

UNIVERSITY OF DAR ES SALAAM
COLLEGE OF INFORMATION AND COMMUNICATION TECHNOLOGIES
(CoICT)
DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATIONS
ENGINEERING (ETE)



TE 499: FINAL YEAR PROJECT REPORT

**IP NETWORKS QoS OPTIMIZATION USING ML POWERED SDN
THE CASE STUDY OF UDSM LAN**

A Project Report in Partial Fulfillment of the Requirements
for the Award of Bachelor of Science in
Telecommunications Engineering

Student Name: SAMSON, Emmanuel Julius
Registration Number: 2016-04-01802
Supervisor's Name: Dr. Prosper Mafole
Supervisor's Signature:
Submission Date: August 5, 2020

Declaration of Authorship and Originality

I, **SAMSON, Emmanuel Julius**, with registration number **2016-04-01802**, declare that this report and the work described in it are my own work, with any contributions from others expressly acknowledged and/or cited.

I declare that the work in this report was carried out in accordance with the regulations of the university of Dar es Salaam and has not been presented to any other University for examination either in Tanzania or Overseas. Any views expressed in this report are those of the author and in no way represent those of the University of Dar es Salaam.

SIGNED:

DATE: August 5, 2020

This report may proceed for submission for assessment for the award of B.Sc. in Telecommunications Engineering at the University of Dar es Salaam.

Supervisor's Signature:

Date:

Abstract

The fourth industrial revolution has brought about invaluable developments both in the way the information is shared within localities and around the world. The internet is the tool that is powering this revolution. The academic institutions have not been left out in adopting the uses of the internet and developing necessary algorithms for enhancing resources sharing around the globe.

How fast the information is shared remains one of the very important aspects of the computer networks that meet the expectations of the users and the demands of the time. Considering the LAN at UDSM (the case study of this work), Quality of Service (QoS) is a varying quantity depending on the place (college, school, or institution) one is at, hence the variations in the Quality of Experience (QoE) too. This is an important implication in a way that while other parts of network might be facing congestion problems, other parts of the same network might have some unutilized bandwidth.

The QoS of the LAN is assessed in three different aspects; the delays and delay jitters with which the information shared experiences, and the throughput of such data sharing. Considering that it is not the focus of this work to get down to the way the information travels through wires, the delay and the delay jitters were not considered. Hence, it was the focus of this work to devise and model the way the bandwidth—in shared environments like that of the UDSM—can be used to a greater advantage (increased throughput). To add on, it was not in the focus of this work to consider the firewall rules impacts in the network despite their relatively necessity in delays associated with the decision making process for routing packets.

The report herein introduces the problem, discusses different literature and findings that power the novelty and necessity for this undertaking, and highlights the methodology that was employed in order to accomplish the objectives of this work in chapters 1, 2 and 3 respectively. It then goes on to discuss the design considerations, and data analysis in chapters 4 and 5. The conclusions and recommendations are drawn in chapter 6.

Acknowledgement

First and foremost, the life we have and the moments we are sharing are simply the gift from the Almighty God without whom this work would have never been possible to complete. It is, therefore, with gratitude that I thank him for all the grace, energy and purpose he has granted me with.

My family has been so much supportive during my Academic journey. Now that this has come to an end, I take this opportunity to thank them for their involvement and support, inclusiveness and compassion and at most times unceasing encouragement during my stay at the University. Special thanks to my Mom, Ms. Adelina Kajubu, and my brother, Dr. Charles Nyaitara.

The Vi Team I have been around with from the First year has been very helping. Sincerely, it is due to the long nights we had together, short days we shared and unwavering support we freely granted to one another that motivated me for this work at first. The challenges that we always posed to one another and silent geeks within everyone in the team are the ones that not only made this work possible but also other works of the like. The future holds great things for this team.

Dr. Prosper Mafole has been one of the best teachers I have ever had. Apart from being a teacher, he has been able to act as a Mentor and an Advisor without crossing any of the lines whenever the situations changed. He has been an inspiration and a great living legacy. His commitment, and hard work would never have left anyone working with him relaxed. Therefore, it is with great respect that I honor him. This work should be one of those he can stand up there with and say, "The DIY principle isn't just words. It can be done; it has been done; and it will be done by anyone who puts their heart to it."

Lastly, I appreciate everyone who participated in this work either directly or indirectly. Their comments, challenges, corrections and every other piece of information provided have been invaluable. Special appreciation to the community around CoICT for the brains they put into use are one of the greatest assets and one that they do not just let rest and settle for small achievements.

Table of Contents

Declaration of Authorship and Originality	i
Abstract	ii
Acknowledgement	iii
Table of Contents	iii
List of Acronyms and Abbreviations	vi
List of Tables	vii
List of Figures	viii
1 Introduction	1
1.1 Introduction	1
1.2 Challenges in Bandwidth Management	2
1.3 How Challenges in Bandwidth Management are Tackled	2
1.4 Statement of the Problem	3
1.5 Motivation	3
1.6 Research Objectives	4
2 Literature Review	5
2.1 Definition of Terms	5
2.2 Current situation	8
2.3 Related Works and Systems	12
3 Methodology	14
3.1 Introduction	14
3.2 Development of a Traffic Learning and Prediction Model	14
3.3 Development of the Bandwidth Allocation Model	15
3.4 Development of the Operational Architecture	15

4 Data Analysis and Algorithm Development	16
4.1 UDSM Network Topology Implementation in NS3	16
4.2 IP Traffic Data Generation	17
4.3 IP Traffic Data Prediction With Poisson Regression	22
4.4 Bandwidth Optimization Algorithm	27
5 Proposed Operational and Architecture Designs	29
5.1 Operational Model Design	29
5.2 Architecture Design	32
6 Conclusion	33
6.1 Contributions	33
6.2 Challenges and Limitations	33
6.3 Conclusion and Recommendation	34
6.3.1 Conclusion	34
6.3.2 Recommendations	35
References	36
Appendices	38
A UDSM LAN Topology Implementation Code	39
B Traffic Data Generation and Cleaning Code	44
C Traffic Data Learning, Prediction, and Visualization Code	47
D Project Budget for Semester II	50

List of Acronyms and Abbreviations

API	Application Programming Interface
ATM	Asynchronous Transfer Mode
BW	Bandwidth
DPI	Deep Packet Inspection
Gbps	Gigabits per Second
GPS	Generalized Processor Sharing
GR	Guaranteed Rate
ICT	Information and Communication Technologies
IETF	Internet Engineering Task Force
IP	Internet Protocol
Mbps	Megabits per second
ML	Machine Learning
NN	Neural Networks
QoE	Quality of Experience
QoS	Quality of Service
RFC	Request for Comments
SDN	Software Defined Networking
SNMP	Simple Network Management Protocol
SNMPVx	SNMP Version x
STM-x	Synchronous Transport Module level-x
TCP	Transmission Control Protocol
TSF	Time Series Forecasting
UDP	User Datagram Protocol

List of Tables

2.1	The Allocations at some of the subnets in the UDSM LAN	8
2.2	Ping statics to UDSM server (196.44.161.196) and Google (8.8.8.8) over the period of 3 days in February.	10
4.1	Traffic Data Excerpt from Figure 2.4	18
4.2	The Allocations at some of the subnets in the UDSM LAN	19

List of Figures

2.1	TCP/IP Model	5
2.2	Software-defined Networking Architecture [1]	8
2.3	Traffic Profile in a range of 3 Days	9
2.4	Traffic Rate in a range of 3 Days	9
2.5	Sum of traffic in a range of 3 Days	9
2.6	Some of the ping command responses on Feb 5, 2020 at one of the computers in the network	11
2.7	The current network topology at UDSM	11
3.1	The Methodology	14
4.1	UDSM Network Topology Implementation extract in NS3	16
4.2	UDSM Network Topology Implementation Configuration extract in NS3	17
4.3	Data Generated by the traffic generation algorithm	20
4.4	Generated Packets Distribution	21
4.5	Generated Inter-arrival times Distribution	22
4.6	Data extracts after dividing them into sequences	23
4.7	Predicted Traffic against Actual Traffic with respect to the past THREE data points	24
4.8	Generalized model results for 3 past data points	25
4.9	Predicted Traffic against Actual Traffic with respect to the past NINE data points	26
4.10	General model results for 9 past data points	26
4.11	Bandwidth Allocation Algorithm Flowchart	28
5.1	The Sequence Diagram for the operations	30
5.2	The Router and Server Operational Model	31
5.3	The Architectural Design of the Operational Environment	32

Chapter 1

Introduction

1.1 Introduction

Internet has been the very important technology since its release for the educational research community earlier and later for the general public in 1983 [2]. It first began with the simple file sharing amongst ARPANET computers. Since its adoption and its release to the whole world, Internet has proved to be an essential part and parcel of all activities in the world. Information sharing, online businesses, social networks, and education materials delivery (a few of the many value added services) happen due to enhanced connectivity around the world. The Internet has made it possible to find or deliver or search for or share information, make transactions, and so many others, in just a matter of seconds or at most a minute.

Like any other institutions out there, the University of Dar es Salaam (UDSM) has made its efforts to make sure that the colleges, school, and institutions within it are connected to the outside world via the Internet. However, the quality of experience for the individual users of the network using wired and wireless means around the university to access the Internet are varying from time to time. Some colleges and schools experience slow Internet connections while some other colleges experience faster connections. CoICT, for example, experiences very slow Internet connections during the day while SoED experiences the opposite. Nevertheless, the Internet connection speed at CoICT is always at its peak during night hours.

