Thesis no: MSEE-2016-27

OpenFlow Software Switch Performance on Network Simulator-3

Network, Network Performances Evaluation, Network Simulator

SRIRAM PRASHANTH NAGURU

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the Degree of Masters in Electrical Engineering with Emphasis on Telecommunication System. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author(s):
SRIRAM PRASHANTH NAGURU
Srnb15@student.bth.se

University Advisor:
Dr. Yong Yao
Department of Communication Systems

University Examiner:
Dr. kurt Tutschku
Department of Communication Systems

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

ABSTRACT

Context. In the present network inventive world, there is a quick expansion of switches and protocols, which are used to cope up with the increase in customer requirement in the networking. With increasing demand for higher bandwidths and lower latency and to meet these requirements new network paths are introduced. To reduce network load in present switching network, development of new innovative switching is required. These required results can be achieved by Software Define Network or Traditional layer-3 technologies.

Objectives. In this thesis, the end to end transmission performance of OpenFlow and Layer-3 switches are investigated and studied using simulation. The dynamic behavior of OpenFlow and layer-3 switches motivated to study in detail their behavior and motivated this research.

Methods. To replicate real life network topology and evaluate e2e transmission performance, a simulation based test-bed is implemented for both OpenFlow switch and layer-3 switch. The test beds are implemented using Network Simulator-3 (NS3). A two-tire network topology is designed with specified components for performance evaluation.

Results. The performance metrics like throughput, average delay, simulation time and PDR are measured, results are analyzed statistically and have been compared. The behavior of network traffic in both the topologies are understood using NS-3 and explained further in this document.

Conclusions. The analytical and statistical results obtained from simulation output helped me to choose better switch. From the simulations conducted and observations done, it is concluded that OpenFlow switch performs relatively better when compared to the layer-3 switch.

Keywords: OpenFlow switch. Layer-3 switch, Network Simulator-3, SDN.

ACKNOWLEDGMENT

Firstly, I would like to express my sincere gratitude to Professor Dr. Yong Yao for his help and support behind the thesis. I have learnt a lot from him in the whole process which will be help me in my career and the research work have been successful with his support and guidance throughout the thesis. I would like to thank my parents, my friends and my well-wishers for their unconditional love and support. I am thankful to the all the viewers who played an active role in the successful completion of my thesis.

-Naguru Sriram Prashanth.

ABBREVIATIONS

API - Application Program Interface

AIM - Advanced Integration Module

BGP - Border Gateway Protocol

CSMA - Carrier Sense Multiple Access

GUI - Graphical User Interface

GNU GPL - GNC-General Public License

IP - Internet Protocol Address

LAN - Local Area Network

LLC - Logical Link Control

LSA - Link State Advertisement

MPOA - Multi Protocol Encapsulation over ATM

MPLS - Multi Protocol Label Switching

MAC - Media Access Control

NS-3 – Network Simulator-3

OSPF – Shortest Path First

OFSID – OpenFlow Switch Library

PDR – Packet Delivery Ratio

QOS – Quality of Services

SI – Software Independences

SNMP – Simple Network Management Protocol

SDN – Software Defined Network

TCAMS – Ternary Content Addressable Memory

TCP – Transmission Control Protocol

UDP – User Datagram Protocol

VLAN – Virtual Local Area Network

WAN – Wide Area Network

TABLE OF CONTENTS

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

In the present telecommunication systems, network switching and switching techniques are growing rapidly and communication between the switches in networks are becoming more and more complex. Due to advanced growth in network traffic and adversity to handle the flow, there is a need for new switching models and an efficient way to approach the present trend switching to control the traffic of a network.

The Layer-3 traditional switch is the upgraded version of layer-2 switch. In the OSI model, a traditional network router promotes layer-3 routing. To improve network routing performances of Local Area Network (LAN), traditional layer-3 routing is proposed. This routing works within the network using IP packets. It is based on destination IP address of a complete network layer-3 switching.

Software-Defined Networking (SDN) is an emerging networking paradigm in the present telecommunication research and industry area to deal with the demand on network flexibility and controlling traffic. SDN architecture includes control plane and data plane. Data plane consists of switches and routers which promote packets transmission in a network, while the control data plan interface acts as an interconnection between controller and switches in the date plane and explains the communication between them. Due to this type of architecture, separation in the network packet forwarding is quite simple for switching under the control of the controller. Controller enforces a control logic to the switches or routers for network structure expansion and modification when network traffic increases.

Create a test bed to our research work includes numerous computers, interconnections between switches and performing the relation to verification and validation of a network is too expensive. There are new simulation techniques for verification and validation to research this kind of expensive networks with less money and time. The researcher can develop a model similar to test bed in the network simulator.

Thus the traditional switch has the similar appearance of the open flow switch makes an interesting research work for my thesis work. Performance comparison of this two switching approaches to be implemented with the help of Network Simulator-3(NS-3). We measure the throughput and average delay to the network switches in this research work.

1.1 Motivation

The present data transmitting network needs better switching systems to handle tremendous traffic generated everyday by millions of users. After a broad literature review on various switching systems, OpenFlow and Layer-3 switches are selected. The flexibility and scalability of these two switches are the feature made them trustworthy. So we are influenced to study open flow switching and traditional layer-3 router networking, as it is one of the most advanced switches and ensure the performance of two switches, which works on decreasing network traffic complexity and have an efficient end to end switching routing and network flexibility.

To evaluate this idea, the end to end transmission performance of OpenFlow switch and traditional switch is compared using simulation.

1.2 Scope of Thesis

This thesis study describes the initial setup of two different networks using OpenFlow switch and layer-3 routers. A performance study of traditional layer-3 router and OpenFlow switch for initial setup network is done using Network Simulator-3. The research study includes the experimentation and measuring the throughput, average delay, simulation time and packet deliver ratio.

A network simulator study helps the author in understanding the network set up, experimentation and evaluating the network for the considered performance metrics. From this performance evaluation, we can have an idea on which switching network is efficient for better controlling of the traffic.

1.3 Problem Statement

Today's most existing works are on network traffic, which is facing many challenges due to network handlers. It is basically impossible to meet the requirements of present organized networks due to lack of accessible interface limits to work on present network architecture.

The network operators are forced to scale their hardware to meet every traffic spikes. Hardware scaling is expensive and most in efficient way of using resources. To solve this problem SDN switching systems like OpenFlow are introduced to reduce time consumption and money invested in installation of new switches. The problem with implementing them is, not every network operator is aware of its advantages. We as telecommunication engineers and researcher should provide enough information and guide lines. Which help network operators adapt to SDN switching systems.

In such critical situation, we focused on OpenFlow switch and traditional layer-3 routers. The issue here is to get familiar with which switch is to use for the present network traffic when both has its own importance in a network.

1.4 AIM and Objectives

The basic ideology of this work is to focus on the communication between the switch and router connected in a network for observing the network traffic.so there is a need for performance metrics based on network traffic monitoring. The idea was implemented in a network model compatible with both the scenarios.

The main goal of this research is to evaluate the performance of traditional layer-3 switch and OpenFlow switch for an end-to-end network topology. The main reason behind selecting these two switches for my research work investigation is because of the energetic attitude towards these switching or routing in the existing network.

-To reach the goal of this project the main objectives are as follows:

1. Understanding the main concepts of OpenFlow switch and traditional layer-3 router through a detailed literature review.
2. Understanding Network Simulator-3
3. Implementation of switch and router in Network Simulator-3 for performances of end-to-end transmission in a network.
4. For the considered performances metrics, test is run for layer-3 router using Network Simulator-3.

5. For the considered performances metrics, test is run for OpenFlow switch using Network Simulator-3.
6. Compare and discuss the obtained results.

1.5 Research Questions

RQ1: How is the link established between end users when the OpenFlow-switch is implemented in Network Simulator-3?

RQ2: How is the link established between end users when the traditional switch is implemented in Network Simulator-3?

RQ3: What are the metrics needed for analyzing the performance of OpenFlow-switch to traditional switch? How can be the performances further be improved?

1.6 Research method

The research method that would be used to conduct this study would be a combination of both literature review and experimental work on OpenFlow and traditional switch.

In the literature review, we gather data about understanding the network routing and switching of a network. Using this data, we would find an efficient switch that could be used for this research work. A detailed study on network simulator could help in conducting our research work.

A network is set up on a testbed for throughput and delay performance metrics. Network simulator is considering as the testbed for the network due to feasibility for collecting the packet data and destination.

The implementation of OpenFlow network is done by installing OpenFlow switch module and controller in Network Simulator-3 using Linux operating system. OpenFlow switch is configured with source and destination nodes using csma module for end-to-end transmission.

The implementation of layer-3 routing network is done by installing the routing module in Network Simulator-3 in Linux operating system. Routers are configured with source and destination nodes using IPv4 forwarding which is static routing API module for end-to-end transmission.

1.7 Thesis Outline

Chapter2 This chapter is about related work. In order to strengthen our background work we referred many research works

Chapter3 This chapter is about background work. This chapter gives a clear explanation about SDN, OpenFlow, traditional switch and network simulator.

Chapter4 This chapter is about method and experimentation of OpenFlow switch and layer-3 router in simulator and functionally of both networks is explained.

Chapter5 this chapter is about numerical results. Results across the both networks are explained with graphical representation and values are tabulated.

Chapter6 this chapter is about conclusion, answering research questions and future work.

2 RELATED WORK

This section presents the related research works done so far, in the area of SDN and OpenFlow.

In [1], OpenFlow data plane performance is observed. OpenFlow is compared with Ethernet switching and IP routing functionalities in a Linux environment. The experiments are carried out on a Linux PC, and a HW traffic generator is used to stress the packet forwarding capabilities of this test PC. Switching and fairness tests are investigated taking throughput and packet latency as the main metrics for performance comparison. The Linux implementation of the layer-2 switching resulted in a 40 percent performance drop compared to layer-3 routing and 30 percent compared to OpenFlow for single 64-byte packet flows. It is observed that the layer-3 routing and the hash table performance are almost match all the time except in cases of system overload due to small packets load, where the throughput slightly degrades. From the results, for 64-byte packets, a performance drop of around 11 percent is observed. For 96-byte packets and 18K forwarding table entries it is reduced to 3.5 percent. It is observed that for the linear table, though the performance is worse than the hash case, full wire speed is achieved for packets starting from 256-bytes. When multiple flows are considered, performance degrades with the increasing size of the forwarding tables. It is observed that the performance degradation is limited for routing and OpenFlow, but has a disastrous effect on switching. While routing and OpenFlow needed at least 256-byte packets to achieve wire speed for small table size, switching needed 1024 bytes.

