon*. Retrieved December 16, 2020, from gigamon.com: https://blog.gigamon.com/2019/09/17/ipfix-vs-netflow/

Adnan, A.-T. A. (2000). MPLS-Based Performance Modeling Using Traffic Engineering Approach to Improve QoS Routing on IP Networks. *IEEE Networks*.

ALAXALA Networks Corporation. (2012). *AX-Series Policy-based Routing Configuration Guide.* ALAXALA Networks Corporation.

Alemayehu, K. (2019, December). Analyzing Impact of Segment Routing MPLS on QoS(Unpublished Master's thesis). Addis Ababa University, Addis Ababa, Ethiopia.

Alexandre , O. T., Bruno , C. J., Marcelo, M. F., Alex , V. B., Antonio , G. T., & Artur , Z. (2019). SDN-Based Architecture for Providing QoS to High Performance Distributed Applications. *International Journal of Network Management*.

Astuto, B. N., Mendonça, M., Nguyen, X. N., & Obraczka, K. (2014). A Survey of Software-Defined Networking: Past,Present, and Future of Programmable Networks. *IEEE*, 1617 - 1634.

Azodolmolky, S., Nejabati, R., Escalona, E., Jayakumar, R., Efstathiou, N., & Simeonidou, D. (2011). Integrated Openflow–GMPLS control plane: an overlay model for software defined packet over optical networks. *Optics Express*, B421-B428.

Bellessa, J. (2015). *Implementing MPLS with Label Switching in Software-Defined Networks(unpublished master's thesis).* University of Illinois, Urbana, Illinois, USA.

Bhat, D., Rao, K., & N R, L. (2015). A Survey on Software-Defined Networking Concepts and Architecture. *International Journal of Science, Technology & Management*, 123-141.

Burno, N. A., Marc, M., Xuan , N. N., Katia , O., & Thierry , T. (2014). A Survey of Software-Defined Networking: Past,Present, and Future of Programmable Networks. *IEEE*, 1617-1634.

Chen, R. S., Tsai, Y. S., . Yeh, K., & Chen, H. (2008). Using Policy-based MPLS Management Architecture to Improve QoS on IP Network. *WSEAS Transactions On Computers*.

Cisco. (2019). WSEAS Transactions on Computers. Inc. A. quarter, *IP Routing: Protocol-Independent Configuration Guide, Cisco IOS* (pp. 217-236). San Jose: Cisco Systems, Inc.

Coker, O., & Azodolmolky, S. (2017). *Software-Defined Networking with OpenFlow - Second Edition.* Packt Publishing.

Faisal , A. (2018). SDN-Based Mechanisms for Provisioning Quality of Service to Selected Network Flows(Doctorial Dissertation, University of Kentucky, Lexington, Kentucky, USA). Retrived from https://www.semanticscholar.org/paper/SDN-BASED-MECHANISMS-FOR-PROVISIONING-QUALITY-OF-TO-Alharbi/b640743c0968e328c880b00d0b02858c3bfd43ff

Jacob , A. (2015). *Performance Management in Software Defined Networking(Unpublished Master's thesis).* University of Gothenburg, Goteborg, Sweden.

Karamjeet , K., Japinder , S., Navtej , S., & Ghumman. (2014). Mininet as Software Defined Networking Testing Platform. *International Conference on Communication, Computing & Systems (ICCCS)* (pp. 139-142). Punjab: SBS.

Kempf, J., Whyte, S., Ellithorpe, J., Kazemian, P., Haitjema, M., Beheshti, N., et al. (2011). OpenFlow MPLS and the Open Source Label Switched routers. *23rd International Teletraffic Congress* (pp. 8-14). International Teletraffic Congress(ITC).

Khalili, R., Despotovic, Z., & Hecker, A. (2018). Flow setup latency in SDN networks. *IEEE Journal on Selected Areas in Communications*, 1-9.

Larry, P. (2018, July 15). *United Global Group*. Retrieved december 15, 2020, from www.unitedglobalgrp.com: https://www.unitedglobalgrp.com/uncategorized/software-defined-networks-a-history/

Latif, Z., Sharif, K., Li, F., Karim, M. M., & Wang, Y. (2019). A Comprehensive Survey of Interface Protocols for Software Defined Networks. *Journal of Network and Computer Applications*, 1-30.