While it is true that the number of Internet service users affects the speed with which the individual users in a certain subnet access the Internet, and that it is deemed necessary to provide each subnet (in this case college LANs) with estimated needed bandwidth, it remains necessary to find the way that we can share the bandwidth that does not get used in other subnets during the times of congestion in one or more of them. It is, therefore, the focus of this work to devise the method that will optimize the allocated bandwidth in the subnet with respect to traffic in the other subnets.

1.2 Challenges in Bandwidth Management

The need for a faster Internet connection remains pure. It proves even much more important when the study involves the academic institutions. Therefore, it was deemed necessary to look for the possible causes of a slow Internet connection from the experiences of other people. As mentioned in [3], below are bandwidth challenges faced by the Internet operators around the globe:

- (i) Constant or increasing rates of traffic growth are commonplace – growth that is not always visible from publicly available sources.
- (ii) We can't build infinite bandwidth to the edge everywhere. It is therefore necessary to engineer and deploy mechanisms that allow performance to degrade much more gracefully than is typically the case today.
- (iii) Better architectural support for technical collaboration between network and content aggregators and application providers to deliver more network and application management data is a high priority.

However, the causes of the bandwidth instabilities and degradation of the QoS within the UDSM LAN have been analyzed and summarized to the following:

- (i) Online video streaming during daytime.
- (ii) Concurrent downloads of ultimate huge files during daytime.
- (iii) Network elements misconfigurations—these lead to unavailability of internet at times.
- (iv) Absence of monitoring tools for each entity at the University to monitor their infrastructures internally. Hence, the means for dynamic allocation of bandwidth remains necessary.

1.3 How Challenges in Bandwidth Management are Tackled

Due to highlighted concerns for the improved Quality of Service of the UDSM LAN, different measures have been implemented including:

- (i) Blocking of some sites that are known to consume much more bandwidth when the data traffic to the sites are allowed, e.g. most torrent sites.
- (ii) Prioritization of traffic during very important events—particularly meetings—which trades the bandwidth sharing purposes to serving one particular event at certain times.
- (iii) The use of passive network in providing effective solution to bottleneck problem [4].
- (iv) The use of traffic shapers in the network.

Items (i) and (ii) above are both static approaches while (iii) is one case of dynamic solution.

1.4 Statement of the Problem

The reliability of the Internet remains an important factor for making sure that the activities around the university go on. Nowadays, most academic activities require that the students and instructors be connected to the Internet. Due to that requirement, a slow network would never serve the purpose. Neither would the fast network at maximum utilization or overutilization. Therefore, to ensure that the available bandwidth gets well utilized, and serves the connected institutions, with relatively the same quality of service to individual users without experiencing delays and jitters in accessing the services remains the question that needs immediate answers.

This work aimed at answering that question and propose the solution. Specifically, this study discusses the potential of employing machine learning in forecasting the future Internet traffic values from specific subnets so that the bandwidth can be allocated dynamically according to the demands of the institution at certain instances of time. On top of that, this study aimed at proposing a feasible and deployable operational architecture of the solution.

1.5 Motivation

To experience a better Internet connection, one must be able to appreciate the speed with which the information comes to them. However, varying levels of services between two or more subscribers that are clients of the same service provider will definitely lead to dissatisfaction—which is exactly what is happening with CoICT. So, improving the QoS in the UDSM LAN would not only save a lot of bandwidth that does not get utilized but also improve the quality

of experiences from the users at highly congested areas.

It is that concern and the exploration of the possible available solutions to bandwidth sharing challenges that motivated me to explore the science and mathematics behind bandwidth allocation and management, especially in a shared environment.

1.6 Research Objectives

The main objective of this study was to address and provide solution to varying QoS experienced at different locations of the UDSM and propose a generalized solution for use in similar environments. The specific objectives of this study were:

- (i) To study the past traffic data at the UDSM LAN and develop a dynamic bandwidth sharing mathematical model from which a bandwidth optimization algorithm will be derived from.
- (ii) To emulate the current UDSM network and point out the required changes.
- (iii) To propose the architecture that would support the operation of the model and the algorithm.

Chapter 2

Literature Review

2.1 Definition of Terms

(i) IP Networks

Considering the TCP/IP model, Figure 2.1, the Internet Protocol (IP) is one of the components on layer 3 of this layered architecture. It is responsible for relaying datagrams across network boundaries. Its routing functions enables internetworking and essentially establishes the Internet.

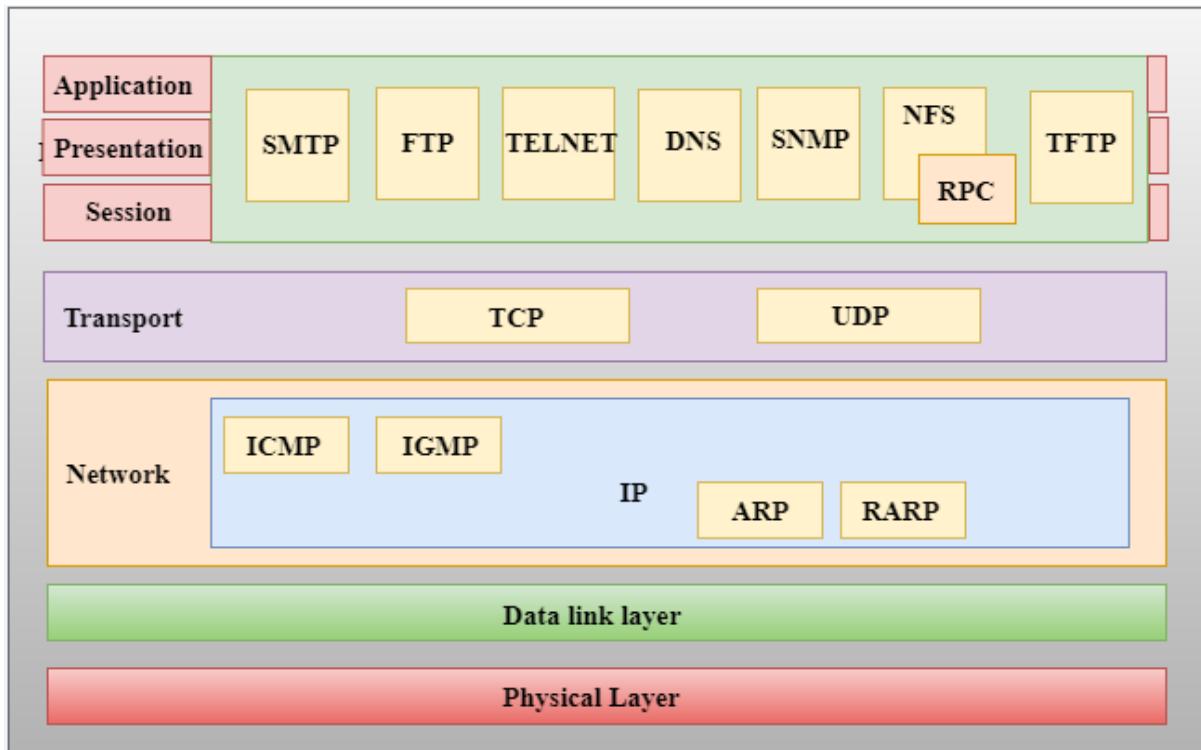


Figure 2.1: TCP/IP Model

(ii) Quality of Service

The definition of QoS depends on the perspective adopted [5]. Considering the IP networks, QoS can be looked at from three main points of view namely:

- (i) Packet drop probability
- (ii) Delays
- (iii) Delay Jitters

Therefore, with respect to the context of this work, QoS is defined as the probability of packet drop in the network. That is, the percentage loss of packets. This definition is adopted due to its close relationship with the bandwidth allocated for data sharing in the network. Narrow bandwidth would mean a limited number of packets flowing in and out the central router, hence resulting to congestion, higher latency levels and higher packet drop probabilities during peak times.

(iii) Machine Learning

Machine Learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without explicitly being programmed to. Machine learning falls into two categories; supervised learning, and unsupervised learning. Supervised learning involves providing the computer program with the past data that has certain inputs with some outputs (labels). The provided data helps the computer program to determine the meaning of the data. On the other hand, unsupervised learning utilizes the provision of the data without any labels and therefore has much more work to finding out what the data means. To do this unsupervised learning uses certain number of algorithms including K-Means algorithm.

Considering the nature of the problem for the study centered at how past traffic was at the UDSM LAN, a number of supervised learning models are fine to be used; one being the linear regression.

(iv) Network Calculus

This is a theory of deterministic queuing systems found in computer networks. It is a set of recent developments that provide deep insights into flow problems encountered in networking. The foundation of network calculus lies in the mathematical theory of dioids, and in particular, the Min-Plus dioid (also called the Min-Plus algebra). With network calculus, we are able to understand some fundamental properties of integrated services networks, window flow control, scheduling and buffer or delay dimensioning. It presents the statistical network calculus in a setting where both arrivals and service are specified in terms of probabilistic bounds.