Benamrane et al. [2], in their work, provide an overview of the main studies and performance enhancement propositions for SDN networks, from the existing literature. The paper describes the concepts of SDN and OpenFlow protocols, and explains the performance of OpenFlow switches and controllers. Various techniques that are used to evaluate the OpenFlow network performances are discussed. It is observed that in SDN/OpenFlow architecture, handling requests of a large number of network devices affect the controller performance. Throughput and latency are taken as the main performance metrics. From the existing literature, it is observed that the OpenFlow switch offers good performance when compared to the traditional switches. They discuss the flow director technique, a software enhancement in which the Flow Director is taken as another lookup table. It is observed that the flow director table size and the hash table size does not affect OpenFlow switch performance and result in an increase of 25 percent throughput when compared to that of the regular software based switching. Hardware based enhancements to the OpenFlow switch, by implementing network processor based acceleration cards to perform the switching showed a 20 percent reduction in the packet delay compared to the traditional designs. For the OpenFlow controllers, virtualization-based and multithreading-based improvements are discussed. FlowVisor and FlowN are compared in scalable virtual networks (>=100 networks) and it is observed that though FlowN has higher latency, it scales better than FlowVisor. Multithreading is observed to improve the controller throughput as it has the advantage to concurrently handle multiple tasks at the same time, thereby making optimal use of the available resources. The authors suggest deeper study and analysis for multiple controller deployments, and coordination tasks across multiple controllers, and their impact on performance and scalability.

Hidell et al. [3], in their research, proposed an architectural design for improving the lookup performance of OpenFlow Linux kernel module

implementation. A 25 percent increase in throughput is observed when compared to the regular software based OpenFlow switching. The key enhancement feature used is the Flow Director, which is used as a lookup shortcut by pre-assigning the received queues to direct the traffic flows. An additional table called the flow director extension table, in addition to the existing hash and linear tables, is used to keep track of the mapping of receive queues to the output ports. The main performance metric used is the packet switching rate (packets per second). Two directly connected PCs, one for generating and receiving traffic, and other as a dedicated switch are used for the experimentation. It is observed from the experimentation results that the performance of OpenFlow switching degrades with the increase in the linear table size, whereas hash table size and the flow director table size has no effect on the switching performance. However, an overall performance degradation is observed when the number of CPU cores is increased to 5, 6 and 7, most likely due to uneven distributed load.

With respect to data plane related research, Farhady et al. [4] point out the gap that needs to be addressed in the data plane related contributions in SDN. Since the mostly addressed in the recent SDN research is the control plane and OpenFlow related work, the authors reviewed the existing technologies and introduced existing blocks of these technologies that focus on building a better SDN data plane. They present a strong case that SDN requires not only control plane, but also data plane programmability for a flexible and comprehensive coverage of applications. Research along the direction of developing APIs similar to OpenFlow for control plane, and platforms and frameworks that help in developing such APIs is suggested. They suggest that developing the API frameworks, like letting the user define switch actions and deploying onto the switch, can speed up the SDN development progress.

In [5] Chaves et al. give details about the OFSwitch13 module, a free software, which enhances the Network Simulator 3 with OpenFlow 1.3 support. Its design and implementation are discussed and a case study is provided to explain the module features. NS-3, a discrete-event simulator is used to model OpenFlow switches through the OpenFlow module. This module implements OpenFlow protocol version 0.8.9, which don't support many of the new features introduced in the latest versions. So, the authors introduced OFSwitch13 module for NS-3, which supports OpenFlow protocol version 1.3, which makes interconnecting NS-3 nodes for sending and receiving traffic using existing CSMA network devices possible. The OpenFlow 1.3 switch network device is used for this purpose. The OpenFlow 1.3 controller application interface is used to provide basic functionalities for controller implementation, like handling the OpenFlow switches. An OpenFlow channel is used to connect each switch to a controller. It is through this interface that the controller manages the switch and configures it. The authors list out the limitations for the module which include its availability for only GNU/Linux platforms, lack of in-band control (since the controller can only manage the switches over a separate dedicated network). Also, only a single connection is available between the controller and the switch.

3 BACKGROUND WORK

This chapter gives an introduction to Software Defined Networking (SDN), OpenFlow technology which is the main aim of this project. A brief description of layer-2 switch, layer-3 switch and routing are provided in this chapter. Also, Network Simulator-3 is primarily described in this chapter.

3.1 Software Defined Networking

Software Defined Networking (SDN) is a new networking paradigm. The main goal behind this technology is to make network open and directly programmable. It is done by physical separation of control plane from data forwarding plane, where control plane controls the network devices. SDN is an emerging dynamic and cost-effective architecture which allows network administrators and programmers to respond quickly in order to meet variable requirements. OpenFlow protocol is a fundamental element for building SDN architecture, which is used for communication between data plane and control plane. [6]

In its strictest structure, SDN is about setting on the choice on how a stream should be set up over the system and designing the network, as needs to the choices can be enhanced taking network into account system asset state, striges and application request. Once a choice is made by the control plane and the important activities to the exchanging plane by means of a system control convention. The principle out comes of SDN is the usefulness of the system is characterized after it has been conveyed under the control of the system proprietor and administrators. New components includes programming without changing the switches permitting the conduct to advances at programming speeds as opposed to at norms body speed[7]

The SDN architecture is:

- Directly programmable, because control plane is decoupled from data forwarding plane.
- Centrally managed, where network intelligence lies behind SDN controllers.

3.1.1 Architecture of Software Defined Networking (SDN):

SDN architecture is composed of three layers[8] as shown in figure1

3.1.1.1 Transmission Layer (Data Plane)

Transmission layer also known as data plane consists of switches, routers. It enables transfer of data from one end to other end of the layer. Its main duty is to execute the orders given by the controller in control plane.

3.1.1.2 Control Layer (Control Plane)

Data controlling lies in the control plane. It consists of one or more data controllers. Software Defined Network controller is a logically centralized entity. Software Defined Networking (SDN) controller in control plane translates the

requirements from the application layer to the transmission layer. A Software Defined Networking (SDN) controller is composed of SDN control logic, NBI agents and the CDPI driver. This layer facilitates the centralized view of the entire network and also hides the difficulty of the physical network to the data applications in the application layer.

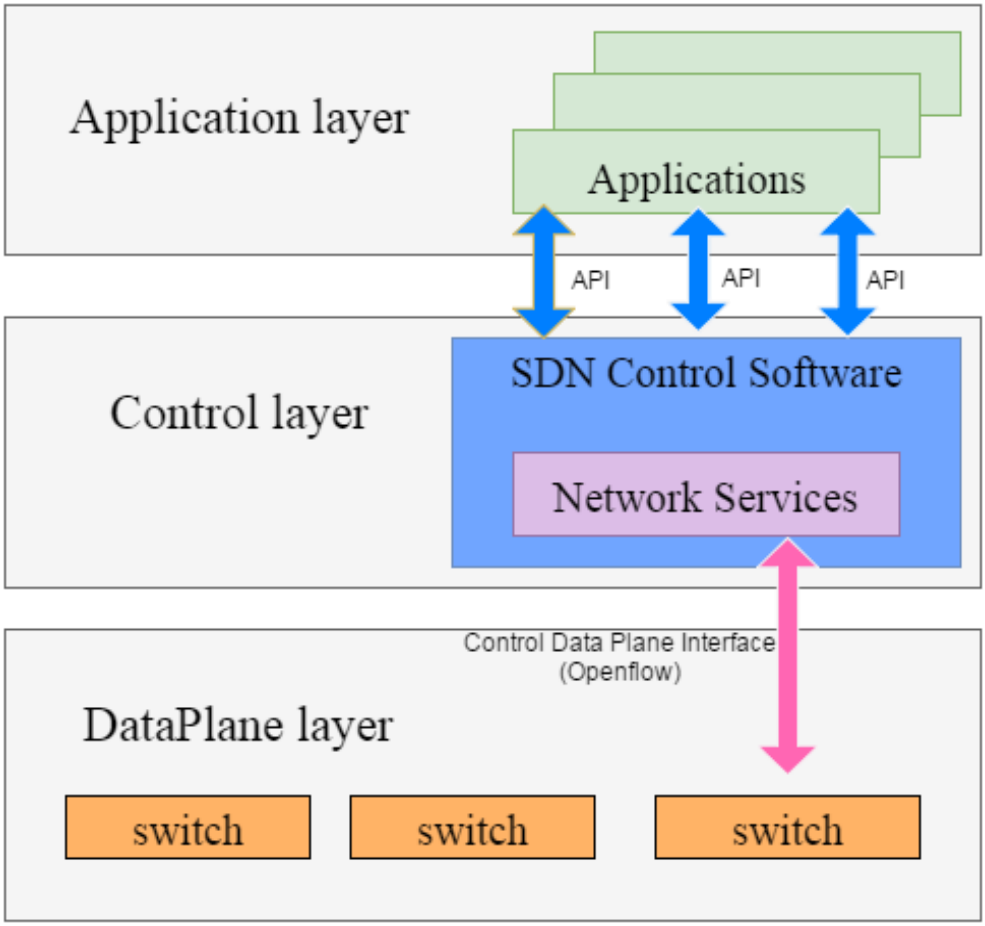

Figure 1 SDN Architecture

3.1.1.3 Application Layer

Application layer provides a platform for implementing all types of services and applications which are designed and developed for specific user needs. Application layer performs its functionality by utilizing the logical and centralized view provided by the control layer.

3.2 OpenFlow

OpenFlow technology is based on the Software Defined Networking (SDN) principle of decoupling control plane and data forwarding plane. The information exchange between data plane and control plane is standardized by OpenFlow technology. OpenFlow is a promising and emerging technology that can drive the traditional telecommunication networks into more flexible and scalable future internet architecture[9].

In OpenFlow based Software Defined Networking (SDN) architecture, data plane consists of one or more OpenFlow switches that can communicate with a Software Defined Networking (SDN) controller via an OpenFlow protocol. OpenFlow protocol is an open source protocol introduced by Open Networking Foundation (ONF), founded by the members of the organizations like CISCO, Facebook, Microsoft, and Google etc.[6].

