

SOFTWARE DEFINED NETWORKING- A CASE STUDY IN CST



Project Report
submitted in partial fulfillment of the requirements for the award
of the degree Bachelor of Engineering
in
Information Technology

Berandra Rai (02180400)

Ngawang Choden (02175019)

Kencho Bidha (02180410)

Tshering Dorji (02180430)

Under the Guidance of:

Mr. Yeshe Wangchuk

INFORMATION TECHNOLOGY DEPARTMENT
COLLEGE OF SCIENCE AND TECHNOLOGY
PHUENTSHOLING, BHUTAN

June, 2022

ROYAL UNIVERSITY OF BHUTAN
COLLEGE OF SCIENCE AND TECHNOLOGY
INFORMATION TECHNOLOGY DEPARTMENT



CERTIFICATE

This is to certify that the B.E. project titled “**Software Defined Networking-A Case Study in CST**”, which is being submitted by **Berandra Rai (02180400)**, **Kencho Bidha (02180410)**, **Ngawang Choden (02175019)**, and **Tshering Dorji (02180430)** in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering in Information Technology, is a record of student work carried out at the College of Science and Technology, Phuentsholing under my supervision and guidance.

.....
Mr. Yeshi Wangchuk
Project Guide

Acknowledgement

We offer our utmost gratitude to the College of Science and Technology, Department of Information Technology for giving us such a tremendous opportunity to do our bachelor's degree project on “Software Defined Networking-A Case Study in CST”. We were facilitated with all the necessary resources, including internet access, free library access, and all of the project materials.

For always being there to provide overall guidance and support throughout this project, we would like to firstly express our utmost respect and gratitude towards the Project Guide, Mr. Yeshi Wangchuk, without whom we would not have been able to drive this project to completion.

Along with the guide, we would like to thank the Sr. ICT Officer, CST, Mr. Jiwan Gurung, for his time and for supplying us with all the necessary resources required for this project. We also would like to express our special thanks to the HoD, Mr. Tandin Wangchuk, for taking an interest in our project and helping us find the related courses for the project.

Next, we would like to extend our heartfelt gratitude towards the Project Coordinator, Mr. Karma Wangchuk, for the overall guidance and constant support. In line with this, we would like to thank all the panel members for providing us with their valuable feedback, which helped us refine our project through each review.

Mr. Berandra Rai

Miss Kencho Bidha

Miss Ngawang Choden

Mr. Tshering Dorji

Abstract

As a means to ease the functionality of networking, Software Defined Networking, commonly abbreviated as SDN, emerged as an architecture that helps to optimize the network resources via dynamic and automated SDN programs, along with providing ease of management and configuration. This paper consists of a series of steps that were taken to conduct the feasibility of implementing SDN at CST by studying and analyzing the existing network topology of the college through simulation. The software tools used for the simulations are GNS3 as the network simulator and OpenDayLight as the SDN controller. The simulation was held in two phases, whereby the first phase included simulating the existing network topology of the college using the GNS3 network simulator, and the second phase included simulating using the ODL SDN controller. The simulated existing network topology could be used by the CST network administrator as a test bed for real time implementation, and the SDN simulation will give the basis for running an SDN network at the college. The purpose of two simulations was to help realize the bottleneck in the current networking system of the college.

List of abbreviations

SI. No	Abbreviations	Full Forms
1	CST	College of Science and Technology
2	GNS3	Graphical Network Simulator
3	SDN	Software Defined Networking
4	API	Application Programming Interface
5	GUI	Graphical User Interface
6	CLI	Command Line Interface
7	ODL	OpenDayLight
8	NFV	Network Function Virtualization
9	VM	Virtual Machine
10	NBI	Northbound Interface
11	SBI	Southbound Interface
12	RUB	Royal University of Bhutan
13	DITT	Department of Information Technology and Telecom
14	DHCP	Dynamic Host Control Protocol
15	OS	Operating System
16	DrukREN	Druk Research and Education Network
17	GPL	General Public License
18	JVM	Java Virtual Machine
19	OFM	OpenFlow Manager
20	IP	Internet Protocol
22	TCP	Transmission Control Protocol
23	REST	Representational State Transfer
24	IOS	The Internetworking Operating System
25	PoE	Power of Ethernet

List of Tables

Table 1 - Allied Telesis devices.....	7
Table 2 - Cisco networking devices.....	10
Table 3 - D-Link switch.....	11
Table 4 - Cost structure for SDN supported	12
Table 5 - Cost structure for SDN non-supported.....	12
Table 6 - Comparison table for Network Simulation tools.....	20
Table 7 -Comparison table for SDN Controller	27
Table 8- Sample Flow table	44

List of Figures

Figure 1 – Methodology	5
Figure 2 - Allied Telesis Switches.....	8
Figure 3 - Cisco Switch	10
Figure 4 - D-Link switch	11
Figure 5- Lightning affected SDN-Non-supported switches.....	13
Figure 6 - GNS3.....	15
Figure 7 - Cisco Packet Tracer	16
Figure 8 - NS3	16
Figure 9 – Mininet	17
Figure 10 - OMNeT++.....	17
Figure 11 - NS2	18
Figure 12 - JiST	18
Figure 13 - OpenDayLight Architecture.....	21
Figure 14 - Floodlight Architecture	22
Figure 15 - Ryu Architecture	22
Figure 16 - Architecture of Open Network Operating System.....	23
Figure 17 - Architecture of NOX.....	24
Figure 18 - Architecture of POX	24
Figure 19 - General Public License	28
Figure 20 - Apache License	28

Figure 21 - Ms-P License	29
Figure 22 - BSD License	29
Figure 23 - CDCL License	30
Figure 24 - EPL license	30
Figure 25 - MIT License.....	31
Figure 26 - A detailed block diagram of the existing network topology.....	32
Figure 27 - Simplified block diagram of the existing network topology.....	33
Figure 28 - College Network Simulation using GNS3.....	34
Figure 29 - SDN Architecture.....	35
Figure 30 - VMWare Workstation Pro (Version 15.5.1).....	37
Figure 31 - Ubuntu Server (Version 20.04.4 LTS).....	38
Figure 33 - OpenFlow Manager	38
Figure 34 - Starting ODL in Ubuntu.....	39
Figure 35 - Opening Mininet editor.....	40
Figure 36 - Changing the Controller to OpenFlow	40
Figure 37 - Connecting Switches to ODL	41
Figure 38 - College Network Simulation using the ODL Controller in Miniedit	41
Figure 39 - Connection Testing	42
Figure 40 - ODL DLUX	43
Figure 41 - Network topology view from OpenFlow Manager.....	43
Figure 42 - Flow Management	44
Figure 43 - Modifying the flow entries.....	45
Figure 44 - Attempt to get the GNS3 virtual environment and physical device to communicate	47
Figure 45 - Using Ethernet adapter cable to connect the virtual environment with the physical device.....	48
Figure 46 - VM Server Summary	49
Figure 47 - VMware Network adaptor inside local PC	49
Figure 48 - Error in GNS3 VM.....	49
Figure 49 - VMware Workstation Pro	50

Table of Content

Acknowledgement	i
Abstract	ii
List of abbreviations	iii
List of Tables	iv
List of Figures	iv
Table of Content	vi
CHAPTER ONE: INTRODUCTION	1
1.1 Background.....	1
1.2 Aim	2
1.3 Objectives.....	2
1.4 State of the Art.....	2
1.4.1 Software Defined Network.....	2
1.4.2 SDN Controller	3
1.4.3 Network Simulator	3
CHAPTER TWO: METHODOLOGY	5
CHAPTER THREE: SDN SUPPORTIVE AND NON-SUPPORTIVE DEVICES IN CST... 7	
3.1 SDN Supportive Devices	7
3.1.1 Allied Telesis	7
3.1.2 Features.....	7
3.1.3 Description	7
3.2 SDN Non-Supportive Devices.....	9
3.2.1 Commercial & Industrial Security Corporation (CISCO).....	9
3.2.2 Features.....	9
3.2.3 Description	9
3.3 D-Link	11
3.3.1 Features.....	11
3.3.2 Description	11
3.4 Cost-Benefits Analysis Between SDN Supportive and Non-Supportive Devices	12
CHAPTER FOUR: OPEN SOURCE SIMULATORS AND CONTROLLES	15
4.1 Open source network simulators.....	15

4.1.1 Types of Open Source Network Simulators	15
4.1.2 Comparison Parameters for Network Simulators	19
4.2 Open source controllers	20
4.2.1 Types of Open Source SDN Controller	21
4.2.2 Comparison Parameters for SDN Controllers	25
4.3 Types of Open Source Licenses	28
4.3.1 GNU GPL (General Public License)	28
4.3.2 The Apache License	28
4.3.3 Microsoft Public Licenses (Ms-PL)	29
4.3.4 Berkeley Software Distribution (BSD)	29
4.3.5 Common Development and Distribution License	30
4.3.6 Eclipse Public License (EPL)	30
4.3.7 MIT License	31
CHAPTER FIVE: COLLEGE NETWORK TOPOLOGY	32
5.1 College Network Description	32
5.2 College Network Simulation (GNS3)	34
CHAPTER SIX: SOFTWARE DEFINED NETWORKING	35
6.1 SDN Architecture	35
6.1.1 Application Layer	35
6.1.2 Control Layer	36
6.1.3 Infrastructure Layer	36
6.2 Setting up an environment for SDN Simulation	36
6.2.1 VMWare Workstation Pro	37
6.2.2 Ubuntu Server	37
6.2.4 OpenFlowManager	38
9.3 SDN Simulation	39
CHAPTER SEVEN: CHALLENGES	47
CHAPTER EIGHT: RECOMMENDATION	51
CHAPTER NINE: CONCLUSION	52
REFERENCES	53

CHAPTER ONE: INTRODUCTION

1.1 Background

For two decades, our country has been bringing in a digital civilization in which everything is linked and integrated into the internet, which is available from any part of the world. Most of the developing and underdeveloped nations are still using the legacy network despite knowing the fact that new technologies and approaches had already been invented. Most organizations are found to be reluctant to migrate towards the trending technologies since they cannot afford to dump the already existing infrastructures and simply invest in the new technologies.

According to Mr. Jiwan Gurung, Network and System Administrator, CST, the college still used the legacy network in most of the blocks. By legacy network, it means that the configuration must be done on each and every device manually. There are a handful of disadvantages associated with this, owing to which most companies and institutes in other parts of the world have migrated to other alternatives to networking. One such alternative is Software-Defined Networking.

SDN, introduced in 2011, is a trending technology in the field of Networking, which works with the use of 'software-based controllers or APIs for interacting with the base physical infrastructure and routing the internet traffic' (VMware, 2021). SDN allows the administrator to control the entire network from a centralized software.

For the purpose of checking the feasibility of implementing SDN in CST, a simulation of the existing network topology was conducted using the GNS3 network simulator and the ODL SDN controller. The main motive is to simplify the current network topology and to migrate towards SDN through simulation to obtain a faster, scalable, and centralized networking environment, which is cost-efficient and also provide a more reliable and efficient network system.

1.2 Aim

To analyze the implementation of SDN at CST through simulation

1.3 Objectives

1. To sort out the SDN enabled devices and their numbers in CST
2. To explore the different types of open source software available for SDN
3. To simulate the college network topology using the GNS3 simulator
4. To simulate the implementation of SDN for the CST network by analyzing the existing network topology using an ODL Controller
5. To conduct the feasibility of implementing SDN at CST
6. To analyze and recommend the implementation of SDN for CST and beyond by publishing the paper

1.4 State of the Art

1.4.1 Software Defined Network

For the future Internet, SDN is considered as one of the most favorable options. It is a cost-effective, dynamic, flexible, and controllable architecture having a decoupled network infrastructure and the forwarding tasks that allows the abstraction of network infrastructure for various functions and services of the network (Christodoulopoulos et al., 2016).

SDN is differentiated by the distinction of the control and data planes, and the network application development programmability. This gives the control plane more access and intelligence in order to handle various network devices efficiently and with little labor (Shanmugam & Ramya, 2018). To fulfill its infrastructure aims, SDN is increasingly relying on elastic cloud architectures and dynamic resource allocation (Reddy & Sivakumar, 2018).

In contrast to that, the traditional networking systems that rely on configuring the proprietary devices manually are found to be time-consuming and susceptible to mistakes (Xia et al., 2015).

1.4.2 SDN Controller

According to Rowshanrad and his team (2016), OpenDayLight is superior in light traffic networks and in tree topologies with medium traffic networks in terms of latency with 95 percent confidence. When it comes to packet loss and topology, FloodLight is found to be performing better in heavy traffic networks with tree topology (Rowshanrad et al., 2016).

Amiri and her friends (2020) used both qualitative and quantitative approaches to identify appropriate controllers. The findings revealed that the best controllers were ONOS and ODL, whereas POX, NOX, and Beacon were at the bottom (Amiri et al., 2020). Similarly, Stancu (2020) compared POX, Ryu, ONOS, and ODL based on the evaluation performed using Round Trip Time (RTT) of ICMP packets between two hosts. The ONOS RTT was the shortest in switch mode, whereas the POX RTT was the longest (Stancu et al., 2015).

According to Neto (2021), SDN controllers were analyzed based on their support for multiple protocols in the SouthBound Interface and different applications at the North Bound Interface and wide documentation availability. It was concluded that ONOS, ODL, and CISCO ACI were the top 3 controllers accepted by the market according to the adopted criteria.

1.4.3 Network Simulator

In the paper written by Khan and his friends (2012), four network simulators were evaluated based on different comparison parameters from which it was concluded that NS3, GloMoSiM, and OMNET++ had the capability to carry out simulations of large-scale networks. Similarly, Augustine (2017) compared two different simulators where OMNET++ performance was slightly inferior to that of the NS2 simulator. NS2 is generally common in academic research, but its main limitation is its complicated architecture. OMNET++ is generally used in industrial areas and has a well-designed simulation engine and a powerful GUI (Augustine, 2020).

Sara (2018) conducted a comparative analysis of GNS3 and Packet Tracer simulation tools, from which it was found GNS3 stood out to be better since it allows the creation of a more realistic topology with the use of Cisco IOS. This topology can interact with other systems

such as the existing OS in a virtual box that gives the feel of configuring the real networking device (Sari, 2018).

Lessmann and his friends (2008) compared J-Sim, OMNeT++, NS2 and ShoX based on strengths and weaknesses in terms of installation, implementation, and issues, and visualization capabilities. Based on the amount of time and effort that individuals take to familiarize themselves with the simulators, the tools were ordered as NS2, OMNeT++, J-Sim, and ShoX (Lessman et al., 2008).

CHAPTER TWO: METHODOLOGY

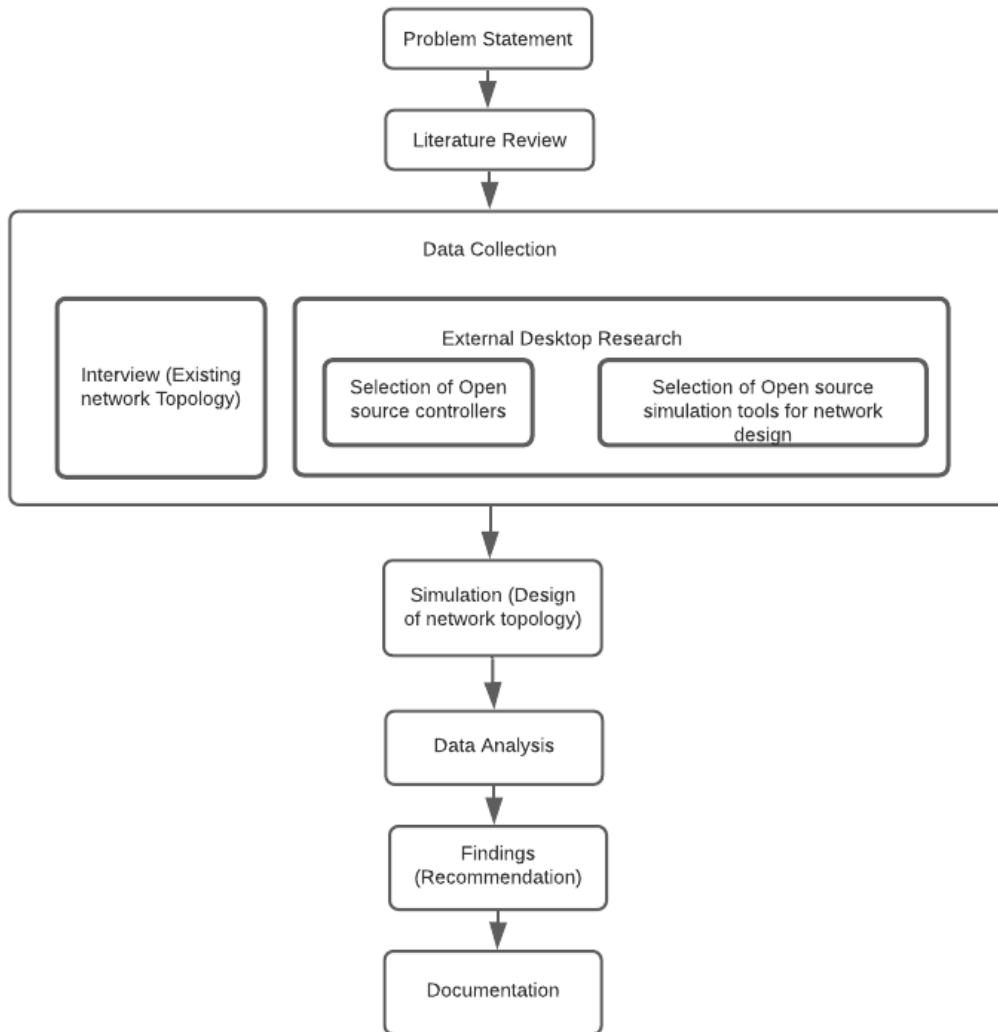


Figure 1 – Methodology

The current networking at the college campus has been deployed with the legacy networks that requires manual configuration and a huge amount of costs are incurred for the maintenance. SDN, a better alternative, was suggested to address this problem. To supplement the alternative suggested, a literature review on relevant papers and findings related to the project were carried out.