When a node performs a reservation, it is necessary to check whether local resources are sufficient. In general, the method for this consists in breaking the node down into a network of building blocks such as schedulers, shapers, and delay elements. Generally, network calculus considers the arrival and service curves in determining the required level of resources allocation for specific sessions. It considers the probabilistic arrival rates and develops the necessary models for optimization of the allocation of the service sessions in the routers.

According to [6], to exploit statistical multiplexing gain of traffic resources in a network, service provisioning requires a framework for the stochastic analysis of network traffic and commonly- used scheduling algorithms. The most influential such framework is the effective bandwidth which, from a qualitative point of view, describes the minimum bandwidth required to provide an expected service for a given amount of traffic. However, applications of the effective bandwidth approach have generally been limited to large buffer asymptotes and other asymptotic approximations.

(v) **Software-defined Networking (SDN)**

While the fundamentals of network engineering rely on the traditional approach for the design, deployment and management where the forwarding, control and data planes are all embedded in one hardware [7], the networks are growing big and it becomes hard to manage each and every component of the network elements. SDN, however, brings about the separation between the three planes (Figure 2.2). At the control plane, that is where most of the configurations, network programmability and management take place.

It is the easiness associated with the architecture that makes it fine for the networks that are prone to growing big over time.

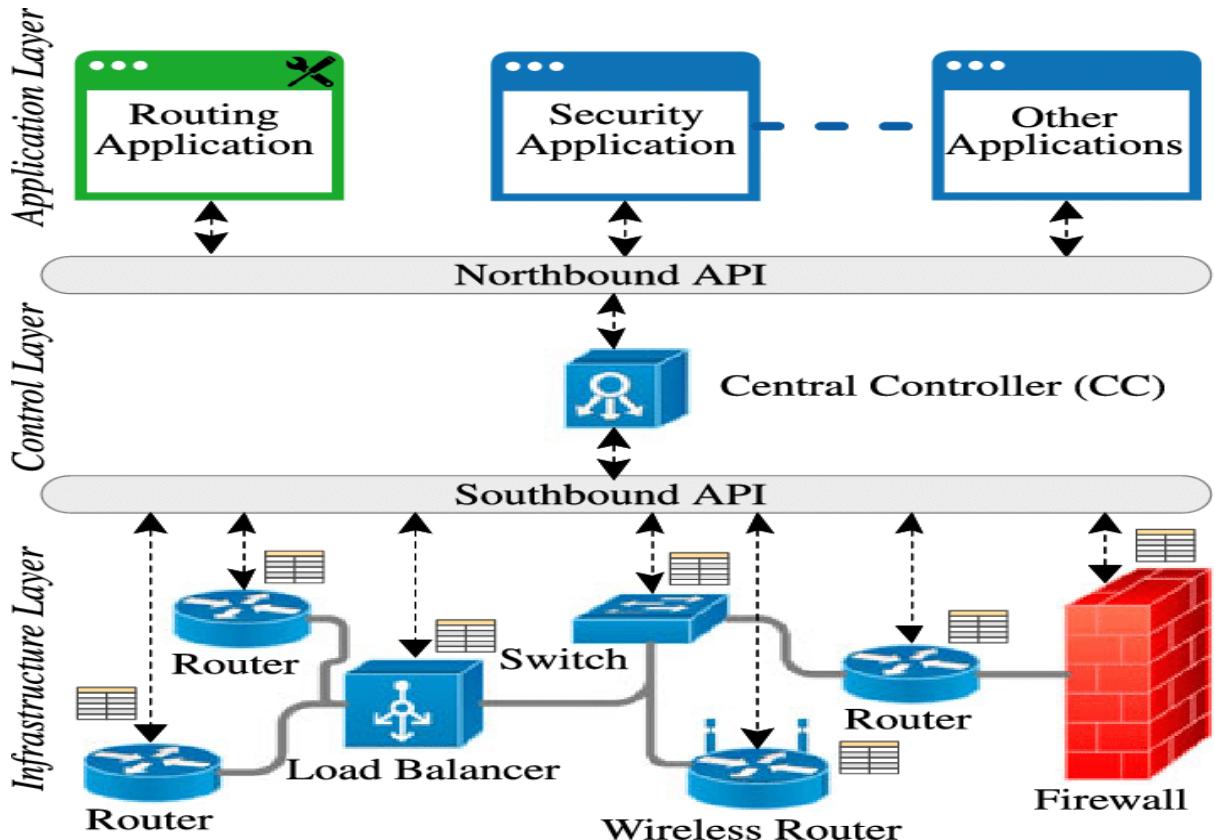


Figure 2.2: Software-defined Networking Architecture [1]

2.2 Current situation

Considering the related situation with the UDSM network, it was deemed necessary to find out how the bandwidth is allocated for different subnets within the LAN. Table 2.1 shows how the bandwidth allocation is at the writing of this project report.

Subnet	Bandwidth Allocated (Mbps) (Download/Upload)	Burst (Mbps)
Maths	32/32	35
Lib	5/5	10
CoET	20/20	25
Hall5	10/8	16/8
CoICT	18/8	20
Library (New)	15/15	15

Table 2.1: The Allocations at some of the subnets in the UDSM LAN

The figures in Table 2.1 are always at the peak most of the times. To add on, the total bandwidth allocated for the University is 155 Mbps (STM-1 allocation), and it overshoots (bursts) to 200 Mbps at times.

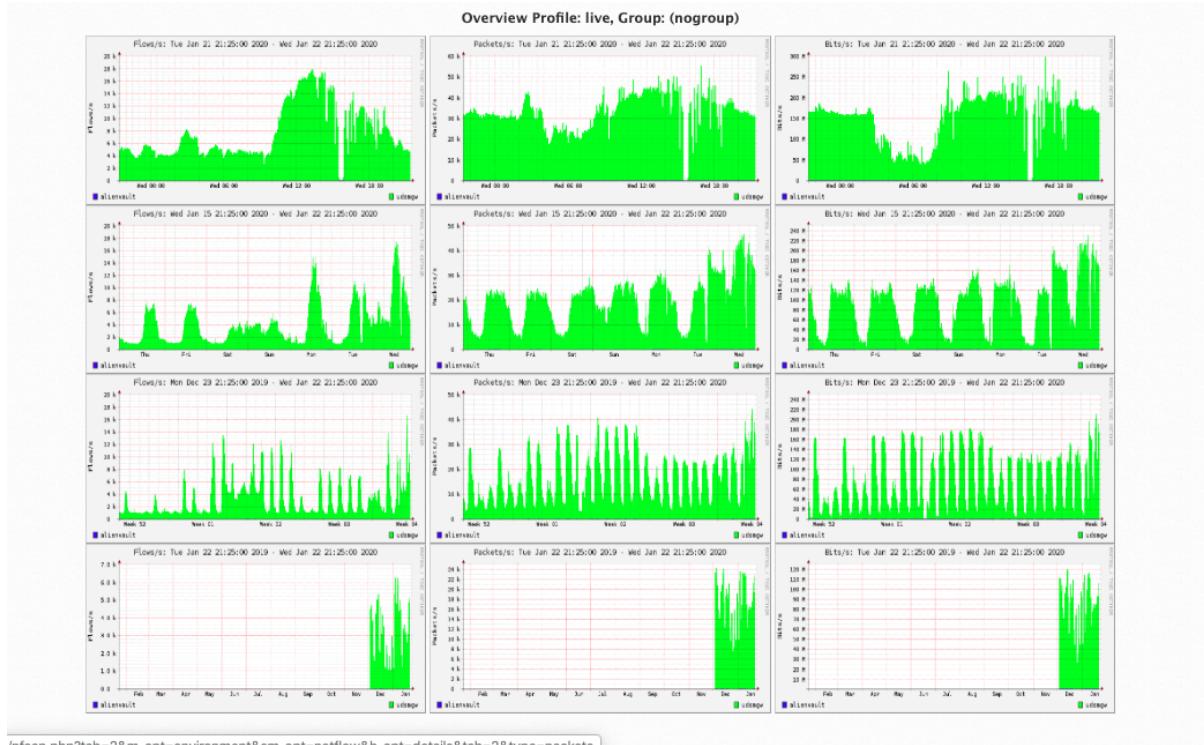


Figure 2.3: Traffic Profile in a range of 3 Days

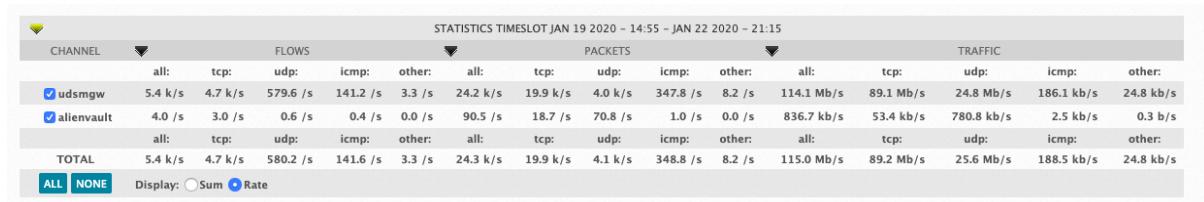


Figure 2.4: Traffic Rate in a range of 3 Days

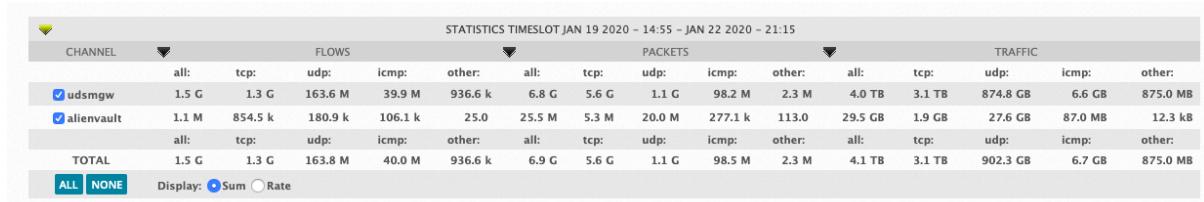


Figure 2.5: Sum of traffic in a range of 3 Days

The figures above, 2.3, 2.4, and 2.5 depict the situation with which the UDSM traffic is in with respect to the protocols associated with the Internet. The figures show that the network is at the

peak most of the times at daytime. Therefore, anyone modeling this network must make sure that it depicts the current situation.