3.2.1 OpenFlow Controller

OpenFlow controller is an OpenFlow network element for remotely controlling the forward tables in switches and routers which are present in the data plane or transmission layer. It is a type of Software Defined Networking (SDN) controller that uses the OpenFlow protocol[10]

Software Defined Networking OpenFlow controller handles communication between applications in the application layer and devices (routers, switches) in the data layer. A Software Defined Networking (SDN) OpenFlow controller can manage network flows in order to meet dynamic requirements of the end users, thereby simplifying the network management.

Some of the OpenFlow network controllers include:

- **NOX**: NOX is a Networking Operating System, a first OpenFlow controller. It provides a C++ OpenFlow 1.0 API. NOX controller supports Linux distributions.

- **BEACON:** Beacon is a Java based open source OpenFlow controller. Beacon supports both event based and threaded operations.

- **Helios:** Helios is an extensible e-based OpenFlow controller. It was invented by NEC researchers.

- **BigSwitch:** BigSwitch is a closed source OpenFlow controller. BigSwitch is build based on Beacon. It provides central management of the entire network.

- **Maestro:** Maestro is an extensible Java-based OpenFlow controller. Maestro was introduced by Rice University. Maestro supports Java-based multi-threading operations. Target researchers are also well supported by Maestro.

- **SNAC:** SNAC is an OpenFlow controller intended for enterprise networks. SNAC is build based on NOX 0.4. SNAC offers a user-friendly interface to configure and monitor the devices present in a network.

3.2.2 OpenFlow Switch

Ethernet switches now a days are replacing by OpenFlow switches due to their less flexibility. Flow based switch (OpenFlow) offers more flexibility compared to traditional Ethernet switches[2].

OpenFlow switches are classified into two types:

- **Hardware based OpenFlow switch**: Hardware based OpenFlow switches uses TCAMS and flow tables in order to store and process the flow information.

- **Software based OpenFlow switches:** OpenFlow switch operations are performed by these switches using Linux systems

OpenFlow switch consists of one or more flow tables. Flow tables determines data processing and forwarding with the help of flow entries as shown in figure.2. Each flow entry determines how data will be processed and forwarded in a network. Each flow entry consists of the following:

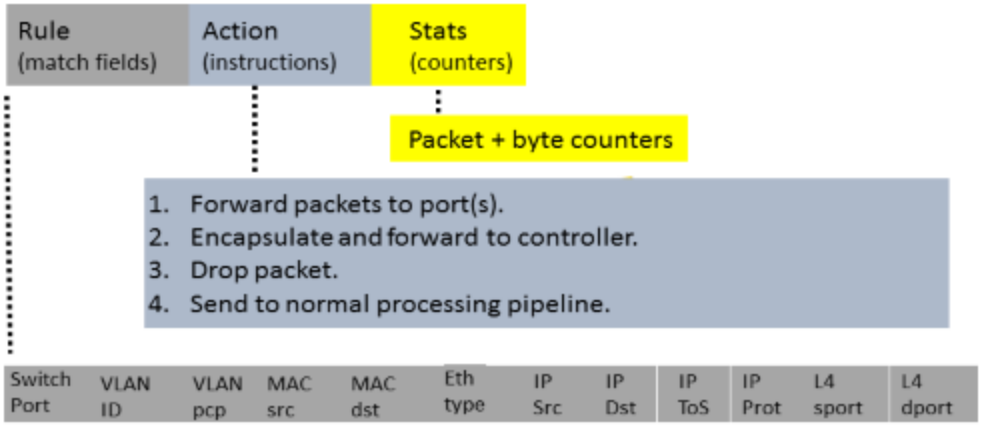

Figure.2 Flow Table[11]

- **Match Fields:** Match fields also known as matching rules, determines rules for matching incoming packets. Match fields contain packet header information and also the information observed in the ingress port and metadata.

- **Statistic Fields:** Statistic fields also known as counters collects the information of particular flow. The information gathered by counter is number of packets received, number of bytes received and the time duration of a particular flow.

- **Action Field**: Action field contains set of instructions or actions which are necessary for handling matching packets.

OpenFlow switches utilises these flow tables in order to forward packets with the help of OpenFlow protocol. OpenFlow protocol may set up new rules into OpenFlow switch flow tables if it is necessary for data processing and forwarding and

the communication between controller and switch is done and maintained over a secure channel.

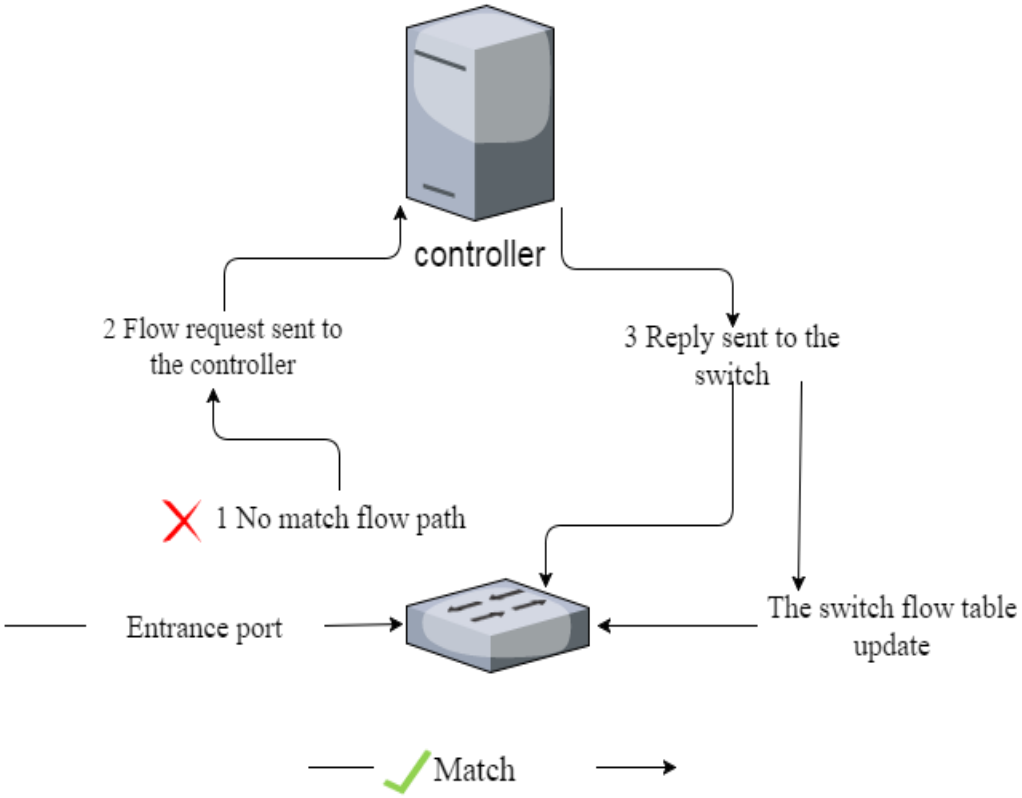

Figure: 3 OpenFlow switch Architecture

The communication mechanism between OpenFlow switch and controller is illustrated below:

When a new rule needs to be implemented in the flow table, the following procedure takes place. After a packet arrives to an SDN switch, it's header field is observed and compared with the match field in the flow table. If the matching is identified, then the corresponding action mentioned in the specific flow table entry is executed by the switch. If the matching is not identified, a flow request is sent to the controller and the controller decides the corresponding action to be performed and sends a new rule to the switch. Finally, the switch's flow table is updated with the required fields.

A dedicated OpenFlow switch is a stupid data path component that advances packet flow between switches as characterized by a remote control process figure2 demonstrates a flow table's creation in OpenFlow switch. In this connection are comprehensively characterized and packet flow are constrained just by the capacities of the specific usage of the flow tables for instance, a packet can be flow to tcp associations or all packets from a specific MAC location or IP address for trials including nov-IPv4 packet flow coordinating a particular header every flow-entry has a straight forward route there are 3 fundamental follows indicated flow path.

- Forwarding current flow packets to an installed port this packets to be directed through operation system in many switches is relied upon to occur.
- Exemplify and forwarding this current flow packet to a controller packet is conveyed to secure channel where it is embodied and sent

back to controller ordinarily utilized for the primary packet as a part of other flow. So a controller can choose if the flow ought to add to flow tables on the forward all packets to controller.

- This packet can be utilized for security, to control refused operating assaults or to flow traffic for end to end transmission.

OpenFlow-enabled switch is industrial switch, business router and switch access points that need to be improved with OpenFlow feature which includes flow tables, secure channels and OpenFlow protocols. Regularly flow tables are reuse for the next flow. In this illustration, all the flow table can be overwritten by controller only the OpenFlow protocol permits a change to be controlled by two or more controllers. All OpenFlow-enabled switches are to bolster on one methodology or the other some will support both methods[12]

3.2.3 OpenFlow Protocol

An OpenFlow protocol can handle high level routing, packet forwarding and secure connection between control plan and data plane. Protocol is used as a communicator between OpenFlow and all the other components on the network above the flow tables, and group table decision between switch and controller is done by OpenFlow protocol. This protocols manages the traffic flow in network. A new flow path in network received by the switch is forwarded to controller via protocol. Controller makes the decision about the path and modifications were done in the flow table.

3.3 Traditional switch

In a network system packet forwarding between the source and destination is done by switch and router where switching is done layer-2 and routing is done layer-3 in OSI model. A customary switch performs noteworthy in sending upward the packets IP address in the switch, date base and changing the packet format from an approaching connection to one of the other active connections with properly that undertaking of exchanging is surely understood further most merchants utilized quick transport or crossbar switches. Here we give a detail view on layer-2 switching and layer-3 routing.

3.3.1 Layer-2 Switching

A bridge connection is established in data link layer or layer-2 in the OSI model. Data link layer handles packet forwarding, provide link with another switch, providing MAC address and logical link. A Switch typically connect with other switches Media Access Control (MAC) address and Logical Link Control (LLC) on each switch. MAC address gives permission for forwarding/receiving of packet. LLC is used for flow control, packet error control, MAC addressing [13].