The data collection process included interviewing the ICT officer of the college from time to time and exploring various open-source software requirements for the project using external desktop research. From this external desktop research, GNS3 and ODL were selected as the tools that were to be used for this project.

Based on the research carried out, the outcomes and the findings were recommended for a reliable and robust network system. Finally, the findings were compiled into the documentation.

CHAPTER THREE: SDN SUPPORTIVE AND NON-SUPPORTIVE DEVICES IN CST

3.1 SDN Supportive Devices

3.1.1 Allied Telesis

The switches from Allied Telesis have given a highly solid foundation for this basic necessity. Besides being consistent and dependable, Allied Telesis switches help to make networking more secure and much easier to use. Both managed and unmanaged switches cater to the demands of enterprises of all sizes (Allied Telesis, 2022).

3.1.2 Features

- Simple Administration: an automated network administration technology that saves time and money.
- Intelligent and secure: they create a network that protects itself in order to respond to threats instantly and without the need for manual intervention.
- Certification: an independent evaluation to ensure that you are utilizing the best product possible.

3.1.3 Description

Given below are the model number, number and description of the Allied Telesis devices that are currently deployed in CST which compose up to 50% of the networking devices of CST.

Table 1 - Allied Telesis devices

Sl. No	Model no. of networking device	No. of devices	Description
1	x230-10GP	6	It has 8x 10/100/1000T PoE+ (Power over Ethernet Plus) ports and two SFP Gigabit uplink ports. PoE+ gives about 30 Watts for handling maximum power in connection with edge devices such as multi-band WAPs (Wireless Access Points), Zoom digital

			security cameras. The compact design allows for flexible deployment.
2	x230-18GP	7	It has 16x 10/100/1000T PoE+ ports and two SFP Gigabit uplink ports. It is excellent for network edge applications, enabling the rise of wireless networking and digital video surveillance.
3	x230-18GT	3	It has 16x 10/100/1000T ports and two SFP Gigabit uplink ports. It supports Gigabit to the desktop for optimal network performance.
4	x230-28GT	3	It has 24x 10/100/1000T ports with four SFP Gigabit uplink connections. Its comprehensive feature set makes it excellent for network edge applications.
5	x510-28GSX	3	It has 24x 100/1000X fiber access ports and 4x 1G/10G SFP+ uplink ports, making it ideal for fiber networks.

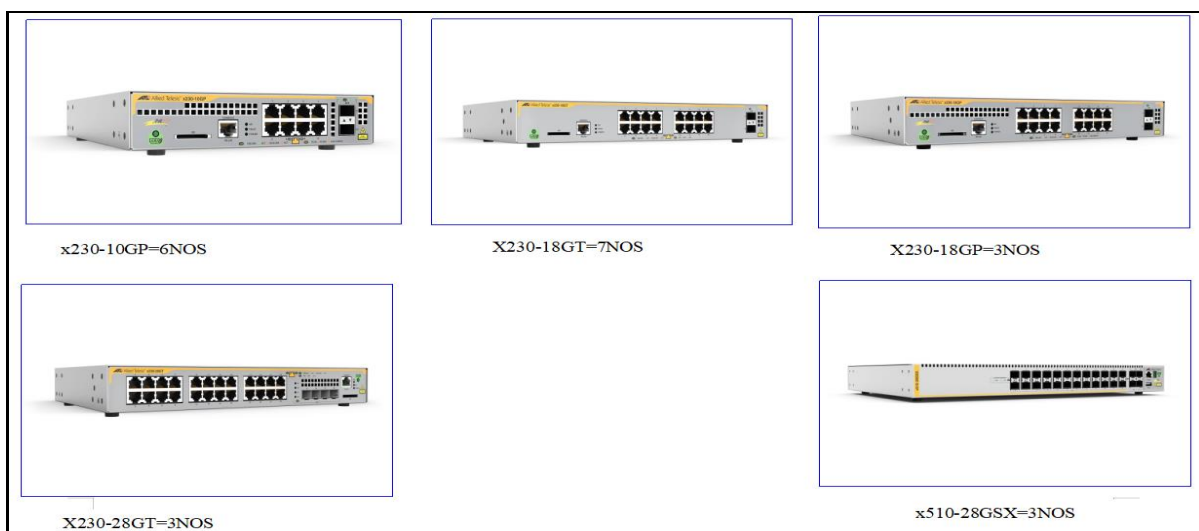


Figure 2 - Allied Telesis Switches

- *X230 Series-Gigabit Managed Edge Switches*

The x230 series delivers an outstanding collection of capabilities in a small design, suitable for flexible deployment, with maximum network performance for even the most demanding applications.

- *x510-Gigabit Stackable Managed Switches*

A high-performance and scalable switch family designed to meet today's stringent network needs. The x510 series' large range of models makes them suitable for a variety of applications, from tiny cores to network edges.

3.2 SDN Non-Supportive Devices

3.2.1 Commercial & Industrial Security Corporation (CISCO)

The organizations to which Cisco offers a complete spectrum of switching solutions include Data Centers and Enterprise Networks. These solutions are tailored to a variety of businesses, including service providers, financial services, and governments (Cisco, 2022).

3.2.2 Features

- 1) Intent-Based Networking Design: Cisco's switches integrate corporate purpose into consistent, automated network policies.
- 2) Intelligent and safe: With integrated security measures, it helps stay ahead of developing threats.
- 3) Adaptable and programmable: Provides complete stack programmability from ASIC to OS.

3.2.3 Description

Given below are the model number, number and description of Cisco devices that are currently deployed in CST.

Table 2 - Cisco networking devices

Sl. No	Model no. of networking device	No. of devices	Description
1	Catalyst3560	1	It has 10/100/1000 and PoE configurations which helps to make its productivity optimal. Hence, it can be used as an appropriate access layer switch for small business LAN access or branch-office environments.
2	Cisco Nexus 3064	1	It has several 1/10 Gigabit Ethernet connection options. SFP+ transceivers in the first 48 ports provide 1 and 10 Gigabit Ethernet connections, while QSFP+ transceivers in the last 4 ports provide 4 x 10GbE connectivity.
3	Cisco SG300-52	1	It has 48 PoE ports with a Gigabit Ethernet connection. With the help of a single Ethernet with this connection, the users can connect the power network endpoints.

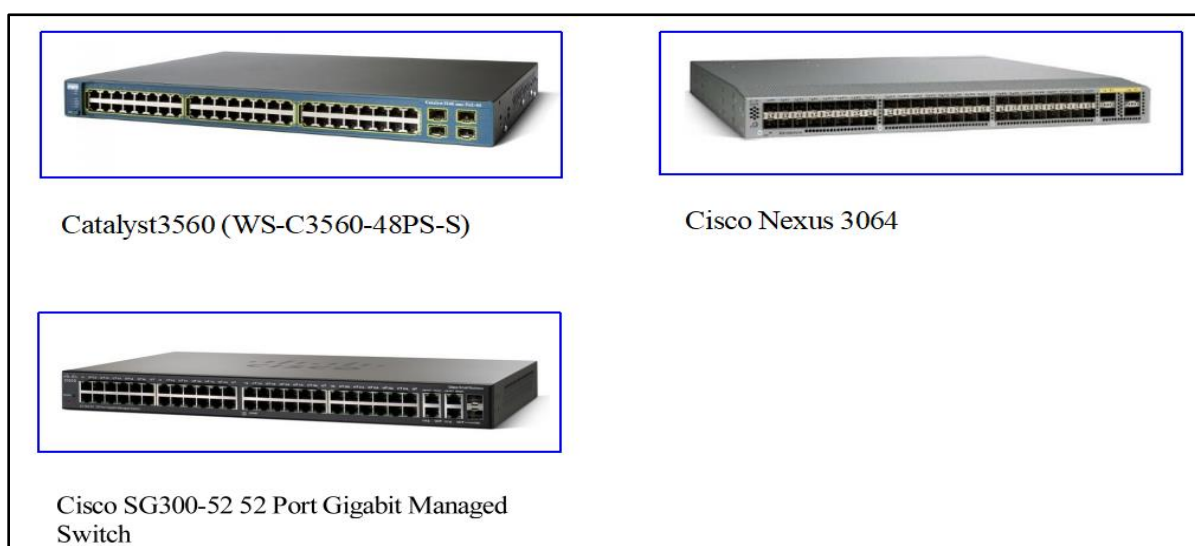


Figure 3 - Cisco Switch

3.3 D-Link

D-Link switches connect into several verticals at all scales, allowing the extension of network more cost-effectively (D-Link, n.d).

3.3.1 Features

- 24x 10/100/1000Base T ports
- 4x Gigabit RJ-45/SFP Combo parts
- Advanced L2 Switching and security Features
- L2+Static Routing
- Optional “Standard Mode” or “Surveillance Mode” management user interface

3.3.2 Description

Given below are the model number, its numbers, and description of the D-Link device that are currently deployed in CST.

Table 3 - D-Link switch

Sl. No	Model No.	No. of devices	Description
1	DGS-1210-28	8	It's a PoE Web Smart Switch with 24 Gigabit SFP ports and 24 10/100/1000 Base-T PoE ports. In addition to IEEE 802.3af compliance, ports 1-4 support IEEE 802.3at power output of up to 30 watts.



Figure 4 - D-Link switch

3.4 Cost-Benefits Analysis Between SDN Supportive and Non-Supportive Devices

The reason for sorting out devices was to carry out a Cost-Benefit Analysis on the implementation of SDN in CST. The cost-benefit analysis is an assessment to check whether the estimated costs exceed the estimated income and other benefits. As of 2022, more than 50% of the networking devices in CST were found to be supporting SDN. The following are the list of SDN supportive and SDN non-supportive networking devices along with their number and cost per device (PriceSpy, 2022).

Table 4 - Cost structure for SDN supported

SDN supported devices (Allied Telesis)			
Model Number	Cost per device (Nu.)	No. of device	Total cost
x230-10GP	59,497.97	6	356987.82
x230-18GT	50,310.06	7	352170.42
x230-18GP	68,326.83	3	204980.49
x230-28GT	68,326.83	3	175873.86
x510-28GSX	68,326.83	3	1024498.62
<i>The grand total of all Allied Telesis devices</i>			<i>2114511.21</i>

Table 5 - Cost structure for SDN non-supported

SDN non-supported devices (Cisco and D-Link)			
Model Number	Cost per device (Nu.)	No. of device	Total cost
Catalyst 3560	6,161.04	3	18483.12
Nexus 3064	94,690.37	3	284071.11
Cisco SG300-53	33,043.72	8	264349.76
DGS-1210-28	10,615.94	8	84927.52
<i>The grand total of all Cisco and D-Link devices</i>			<i>651831.5</i>

As per the ICT, the total expenditure (approximately):

1. If all networking devices are replaced by SDN supported devices (Allied Telesis) are Nu. 55,00,000 with a warranty period of 5 years.
2. If all the networking devices are replaced by SDN non-supported devices (Cisco/D-Link), the total cost is Nu. 35,00,000 with a warranty period of 1 year.

Based on the warranty of the networking devices in CST, we if make the assumption, then

$$\begin{aligned}\text{Total cost of SDN non-supportive devices in 5 years} &= 5 * 35,00,000 \\ &= \text{Nu.175,00,000 (approximately)}\end{aligned}$$

$$\text{Total cost of SDN supportive devices in 5 years} = \text{Nu 55,000,000 (approximately)}$$

Hence, since **55,00,000 < 175,00,000**, the cost of SDN supportive devices, which will last for 5 years, is less compared to the cost of SDN non-supportive devices in 5 years. Approximately, an amount of Nu.120,00,000 can be saved in a period of 5 years.

The following are the pictures of switches that got damaged due to lightning.

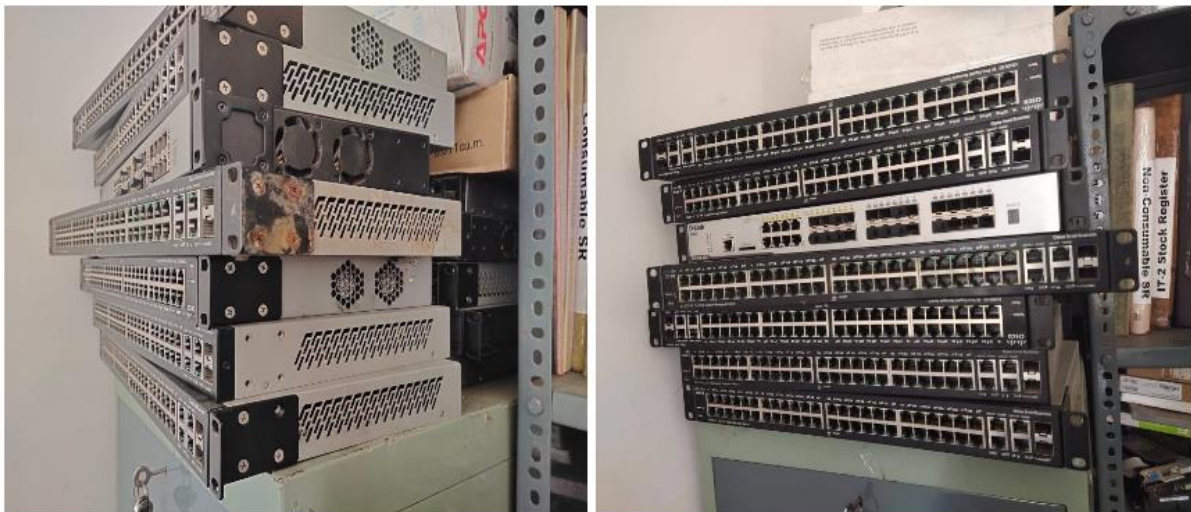


Figure 5- Lightning affected SDN-Non-supported switches

In addition to the above analysis, there are certain benefits of Allied Telesis devices which adds to SDN supportive devices more advantages compared to the non-supportive ones. The following are the respective benefits:

- Allied Telesis devices are given a warranty of five years
- It provides more efficient network management
- It facilitates greater reliability via Automation

In contrast to these benefits, the SDN non-supportive devices has the following drawbacks:

- The SDN non-supportive devices in CST have a warranty of just one year
- There is no reliable mode of network automation
- These devices are more vulnerable to damages

CHAPTER FOUR: OPEN SOURCE SIMULATORS AND CONTROLLERS

In this section, a number of open source network simulators and SDN controllers were explored and compared based on a number of parameters. This was done to select the appropriate software tools for the simulation which rightfully fulfilled the requirements of this project.

4.1 Open source network simulators

A network simulator is a software application that is used to facilitate an efficient and cost-effective method of determining the behavior of a network under numerous operational conditions. situations. Conducting simulations before the actual implementations, such as installing a network or making modifications to an existing network, helps in studying. several possibilities through testing network performance, which ultimately aids in detecting possible problems and rectifying those problems without having the risk of causing any accidental damage to the existing network.