To make sure that the above scenario depicted in the figures above was correct, more work was done to capture how much data could be exchanged over the day time compared to that exchanged over the night time. A **ping** command was set at one of the computers—at CoICT—in the network for some days in Feb 2020, and the following data extract on Table 2.2 was obtained.

Date	Destination Address	Time of the day	Packets sent	Packets received	Average Delay	%ge Loss
Feb 3	196.44.161.196	Evening	4027	3453	2ms	14
Feb 3	8.8.8.8	Evening	3466	2663	113ms	23
Feb 4	8.8.8.8	Evening	1572	1269	464ms	19
Feb 4-5	8.8.8.8	Overnight	31701	29415	24ms	7
Feb 5	8.8.8.8	Daytime	9987	6267	UNDEFINED	37

Table 2.2: Ping statics to UDSM server (196.44.161.196) and Google (8.8.8.8) over the period of 3 days in February.

From the data extract above, it was observed that:

- (i) The network is generally slow during daytime and it is characterized by higher numbers of packet losses.
- (ii) The network has relatively low delays in the evening compared to daytime. This is,in one way, due to offices closing around the University, thus decrements in congestion.
- (iii) The network is relatively fast at night and has less packet losses compared to both daytime and evening. This happens to be due to less number of user nodes in the network during night time.
- (iv) There are times where the network is not reachable at all (*as depicted in Figure 2.6*). This situation dominates a lot at CoICT to the point that in order to obtain the data on general network performance and latency, I had to wait for the time when the network was available.

Figure 2.6: Some of the **ping** command responses on Feb 5, 2020 at one of the computers in the network

To add on, the details on the preceding paragraphs were obtained from the current network topology at the University of Dar es Salaam in Figure 2.7.

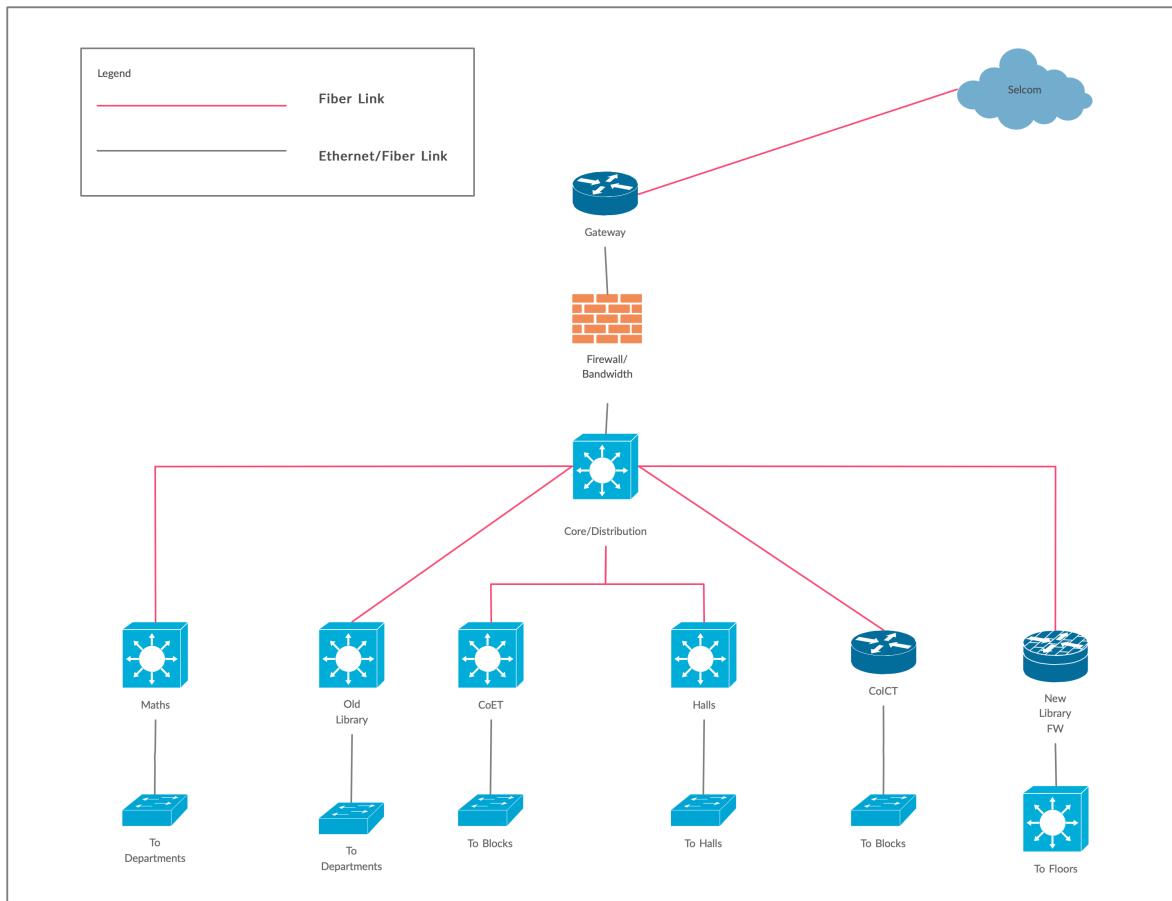


Figure 2.7: The current network topology at UDSM

2.3 Related Works and Systems

The literature review on different bandwidth allocation models is an ongoing work. Up to this point, a range of studies suggested a possible hack on how to allocate the bandwidth depending on what IP addresses accessed what content and how often. One of the studies [8] sought to propose the dynamic bandwidth allocation model based on how the users rated the websites and how often the users would visit the websites. The work went further to develop the bandwidth allocation model and ended up comparing it to originally allocated bandwidth for the chosen websites, which proved to be an efficient approach. The idea in this work means that it could be extended to dealing directly with the IP addresses, which is obviously correct, and that the bandwidth allocation could be done depending on who access what and from what context. However, that is not the focus of this work.

To address the problems associated with bandwidth management and allocation, different network monitoring systems were developed and are in operation right now. Some are:

(i) Nagios

It is one of the leading in industry providing monitoring solutions for small and enterprise level of infrastructure. It monitors different components including network protocols, operating systems, system metrics, applications, services, web servers and middle ware systems. It also provides detailed status information on all devices, alerts on configured features of the system via e-mail, SMS, or custom scripts, and generally helps the users plan for future upgrades and even integrate with in-house or third-party applications due to multiple APIs.

Nevertheless, one has to be very specific when telling the system what to monitor.

(ii) Cacti

This is another open source network monitoring tool which can be installed on Linux and Windows operating systems. It can be connected to round robin database tool (RRDT) which allows to generate graphs related to relevant network data. It requires MySQL database in order to work fine.

(iii) Zabbix

Like Nagios, Zabbix was designed to monitor everything from performance and avail-

ability of servers, network equipment to web applications and databases. Its server-agent system architecture where an agent is installed on a server (a client being monitored). It has an advantage that it can be installed on Linux, Windows, Solaris, MacOS, FreeBSD, OpenBSD and supports SNMP for better reporting.

(iv) **Icinga**

It started as a fork of Nagios in 2009. Icinga monitors network and host services, servers and components, provides support for event handlers and notifications, and has a choice between Classic UI and Icinga web in operational user interfaces.

(v) **OpenNMS**

Like the above stated, OpenNMS allows to collect systems information using SNMP, HTTP, and others. With OpenNMS one can discover layer two network topologies in their network. It is built on event-driven architecture. Its full package comes in with device temperature monitoring, IPv4 and IPv6 support and geographical node map to show nodes and service outages using Open Street Map, Google Maps or MapQuest. It has limited resources for development and maintenance.

Although a thorough review was done and the mentioned tools were tested, none of them provides the ability to manage the bandwidth. The best that they can do is to provide graphs for visualization of bandwidth utilization in the network. None of them does analysis of the utilization of the bandwidth and therefore requires human intervention in trying to manage the bandwidth. Therefore, the goal of this work remains pure and necessary to be achieved.

However, there are a number of companies that use bandwidth scaling, which is the feature that allows the network operators to increase or decrease the bandwidth allocations based on how much traffic there is in the network [9]. The technique relies on the insights of the network operators, hence no intelligence in traffic is associated with the bandwidth allocations.

The focus of this work, therefore, was to devise the method which employs the machine learning model that determines where and when to allocate the resources without involving the network operators in the process.

Chapter 3

Methodology

3.1 Introduction

The methodology in this study consisted of three activities as summarized in Figure 3.1 and explained in the next sections.