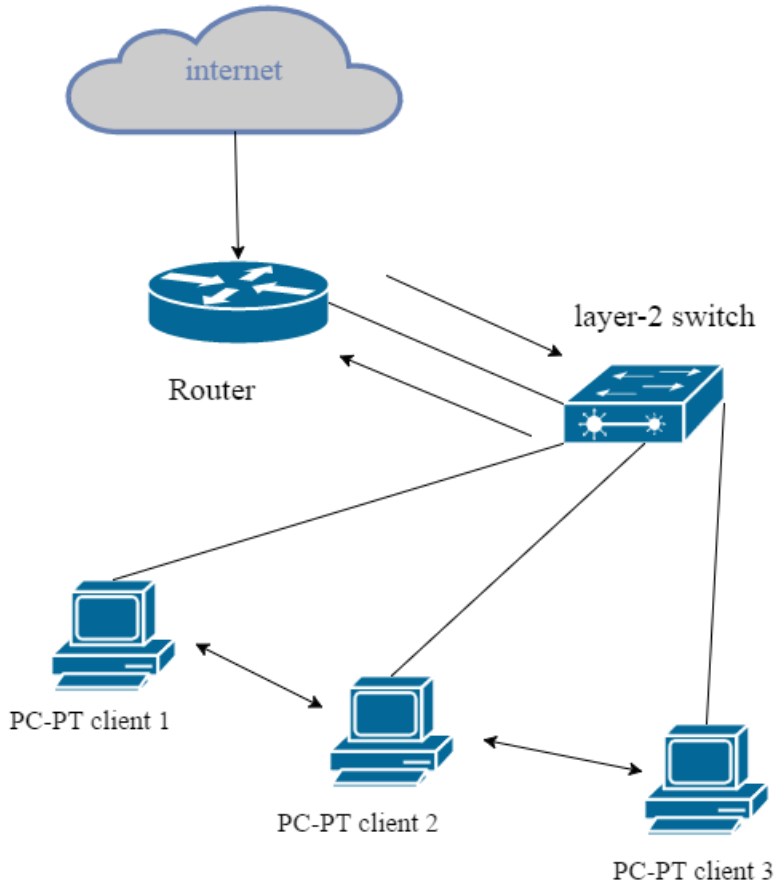

Figure-4: Layer-2 Switching Architecture

Layer 2 switches themselves go about as IP hubs for simple network management protocol administration telnet and web GUI. Such administration usefulness includes the nearness of an IP stack on the switch alongside User Datagram Protocol (UDP), transmission control protocol (TCP), telnet and SNMP capacities. The switches themselves have a MAC address with the goal that they can be tested to as a layer2 end date port while additionally giving straight forwarding switch capacities. Layer2 exchanges does not includes changes in MAC outline. The IEEE802.1Q committee is dealing with VLAN standard that includes labelling.

3.3.2 Layer-3 Routing

IP addressing is performed at network layer of the OSI model that implies the network layer is in charge for end-to-end packet delivery including directing through transitional host while the data link layer is in charge of node to node connections. Layer -3 gives the procedural method for transmitting information with sequence of time from server host to end user by means one or more network by maintain the Quality of Service (QOS) and error maintains capacity.

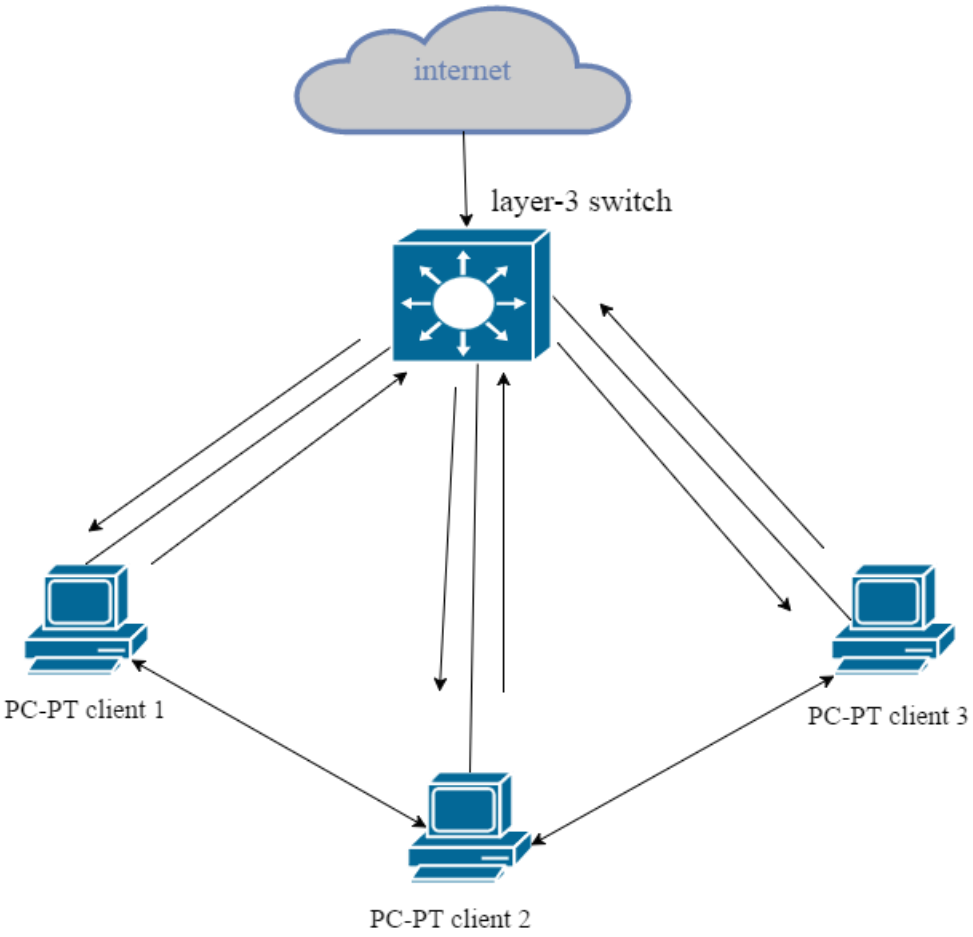

Figure-5: Layer-3 Switching Architecture

Administration of the layer 3 switch is ordinarily done by means of SNMP. In additionally have MAC address for their port. This setup can be one for each port or all ports can utilize the same MAC address the layer 3 switch regularly utilize this MAC address for SNMP, talent and web operator.

Design of the layer-3 switch is an essential issue at the point when layer-3 switch additionally perform layer-2 exchanging, they take in the MAC address on the ports the just configuration required Vlan design for layer-3 exchanging. The switches can be arranged with the part of comparing to each of the subnets or they can perform IP address learning this procedure includes snooping.

Adroitly, the ATM forum's LAN emulation LANE specificity particle is near to the layer 2 exchanging model, while MPOA is nearest to the layer 3 exchanging model various layer2 switches are outfitted with ATM interface and give a LANE customer capacity on the ATM interface. The situation permits the connecting of MA edges over an ATM system from switch to switch

Layer 3 switches totally take out requirement for the customary switch. No switches are still required particularly where associations with the wide range are required layer 3 switches may at present interface with such switches to take in their tables and course packets to the switch when these packets should be sent over the WAN the switches will be exceptionally powerfully on the workgroup and spin inside an undertaking on switch at the edge of the WAN. Switches play out various different capacities like shifting with access records, between Autonomous system AS directing with conventions for example, the Border Gateway Protocol BGP, etc. some layer3 switching may totally swap the requirement for a switch in the event that they can give every one of these capacities[13].

3.4 Network Simulator-3

The Network Simulator-3 is test system focused on fundamentally for exploration and instructive use. The NS-3 venture, began in 2006 is an open source venture creating network simulators. The motivation behind this instruction work of exercise is to acquaint NS-3 framework and way of troubleshooting the problem in network. For new clients to gather fundamental data from the definite manuals and changes over this data into working recreations. We mainly focuses on the task endeavours to keep up a situation for analysis to contribute and share the working process[14]

Network Simulator-3 is discrete-event computer network simulator. It is a free software for public use under GNU GPLv2 license compatibility. NS-3 is the latest version to take over network simulator-2. The main target of this simulator is for research development and educational need. This simulation software is available in all operating systems. All the modules in simulator are written in C++ object oriented script with some python binding. This module scripts can be run by C++ or python scripts. NS-3 got upgraded with new animation and anticipates are available for internal working process of the network. Outputs are generated in pcap packet files for storing packet transmitting/receiving for a particular event.

3.4.1 Simulator Installing Requirements

This section gives a detail description on installation requirements of the thesis work.

- **The web:** A detailed documentation and website is available for basic information about the file and downloading the simulator relating to operating system.

- **Waf:** To compile the given source code which is downloaded in operating system. There are many compiling tools are available probably due to the most well know tool is *make*. To make easy configuration for high configurable system. (Appendix D)

- **Development environment:** NS-3 can be run in any operating system due to feasibility but the author preferred using it in Linux environment due its advantages with other interconnecting outer modules.

3.4.2 Network Animation

Animation is an imperative instrument for network architecture simulator which there is no personal default graphical animation tool. We have two approaches to give activity to be specific utilizing the Pyviz technique or NetAnim technique[15].