Lawal, A., Bull, P., & Abdallah, A. (2017). Performance Implication and Analysis of the OpenFlow SDN Protocol. *international conference on Advanced Information Networking and Applications Workshops.* IEEE.

Maim, S., N , Y., A , A., R B , A., M N M , W., & Z , I. (2019). Performance analysis of Software Defined Network (SDN) in link failure scenario. *IOP Conference Series: Materials Science and Engineering* (pp. 1-5). Bogor: IOP Publishing.

Mamushaiane, L. (2019). *Classification of SDN distributed controller approaches: a brief overview*.

Manmeet , S., Nitin , V., Jobanpreet , S., & K. , H. (2018). Estimation of End-to-End Available Bandwidth and Link Capacity in SDN. *International Conference on Ubiquitous Communications and Network Computing* (pp. 130-141). Pilani: Springer, Cham.

Marc, W. (2020, january 15). *pcwdld*. Retrieved December 16, 2020, from www.pcwdld.com: https://www.pcwdld.com/what-is-ipfix

Mittal, S. (2017). Performance Evaluation of Openflow SDN Controllers. *International Conference on Intelligent Systems Design and Applications* (pp. 913-923). Springer.

Mohammad , T. M., Behzad , A., Nader , M., & Luca , C. (2018). SDN-Based Resource Allocation in MPLS Networks: A Hybrid Approach. In M. M. Tajik, B. Akbar, N. Mokari, & L. Chiaraviglio, *concurrency and computation practice and experience* (pp. 1-11). 2018 John Wiley & Sons, Ltd.

open networking foundation. (2017). *OpenFlow Switch Specification.* open networking foundation.

Paliwal, M., Shrimankar, D., & Tembhurne, O. (2018). Controllers in SDN: A Review Report. *IEEE*, 1-14.

S. M., S., Sujan , B., Hossain, A., Mohammad, A. L., & Ariful, A. (2018). ENHANCING AND MEASURING THE PERFORMANCE IN SOFTWARE DEFINED NETWORKING. *International Journal of Computer Networks & Communications (IJCNC)*, 27-40.

Salman, O., Elhajj , I., Kayssi, A., & Chehab, A. (2016). SDN Controllers: A Comparative Study. *18th Meditranian Electronical conference* (pp. 1-6). IEEE.

Sugeng, W., Istiyanto, J. E., Mustofa, K., & Ashari, A. (2015). The Impact of QoS Changes towards Network Performance. *International Journal of Computer Networks and Communications Security*, 48-53.

Themba, S., Sabelo, S., Matthew, O., & Pragasen, M. (2019). An SDN Solution for Performance Improvement in Dedicated WideArea Networks . *Conference on Information Communications Technology and Society (ICTAS).* Pretoria: IEEE.

Tootoonchian, A., Gorbunov, S., Ganjali, Y., & Casado, M. (2012). On Controller Performance in Software-Defined Networks. *2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services.* San JOSE: Hot-ICE '12 and USENIX.

Xenofon , F., Mahesh, M. K., & Kimon , K. (2015). Software Defined Networking Concepts. In L. Madhusanka, G. Andrei , & Y. Mika , *Software Defined Mobile Networks (SDMN): Beyond LTE Network Architecture* (pp. 21-44). John Wiley & Sons, Ltd.

Zhu, L., Karim, M. M., Sharif, K., Li, F., Du, X., & Guizani, M. (2019). SDN Controllers: Benchmarking & Performance and Evaluation. *IEEE*, 1-13.

# APPENDIX

## Initiative questions

Where is the state of the art of software defined network now?

What are the hot issues to be explored in software defined network?

How researchers tried to address the performance problem in software defined network?

What are not considered yet in improving the performance of software defined network?