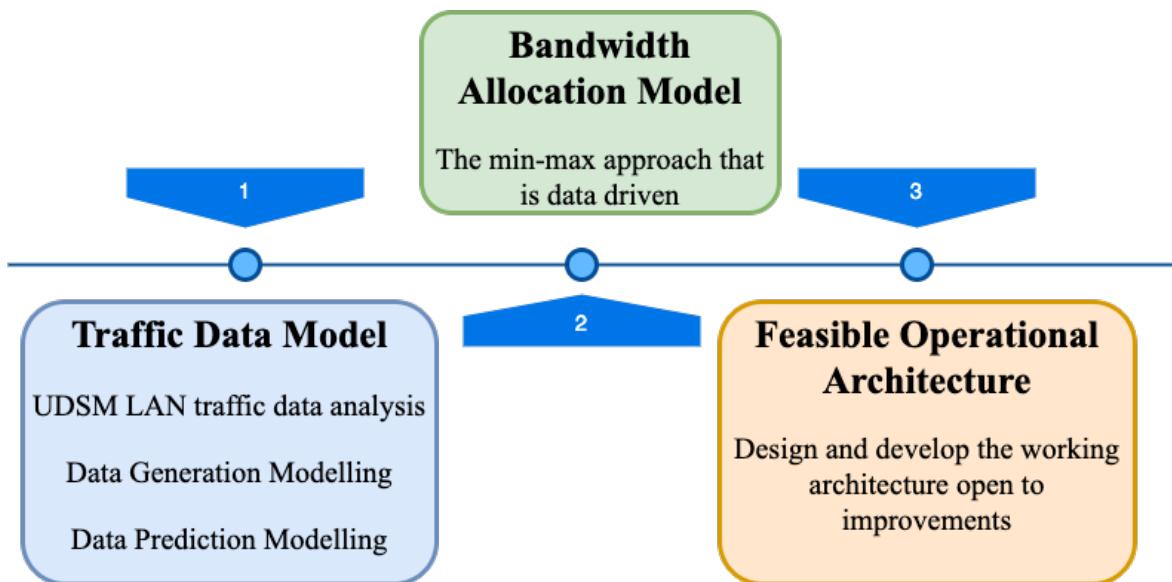


Figure 3.1: The Methodology

3.2 Development of a Traffic Learning and Prediction Model

To attain the sole purpose of bandwidth allocation in the subnetworks, the insights from the data obtained—in chapter 2—had to be extracted. These insights were then used to develop network traffic generation and prediction models. The data generation model was implemented in python, taking into account the stochastic nature of the IP traffic data.

The traffic learning and prediction model was designed, developed and implemented in python. The model took advantage of the available python **statsmodels** and **patsy** libraries. These libraries helped to model the poisson regressor without having to write one from scratch. The learning was done in three different phases. Each of these phases was done independently while the simulation time was differentiated from each of them. At first, it was determined how much

the current traffic would depend on the past two data points—which performed worse in the prediction, while in the next two the predictions were made with respect to three and nine past traffic data points respectively. .

Despite their better performances in other kinds of data, Time Series Forecasting Models (TSF) and Neural Networks (NN) were not used. This was due to the fact that the data produced for study would do better with a poisson regressor due to its nature.

3.3 Development of the Bandwidth Allocation Model

The analyses done from the development of a traffic learning and prediction model made it easy to understand the nature of the traffic and the underlying predictions. This was followed with the mathematical analysis which produced a simple equation that would later be used to divide the bandwidth amongst the network elements in the network core depending on the amount of traffic that comes in and goes out of such elements. The developed equation considered the mean of the predictions as observed in the outputs from running simulations.

The implementation in this phase was done in python.

3.4 Development of the Operational Architecture

The feasible operational architecture was developed after a thorough review of SDN and associated technologies. The operations involving bandwidth allocation, traffic analysis and prediction in the core network were centralized. This was due to the fact that the bandwidth allocated for an institution is configured along the transmission line between its point of presence (POP) and the Internet service provider (ISP) premises. Therefore, it would be the central router or the core network that managed the bandwidth resources allocated for the institution.

Chapter 4

Data Analysis and Algorithm Development

4.1 UDSM Network Topology Implementation in NS3

In order to accomplish the sole purpose of implementation, analysis, and bandwidth optimization algorithm development, the UDSM topology in Figure 2.7 had to be implemented in NS3. The following comment extracts from the code—Figure 4.1 and Figure 4.2—show the considerations made in writing the code for the configuration of the network.

```
.  
5 ======  
6 Author: Emmanuel Julius Samson  
7 Reg #: 2016-04-01802  
8 Degree: B.Sc. Telecommunications Engineering (2020)  
9 FYP Title: IP Networks QoS Optimization using ML and SDN  
10 ======  
11  
12  
13 A simplified extract of the UDSM LAN  
14 NB: The FW/BW node has not been implemented in the code  
15  
16 |  
17 | Gateway Router to Selcom  
18 | |  
19 | |  
20 | | FW/BW  
21 | | |  
22 | | |  
23 | | Core Distribution switch  
24 | | |  
25 | | |  
26 | ======  
27 | | | | | | | |  
28 | | | | | | | |  
29 | Maths_Switch Old_Library CoET Halls CoICT_Router New_Lib_Router |  
30 | | | | | | | |  
31 | | | | | | | |  
32 | Switch Switch Switch Switch Switch Switch |  
33 | To_Depts To_Depts To_Blocks To_Halls To_Blocks To_Floors |  
34 | | | | | | |  
35 | | | | | | |  
36  
37  
38 ======  
39  
40 Tasks:  
41 1. Create the network nodes  
42 2. Install the internet stack on each of the nodes  
43 3. Attach one or more network devices on each node and attach the device to channel  
44 4. Create the network interface for each network device  
45 5. Assign an IP address to each network interface  
46  
47 ======
```

Figure 4.1: UDSM Network Topology Implementation extract in NS3

```

47 =====
48
49 Necessary Configurations:
50 -----
51 Routers:
52   1. Gateway      : routers[0]    -- subnet1
53   2. CoICT_Router : routers[1]    -- subnet6
54   3. New_Lib_Router : routers[2]   -- subnet7
55
56 Distribution Switches:
57   1. Core Distribution Swich : distribution_switches[0]  -- subnet1 || subnet 2 - 7
58   2. Maths                  : distribution_switches[1]  -- subnet2
59   3. Old Library            : distribution_switches[2]  -- subnet3
60   4. CoET                   : distribution_switches[3]  -- subnet4
61   5. Halls                  : distribution_switches[4]  -- subnet5
62
63 Access switches:
64   1. Maths/To_Departments   : access_switches[0]    -- subnet20
65   2. Old_Library/To_Departments : access_switches[1]  -- subnet30
66   3. CoET/To_Blocks          : access_switches[2]    -- subnet40
67   4. Halls/To_Halls          : access_switches[3]    -- subnet50
68   5. CoICT_Router/To_Blocks  : access_switches[4]    -- subnet60
69   6. New_Lib_Router/To_Floors: access_switches[5]    -- subnet70
70
71 -----
72
73
74 =====
75
76 */
77

```

Figure 4.2: UDSM Network Topology Implementation Configuration extract in NS3

To begin with, the point-to-point connection between the Gateway and the core distribution switch was put into a single subnet (*subnet1*). This allows for the assignment of IP addresses to their specific interfaces which allows for communication between them. This configuration has been carried out throughout the network; bringing together the network nodes—switches and routers—that were needed to be together in single or multiple subnets. However, any subnet that came at the access level, it was named, hence configured by the preceding subnet at the distribution nodes with a leading zero (0).

4.2 IP Traffic Data Generation

The amount of data exchanged between servers and clients in server-client configurations in IP networks is purely stochastic and follows the count based scenario—the on-off traffic generation sources that come and go during network operations. That means, the timing of the generation of traffic data is purely random but the average of traffic generated/exchanged (μ) per unit time is known. This is the characteristic of the Poisson distribution as depicted in

equation 4.1. It is this feature, this work considers, that suggests closely related traffic shapes during the day and night times.

$$P(X = x) = \frac{\mu^x}{x!} \exp(-\mu) \quad (4.1)$$

In the equation above, μ is the average of the amount of data traffic generated per unit time and x is the amount of data traffic generated in a given time. $P(X = x)$ is the probability that variable X is equal to value x at a given interval of time.

Considering the data in Figure 2.4, the following information was found useful to consider in the generation of traffic data points.

	TCP	UDP	ALL
Flows	4.7k/s	579.6/s	5.4k/s
Packets	19.9k/s	4.0k/s	24.2k/s
Traffic	89.1Mb/s	24.8Mb/s	114.1Mb/s

Table 4.1: Traffic Data Excerpt from Figure 2.4

Having determined the amount of data flowing through the network per second, i.e. $114.1 Mbps$, the traffic was then distributed among the major network access nodes with respect to the allocations in Table 2.1. It is also from this table that it was observed that the amount of bandwidth allocated for each sub-network in traffic burst times was $124 Mbps$, the value that is a little higher than the average traffic in three days as shown in Table 4.1. The traffic occupied at least 92% of the bandwidth allocated for each for burst times. This further notices that the allocated bandwidth for each sub-network did not consider the fact that the traffic would, in most times, be higher than the allocated bandwidth.