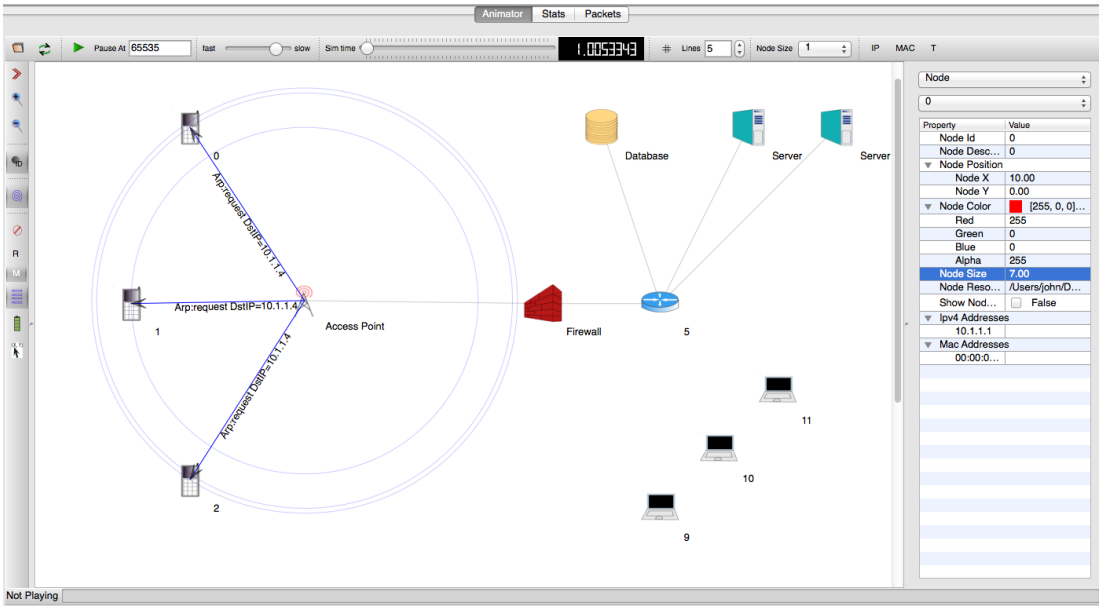

Figure-6: Animation

3.4.2.1 NetAnim

NetAnim is stand-alone program which utilizes the custom follow document produced by the moment interface to graphically show re-enactment. NetAnim depends on the multi-platform QT4GUI tool box. The NetAnim GUI gives play, respite and record button which can play and respite begin and stop the re-enactment. The record catch begins a progression of screenshots in which the follow document was run two sliders for usefulness, which permits a client to skip to any minute in the recreation. The base slider changes the granularity of the time venture for the movement. At last, there is stop button to stop the reproduction and quit the illustrator.

3.4.2.2 Pyviz

Pyviz utilizes GTKT and GOO CANVAS for the GUI part furthermore gets to the NS-3 programming interface specifically all the by means of individual python code. ITE gadgets don't bolster visualizer get class is a singleton that controls the entire visualizer GUI. It contains a run-down of ports and channels, the canvas, a run-down of bolts used to speak to the packet transmission and arranged GUI components, similar to the play and stop button.

4 METHODOLOGY

A literature study was conducted on different topologies using references in [2] [3] [1] [4] [5] and among these OpenFlow and layer-3 switching are selected for performance evaluation. Considering performance metrics like throughput and delay as the main metrics for evaluation and other metrics like band width, packet delivery ratio and simulation time. Using NS-3 simulator, these topologies are replicating real time environment. NS-3 simulator provides a significant number of modules out of that box.

4.1 Selecting parameters

Before implementing NS-3 simulation on OpenFlow and layer-3 router a research study on both switches for the traffic effecting parameters. For that factor we gain some knowledge on what simulation parameters to be considered for this research are listed below:

1. Performance of throughput according to packet size
2. Performances of delay according to simulation time

4.2 Creating a suitable testbed

To make this thesis experimentation work, NS-3 simulator is installed on Linux based operating system. To conduct simulation part on this NS-3 software there is a need for pre request external software to be installed like gcc-c++, python. NS-3 development tree, statistic framework, Etc. (appendix A) a manual installation is required for NS-3 installation a repository using mercurial. This repository has a use of NS-3 allinone environment[16] (Appendix B).

All these required software's are downloaded and configured. NS-3 simulator is ready to for executing of switches

4.3 Experimental setup

4.3.1 Two Tier Architecture

A simulation test bed is created to evaluate the network performance of OpenFlow switch and layer-3 router, focusing on the effect of switch and router connecting end user with respective to network switch. A two tier network architecture was consider for our research work which consists of switches and end host. Centralized networking of the switches is main reason for considering this network architecture of my work. The figure 7 & 9 show the experiment setup of two different vendor switches. For the both case centralized architecture was consider so as to have same throughput and delay so as to have better comparison results

4.3.2 OpenFlow setup

This section explains how OpenFlow switch and controller module are dumped into NS-3 for end-to-end transmission in a network.

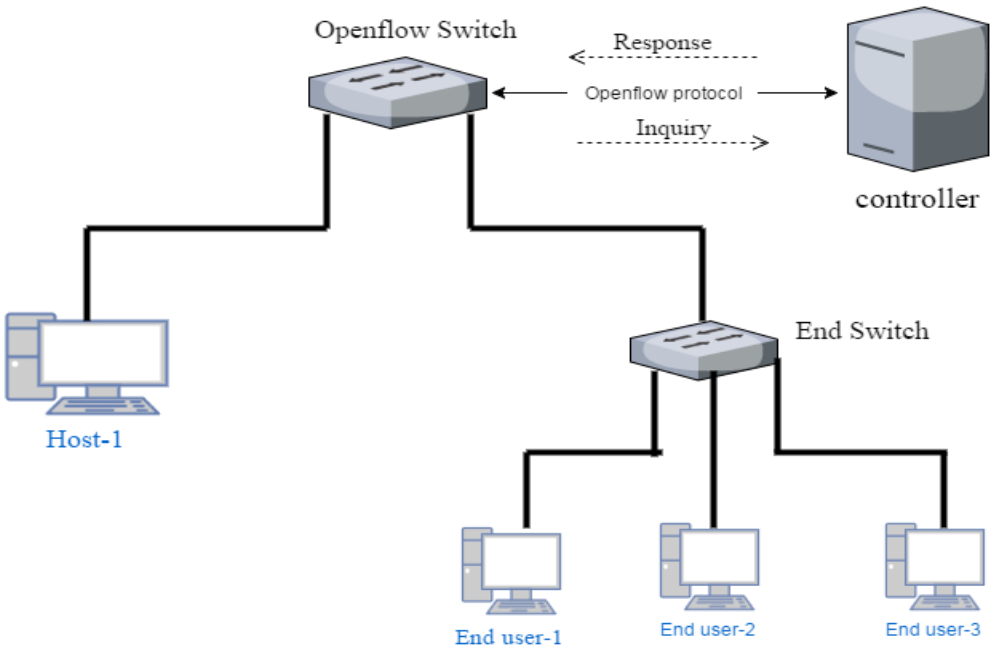

Figure-7: OpenFlow Experimental Setup

The above figure-7 show two tire network architecture of OpenFlow we are considering. It mainly consists of one sources, two OpenFlow switches, one controller and 3 destination nodes

4.3.2.1 Configuring OpenFlow switch module

An OpenFlow switch module is downloaded and configured via OpenFlow API and MPLS. These also supports for the extension of QOS and SLA. For experimentation of Module used in NS-3 is software based. So it is refereed as OFSID. The OFSID model library was built for installing OpenFlow module on the node. For this SwitchNetDevice and OpenFlowSwitch-Helper library used this module also consist of flow tables to match the packet by their header file for transmission. NetDevices library are used on nodes which acts as a packet forwarding and receiving in the between the nodes[17] (Appendix C)

4.3.2.2 Configuring OpenFlow controller module

NS-3 does not support any external OpenFlow controllers like NOX, floodlight, etc. so an internal controller was created using IP, mask, and MAC address to manage the network using OpenFlow SwitchHelper NS-3::ofi:controller. This drop controller was created with computer objects and because the controller is connected locally to the OpenFlow switch there won't be any channel breaking down. The instructions between the switch are configured in controller for every single packet make the switch by bridge NetDevice.

4.3.2.3 Configuring OpenFlow protocol

OpenFlow-enabled switch protocol is designed to express the basic interconnection between the switch and controller of the in the network. The message of the protocol consists of flow path, flow tables and TCAM which make perfect OpenFlow environment.

Figure-8: OpenFlow Protocol Configure

Once the OpenFlow environment was created, a C++ script is written for designing the consider network architecture in ns-3

4.3.3 Layer-3 routing setup

This section explains how layer-3 switch module is dumped into NS-3 for end-to-end transmission in a network. The below figure-9 show two tire network architecture of layer-3 routing we are considering. It mainly consists of one source, two routing switches, and 3 destination nodes.

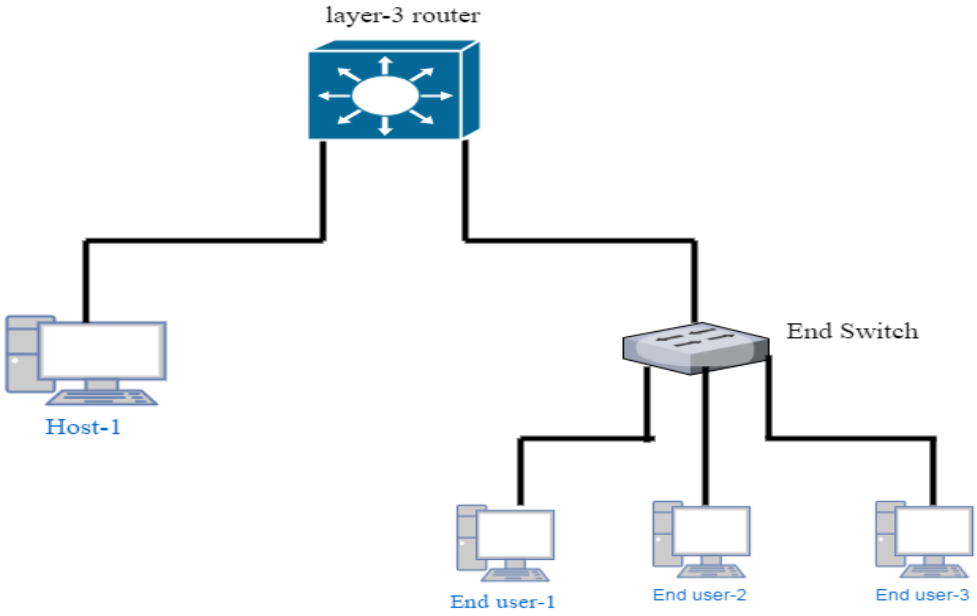

Figure-9: Layer-3 Routing Setup

4.3.3.1 Configuring routing module

NS-3 support layer-3 traditional routing and protocols. The implementation of this routing is done in OpenFlow source and overall architecture was designed of routing approaches which has router and on-demand routing protocols. The router was being used as unicast routing. The unicast routing has two different support point to point and CSMA linking. NS-3 has default internet stack helper, global routing capability this modules are added to the network-nodes and connected with other nodes by routing protocol[18]

This routing uses IPv4 static routing API which is responsible to link establishment between the nodes. API and LSA are responsible for executing the shortest path first OSPF computing in the network. OSPF uses a point to point linking for packet forwarding and receiving.