4.1.1 Types of Open Source Network Simulators

- GNS3



Figure 6 - GNS3

Although it is quite difficult to set up GNS3, in comparison to Cisco Packet Tracer, GNS3 is more advanced and offers greater flexibility. Besides, GNS3 is open-source, which allows users to dive into its source code and extend its base functionality. It also has support for more device options, such as emulated devices (Andrea, 2020).

Aside from simulations, GNS3 can run Cisco IOS images in an emulated virtual environment by supporting most of the features that are available in the real devices, allowing GNS3 to integrate with the real, physical devices. GNS3 has support for different vendors such as Arista and Mikrotik in addition to the IOS.

- Cisco Packet Tracer



Figure 7 - Cisco Packet Tracer

One of the comprehensive network simulators that provides realistic simulation and visualization experiences is Cisco Packet Tracer. Cisco Packet Tracer provides innovative features such as assessment, activity authoring capabilities, and multi user collaboration and competition opportunities, making it ideal to be used as a tool for teaching and learning (Cisco NetAcademy, 2022).

- NS3



Figure 8 - NS3

NS3 is an event-driven network simulation tool which is strictly used for research, development, and educational purposes since it is licensed under the GNU GPLv2 license. One

of the most captivating things about NS3 is that it contains an extensive Wiki documentation for beginners to assist them with the setup. NS3 is said to be satisfactorily intuitive, even for first-time users. NS3 also works with platforms such as Eclipse IDE, NetBeans and many other platforms (Andrea, 2020).

- Mininet

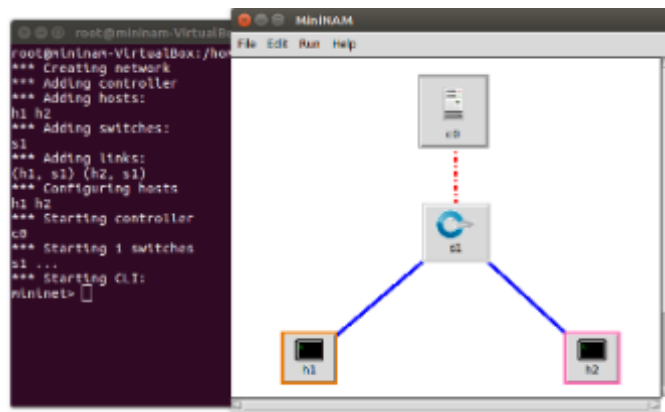


Figure 9 – Mininet

Mininet is also one of the open-source software for network simulation based on OpenFlow making it ideal for building OpenFlow solutions. With sufficient technical knowledge, Mininet is known for its excellent flexibility when it comes to setting it up. Mininet requires Linux to be installed either natively or through Virtual Box or VMWare in case of Mac and Windows operating systems (Andrea, 2020).

- OMNeT++



Figure 10 - OMNeT++

OMNeT++ (Objective Modular Network, Testbed in the C++ programming language), is a platform for simulation of networks mainly utilized for discrete-event systems and is primarily targeted to simulate distributed systems and computer networks. It is to be noted that OMNeT++ cannot be used without any extensions for wireless communication (Fokus, 2019).

- NS2



Figure 11 - NS2

NS2 (Network Simulator, Version 2) is one of the event-driven network simulators. NS2 is immensely useful for studying the dynamic nature of communication networks. It can be used to simulate both wired as well as wireless network functions and protocols (Issariyakul, 2011).

- JiST



Figure 12 - JiST

JiST is a simulation engine for discrete events running on a standard JVM. It basically presents a prototype that demonstrates the “virtual machine-based simulation”, which is a new technique that aims to build platforms for discrete event simulation by combining traditional systems and simulation designs that are based on programming language.

4.1.2 Comparison Parameters for Network Simulators

To select the most appropriate network simulator for this project, the following are the parameters based on which the comparison among the network simulators was carried out:

- Open source

The source code of a piece of software that is said to be open source can be available on the internet freely so that the general public can use or modify the source code. One of the main uses of open source network simulators would be that the source code comes with affiliated packages and no limitations have been set on its interfaces, which will keep it open for future improvements as well (Mahmood et al., 2013).

- Suitable for wireless networks

Wireless network means network connection done without wired connection. WiFi is an example of a wireless network. This parameter is verified to see if the network simulator has support for wireless network connectivity.

- Emulates real network interface

Network emulation is a process of evaluating the performance of a network or forecasting the impression of any possible changes or optimizations by simulating a network that is already partially under real- world implementation. The purpose of emulation in general is to test and validate a simulation model with its real-world counterparts, or vice versa, in the due course of development of a real-world implementation, to test it in a simulated model (Mahmood et al., 2013).

- User friendly interface

A user interface that allows the users to easily understand and navigate intuitively through the application efficiently is said to be a user-friendly interface (Gupta et al., 2013).

Table 6 - Comparison table for Network Simulation tools

Parameter	Packet Tracer	GNS3	NS3	Mininet	OMNeT++	NS2	JisT
Open Source	No	Yes	Yes	Yes	Yes	Yes	Yes
Program Language	Javascript	Python	C++/Python	Python	C++	C++	Java
Suitable for wireless network	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Emulates real network interface	Yes	Yes	No	Yes	Yes	Yes	Yes
User Friendly Interface	Yes	Yes	No	Yes	Yes	No	Yes
Platform	Linux, Windows, Mac	Linux, Mac, Windows	Linux, Mac	Linux	Windows	Linux, Mac	Linux, Mac, Windows
License	Proprietary	GPL	GPL	BSD	GPL	GPL	Apache

The number of network simulators chosen for this project was narrowed down to the top seven, which were then compared based on the comparison parameters set. The parameters were set based on the needs of the project. Upon comparing the network simulators, it was found that GNS3 qualified the maximum number of parameters among other network simulators. Hence, GNS3 was selected as the network simulator for this project.

The GNS3 is a network simulation tool that allows users to view and import software images from many manufacturers. The GNS3 VM was known to be the most popular method for importing images. The simulator's most exciting feature was that it acted as a whole network by bringing in a network device.

4.2 Open source controllers

The SDN controller is considered to be the network's heart and brain, handling flow management for improved network administration and application performance. It directs traffic based on forwarding policies specified by a network operator, minimizing the requirement for each network device configuration. The control plane of the network device is removed and replaced with centralized controller software to oversee automated network

operations. With the help of protocols such as OpenFlow, the controller is able to connect and communicate with the underlying OpenFlow switches through a southbound interface (English, 2018).

4.2.1 Types of Open Source SDN Controller

- OpenDayLight

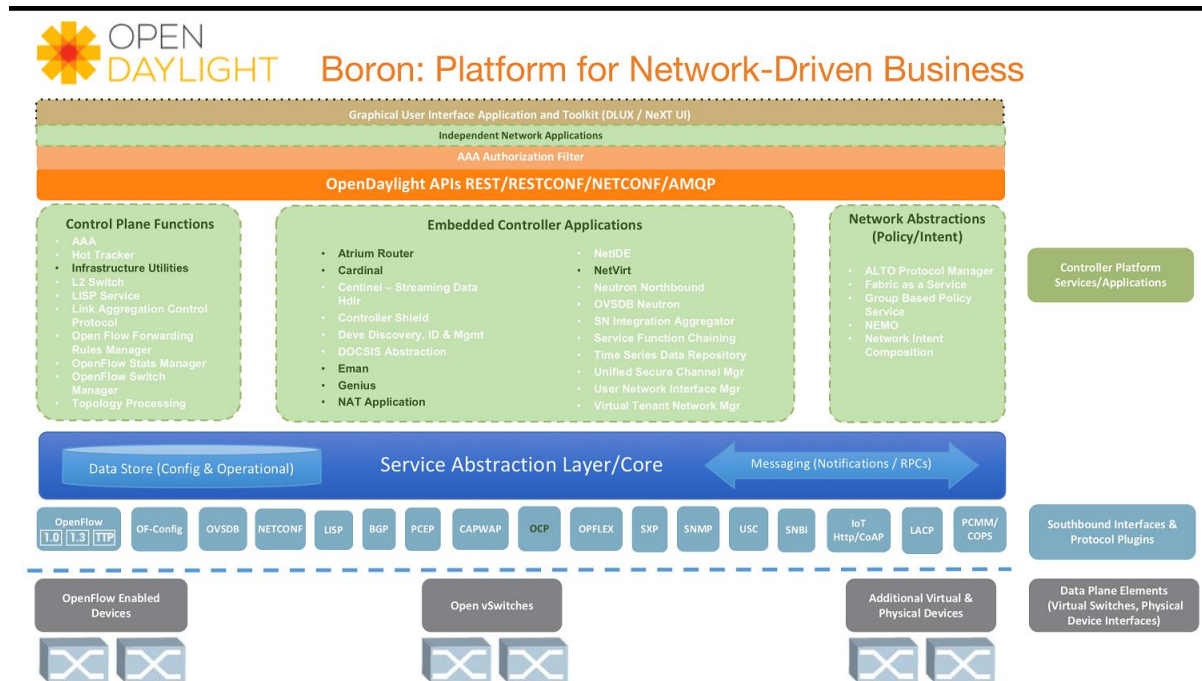


Figure 13 - OpenDayLight Architecture

OpenDayLight or ODL, is a platform for SDN that provides a centralized and programmatic control along with a provision of monitoring the network devices through the use of open protocols. It provides an interface that allows users to control and manage network devices (OpenDayLight, 2021).

- Floodlight

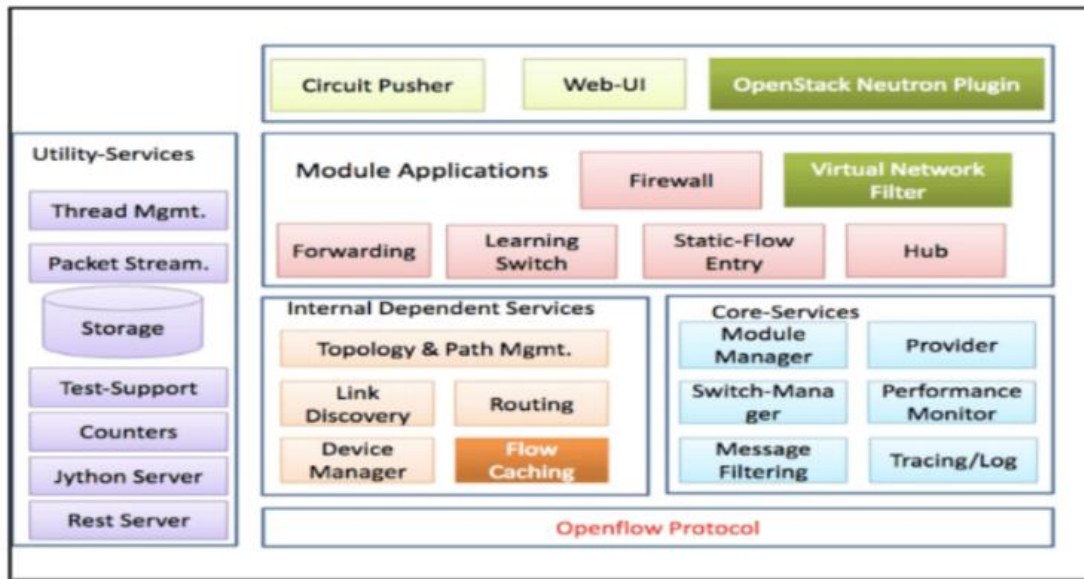


Figure 14 - Floodlight Architecture

Floodlight is an OpenFlow controller based on Java. It was contributed by Big Switch Networks as one of its core elements. The architecture of Floodlight is based on the company's commercial offering called the Big Network Controller (Rao, 2021).

- Ryu

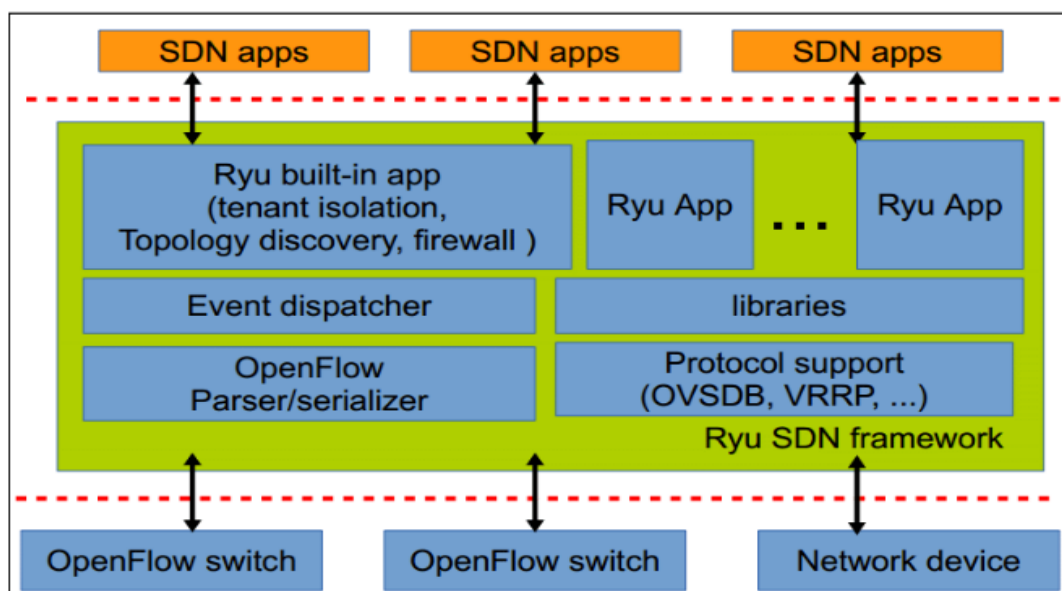


Figure 15 - Ryu Architecture

Ryu is a framework for SDN that is component-based and provides software components consisting of APIs that are well-defined. This provides an easy platform for the creation of new applications for controlling and managing networks. To facilitate the management of network devices, Ryu provides support for various protocols, such as OpenFlow (Ryu SDN Framework, 2017).

- Open Network Operating System

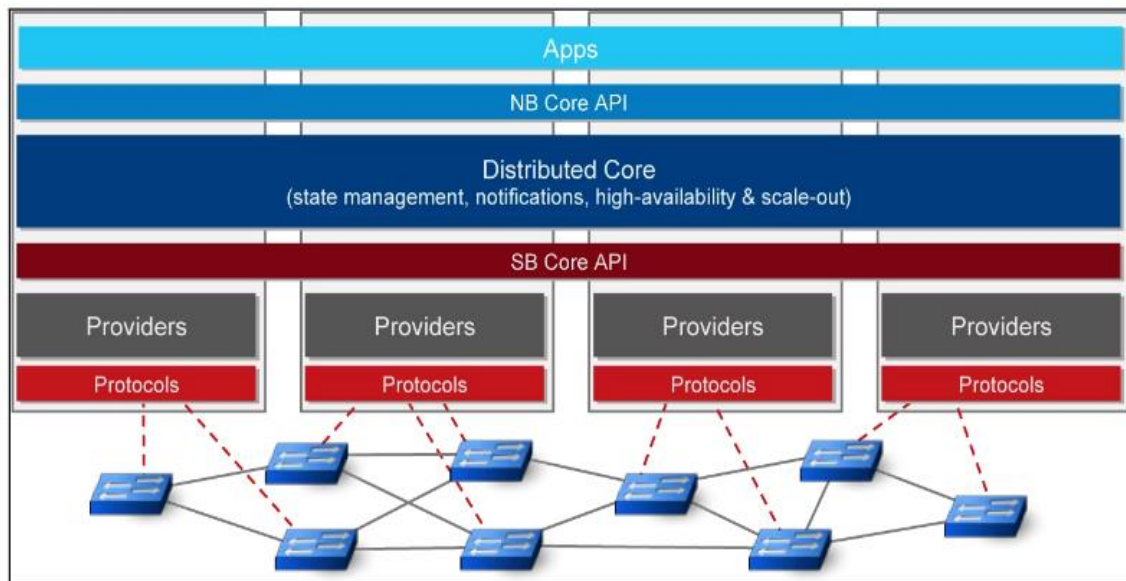


Figure 16 - Architecture of Open Network Operating System

Open Network Operating System (ONOS) is one of the leading open source controllers for SDN for the construction of new-generation SDN/NFV solutions. ONOS offers the flexibility for creating and deploying new dynamic network services through its simplified programmatic interfaces. With ONOS, the need for running switching and routing control protocols inside the network fabric is eliminated since ONOS has support for both configuration and real-time control of the network (OpenNetworkingFoundation, 2022).