Therefore, with above considerations in mind, the traffic generation algorithm considered that the average traffic data in each sub-network was 0.92 times the amount allocated for burst periods, as shown on Table 4.2.

Subnet	Burst (Mbps)	Average Traffic (Mbps)
		$0.92 \times \text{Burst}$
Maths	35	32.2
Lib	10	9.2
CoET	25	23
Hall5	16	14.72
CoICT	20	18.4
Library (New)	15	13.8

Table 4.2: The Allocations at some of the subnets in the UDSM LAN

Since the average delay data from Table 2.2 was not sufficient to calculate the average delays over the period of time, 350ms was assumed the delay of the network during busy times (day-time). Hence, the following data generation program logic was implemented in python (see Appendix B) to generate the traffic data.

```

1 Record program starting_time
2 Initialize transmission_time[]
3 Initialize start_transmission_time[]
4 Initialize packets_transmitted[]
5
6 while(current_time - starting_time) < simulation_time:
7     Append current_time to start_transmission_time[]
8     Append randomly generated packet_number to packets[]
9     Randomly select transmission_time (average = 350ms)
10    Append transmission_time to transmission_time[]
11
12 Pack start_transmission_time[], transmission_time[] and
13    packets_transmitted together
14 Write the data to file for storage and further processing

```

Thereafter, the program was run for five minutes and the following extract of data points—Figure 4.3

913 lines (913 sloc) | 36.8 KB

Raw Blame   

Search this file...

	Start_Time	Transmission_Duration	Packets
1	2020-07-24 17:38:10.658045	0.359	19467
2	2020-07-24 17:38:11.017493	0.316	23150
3	2020-07-24 17:38:11.337716	0.292	24783
4	2020-07-24 17:38:11.630864	0.34600000000000003	20392
5	2020-07-24 17:38:11.977492	0.38	19399
6	2020-07-24 17:38:12.360392	0.36	24277
7	2020-07-24 17:38:12.725418	0.334	20766
8	2020-07-24 17:38:13.063216	0.34400000000000003	20713
9	2020-07-24 17:38:13.410676	0.318	24539
10	2020-07-24 17:38:13.729963	0.376	19597
11	2020-07-24 17:38:14.106243	0.34800000000000003	20403
12	2020-07-24 17:38:14.459585	0.385	22417

Figure 4.3: Data Generated by the traffic generation algorithm

It is, however, most important to note that the data generation algorithm above considered the main features for the prediction of traffic as highlighted in the article [10]. It is stated that there are two major traffic parameters generated by network traffic models:

- (i) Packet length distribution and
- (ii) Inter-arrival distributions

Other parameters such as routes, distribution of destinations, e.t.c. are of less importance.

To prove the stochastic distribution of the generated packets data, the code below was written for reading the data.

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 data = pd.read_csv('data_points.csv')
5
6 plt.hist(data["Packets"], bins=100)
7 plt.xlabel("Number of Packets")

```

```

8     plt.ylabel("Relative number of Generations")
9     plt.title("The Distribution of Generated Packets")
10    plt.savefig("hist_dist.png")
11    plt.show()

```

The output of the code was Figure 4.4, which clearly shows how random the data generation was.

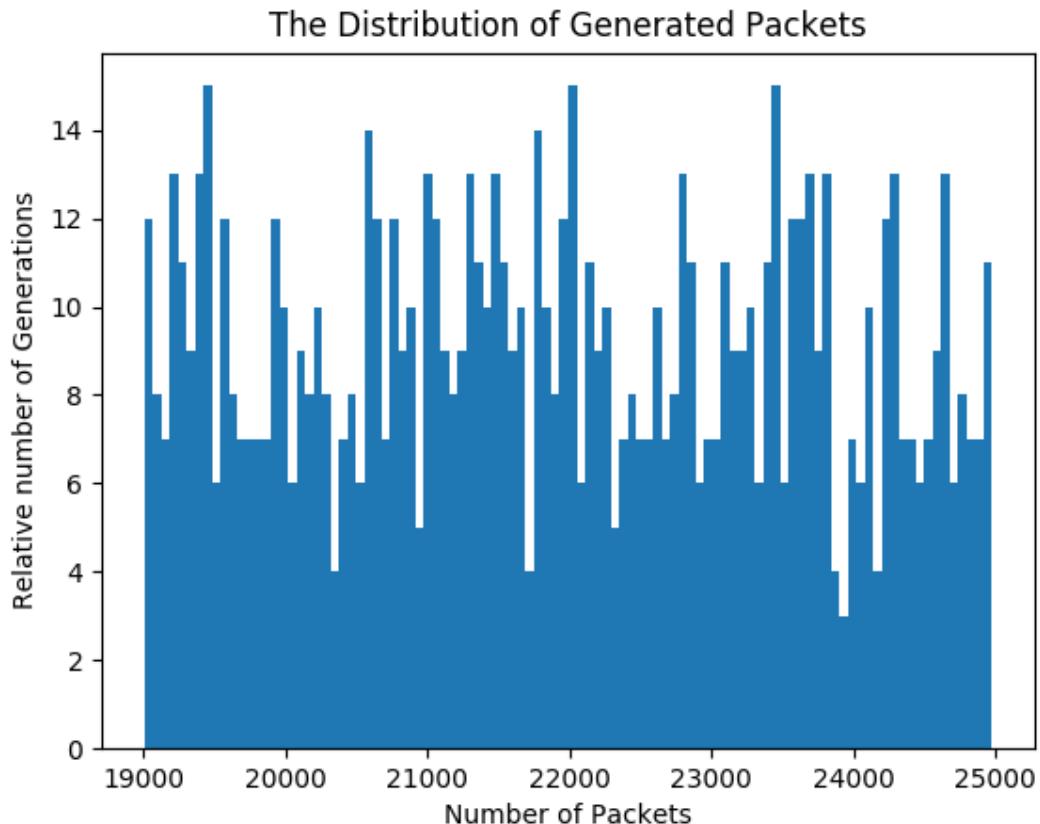


Figure 4.4: Generated Packets Distribution

The distribution of the inter-arrival times was also investigated. The following piece of code was used to determine the distribution of the inter-arrival times, and Figure 4.5 was produced. Therefore, with Figures 4.4 and 4.5, it was to the satisfaction that the data produced correlated with the stochastic nature of the IP networks data generators. Hence, further considerations for the ML algorithm for traffic learning and prediction followed.