4.3.3.2 IPv4 routing

The overall routing was run on key of IPv4 routing protocol. This module is script using C++ and added to its simulation setup time by header file IPv4::set routing protocol. The basic work is to send and receive the packets at the nodes. For this end to end transmissions between the nodes occur with class IPv4L3 protocol: receiver, and IPv4Routing protocol:: RouterInput.

4.4 Simulation and execution

Once the complete network scenario was composed and configured for both the networks topologies, these topologies were run in NS-3 through code C++ on terminal. A waf script is used for executing of this network program in NS-3. During the execution process a pcap files and tracer files are generated in the system which has date of analysis of network traffic behavior during the simulation process.

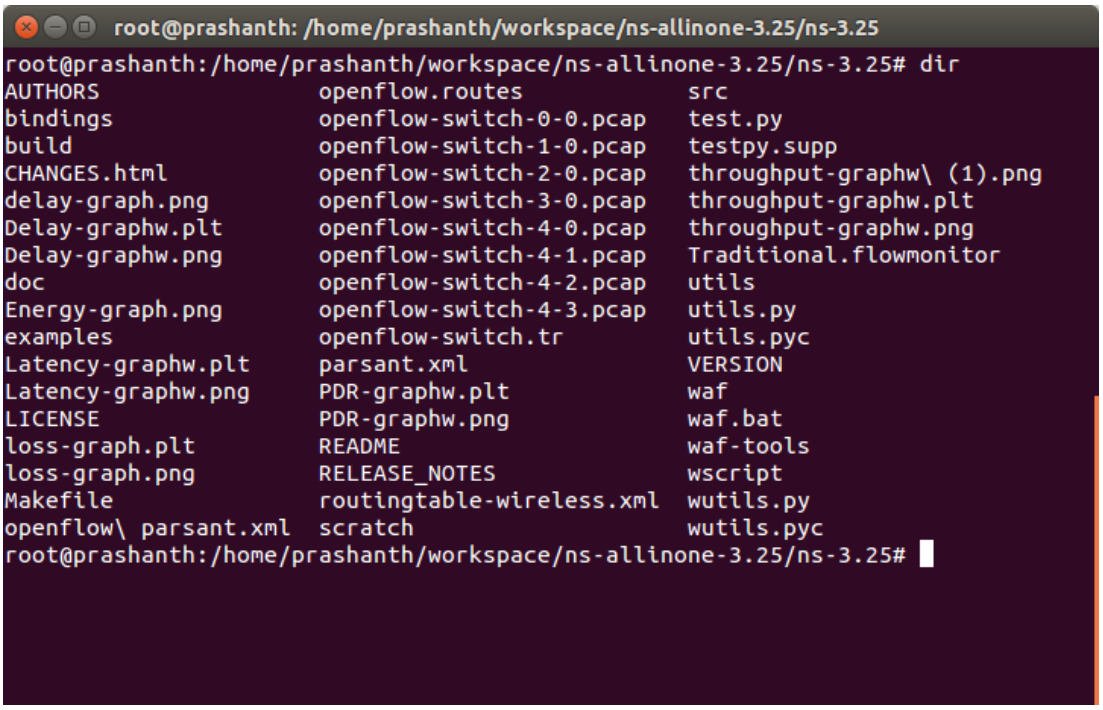

Figure-10: Simulation and Execution

4.5 Graphical animation

Basically NS-3 does not have any internal tool for animation. So their external tools modules are used for animation part. NetAnim is used for graphical animation. NetAnim records the status of every node by default for every 250ms. Includes the header files and statement to automatically running and correcting the data in XML file. Once the XML file are created by the end of simulation. A manual launch of NetAnim application is need. To execute the application by.\NetAnim command is executed on the terminal. Now by adding the XML file by using file adding at left-hand top once the file is added by hitting the green paly button the required simulation animation is run.

4.5.1 OpenFlow Network Animation

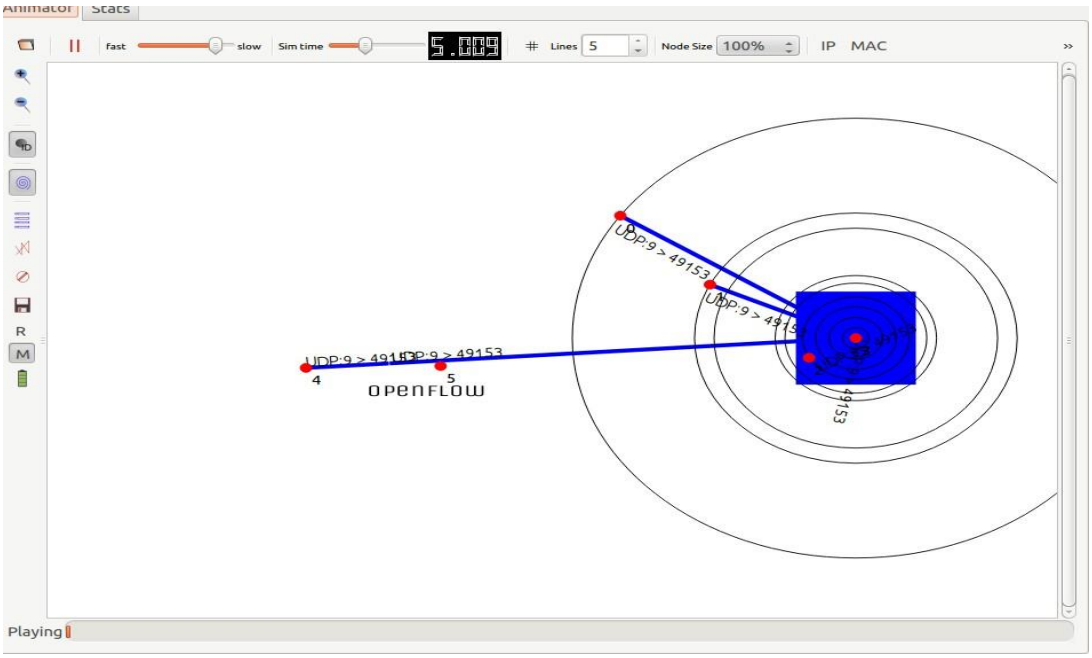

Figure-11: OpenFlow Animation

4.5.2 Layer-3 router Network Animation

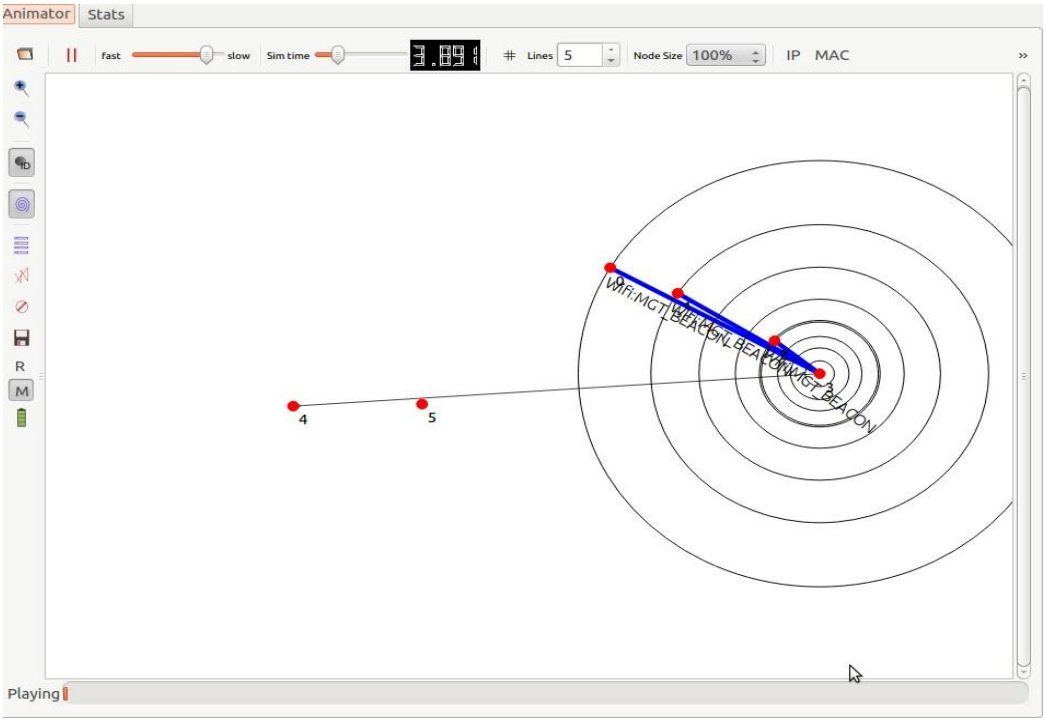

Figure-12: Layer-3 router animation

5 RESULTS AND ANALYSIS

This section deals with the simulation statistical results obtained by considering different metrics which are by performing a literature research on OpenFlow and layer-3 router. This statistical results were used for comparing OpenFlow with layer-3 switching networks.

5.1 Performances of throughput according to packet flow size

Throughput Description:

According to the computer technology, the amount of work that a computer can do in a given time period is defined as throughput. While considering communication networks, the average of successful message delivery over a communication channel is defined as network throughput. This information might be conveyed over a physical channel or logical channel through a specific node in a network. The throughput is typically measured in bits per second or data packets per time space. Throughput can be calculated as normal throughput value, maximum throughput value and peak throughput value.

Formula:

Forwarding Time = Packet size / Bandwidth.
Throughput = Packet size / Forwarding Time.

Source Code to calculate Throughput:

```
Void
ThroughputPerSecond (ptr<application> sink1apps, int totalPacketsThrough,
Ptr<Node> node)
{
Throughput=0.0;
Ptr<packetSink> sink1 = DynamicCast<PacketSink> (sink1Apps);

PacketTransmittingTime = sink1->GetTotalRx ();
Throughput = PacketTransmittingTime*8/ ((237. 0)*1000000.0);

Std : : cout << throughput ;
```

Packet size (bytes)	OpenFlow	Layer-3
64	313,48	420,33
96	447,68	509,15
128	694,31	640,31
192	764,4	734,84
224	853,16	856,76
256	869,89	874,61
512	890,47	899,91
1024	921,54	924,07

Table-1: statistical report of throughput according to packet size

In the above table1 shows the statics of throughput performance of OpenFlow and layer-3 measured with respect to different packet like 64, 96, 128, 224, 256, 512, 1024 bytes.