- NOX

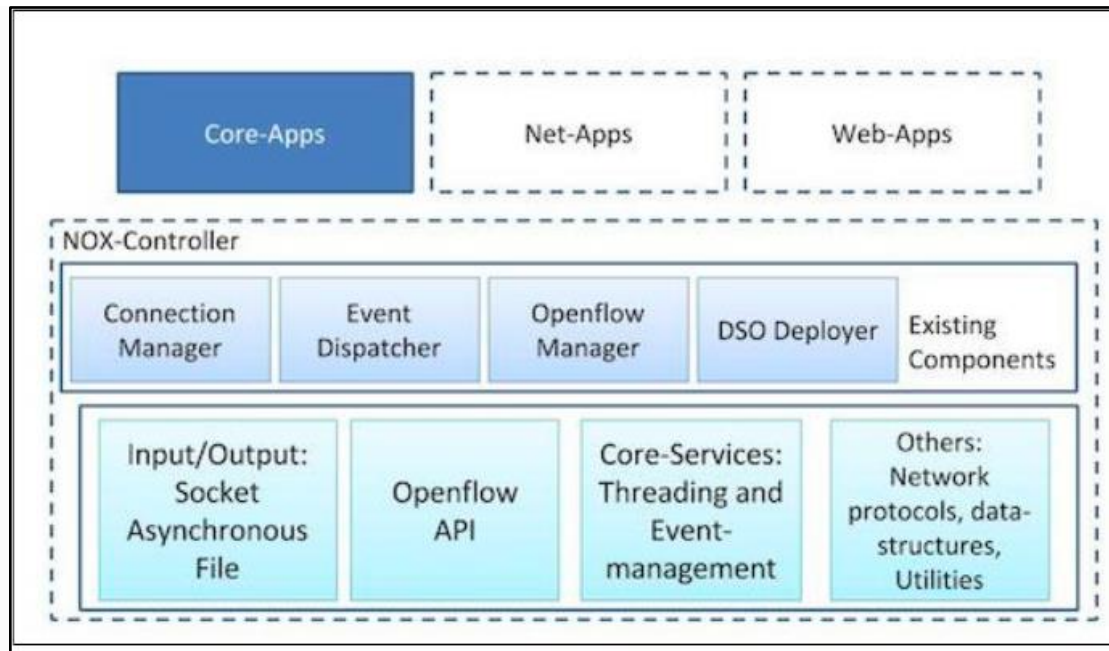


Figure 17 - Architecture of NOX

NOX is the native controller of OpenFlow that is used as a platform for controlling networks with a high-level programmatic interface where the network control applications can be managed and developed (Rao, 2021).

- POX

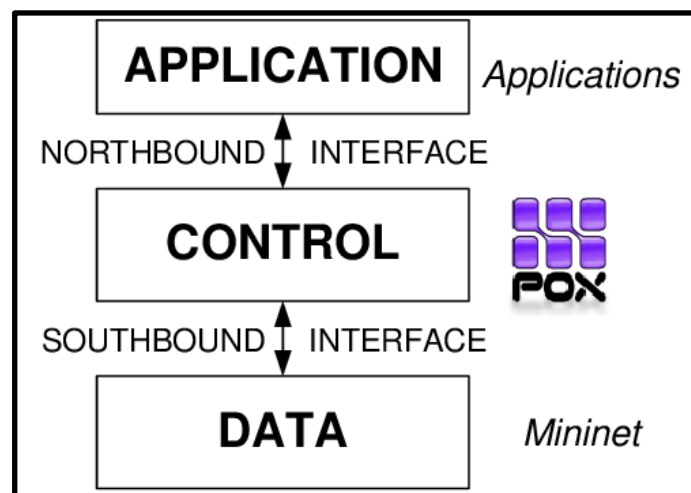


Figure 18 - Architecture of POX

POX uses two protocols, namely OpenFlow or OVSDB to provide a framework to communicate with SDN switches. With the Python programming language, POX can also be used for creating an SDN controller. Nevertheless, with the use of stock components that come bundled with it, POX can be used immediately as a basic controller for SDN (Linkletter, 2015).

4.2.2 Comparison Parameters for SDN Controllers

The following are the parameters based on which the comparison among the SDN controllers was carried out.

- GUI

In contrast to the traditional command-line or text-based interfaces, Graphical User Interface (GUI) is basically an interface that allows users to develop interactions by using electronic devices such as smartphones and computers, which can be accessed with the help of visual indicators, which is inclusive of menus and icons (Stoltzfus, 2021).

- Programming Language

It is a computer language written as a set of instructions that help developers and programmers interact with computers to perform certain tasks. certain tasks. a specific task (Javatpoint, 2021). The controllers are built either by using just one programming language or by using a combination of multiple programming languages in their core and modules. Under certain conditions, a controller that is built using multiple languages was said to have higher performance with efficient memory allocation and the ability to be executed on multiple platforms (Zhu et al., 2021).

- Architecture

The architecture of a controller can either be centralized or distributed, whereby a centralized controller was popular among networks of smaller scale, while a distributed controller was accessible across numerous domains. The architecture of a controller can be further classified as flat or hierarchical. The controller with a flat architecture had equal responsibilities assigned to all its controller instances, whereas a hierarchical controller consisted of a root controller.

- Programmable Interface

Controllers can have different APIs, each with different functions. The Northbound API contributed to the controllers by facilitating various types of useful activities such as topology monitoring, clusters, network virtualization, intrusion detection, and forwarding of data flow, all of which were performed using the network events that were generated by the data plane devices. The Southbound API enables the interactions between the controller and the SDN that enables the routers and switches, whereas the east-west API is used in a distributed or hierarchical environment to form peers coming from different domains of multiple controllers. It is to be noted that only a few selected controllers had customized APIs for specific use while the rest of the controllers did not provide all APIs.

- Platform and Interface

The compatibility of the controllers with specific operating systems during the implementation is defined by platform and interface properties. It is to be noted that Linux distributions are popular among the majority of the controllers. Some of the controllers have graphical interfaces while others have web-based interfaces for the use by administrators to configure and view statistical information.

- Threading and Modularity

Controllers can be single-threaded or multiple-threaded based on the suitability for deployments. For lightweight SDN deployments, a single-threaded controller was preferred, whereas in the case of commercial purposes, including optical networks, 5G, and SDN-WAN, multi-threaded controllers were more suitable. In terms of modularity, which is the ability of a controller to allow different applications and functionalities to be integrated, a higher modularity was preferred since a controller with higher modularity permitted faster execution of tasks in a distributed environment.

- License, Availability, and Documentation

When it comes to licensing, some of the controllers discussed here were licensed as open source while others contained a proprietary license. To avail of the proprietary licenses, special permission for research purposes was required. The source code of the open source controllers

is available online. Hence, further changes could be made as per the requirements by anyone. Also, it was observed that the majority of the controllers had a lack of proper documentation while assessing them online. Nevertheless, the few controllers that were updated on a regular basis were found to include current and detailed documentation for all available versions, as well as community-based support.

Table 7 -Comparison table for SDN Controller

Parameters	ODL	FloodLight	RYU	ONOS	NOX	POX
Introduction Year	2013	2013	2013	2014	2009	2009
GUI	Web based	Web based Java	Python	Java	Python +qt4	Python +qt4
Programming Language	Java	Java	Python	Java	C++	Python
Platform support	Linux, Mac, Windows	Linux, Mac, Windows	Linux	Linux, Mac, Windows	Linux	Linux, Mac, Windows
Documentation	Good	Medium	Medium	Good	Medium	Poor
Architecture	Distributed Flat	Centralized	Centralized	Distributed Flat	Centralized	Centralized
North-Bound API	Rest, RestConf,XMPP ,NetConf	Rest, Java RPC,Quantum	Rest	Rest,Neutron	ad-hoc	Rest
South-Bound API	Open Flow	Open Flow	Open Flow	Open Flow	Open Flow	Open Flow
License	EPL	Apache	Apache	Apache	GPL	Apache
Multithreading	Yes	Yes	Yes	Yes	Yes(Nox-MT)	No

As in selecting a network simulator, the same method of comparison was used to choose the SDN controller, which was based on required parameters. According to the comparative table, ODL supported all of the required parameters. As a result, ODL was chosen as the SDN controller, which satisfied the requirements of the project.

ODL provides open northbound APIs that can be used by applications. Those same applications simply used controllers to collect network data, run analytics algorithms, and create new network rules using the ODL Controller. The ODL Controller is a standalone software device with its own Java Virtual Machine.

4.3 Types of Open Source Licenses

4.3.1 GNU GPL (General Public License)



Figure 19 - General Public License

In an attempt to prevent the GNU software from becoming proprietary, Richard Stallman designed GPL. Its type of license is copyleft, which means that any software based on a GPL component must be released as open source (Goldstein, 2021).

4.3.2 The Apache License



Figure 20 - Apache License

Apache Software Foundation introduced the apache license. It is an extensively used license with strong community support. They grant their users the right to use it freely, make changes, and distribute any Apache-licensed product. However, users must adhere to the provisions of the Apache License while making changes.

4.3.3 Microsoft Public Licenses (Ms-PL)

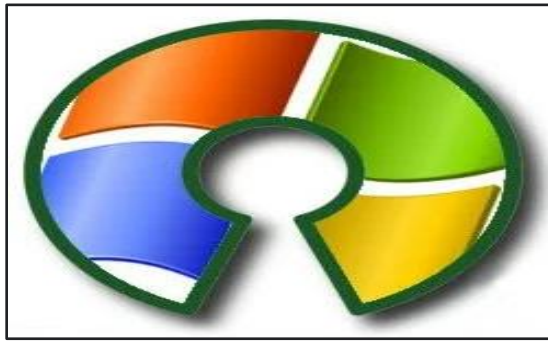


Figure 21 - Ms-P License

It is one of the free and open source software licenses created and owned by Microsoft to be used with open source projects. Any software under this license may be reproduced and distributed in its original or derivative form. However, the users are not allowed to utilize any of the contributors' names, logos, or trademarks when doing so. It is not necessary to distribute the source code of software when it is distributed.

4.3.4 Berkeley Software Distribution (BSD)



Figure 22 - BSD License

In this type of licensing, one is allowed or modify and are also allowed to distribute the source code but must keep the copy of the works are kept available as open.

4.3.5 Common Development and Distribution License



Figure 23 - CDCL License

It is an open source license that was released by sun microsystem and user are allowed to duplicate or distribute any work of this license keeping in mind that one cannot modify or delete others work and also should give credit to the contributor or inventor.

4.3.6 Eclipse Public License (EPL)



Figure 24 - EPL license

It is a copyleft license whereby the users must disclose the updated code under the Eclipse Public License when they alter an EPL-based component and share it in source code form as part of their software. If the users provide such a piece of software in object code form, they must mention that the source code is accessible upon request to the receiver. Users must also offer a mechanism for requesting the source code. If the users redistribute software that has an EPL component, they must include the whole license text as well as the copyrights.

4.3.7 MIT License



Figure 25 - MIT License

It is one of the free licensing provided for the software with the condition that one should be providing the copy of the original work with their source code.

CHAPTER FIVE: COLLEGE NETWORK TOPOLOGY

5.1 College Network Description

Before proposing a network, it is important to know the topology of the network that exists currently. A network topology is the description of how the systems are arranged in a computer network. Almost 50% of the devices support SDN in the college network. Due to the high cost of switches, i.e., 70, 000 to 80,000 per switch, only 80 switches are in the college network currently.

The following figure presents the detailed view of the existing CST network topology, followed by a figure that presents a simplified block diagram.

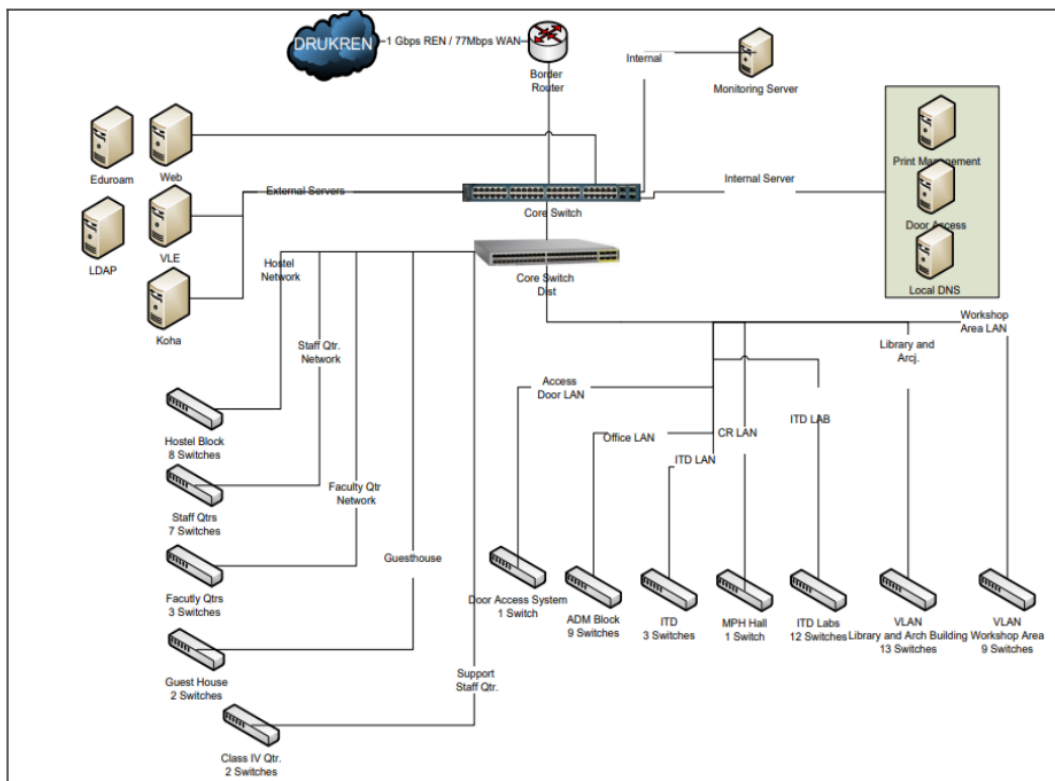


Figure 26 - A detailed block diagram of the existing network topology

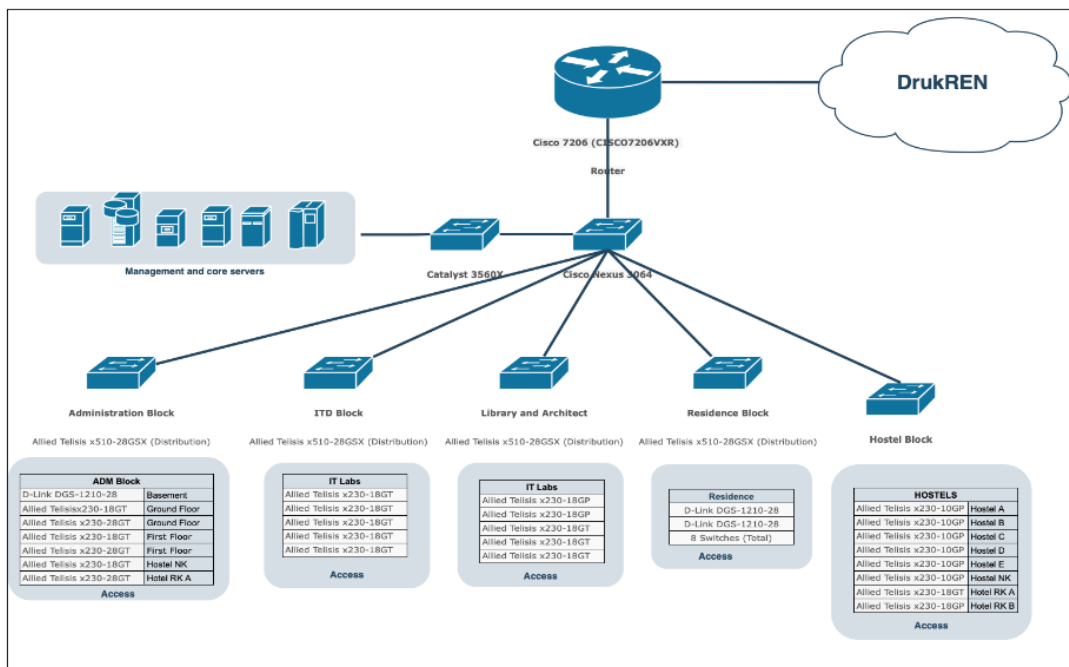


Figure 27 - Simplified block diagram of the existing network topology

Currently, CST follows star topology. In order to study, although it is important to study the logical as well as the physical topology, the consideration was only for the physical network topology.