```

1 import pandas as pd
2 import matplotlib.pyplot as plt

```

```

3
4     data = pd.read_csv('data_points.csv')
5
6     plt.hist(data["Transmission_Duration"], bins=100)
7     plt.xlabel("Inter-arrival times in seconds")
8     plt.ylabel("Relative number of Generations")
9     plt.title("The Distribution of Inter-arrival times")
10    plt.savefig("trans_dist.png")
11    plt.show()

```

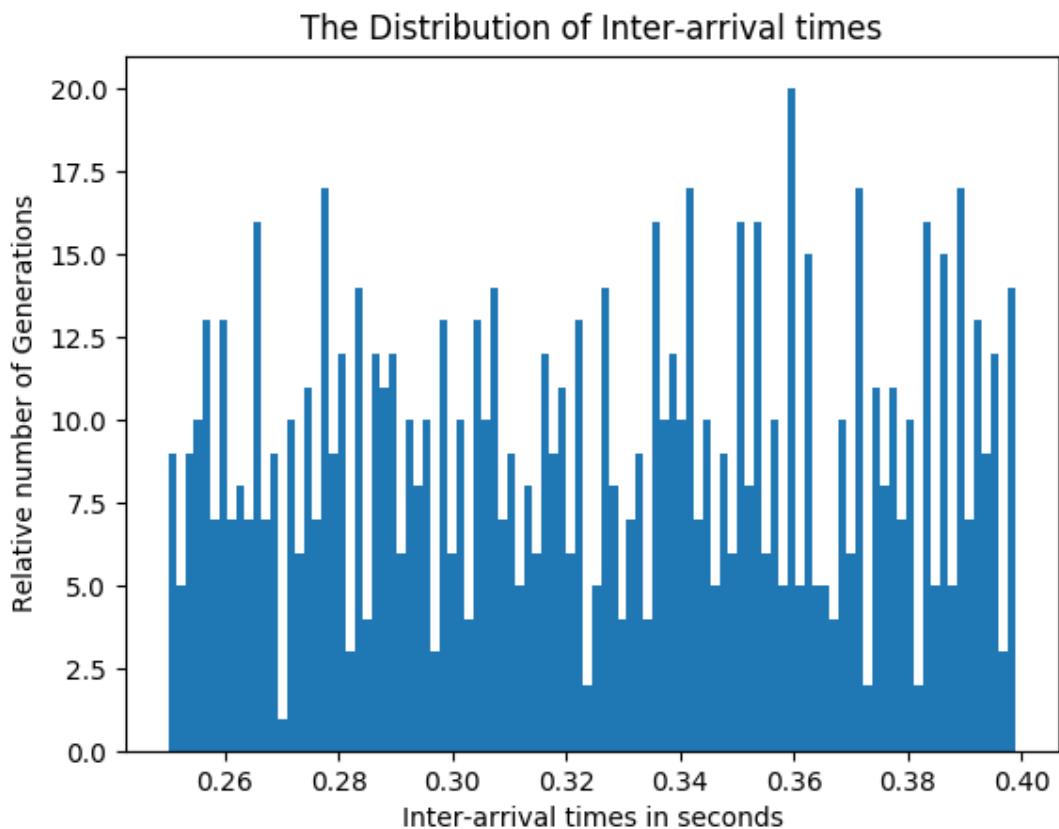


Figure 4.5: Generated Inter-arrival times Distribution

4.3 IP Traffic Data Prediction With Poisson Regression

Considering the context of the IP networks, and the nature of the data that gets produced over time, it was necessary to consider the fact that the sequence with which the data gets produced is important in modeling the data traffic prediction algorithm. It is with this in mind that the

data produced in the previous section was further modified to produce data extract in Figure 4.6. In this, *packets_4* was made the dependent variable while *packets_1*, *packets_2*, and *packets_3* were made the independent variables; making it a necessity that the next amount of traffic would depend on the past three observed values.

	packets_1	packets_2	packets_3	packets_4
1	19467	23150	24783	20392
2	23150	24783	20392	19399
3	24783	20392	19399	24277
4	20392	19399	24277	20766
5	19399	24277	20766	20713
6	24277	20766	20713	24539
7	20766	20713	24539	19597
8	20713	24539	19597	20403
9	24539	19597	20403	22417
10	19597	20403	22417	22690
11	20403	22417	22690	22512

Figure 4.6: Data extracts after dividing them into sequences

Hence, the prediction used the **patsy** and **statsmodels** libraries in python to develop the training and prediction model (see Appendix C). The code produced the results in Figures 4.7 and 4.8. This—despite meeting the prediction goal—didn’t prove to be a fine model for prediction of individual traffic. That is because due to the stochastic nature of the data, the model didn’t fit perfect and ended up predicting the average traffic most of the time.

Nevertheless, the prediction correlated with the poisson model with unchanging mean. In networks that need individual packet flow level monitoring, however, it would perform worse; it would recommend almost the same amount of bandwidth every time—despite some gradual variations—which would be unnecessary because the static allocation would do better on that.

The analysis of networks, however, considers the contingencies associated with the networks. This guess work (supported by analysis and client needs) that could, in one way or the other, determine the mean traffic that would be produced by a subnetwork during its operation should

suggest the amount of bandwidth necessary to be allocated. It is, therefore, this knowledge that would later be implemented in **typical data driven network operations** to determine the trends thereby predicting the **mean** of the traffic that would necessarily be allocated bandwidth for depending on what has been happening in the past (in minutes, hours, e.t.c). Consequently, the mean of the traffic would keep changing from time to time depending on the number of access nodes at the network access layer (Figure 5.3) and the amount of traffic that one initiates.

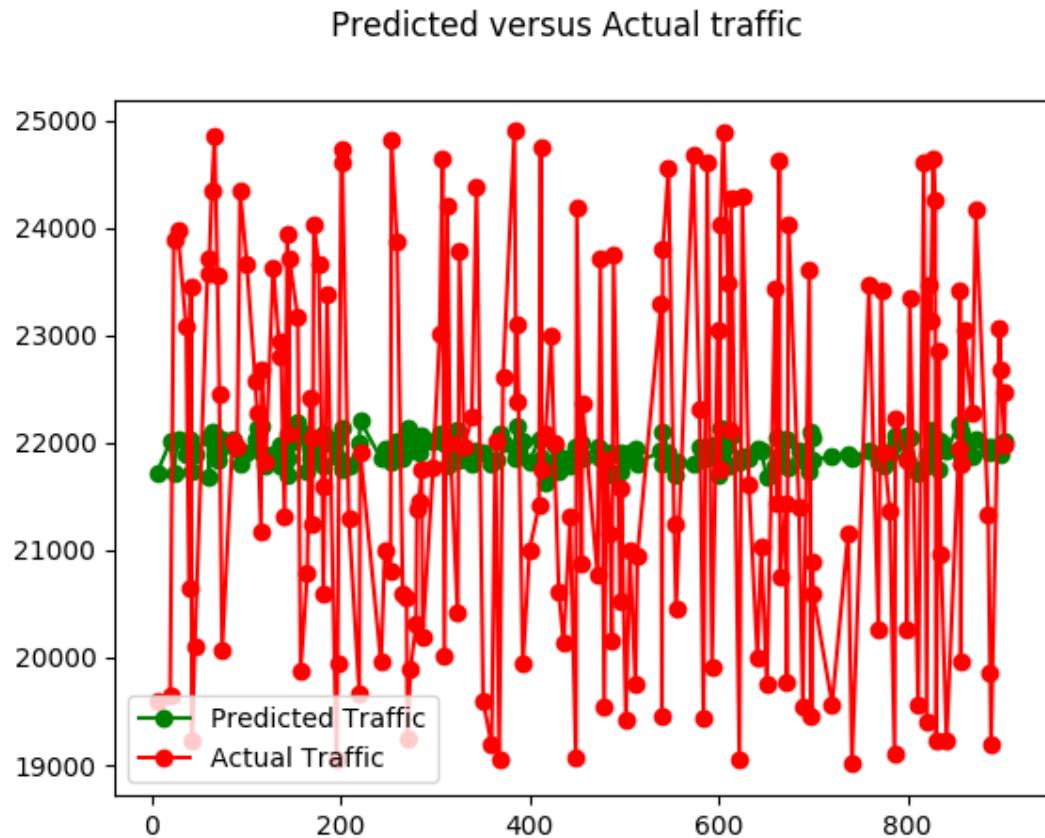


Figure 4.7: Predicted Traffic against Actual Traffic with respect to the past THREE data points

Generalized Linear Model Regression Results						
Dep. Variable:	packets_4	No. Observations:	723			
Model:	GLM	Df Residuals:	719			
Model Family:	Poisson	Df Model:	3			
Link Function:	log	Scale:	1.0000			
Method:	IRLS	Log-Likelihood:	-52059.			
Date:	Sat, 25 Jul 2020	Deviance:	95564.			
Time:	12:04:39	Pearson chi2:	9.54e+04			
No. Iterations:	4					
Covariance Type:	nonrobust					
coef	std err	z	P> z	[0.025	0.975]	
Intercept	9.9670	0.006	1728.156	0.000	9.956	9.978
packets_1	1.25e-06	1.47e-07	8.480	0.000	9.61e-07	1.54e-06
packets_2	2.251e-06	1.48e-07	15.253	0.000	1.96e-06	2.54e-06
packets_3	-2.188e-06	1.47e-07	-14.863	0.000	-2.48e-06	-1.9e-06

Figure 4.8: Generalized model results for 3 past data points

From Figure 4.8, it can be noted that the mathematical model could be summarized by the general regression equation 4.2:

$$\lambda_n = \beta_0 + \beta_1 \lambda_{(n-1)} + \beta_2 \lambda_{(n-2)} + \beta_3 \lambda_{(n-3)} \quad (4.2)$$

Where:

- λ_n Traffic generated at point n
- β_0 The intercept of a regressor = 9.9670
- β_1 The coefficient for traffic at $n-1$ = 1.256e-6
- β_2 The coefficient for traffic at $n-2$ = 2.251e-6
- β_3 The coefficient for traffic at $n-3$ = -2.188e-6

Therefore equation 4.2 becomes:

$$\lambda_n = 9.9670 + 1.25e-6\lambda_{(n-1)} + 2.25e-6\lambda_{(n-2)} - 2.188e-6\lambda_{(n-3)} \quad (4.3)$$

The coefficients in equation 4.3, however, will change from time to time depending on the available data for processing. This was determined when the simulation was run for 15 minutes, producing 2743 data points. Then, the prediction algorithm was run on the data taking the past 9 traffic points as the determinants of the 10th. Figure 4.9 was the output of such training and prediction while Figure 4.10 was the output of the trained model.

Predicted versus Actual traffic

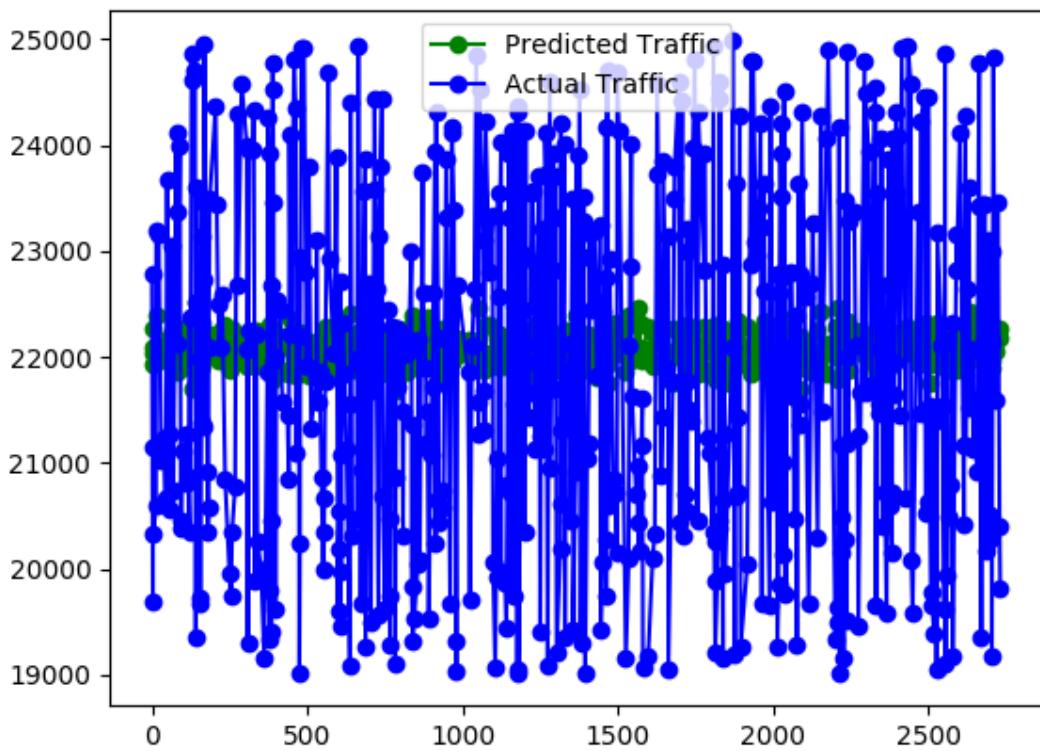


Figure 4.9: Predicted Traffic against Actual Traffic with respect to the past NINE data points