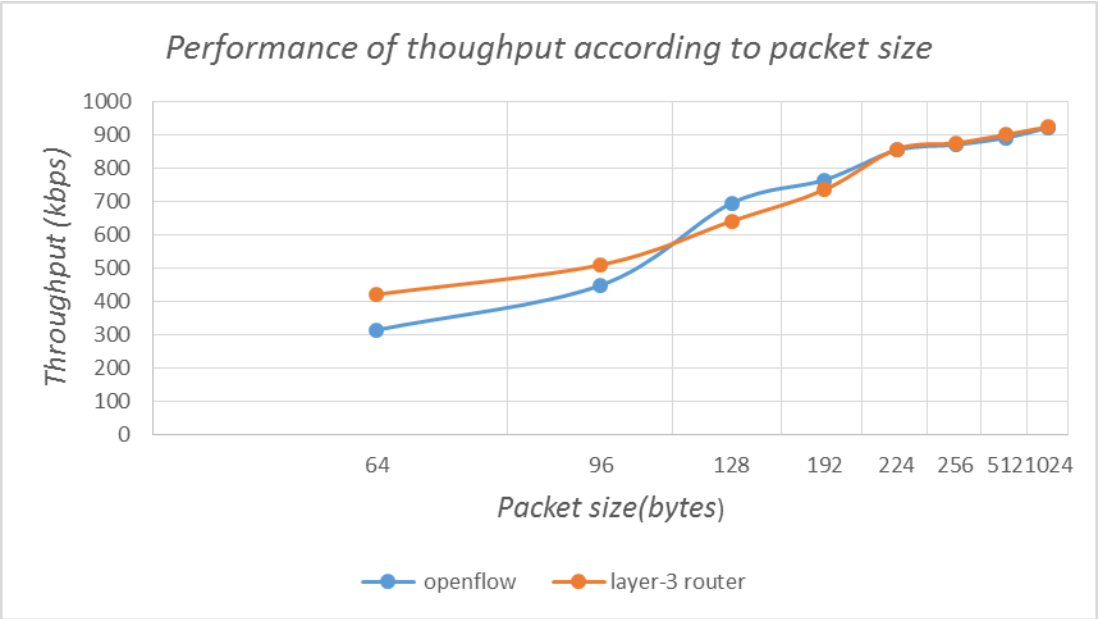

Figure-13: Comparison Layer-3 Router over OpenFlow Switch

From the above Figure-13 the performances of two network technologies it can be observed that the throughput increases with the packet size. We know this due to the fact that the relative overhead decrease with packet flow size increase when forwarding packet flow scale increase and due to implementation software switch. The transmission error rate also increase which can lead to network failure more frequently. As observed both layer-3 routing and OpenFlow perform well almost the same for the large flow size and whose worse performs for the small packet size. This technique gives approximate results from increase of 224-bytes packet flow.

5.2 Performances of delay according to number of packets.

Delay:

The delay of a network specifies to what extent it takes for a bit of data to travel over from source node to destination node. It is measured in Seconds. End-to-end delay is known as one-way delay and is commonly used in IP network monitoring.

Count = Total packet count
Delay = Received time – sending time
Total Delay = Delay / count

Source code to calculate Delay:

```
Void
Static Void GenerateTraffic (Ptr<socket> socket, uint32_t pktsize,
Uint32_t PktCount, Time pktIntterval ) {
  If (pktCount > 0)
    {

 Socket - > Send (Create<Packe> (pktSize ) ) ;
….
```

No. of Packets	Delay OpenFlow	Delay layer-3 router
2	41	42,5
4	45	45
6	46	51,5
8	52	56,5
10	55,5	58,5
12	55,5	61,5

Table-2: statistical report of average delay to simulation time

In the above table-2 shows the statics of average delay performance of OpenFlow and layer-3 measured with respect to simulation time.

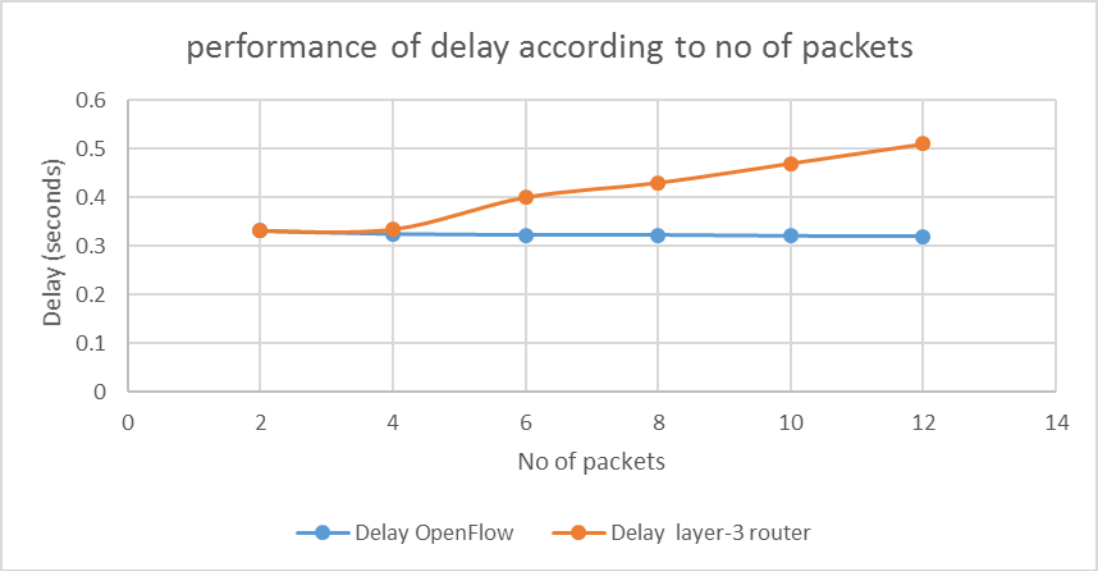

Figure-14: Comparison Layer-3 Router over OpenFlow Switch

From the above figure-14 shows the performances of 2 networks technologies in terms of delay with respect to number of packets. For a specific increase in the traffic load the number of data packets has increased from 2, 4, 6, 8, 10 and 12. In the above figure we can observe the end-to-end delay while changing the number of packets. The decrease of collision decreases the reformation of collide packets which results in decrease in values of end-to-end delay. It depends on many factors alike network traffic, model's complexity. It becomes nearly constant in OpenFlow once it's reaching a maximum delay. As observation OpenFlow has better constant delay when compare with layer-3 router.

6 CONCLUSION

After collecting performance evaluation data of end to end transmission from multiple simulations, a deep analysis on packet transmission is carried out and observed by using ns-3 simulator technics. To make sure analysis is accurate both the switches are implemented on same architecture. And also to avoid any network impact on evaluation of the switch performances. This made verification and validation of results transparent and accurate. It is observed that OpenFlow switch have throughput better and lower delay than Layer-3 switch. The explanation is given below.

Once we have statics of experimental results which show throughput, average delay time comparing the results of both switches. These statics results show layer-3 router for a host to end user is relatively more when compared to OpenFlow. Using Layer-3, packet delay increased linearly with number of packets. But packet delay remained constant with increase in number of packets when OpenFlow switch is used.

This is because of scalability provided by OpenFlow switch. The through-put of both switches is similar through most of the tests but by using proper tunneling protocols the packets can be efficiently used in OpenFlow. From this comparison we can conclude that OpenFlow switching technology is better packet forwarding technology because of its flexible and scalability properties in a network communication.

Furthermore, OpenFlow switch module support in NS-3 and technical support from system made this performances evaluation possible. Implementation of OpenFlow switching module present in this work is completely a software base network switch. This thesis also shows benefits and limitations of software base switch in ns-3.

6.1 Linking to research questions

RQ1: How is a link established between end users when Open Flow-switch is implemented in Network Simulator-3?

Using OpenFlow switch and controller a forwarding link between source and destination nodes is established using ns-3 simulator for collecting statistical results of network. It gives an overview on the performances of switch. These experimental statistical results analyzed for traffic monitoring in the network. These results illustrate the throughput and average delay of complete OpenFlow network.

RQ2: How is a link established between end users when traditional switch is implemented in Network Simulator-3?

Using layer-3 router a forwarding link between source and destination nodes is established using ns-3 simulator for collecting statistical results of network. It gives an overview on the performances of router. These experimental statistical results analyzed for traffic monitoring in the network. These results illustrate the throughput and average delay of complete router network

RQ3: What are the metrics needed for analyzing the performance of OpenFlow-switch with traditional switch? Can the performances further be improved?

Simulation for the given network setup is adopted to a statistical result. An impressive performance metrics throughput and average delay are selected to evaluate the network traffic for the complete network. These are chosen for this experimental setup because the complete focus of this study is based on comparing of effective switch to use in a network traffic control. This will help in achieving some knowledge on pros and cons on OpenFlow and layer-3 routing.

6.2 Future work

The research work performed in this thesis is regarding the performance evaluation on OpenFlow switch and traditional switch using NS-3 from the results it is concluded that OpenFlow is software-based switch is used for comparing the layer-3 router

Firstly, it would be interesting to change with hardware based OpenFlow switch using external controller like NOX and POX controller for the real time experiments.

Secondly the experiment network is two-tire network used in this thesis work by increasing the network area like college campus, the performances traffic metrics can be tested with multiple switches and single controller.

Thirdly by taking this results as base for traffic measurements, it is possible to try and find out new techniques to control traffic in the network.

REFERENCES

[1] A. Bianco, R. Birke, L. Giraudo, and M. Palacin, "OpenFlow Switching: Data Plane Performance," in *2010 IEEE International Conference on Communications (ICC)*, 2010, pp. 1–5.

[2] F. Benamrane, M. Ben mamoun, and R. Benaini, "Performances of OpenFlow-Based Software-Defined Networks: An overview," *J. Netw.*, vol. 10, no. 6, Jun. 2015.

[3] V. Tanyingyong, M. Hidell, and P. Sjödin, "Improving PC-based OpenFlow Switching Performance," in *Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, New York, NY, USA, 2010, p. 13:1–13:2.