The College of Science and Technology is one of the colleges that has access to the DrukREN network with a speed of 1 Gbps/74 Mbps. It is the primary source of internet connection for the college. The network traffic from the DrukREN is received by the border router, which resides at the edge of the network and ensures its network connectivity with external networks or the internet. The college network is divided into three layers; a core switch, which resides within the backbone of the network that serves as the gateway to the internet, and two other layers to deliver frames or packets as quickly as possible in the center of the network. There are various types of servers set up which can be accessed by the students and staff. A distribution switch collects the data from all the access switches and is then forwarded to the core layer switches. At the bottom layer, there are access switches that directly interact with the end-user devices.

5.2 College Network Simulation (GNS3)

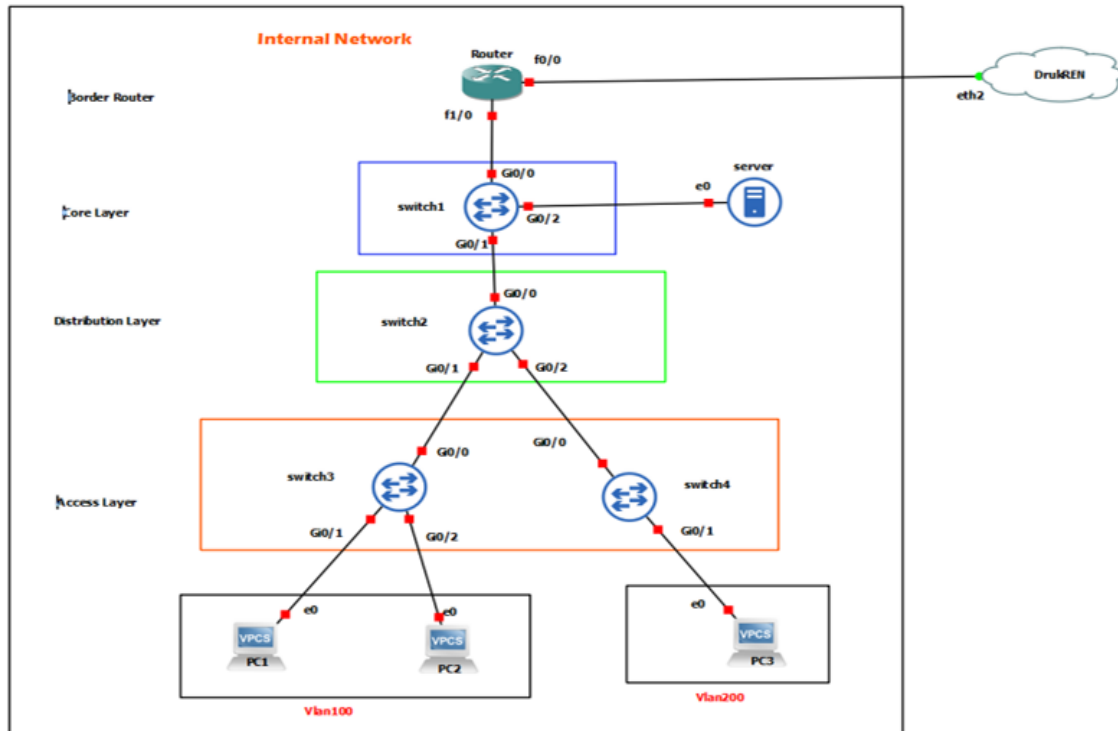


Figure 28 - College Network Simulation using GNS3

The local control plane, which is effectively the device's brain, and the data plane, which forwards traffic, are both present in legacy networking devices. Each device's control plane runs locally. SDN advocates said that this was a difficult approach to figure out the network's topology. There was no single device that could be seen as the complete network; each device had to figure out what the network looked like independently before synchronizing. Since the college has several network devices, using the Command Line Interface from a management standpoint, this necessitates connecting to each device and manually configuring the networking equipment.

Not only was the manual configuration time-consuming, it was also prone to human error. The configuration of the devices should be done on each device, and there was no centralized control of the underlying network. Since networking devices are proprietary, no one can write them. They have an application for them, but they have no access to the vendor's operating system code to make changes.

CHAPTER SIX: SOFTWARE DEFINED NETWORKING

SDN is one of the trending technologies in the field of networking that uses software-based controllers and APIs in order to make connections and communicate with the underlying devices and helps to direct traffic on the network.

6.1 SDN Architecture

The following figure depicts a detailed architecture of SDN.

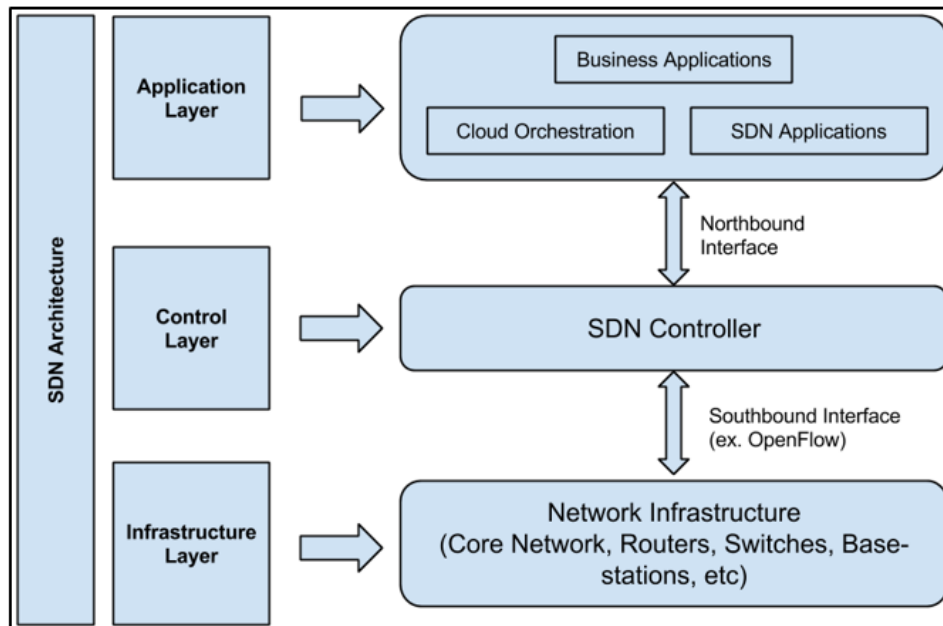


Figure 29 - SDN Architecture

As can be seen in the figure given above, the architecture of SDN is composed of three main layers. They are:

6.1.1 Application Layer

This is a layer of network applications and functions that are used by organizations. The network applications include systems such as intrusion detection systems and functions such as load balancing and firewalls. Additionally, it contains business and SDN applications and orchestration of the cloud.

6.1.2 Control Layer

This layer is referred to as the brain of SDN. The centralized SDN controller software facilitates this layer with the intelligence to manage the flow of traffic in the network.

6.1.3 Infrastructure Layer

This layer is composed of all the networking devices that are used to forward the network traffic to their destinations.

The communication between these three layers is facilitated with the help of the following APIs. They are called Northbound and Southbound APIs. The first type of API, that is northbound API, supports the Applications layer to communicate with the Control layer while the southbound API allows communication process between the Infrastructure layer and the Controller layer (Rosencrance et al., n.d.)

The SDN architecture is built on four pillars (Christodoulopoulos et al., 2016):

1. The data plane and the control plane are separated.
2. Promotion decisions are taken primarily on the basis of the flow rather than the destination.
3. SDN Controller is an external entity that handles the control logic.
4. The network can be programmed using the controller's applications.

Simple network management, easy traffic monitoring; rapid development services, automation settings; network virtualization, reduction of operating costs, assignment of VLAN and QoS settings are done automatically with Software Defined Network.

6.2 Setting up an environment for SDN Simulation

The implementation of the SDN will be done with the help of the ODL Controller, which will be installed on the Ubuntu server, which in turn will be installed on the hypervisor, VMware Workstation. The various tools or platforms that were used in setting up the environment were:

6.2.1 VMWare Workstation Pro

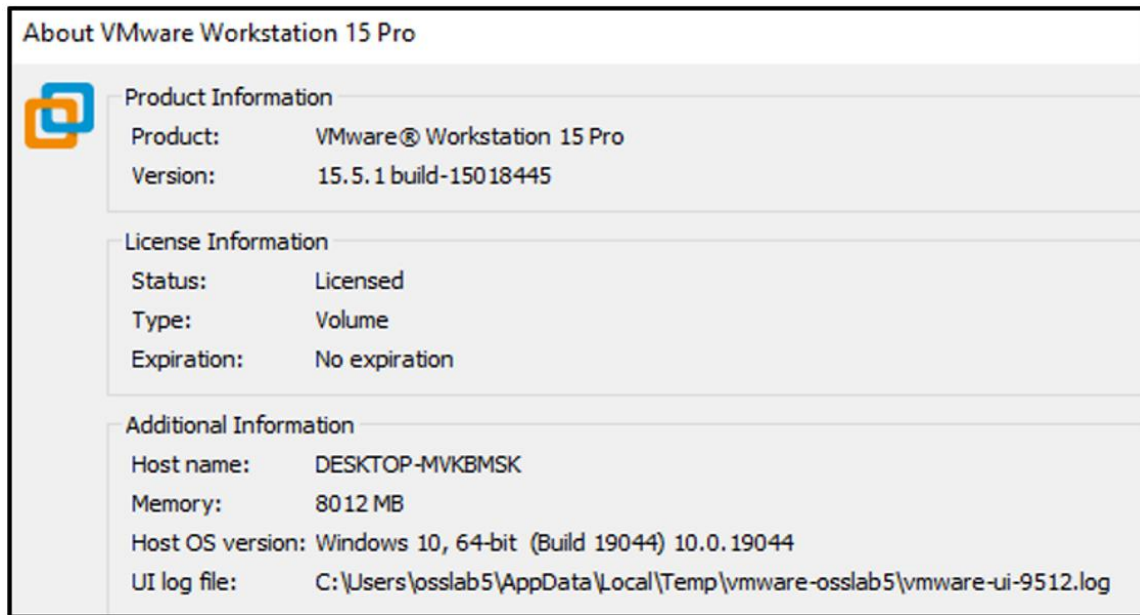


Figure 30 - VMWare Workstation Pro (Version 15.5.1)

It is a hypervisor that is installed on the host operating system and based on the users' requirement for various flavors of operating systems can be installed on it and also dependent on the resources available and one need to have an iso image to boot any machines on it (VMWare, 2020).

6.2.2 Ubuntu Server

It is a server operating system developed by Canonical, which is a free, open- source programming community across the globe that operates on almost any type of hardware or a virtualized platform. This can serve websites, file shares, and containers and also provide organizations with a strong cloud presence (Wallen et al., 2020)

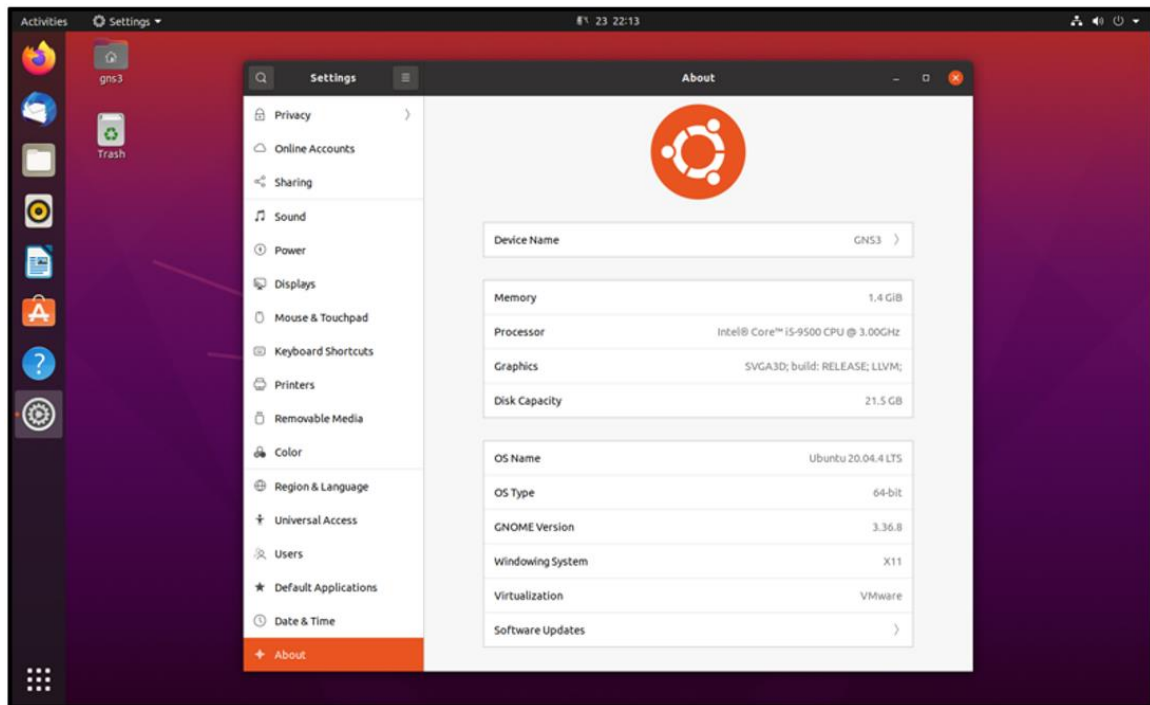


Figure 31 - Ubuntu Server (Version 20.04.4 LTS)

6.2.4 OpenFlowManager

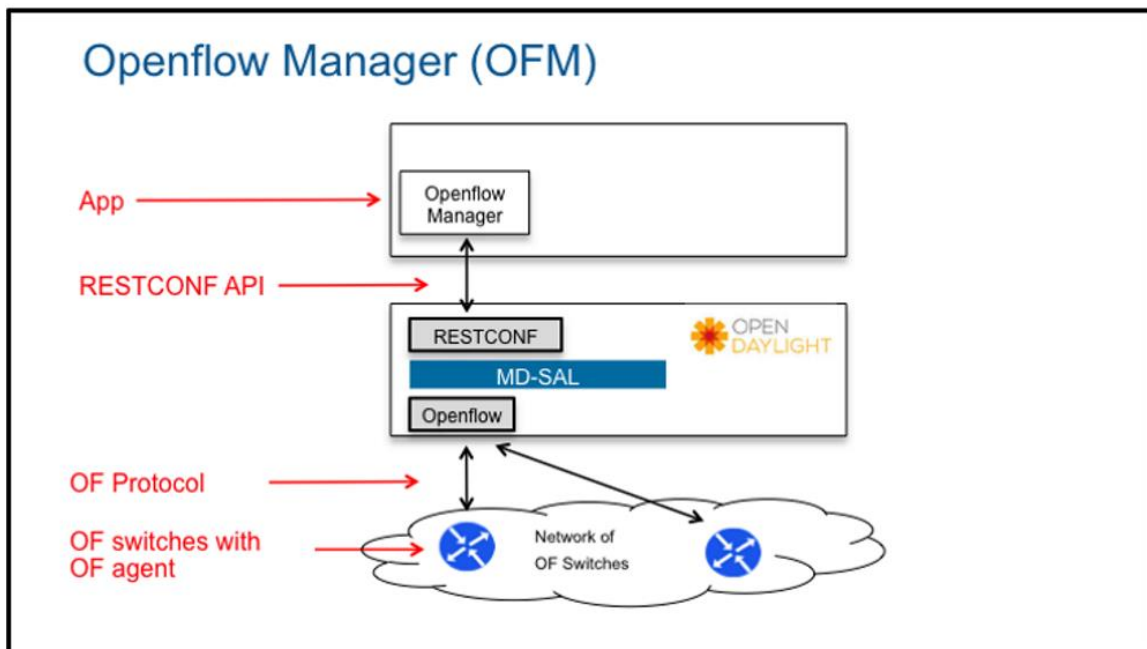


Figure 32 - OpenFlow Manager

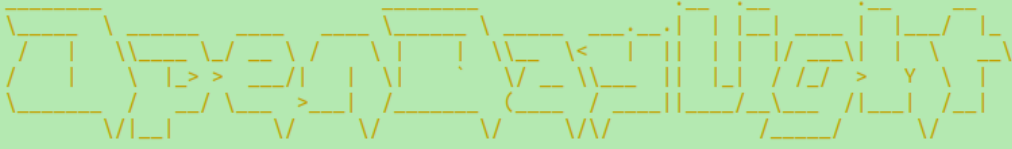
OpenFlow Manager (OFM) is an application for visualizing OpenFlow topologies, programming OpenFlow pathways, and gathering OpenFlow statistics that runs on top of ODL. SDN refers to an application's interaction with a network in order to simplify operations or allow a service. A controller is located between the application and the network and communicates with it. A variety of protocols are used to communicate with devices such as switches in the southbound direction. It presents a network abstraction in the northbound direction by employing generally used REST APIs. The application's controller vehicle is ODL. The OpenFlow Manager (OFM) is a software that manages OpenFlow networks using this concept (chrismetz09 & CiscoDevNet, 2017).