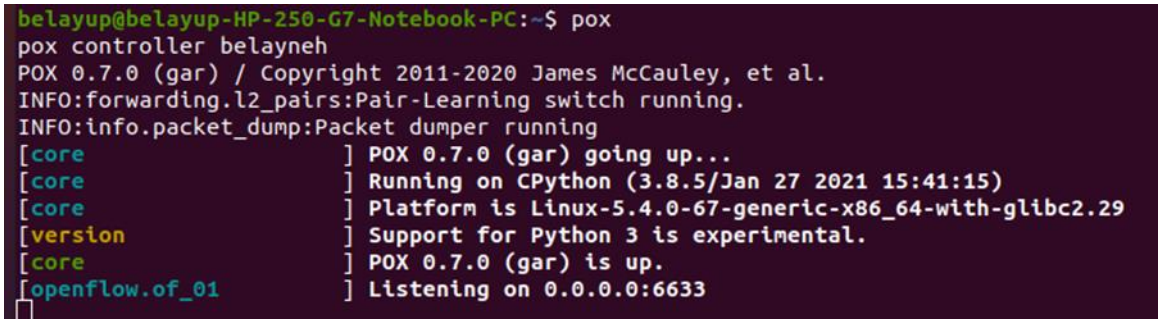


**Figure 1 Starting Pox Controller**

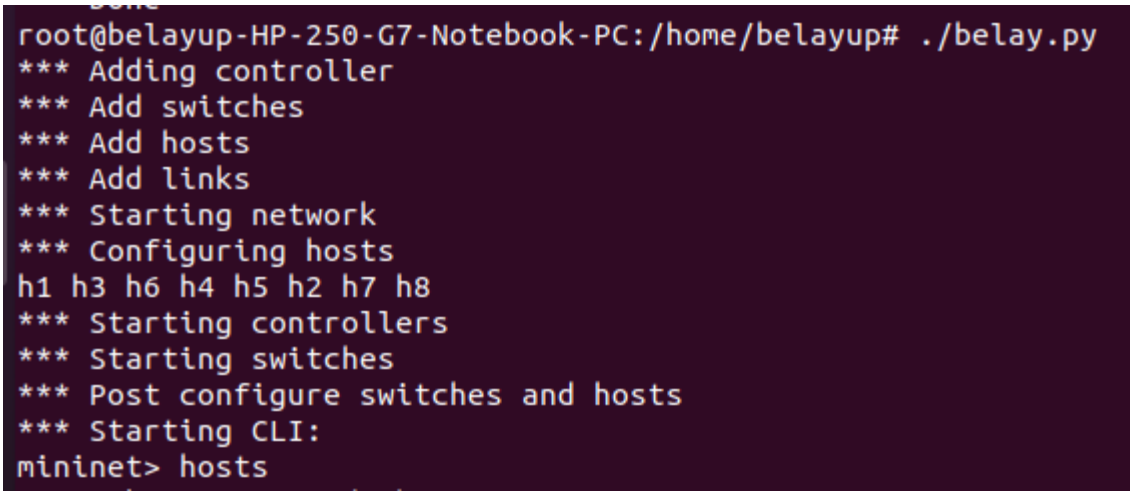


**Figure 2: Creating Network**

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h3 h6 h4 h5 h2 h7 h8
h3 -> h1 h6 h4 h5 h2 h7 h8
h6 -> h1 h3 h4 h5 h2 h7 h8
h4 -> h1 h3 h6 h5 h2 h7 h8
h5 -> h1 h3 h6 h4 h2 h7 h8