[4] H. Farhad, H. Lee, and A. Nakao, "Data Plane Programmability in SDN," in *2014 IEEE 22nd International Conference on Network Protocols*, 2014, pp. 583–588.

[5] L. J. Chaves, I. C. Garcia, and E. R. M. Madeira, "OFSwitch13: Enhancing ns-3 with OpenFlow 1.3 Support," 2016, pp. 33–40.

[6] "Software-Defined Networking (SDN) Definition - Open Networking Foundation." [Online]. Available: https://www.opennetworking.org/sdn-resources/sdn-definition. [Accessed: 09-Sep-2016].

[7] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, p. 19.

[8] "OpenFlow - Open Networking Foundation." [Online]. Available: https://www.opennetworking.org/sdn-resources/openflow. [Accessed: 09-Sep-2016].

[9] "OpenFlow » What is OpenFlow?" .

[10] "What is an OpenFlow Controller?," *SDxCentral*, 23-Jun-2016. [Online]. Available: https://www.sdxcentral.com/sdn/definitions/sdn-controllers/openflow-controller/. [Accessed: 09-Sep-2016].

[11] "Inside OpenFlow! | ConfigNetworks." .

[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[13] "Layer 2 and Layer 3 Switch Evolution - The Internet Protocol Journal - Volume 1, No. 2 - Cisco." [Online]. Available: http://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-19/switch-evolution.html. [Accessed: 09-Sep-2016].

[14] "What is ns-3 « ns-3." [Online]. Available: https://www.nsnam.org/overview/what-is-ns-3/. [Accessed: 09-Sep-2016].

[15] "Animation — Model Library." [Online]. Available: https://www.nsnam.org/docs/models/html/animation.html. [Accessed: 09-Sep-2016].

[16] "Installation - Nsnam." [Online]. Available: https://www.nsnam.org/wiki/Installation#Installation. [Accessed: 09-Sep-2016].

[17] "OpenFlow switch support — ns-3 vns-3-dev documentation." [Online]. Available: https://www.nsnam.org/docs/release/3.13/models/html/openflow-switch.html. [Accessed: 09-Sep-2016].

[18] "ns-3: Global Routing." [Online]. Available: https://www.nsnam.org/doxygen/group__globalrouting.html. [Accessed: 09-Sep-2016].

APPENDIX

Apendix A

The following list of packages should be accurate for Ubuntu 14.10 release; other releases or other Debian-based systems may slightly vary.

minimal requirements for C++ (release): apt-get install gcc g++ python

minimal requirements for Python (release): apt-get install gcc g++ python python-dev

apt-get install qt4-dev-tools libqt4-dev

apt-get install mercurial

apt-get install bzr

apt-get install cmake libc6-dev libc6-dev-i386 g++-multilib

apt-get install gdb valgrind

apt-get install gsl-bin libgsl0-dev libgsl0ldb1

apt-get install flex bison libfl-dev

apt-get install tcpdump

apt-get install sqlite sqlite3 libsqlite3-dev

apt-get install libxml2 libxml2-dev

apt-get install libgtk2.0-0 libgtk2.0-dev

apt-get install vtun lxc

apt-get install uncrustify

apt-get install doxygen graphviz imagemagick

apt-get install texlive texlive-extra-utils texlive-latex-extra texlive-font-utils texlive-lang-portuguese dvipng

apt-get install python-sphinx dia

apt-get install python-pygraphviz python-kiwi python-pygoocanvas libgoocanvas-dev ipython

apt-get install libboost-signals-dev libboost-filesystem-dev

apt-get install openmpi-bin openmpi-common openmpi-doc libopenmpi-dev

Appendix.B

#1 Obtain the OFSID code. An ns-3 specific OFSID branch is provided to ensure operation with ns-3. Use mercurial to download this branch and waf to build the library::

$ hg clone http://code.nsnam.org/jpelkey3/OpenFlow

$ cd OpenFlow

From the “OpenFlow” directory, run::

$ ./waf configure

$ ./waf build

#2 Your OFSID is now built into a libOpenFlow.a library! To link to an ns-3 build with this OpenFlow switch module, run from the ns-3-dev (or whatever you have named your distribution)::

$ ./waf configure --enable-examples --enable-tests --with-OpenFlow=path/to/OpenFlow

#3 Under ---- Summary of optional NS-3 features: you should see::

"NS-3 OpenFlow Integration : enabled"

indicating the library has been linked to ns-3. Run::

$ ./waf build

to build ns-3 and activate the OpenFlowSwitch module in ns-3.

Appendix c

First you need to download Bake using Mercurial, go to where you want Bake to be installed and call

hg clone http://code.nsnam.org/bake

It is advisable to add bake to your path.

export BAKE_HOME=`pwd`/bake

export PATH=$PATH:$BAKE_HOME

export PYTHONPATH=$PYTHONPATH:$BAKE_HOME

After that you can use Bake to find the missing packages, download build and install ns-3 and its modules.

To find out what is missing in your system and may be needed for installing ns-3 you can call bake check:

bake.py check

You should have seen something like:

> Python - OK
> GNU C++ compiler - OK
> Mercurial - OK
> CVS - OK
> GIT - OK
> Bazaar - OK
> Tar tool - OK
> Unzip tool - OK
> Unrar tool - OK
> 7z data compression utility - OK
> XZ data compression utility - OK
> Make - OK
> cMake - OK
> patch tool - OK
> autoreconf tool - OK
> Path searched for tools: /usr/lib64/qt-3.3/bin
/usr/lib64/ccache /usr/local/bin /usr/bin/bin/usr/local/sbin /usr/sbin
/sbin /user/dcamara/home/scripts/user/dcamara/home/INRIA/Programs/bin
/user/dcamara/home/INRIA/repos/llvm/build/Debug+Asserts/bin

Before downloading and building ns-3 you need to configure bake to inform it which are the modules you want added to ns-3, the standard distribution for example.

bake.py configure -e ns-3.17

Then to see the modules it has added, and the specific system requirements for this configuration, you can call bake show:

bake.py show

To download the modules, build and install you can call bake deploy

bake.py deploy

This will download the selected modules, all their dependencies and build ns-3 with all these independent modules. You can also perform this installation step by step, i.e. by calling download and build in different steps.

bake.py download
bake.py build

Manual installation

The ns-3 code is available in Mercurial repositories on the server http://code.nsnam.org (look for the latest release e.g., "ns-3.4"). You can download a tarball of the latest release at http://www.nsnam.org/releases or you can work with our repositories using Mercurial. We recommend using Mercurial unless there's a good reason not to (See the end of this section for instructions on how to get a tarball release).

The simplest way to get started using Mercurial repositories is to use the ns-3-allinone environment. This is a set of scripts that manages the downloading and building of various subystems of ns-3 for you. We recommend that you begin your ns-3 adventures in this environment as it can really simplify your life at this point.

Downloading ns-3 Using Mercurial

One practice is to create a directory called repos in one's home directory under which one can keep local Mercurial repositories. If you adopt that approach, you can get a copy of ns-3-allinone by typing the following into your Linux shell (assuming you have installed Mercurial):

```
cd
mkdir repos
cd repos
hg clone http://code.nsnam.org/ns-3-allinone
```

As the hg (Mercurial) command executes, you should see something like the following displayed,

```
destination directory: ns-3-allinone
requesting all changes
adding changesets
adding manifests
adding file changes
added 26 changesets with 40 changes to 7 files
7 files updated, 0 files merged, 0 files removed, 0 files unresolved
```

After the clone command completes, you should have a directory called ns-3-allinone under your ~/repos directory, the contents of which should look something like the following:

```
build.py*  constants.py  dist.py*  download.py*  README  util.py
```

Notice that you really just downloaded some Python scripts. The next step will be to use those scripts to download and build the ns-3 distribution of your choice.

If you go to the following link: http://code.nsnam.org/ you will see a number of repositories. Many are the private repositories of the ns-3 development team. The repositories of interest to you will be prefixed with ns-3. Official releases of ns-3 will be numbered as ns-3.release.hotfix. For example, a second hotfix to a still hypothetical release nine of ns-3 would be numbered as ns-3.9.2 on this page.

The current development snapshot (unreleased) of ns-3 may be found at http://code.nsnam.org/ns-3-dev/. The developers attempt to keep these repository in consistent, working states but they are in a development area with unreleased code present, so you may want to consider staying with an official release if you do not need newly-introduced features..

To download the most common options type the following into your shell (remember you can substitute the name of your chosen release number instead of ns-3-dev)

```
./download.py -n ns-3-dev
```

After download process completes, you should have several new directories under ~/repos/ns-3-allinone:

```
build.py*  constants.pyc  download.py*  nsc/  README  util.pyc
constants.py  dist.py*  ns-3-dev/  pybindgen/  util.py
```

Go ahead and change into ns-3-dev under your ~/repos/ns-3-allinone directory. You should see something like the following there:

```
AUTHORS  examples/  RELEASE_NOTES  utils/  wscript
```

bindings/ LICENSE samples/ VERSION wutils.py
CHANGES.html ns3/ scratch/ waf*
doc/ README src/ waf.bat*

You are now ready to build the ns-3 distribution.

Appendix D

To see valid configure options, type ./waf --help. The most important option is -d <debug level>. Valid debug levels (which are listed in waf --help) are: "debug" or "optimized". It is also possible to change the flags used for compilation with (e.g.):

CXXFLAGS="-O3" ./waf configure

or, alternately, the gcc compiler

CXX=g++-3.4 ./waf configure

./waf -d optimized configure; ./waf

The resulting binaries are placed in build/<debuglevel>/srcpath. For example, in a debug build you can find the executable for the first.cc example as build/examples/first. You can debug the executable directly by:

./waf --shell

cd build/debug/examples

gdb ns-<version>-first-debug

Of course, you can run gdb in emacs, or use your favorite debugger such as ddd or insight just as easily. In an optimized build you can find the executable for the first.cc example as build/examples/ns-<version>-first-optimized.

In order to forcibly disable python bindings, you can provide the following option:

./waf --disable-python configure

In order to tell the build system to use the sudo program to set the suid bit if required, you can provide the following option:

./waf --enable-sudo configure

To start over a configuration from scratch, type:

./waf distclean

Or if you get stuck and all else fails:

rm -rf build

Followed by changing back into ns-3-allinone and doing:

./build.py

Will basically reset your build state.

To see all waf options:

./waf --help