9.3 SDN Simulation

This simulation was carried out by using two virtual machines installed on VMware Workstation Pro. Ubuntu was used to install ODL controllers and Mininet was used for the infrastructure layer or the switching layer.

After having installed all the requirements, start ODL from the Ubuntu terminal.

```
gns3@GNS3:~/Downloads/odl/bin$ ./karaf
OpenJDK 64-Bit Server VM warning: ignoring option MaxPermSize=512m; support was removed in 8.0



Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.
```

Figure 33 - Starting ODL in Ubuntu

There are two options to create a topology; by using the command line from Mininet or by using the Mininet GUI. In this document, the topology was created using a Mininet GUI called Miniedit, which is basically a Python file. Use Secure Shell to connect to the Mininet from Ubuntu and run the command.


```

mininet@mininet-vm:~$ sudo python ./mininet/examples/miniedit.py
topo=None
Open vSwitch version is 2.5.9
New Prefs = {'ipBase': '10.0.0.0/8', 'sflow': {'sflowPolling': '30', 'sflowSampling': '400', 'sflow
Header': '128', 'sflowTarget': ''}, 'terminalType': 'xterm', 'startCLI': '1', 'switchType': 'ovs',
'netflow': {'nflowAddId': '0', 'nflowTarget': '', 'nflowTimeout': '600'}, 'dpctl': '', 'openFlowVer
sions': {'ovsOf11': '0', 'ovsOf10': '0', 'ovsOf13': '1', 'ovsOf12': '0'}}
Getting Hosts and Switches.
Getting controller selection:ref
Getting Links.
*** Configuring hosts
h5 h6 h3 h2 h1 h4
**** Starting 1 controllers
c0
**** Starting 4 switches
s2 s3 s4 s1
No NetFlow targets specified.
No sFlow targets specified.

NOTE: PLEASE REMEMBER TO EXIT THE CLI BEFORE YOU PRESS THE STOP BUTTON. Not exiting will prevent M
inEdit from quitting and will prevent you from starting the network again during this session.

*** Starting CLI:
mininet>

```

Figure 34 - Opening Mininet editor

On the left panel of the Mininet GUI are various tools like a pointer, hosts or end users, OpenFlow enabled switches, legacy routers and switches, and also a controller. The default controller is the POX controller, and it can be changed to an OpenFlow controller from the edit option by selecting the OpenFlow version 1.3.

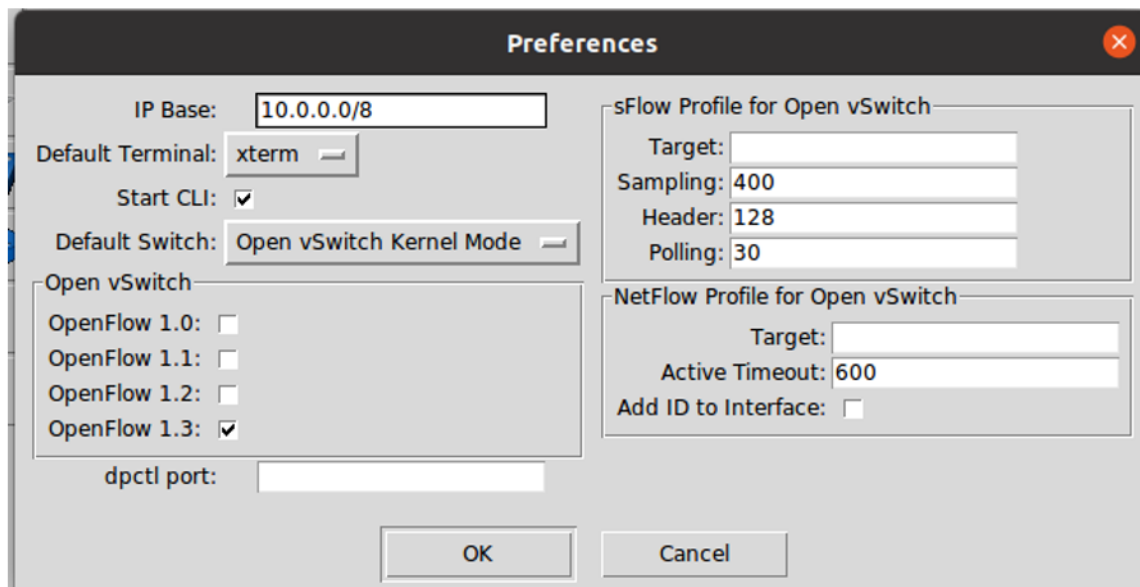


Figure 35 - Changing the Controller to OpenFlow

The topology in Miniedit can be created by dragging and dropping options like GNS3 tools. The open vSwitches or the underlying networking devices will be running on the Mininet. There needs to be a connection between the devices running on the Mininet and the controller. The following command can be used for the connection purpose:

```
mininet@mininet-vm:~$ sudo ovs-vsctl set-controller s1 tcp:192.168.85.130:6633
mininet@mininet-vm:~$ sudo ovs-vsctl set-controller s2 tcp:192.168.85.130:6633
mininet@mininet-vm:~$ sudo ovs-vsctl set-controller s3 tcp:192.168.85.130:6633
mininet@mininet-vm:~$ sudo ovs-vsctl set-controller s4 tcp:192.168.85.130:6633
mininet@mininet-vm:~$
```

Figure 36 - Connecting Switches to ODL

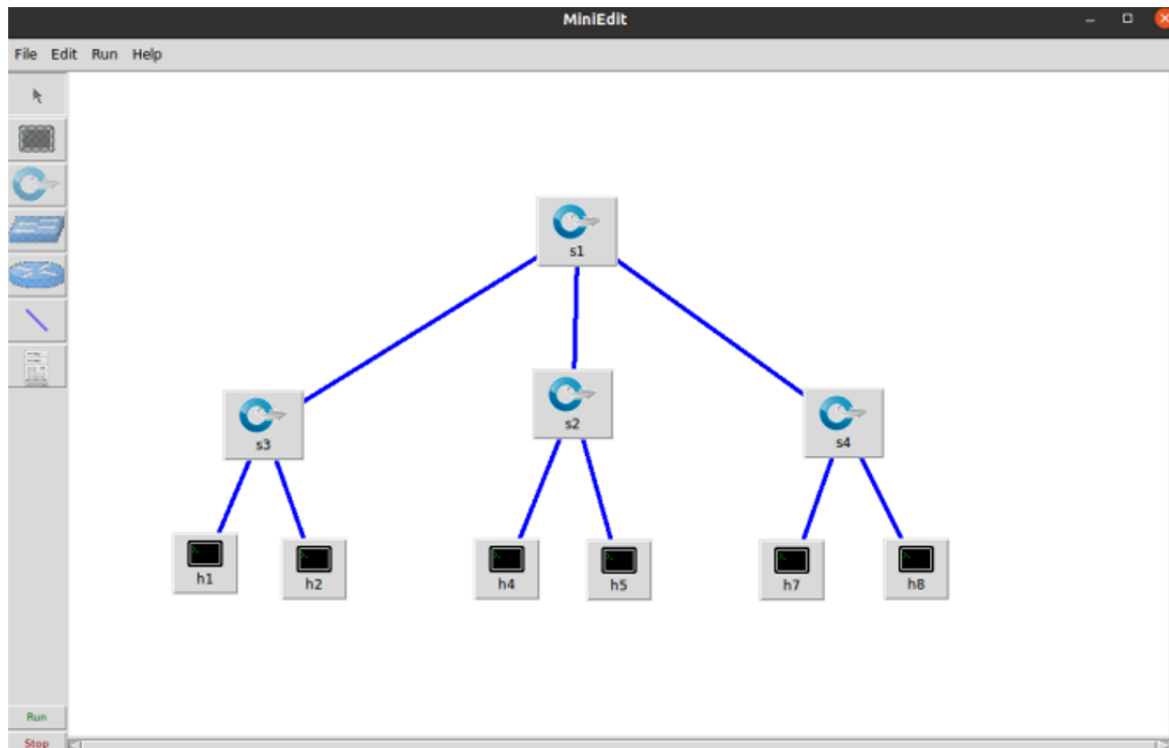
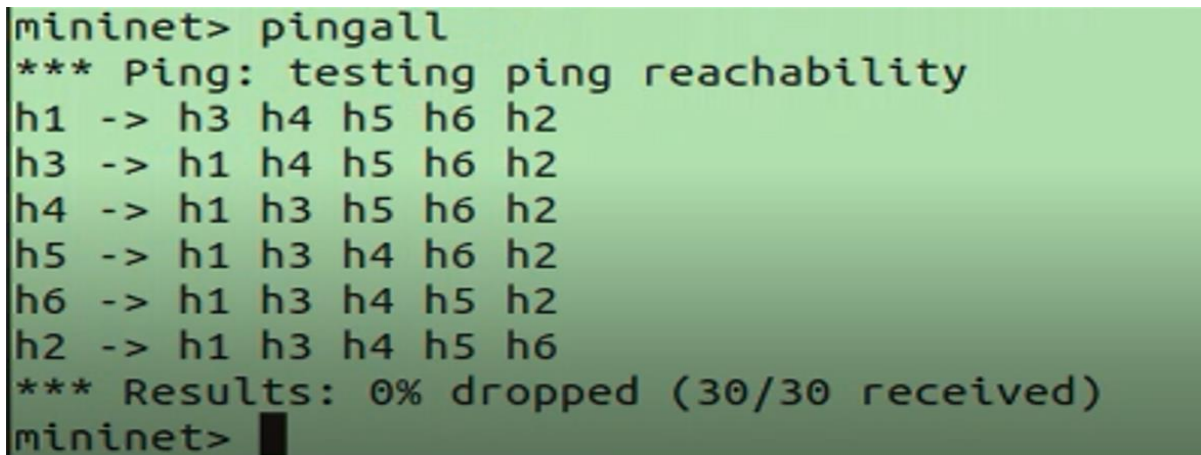


Figure 37 - College Network Simulation using the ODL Controller in Miniedit

A controller coupled to a collection of switches forming a tree topology is used to replicate the design in an SDN architecture. The controller is the network orchestrator and supervises all switches connected to it, hence the first level is dedicated to it, but it is not necessary to use the controller in the topology since, the underlying devices needs to be connected to the controller via Mininet. The controller is regarded as the network's nerve center.

The second level is made up of one switch that are linked to the access switches. These switches serve as the network's distribution layer, ensuring the network's redundancy and availability. The last level includes three switches that connect two host machines. This level is called the Access Level.

The connection testing can be done for the devices booted up with the 'pingall' command.