h2 -> h1 h3 h6 h4 h5 h7 h8
h7 -> h1 h3 h6 h4 h5 h2 h8
h8 -> h1 h3 h6 h4 h5 h2 h7
*** Results: 0% dropped (56/56 received)
mininet>
```

**Figure 3: Testing Connectivity**

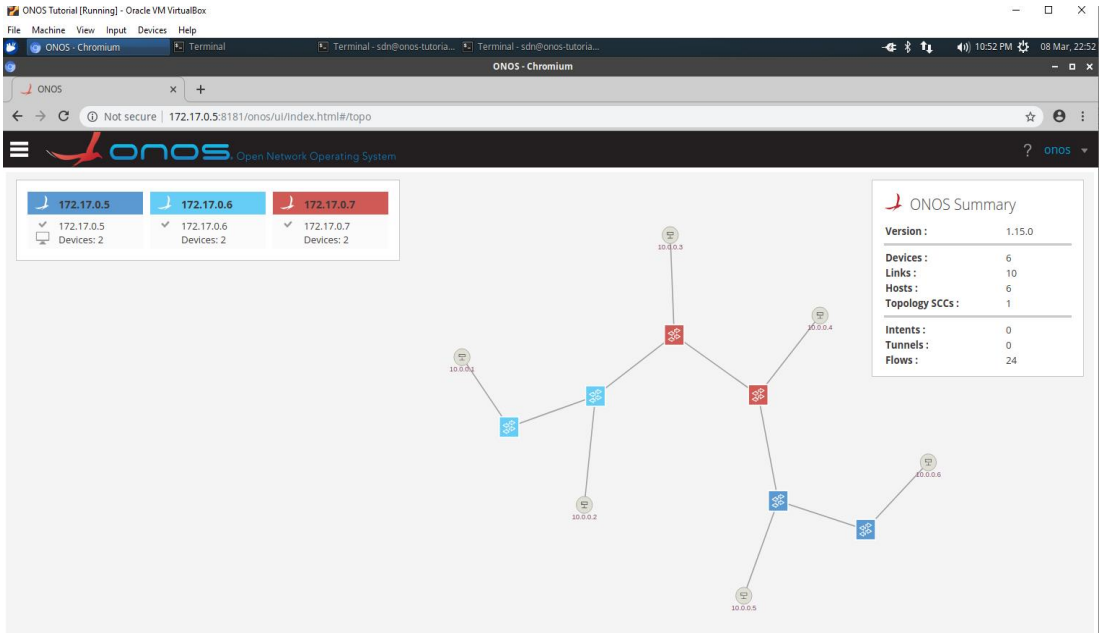


**Figure 4: Topology in ONOS Controller**

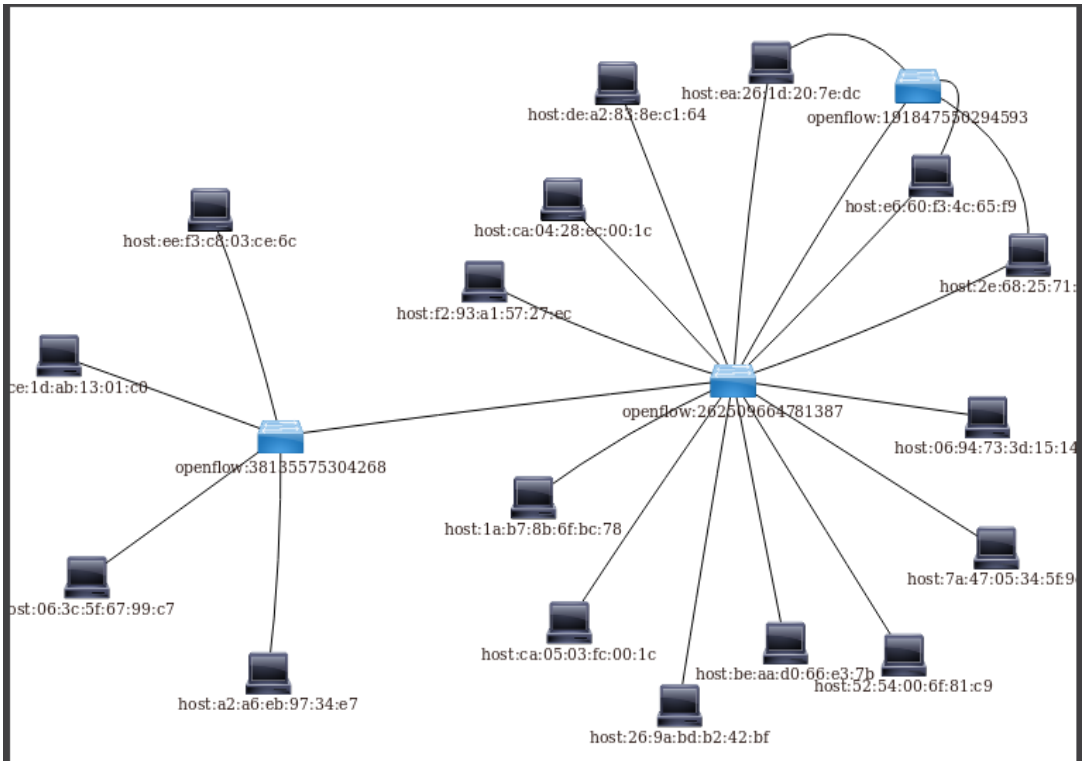


**Figure 5: Topology with Opendaylight Controller**