A terminal window with a green background showing the output of the 'pingall' command in mininet. The output lists connections between hosts h1 through h6 and shows that all 30 ping attempts were successful.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h3 h4 h5 h6 h2
h3 -> h1 h4 h5 h6 h2
h4 -> h1 h3 h5 h6 h2
h5 -> h1 h3 h4 h6 h2
h6 -> h1 h3 h4 h5 h2
h2 -> h1 h3 h4 h5 h6
*** Results: 0% dropped (30/30 received)
mininet>
```

Figure 38 - Connection Testing

The ODL controller orchestrates the SDN controller. It connects directly to all of the network's switches. The Open vSwitch specification is used to implement the switches. This program is intended to provide complete support for the OpenFlow protocol, which will be used by all SDNs for routing.

After that connection, the topology can also be viewed from the ODL DLUX which is the GUI version of ODL running on port number 8181.

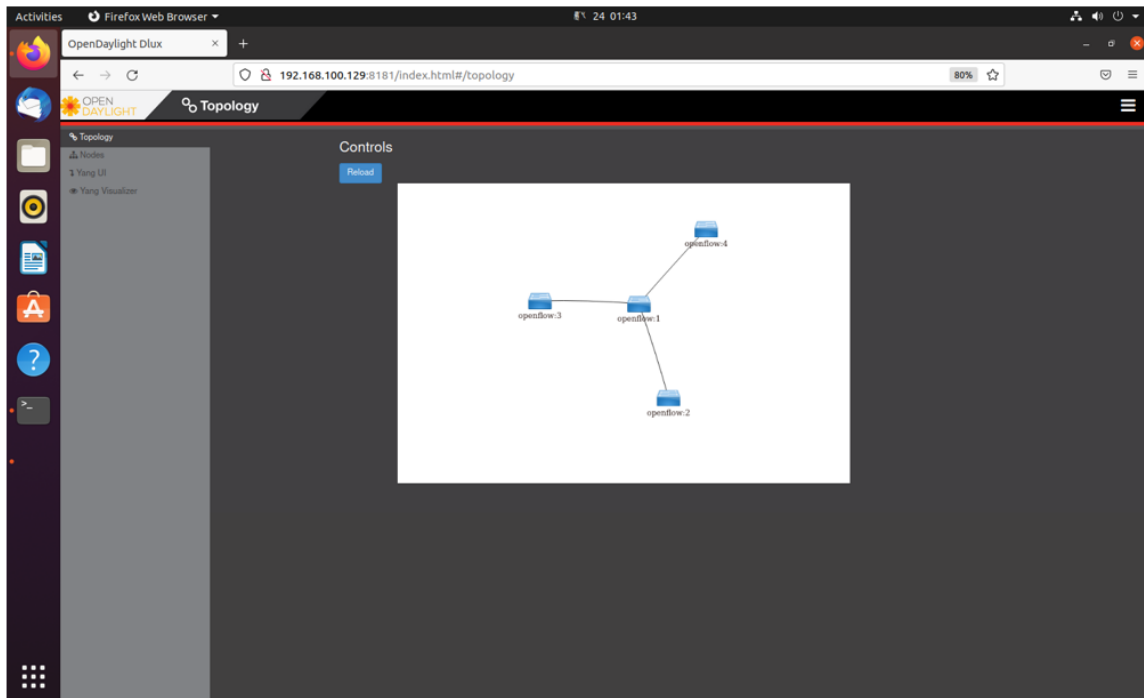


Figure 39 - ODL DLUX

The network administrator can use both the ODL and the OpenFlow manager applications to view the network topology. However, the OpenFlow Manager provides a better environment for the administrator to perform necessary configurations and updates from the centralized controller.

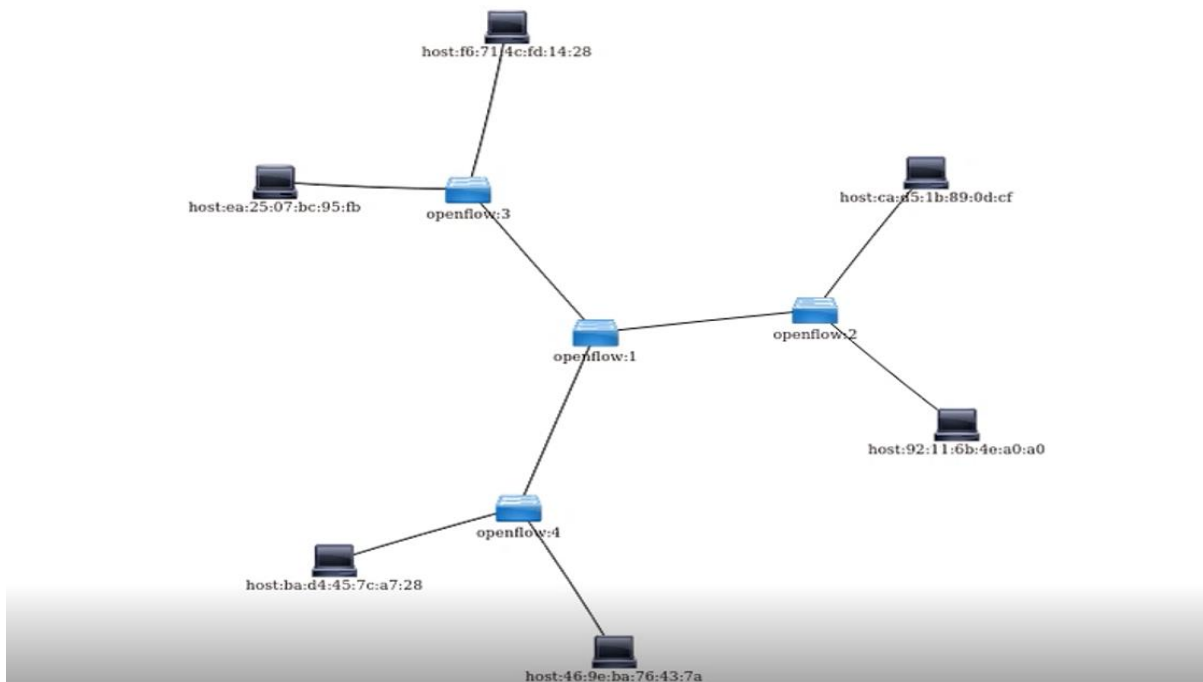


Figure 40 - Network topology view from OpenFlow Manager

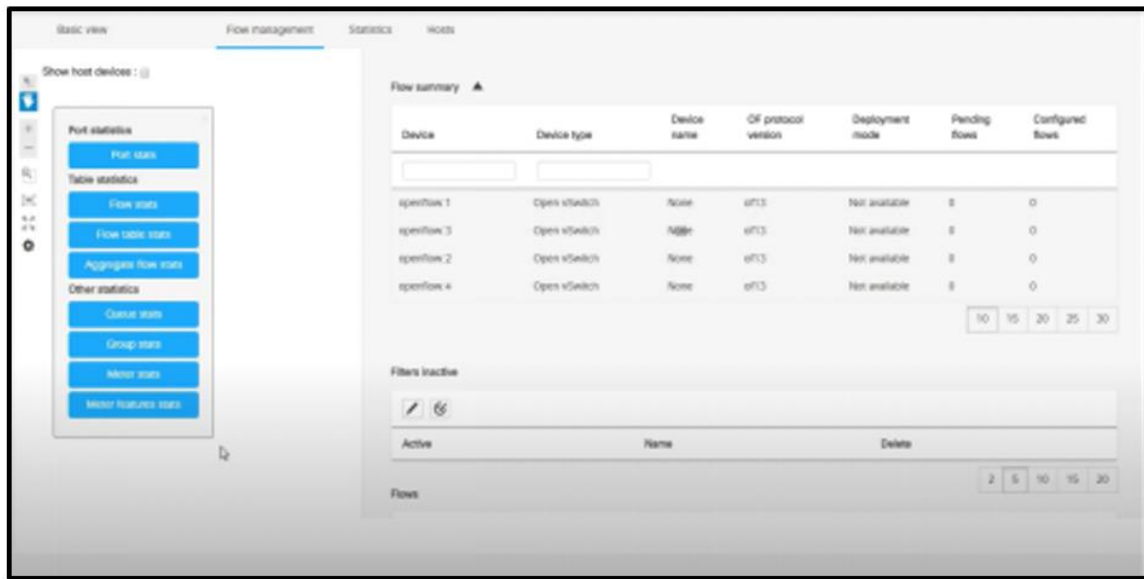


Figure 41 - Flow Management

The SDN controller will interface to the Open vSwitch using the OpenFlow protocol, and it will configure the flow table within the remote switches using standard off-the-shelf silicon or merchant silicon. Flow table of the switches stores flow entries or rules, and packets are matched depending on the priority of matching flow entries. Those flow entries can be added and configured with the OpenFlow manager running in the application layer. The packet forwarding can be done based on their source and destination Mac or IP addresses, which determines what actions to take on specific TCP ports and also maintains statistics. One function could be a switch or route based on a mac address considering the flow table below.

Table 8- Sample Flow table

MAC		IP		TCP	ACTION	COUNT
Src	Dest	Src	Dest			
W	A1B2C3D4E5	W	W	W	Port 1	2800
W	W	W	5.6.7.8	W	Port 2	3000
W	W	W	W	25	Drop	1500

The OpenFlow Manager is used in the application layer, which makes use of the NorthBound API to interact with the controller in order to access the controller's status. The controller modifies the flow tables of networking devices using the SouthBound API, which are typically switches, but could also be routers, load balancers, firewalls, and other infrastructure layer devices.

Within the flow table there will be many rules and based on those rules, certain actions would take place. Here, when one has a particular destination address, one would want to forward the packet out into port 1. Keeping count is also possible by going through the port and feeding this up to the central controller.

This choice may be made based on the IP address. A more nuanced judgment can be made using the destination address and potentially the source Mac address to decide what compromises a flow and, depending on that, measures can be taken, such as which port to forward that flow onto.

It is also possible to have functionalities such as a firewall, and can instruct the flow table. For instance, if one sees TCP port number 25, then drop the packet. Likewise, some other functionalities are also possible. Lastly, if there are no rules for certain packet flow, the decision to be made will be done by the controller and it will select the best route to forward the packet in the network.

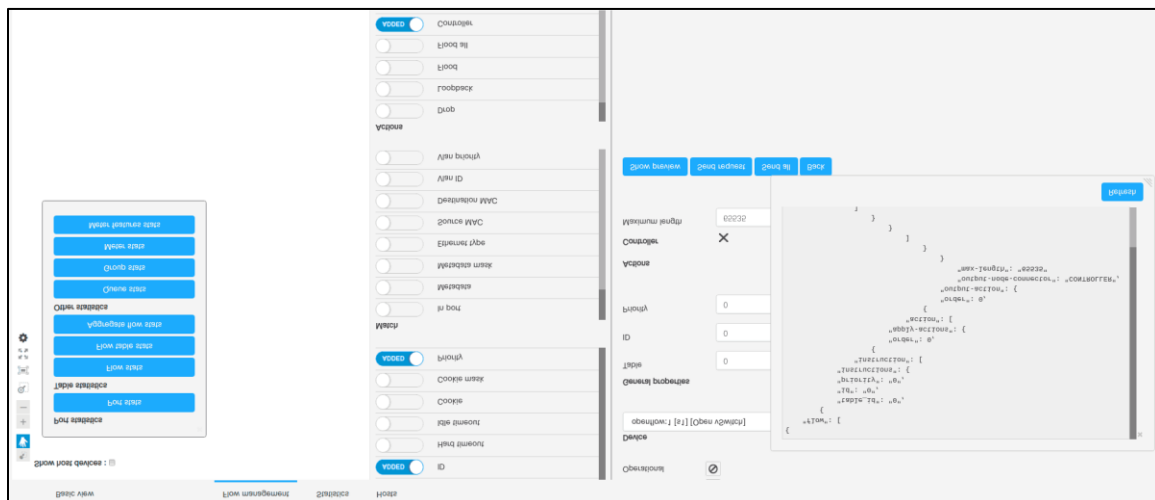


Figure 42 - Modifying the flow entries

Be it in any network, networking devices like routers and switches are used to forward the packets based on IP address and Mac address respectively, and firewalls to block certain traffic flows. For the traditional networking, the devices are required to be configured for forwarding the packets based on certain protocols whereas in the case of software defined networking, there is a centralized controller which configures the flow table containing rules dynamically

or statically of the open vSwitch. The rules that are written in the flow table can determine the functionality of the devices. If the packet is forwarded based on the IP address, the open vSwitch is working as router. In addition to a centralized control, the scalability of the network will also be ensured. Thus, compared to traditional architecture, SDN architecture has lots of advantage.

CHAPTER SEVEN: CHALLENGES

Since SDN is a relatively new concept not only locally but globally as well, and there were certain challenges that the team faced while conducting this research.

The main challenge faced during the entire period of this project was during the simulation of the existing network topology of the college using GNS3. GNS3 required IOS images of the physical devices to be used in the simulation. In CST, the only SDN supported device was Allied Telesis. The IOS images of Allied Telesis devices were obtained from the college ICT officer initially. However, those IOS images were not usable in GNS3. Looking for the images from internet sources such as Docker Hub was also not useful since the images available there were running only on bare metal and not on virtual machines or containers (servers).

As one of the alternative means to this problem, the solution that was proposed was to use a hybrid mode of simulation whereby half of the simulation was to be conducted in the virtual environment using the GNS3 network simulator and half of the simulation was to be conducted by connecting the physical devices to the virtual environment. However, this alternative also proved to be a failure for the reasons stated below.



Figure 43 - Attempt to get the GNS3 virtual environment and physical device to communicate

There was literally no one who had hands-on experience in working on GNS3 network simulator or SDN. While speaking with ICT, technicians, the project coordinator, and even technicians who came to maintain the door access of the laboratory at the time, tried to help with the hybrid mode of simulation. However, none of these efforts were successful.

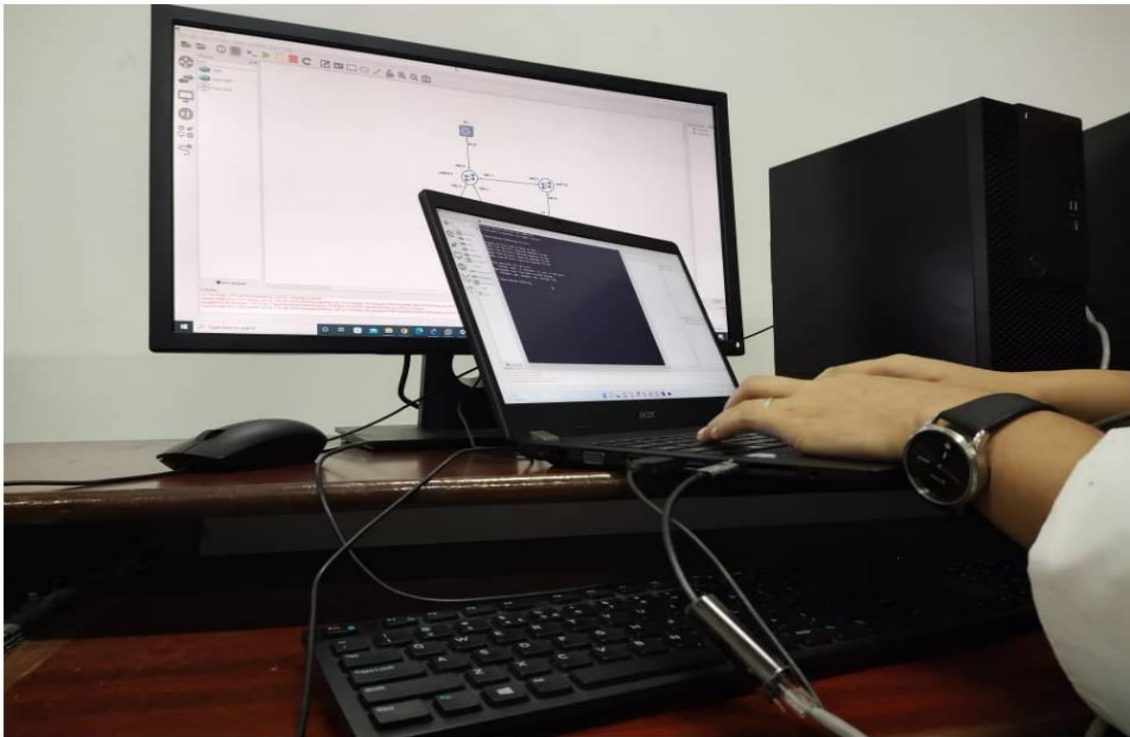


Figure 44 - Using Ethernet adapter cable to connect the virtual environment with the physical device

While communicating between the physical and virtual environments was difficult, another option was to use USB to bridge the network via the cloud. However, establishing the connection between the two environments was not successful.

In addition to the above challenges, there were also some challenges that were faced throughout the simulation in general. The following are some of those challenges:

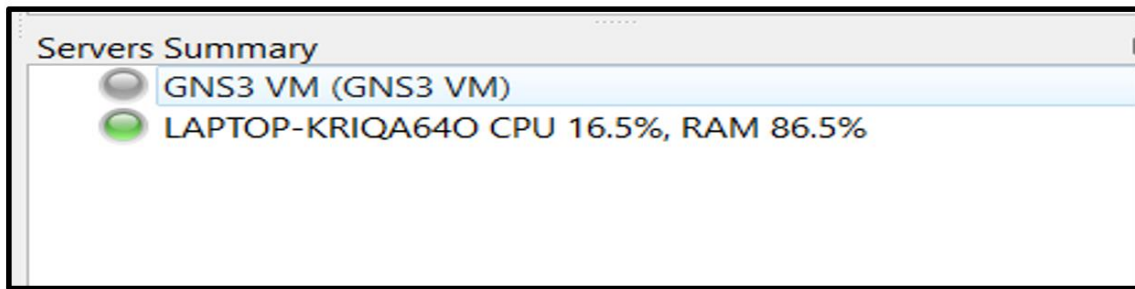


Figure 45 - VM Server Summary

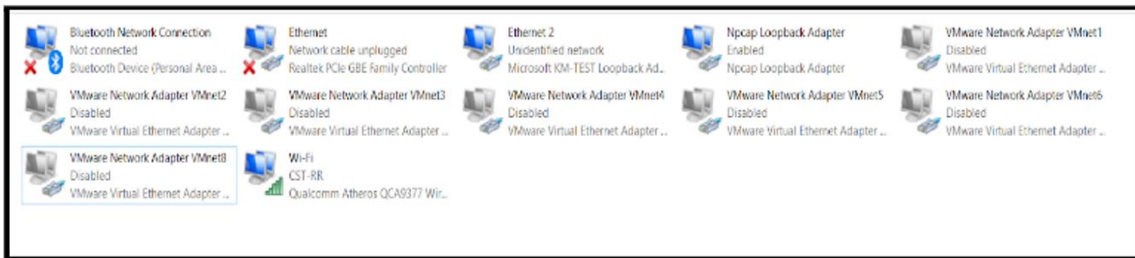


Figure 46 - VMware Network adaptor inside local PC

When the environment was ready to be used, it occasionally returned to square one. Even if the environment was functional, the GNS3 virtual machine did not start; in order to do so, one must restart VMware's network adapter.

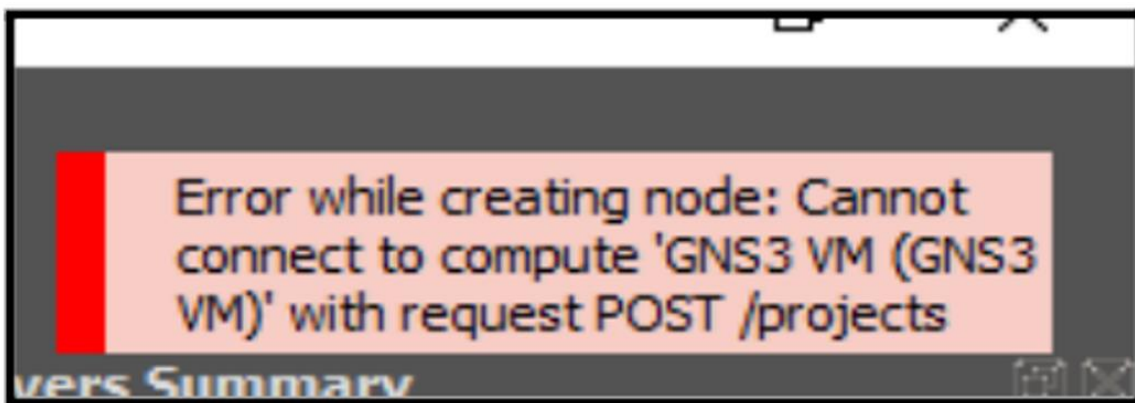


Figure 47 - Error in GNS3 VM

The hybrid mode had no suitable references. It took a long time to load and display problems like the ones as shown below. Restarting the GNS3 solved the problem.

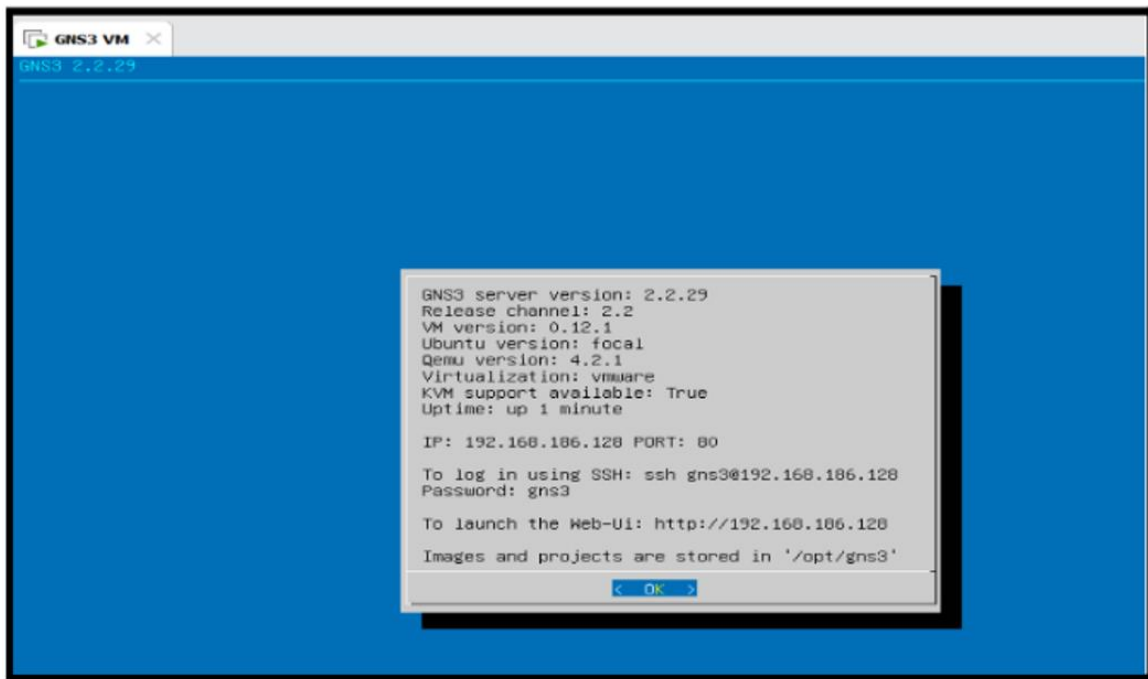


Figure 48 - VMware Workstation Pro

Overall, dedicated computing resources with more than 8GB of RAM, 100GB of ROM, and a powerful processor are required for normal application functionality. Limitation of a good specification computer was one of the constant challenges that was faced.

CHAPTER EIGHT: RECOMMENDATION

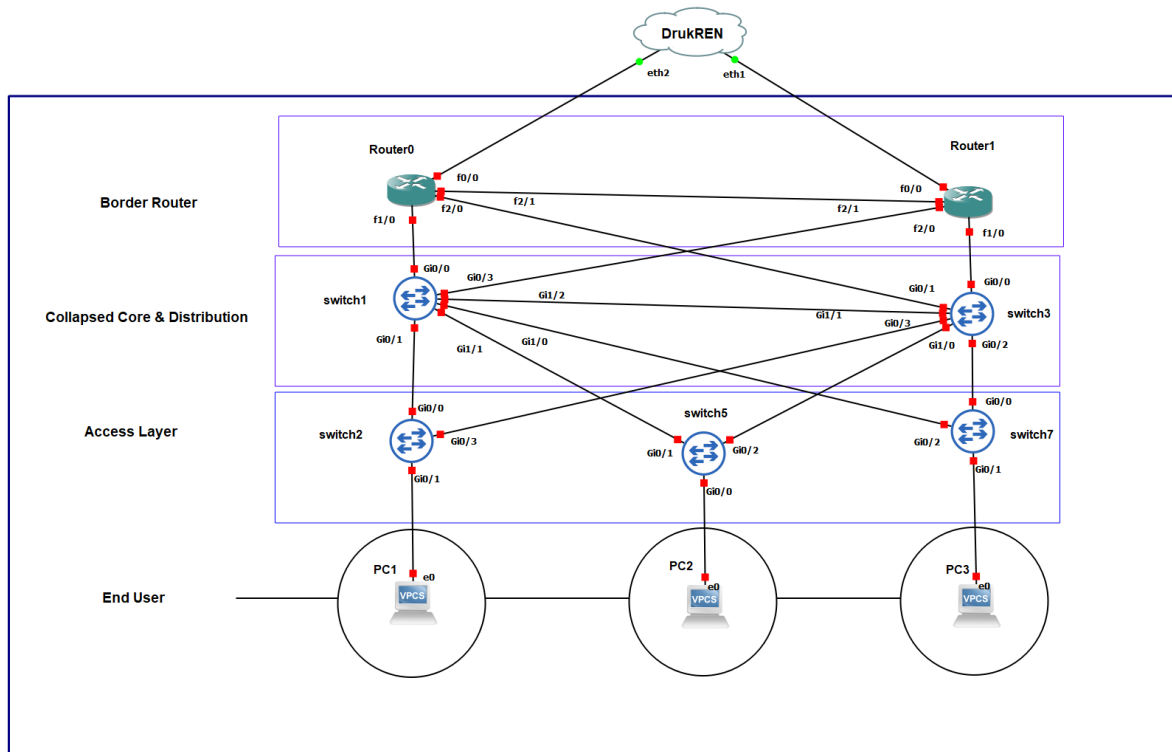


Figure 49 - Recommended topology as an alternative to the existing network topology

As a recommendation to the existing college network, a two-tier architecture or so called the collapsed core architecture where the core and distribution layers are merged as one to the existing three tier architecture is proposed. With the two-tier architecture, not only the cost of the networking devices is reduced but also the benefit provided by the three tiers can be reaped within the two tiers. CST, being a small campus area network and considering its less tendency towards growing in size as compared to large enterprises, makes it more suitable for CST to use two tier architecture in terms of cost as well as benefit. Additionally, this recommended topology provides a redundancy for the border router and the core-distribution layer which serves as a backup whereby if one point fails, the other one will still function.

CHAPTER NINE: CONCLUSION

SDN is one of the trending technologies in the field of networking. The rise of SDN has dominated the traditional form of networking ever since its inception in 2011. The same case is happening in the field of networking across the organizations and institutes in Bhutan as well. CST is currently halfway through its migration from traditional forms of networking towards the SDN, with 50% of the networking devices being SDN supported. However, the majority of the organizations in Bhutan are found to be reluctant to depart from the legacy, mainly due to the cost that would be incurred while migrating towards SDN.

For the reasons stated above, this research aimed to analyze and study the implementation of SDN in CST and check its feasibility, so as to recommend the implementation of SDN throughout the nation. This analysis was performed through two phases of simulations of the existing network topology using the GNS3 network simulator and then by using ODL, which is one of the SDN controllers.

The simulations of the existing network topology of the college helped gain a better insight towards learning the bottleneck of the legacy networking system, which was mainly to do with network congestion, automation, and programmability. From the analysis, the ultimate conclusion drawn was that it is absolutely feasible for CST or any other organization to migrate the network towards SDN in terms of cost efficiency as well as in terms of functionality as a whole.

REFERENCES

- Allied Telesis. (2022, January 21). *Switches*. Retrieved February 25, 2022, from <https://www.alliedtelesis.com/bt/en/products/switches>
- Andrea, H. (2020, November 17). *18 Network Simulation Software Tools for Certification Practice or Research*. Network Training. Retrieved February 23, 2022, from <https://www.networkstraining.com/network-simulation-software-tools/>
- Amiri E., Alizadeh E., & Rezvani M. (2020). *Controller Selection in Software Defined Networks using Best-worst Multi-criteria Decision-Making*. Bulletin of Electrical Engineering and Informatics, 9 (4), 1506-1517. <https://doi.org/10.1159/eei.v9i4.2393>
- Augustine. A.,(2017), *A Comparison of Network Simulators for Wireless Networks*. International Journal of Advanced Research in Electrical, Electronics, and Instrumentation Engineering, 6 (3). <https://10.15662/IJAREEIE.2017.0603001>
- Christmec. C. (2016, August 16). *GitHub - CiscoDevNet/OpenDaylight-Openflow-App*. GitHub. Retrieved on May 23, 2022, from: <https://github.com/CiscoDevNet/OpenDaylight-Openflow-App>
- Christodouloupoulos, J., Paraskevas, M., & Triantafyllou, V. (2016, November). *Software Defined Networks: A case for QoS implementation at the Greek School Network*. In Proceedings of the 20th Pan-Hellenic Conference on Informatics (pp. 1-6).
- Cisco. (2022, February 14). *Cisco-Networking, Cloud, and Cybersecurity Solutions*. Retrieved February 26, 2022, from <https://www.cisco.com>
- Cisco NetAcademy. (2022, January 27). *Packet Tracer FAQs*. Networking Academy. Retrieved on February 24, 2022, from <https://www.netacad.com/courses/packet-tracer/faq>

- D-Link. (n.d.). *D-Link Switch*. Retrieved February 25, 2022, from <https://us.dlink.com/en/business/switches>
- Eltaj, M., & M. Hassan, A. H. (2020). *Performance Evaluation of SDN Controllers: FloodLight, POX and NOX*. International Journal of Engineering and Applied Sciences (IJEAS), 7(8). <https://doi.org/10.31873/ijeas.7.08.01>
- English, J. (2018, December 17). *What is an SDN controller (software-defined networking controller)? - definition from whatis.com*. SearchNetworking. Retrieved February 23, 2022, from: <https://www.techtarget.com/searchnetworking/definition/SDN-controller-software-defined-networking-controller>
- Fokus, F. (2019, May 5). *Network Simulator OMNeT++*. Eclipse MOSAIC – A Multi-Domain and Multi-Scale Simulation Framework for Connected and Automated Mobility. Retrieved on February 24, 2022, from: https://www.eclipse.org/mosaic/docs/simulators/network_simulator_omnetpp/
- Goldstein. A., (2021, July 29). *Open Source Licenses Explained*. WhiteSource. Retrieved on 24/02/2022 from: <https://www.whitesourcesoftware.com/resources/blog/open-source-licenses-explained>
- Gupta, S. G., Ghonge, M. M., Thakare, P. D., & Jawandhiya, P. M. (2013, 04 04). *An overview of open-source network simulation tools*. Retrieved on March 3, 2022, from: <http://ijarcet.org/wp-content/uploads/IJARCET-VOL-2-ISSUE-4-1629-1635.pdf>
- Hyu, T. (2018). *Fiber Optic Solutions*. Retrieved on 2022, from: <https://www.fiber-optic-solutions.com/access-switch-need.html>
- Issariyakul, T. (2011, October 12). *An Introduction to Network Simulator 2 (NS2)*. SpringerLink. Retrieved February 24, 2022, from [https://link.springer.com/chapter/10.1007/978-1-4614-1406-3_2?noAccess=true&error=cookies_not_supported & code=87a75769-561d-4e68-b9b5-fc648c01ae27](https://link.springer.com/chapter/10.1007/978-1-4614-1406-3_2?noAccess=true&error=cookies_not_supported&code=87a75769-561d-4e68-b9b5-fc648c01ae27)

JavaTPoint. (2022, January 1). *Programming language. What is Programming Language - Javatpoint*. Retrieved on May 10, 2022, from: <https://www.javatpoint.com/programming-language>

Khan, A. R., Bilal, S. M., & Othman, M. (2012). *A performance comparison of open source network simulators for wireless networks*. 2012 IEEE International Conference on Control Systems, Computing, and Engineering, 34–38. <https://doi.org/10.1109/iccsce.2012.6487111>

Lessmann, J., Janacik, P., Lachev, L., & Orfanus, D. (2008). *A Comparative Study of Wireless Network Simulators*. Seventh International Conference on Networking (Icn 2008), 517–523. <https://doi.org/10.1109/icn.2008.97>

Linkletter, B. (2015, July 9). *Using the POX SDN controller*. Open-Source Routing and Network Simulation. Retrieved February 27, 2022, from <https://www.brianlinkletter.com/2015/04/using-the-pox-sdn-controller/>

Lutkevich, B., & Lebeaux, R. (2021, October 14). *software license*. SearchCIO. Retrieved February 24, 2022, from <https://www.techtarget.com/searchcio/definition/software-license>

Mahmood, A., Saleem, M. F., & Latif, A. (2013, 09 09). *Key Features and Optimal Performance of (Academia, Compiler)* Retrieved on March 3, 2022, from: https://www.academia.edu/4861597/Key_Features_and_Optimum_Performance_of_Network_Simulators_A_Brief_Study

Neto, B. F. J. V. (2021, February 2). *SDN Controllers - A Comparative approach to Market Trends*. Archive ouverte HAL. Retrieved October 16, 2021, from <https://hal.archives-ouvertes.fr/hal-03133692>

OpenNetworkingFoundation. (2022, February 23). *Open Network Operating System (ONOS) SDN Controller for SDN/NFV Solutions*. Retrieved February 27, 2022, from <https://opennetworking.org/onos/>

- OpenDaylight. (2021, March 9). *Platform Overview*. Retrieved on May 24, 2022, from: <https://www.opendaylight.org/about>
- OpenDayLight. (2021, September 20). *Welcome to OpenDaylight Documentation. OpenDaylight Documentation Phosphorus documentation*. Retrieved February 27, 2022, from <https://docs.opendaylight.org/en/stable-phosphorus/>
- Pakzad, F. (2022, February 4). *A Comparison of Software Defined Networking (SDN) Controllers. Part 3: OpenDayLight (ODL)*. Aptira. Retrieved February 25, 2022, from <https://aptira.com/comparison-of-software-defined-networking-sdn-controllers-part-3-openshift-odl/>
- PriceSpy. (2021, 1 january). *PriceSpy Ltd. PriceSpy UK*. Retrieved on 9 february 2022, from: <https://pricespy.co.uk/>
- Rao, S. (2021, December 2). *SDN Series Part Five: Floodlight, an OpenFlow Controller*. The New Stack. Retrieved February 27, 2022, from <https://thenewstack.io/sdn-series-part-v-floodlight/>
- Reddy, N. D. P., Sivakumar.B (2018, April 24). *Implementing Software - Defined Networking (SDN) in a Campus Environment*. IJERT. International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181. Vol 3 (18). P 1
- Rosencrance, L., English, J., & Burke, J. (n.d.). *Software-defined networking (SDN)*. Nemertes Research. Retrieved 2022, from <https://www.techtarget.com/searchnetworking/definition/software-defined-networking-SDN>
- Rowshanrad S., Abdi V., & Keshtgari M.(2016). *Performance Evaluation of SDN Controllers:FloodLight and OpenDayLight*. IIUM Engineering Journal,17 (2),47-57. <https://doi.org/10.31436/iiumej.v17i2.615>
- Ryu SDN Framework. (2017, October 1). *Ryu SDN Framework. WHAT'S RYU?* Retrieved February 27, 2022, from <https://ryu-sdn.org/>

- Sari, L.M., Hatta, P., Wihidayat, E.S., & Xiao, F. (2018). *A Comparison between the Use of Cisco Packet Tracer and Graphical Network Simulator 3 as Learning Media on Students' Achievement*. Jurnal Pendidikan Teknologi dan Kejuruan, 24 (1), 132-136. <https://10.21831/jptk.v24i1.16042>
- Shanmugam, J., & Ramya, S. T. (2018). *Software Defined Networking: A Paradigm Shift in Networking for Future, Emerging Trends and Applications*. International Journal of Applied Engineering Research, 13(18), 13475-13481.
- Stancu A.L., Halunga S., Vulpe A., Suciu G., Fratu O., & Popovici E.C.(2015) *A Comparison between Several Software Defined Networking Controllers*. 2015 12th International Conference on Telecommunication in Modern Satellite, Cable and Broadcasting Services (TELSIKS). Published. <https://doi.org/10.1109/telsks.2015.7357774>
- Stoltzfus, J., (2021, May 28). *Graphical User Interface (GUI)*. *Techopedia.Com*. Retrieved on May 10, 2022, from: <https://www.techopedia.com/definition/5435/graphical-user-interface-gui>
- Visaac. (April, 22, 2020). *VMware workstation 15.5.2 pro release notes*. VMware Workstation 15.5.2 Pro Release Notes. Retrieved on May 24, 2022, from: <https://docs.vmware.com/en/VMware-Workstation-Pro/15.5/rn/VMware-Workstation-1552-Pro-Release-Notes.html>
- VmWare.(2022, May 5). What is Software-Defined Networking (SDN)?.Retrieved on May 10 from: <https://www.vmware.com/topics/glossary/content/software-defined-networking.html>
- Wallen, J., Staff, T. R., Wolber, A., Whitney, L., Pernet, C., Alexander, M., & Combs, V. (2020, December 11). *Ubuntu server: A cheat sheet*. TechRepublic. Retrieved on May 24, 2022, from: <https://tek.io/3wFpQzR>

- Weingartner, E., vom Lehn, H., & Wehrle, K. (2009). *A Performance Comparison of Recent Network Simulators*. 2009 IEEE International Conference on Communications, 1–5. <https://doi.org/10.1109/icc.2009.5198657>
- Xia, W., Wen, Y., Foh, C. H., Niyato, D., & Xie, H. (2015). *A Survey on Software-Defined Networking*. *IEEE Communications Surveys & Tutorials*, 17(1), 27–51. <https://doi.org/10.1109/comst.2014.2330903>
- Zhu, L., Karim, M. M., Sharif, K., Xu, C., Li, F., Du, X., & Guizani, M. (2021). SDN Controllers. *ACM Computing Surveys*, 53(6), 1–40. <https://doi.org/10.1145/3421764>