Generalized Linear Model Regression Results									
Dep. Variable:	packets_10	No. Observations:	2176						
Model:	GLM	Df Residuals:	2166						
Model Family:	Poisson	Df Model:	9						
[Link Function:	log	Scale:	1.0000						
Method:	IRLS	Log-Likelihood:	-1.5945e+05						
Date:	Thu, 30 Jul 2020	Deviance:	2.9314e+05						
Time:	15:24:20	Pearson chi2:	2.93e+05						
No. Iterations:	4								
Covariance Type:	nonrobust								
	coef	std err	z	P> z	[0.025	0.975]			
Intercept	10.0693	0.006	1761.470	0.000	10.058	10.081			
packets_1	-1.463e-06	8.39e-08	-17.436	0.000	-1.63e-06	-1.3e-06			
packets_2	-1.504e-06	8.4e-08	-17.911	0.000	-1.67e-06	-1.34e-06			
packets_3	2.335e-06	8.38e-08	27.852	0.000	2.17e-06	2.5e-06			
packets_4	-1.131e-07	8.41e-08	-1.345	0.179	-2.78e-07	5.17e-08			
packets_5	6.282e-07	8.42e-08	7.460	0.000	4.63e-07	7.93e-07			
packets_6	-4.585e-07	8.44e-08	-5.430	0.000	-6.24e-07	-2.93e-07			
packets_7	-2.055e-06	8.42e-08	-24.404	0.000	-2.22e-06	-1.89e-06			
packets_8	-5.466e-07	8.48e-08	-6.447	0.000	-7.13e-07	-3.8e-07			
packets_9	1.272e-07	8.38e-08	1.517	0.129	-3.71e-08	2.92e-07			

Figure 4.10: General model results for 9 past data points

Hence, the linear model in equation 4.2 could be simply summarized to produce a modified equation 4.4 as follows:

$$\lambda_n = \beta_0 + \sum_{m=1}^{n-1} \beta_m \lambda_{n-m} \quad (4.4)$$

The symbols have the same meaning as in equation 4.2

4.4 Bandwidth Optimization Algorithm

From the findings in the two preceding sections, it was found out that bandwidth optimization could easily be determined by considering **the mean of the predicted traffic data** from each subnetwork utilizing the total allocated bandwidth. Hence, equation 4.5 was devised; setting a foundation for bandwidth allocation algorithm that allocates resources depending on the usage of one subnetwork against the others.

$$\gamma_i = \frac{\mu_{\lambda_i}}{\sum_{i=1}^n \mu_{\lambda_i}} \times \Gamma \quad (4.5)$$

Where:

γ_i Bandwidth allocated to subnetwork i

μ_{λ_i} Predicted mean of the traffic at subnetwork i

Γ Total allocated traffic at the core network

And,

$$\mu_{\lambda_i} = \frac{\sum \lambda_i}{n} \quad (4.6)$$

n is the number of traffic points in the test set.

With those equations in the mind, and the observations from the preceding sections, the flowchart in Figure 4.11 was devised to depict the typical process that would achieve traffic prediction and bandwidth allocation during the network operations.

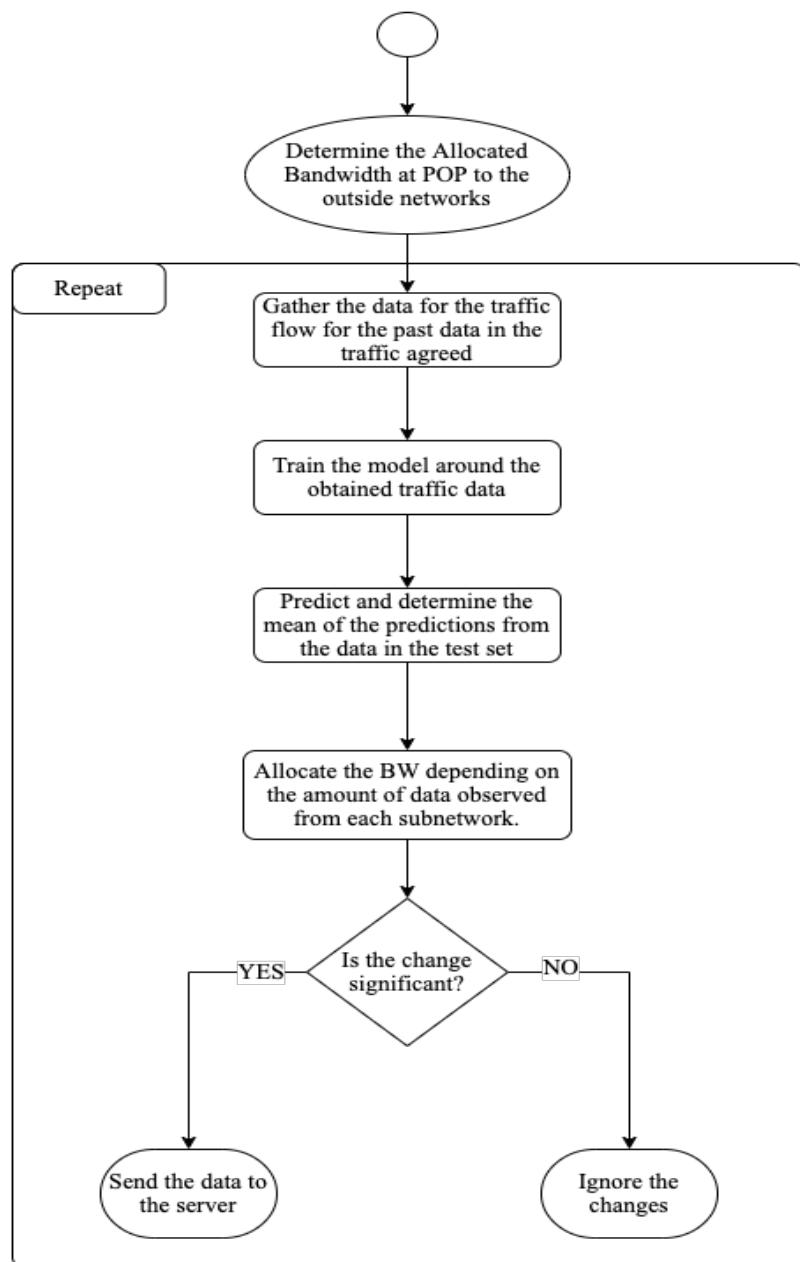


Figure 4.11: Bandwidth Allocation Algorithm Flowchart

Chapter 5

Proposed Operational and Architecture Designs

In chapter 4, this work went into details on the analysis and considerations done in the development of the bandwidth allocation algorithm. Also, as addressed in chapter 1, this work aimed at proposing the operational model and architecture designs which would support the implementation of the solution. Therefore, these designs and considerations are explained. Nevertheless, these designs are open to modifications and corrections depending on the users involved and the equipment vendors specific details.

5.1 Operational Model Design

For production purposes, the proposed Internet traffic data driven bandwidth allocation algorithm is expected to be shipped with other supporting scripts. In so doing, the whole system should run in two different places; on the server and on the router. Internet traffic data processing, learning and prediction, and bandwidth allocation scripts would be run on the server while actual bandwidth allocation would be run on the router.

The server should be responsible for hosting the running scripts and therefore responsible for providing the necessary operational environment including the python interpreter.

The router will also be responsible for providing configuration success notification which will help to tell whether it is needed to repeat the configurations in case they proved failure.

There should be one or more python (or any other language) scripts together with the bandwidth allocation algorithm which, in their collection, will be responsible for the following:

- (i) Calculate and allocate initial minimum bandwidth for each router interface
- (ii) Connect the server to the router via telnet (or any other access method thought necessary)
- (iii) Capture the Internet traffic data from each interface on the router
- (iv) Predict the Internet traffic for each interface in the interval of time agreed depending on how critical the network is deemed to be.

- (v) Call for bandwidth allocation algorithm, and provide it with the predicted traffic values
- (vi) Allocate bandwidth for each interface depending on the provided predicted traffic values
- (vii) Prepare the configuration scripts and send them to the router
- (viii) Configure the router depending on the prepared configuration scripts
- (ix) Determine whether the configurations done were successful
- (x) Prepare and store configuration logs for future reference

Therefore, Figure 5.1 depicts the typical sequence diagram of the operations between the server and the router while Figure 5.2 summarizes the operational model design for the two major entities for the correct operation of the algorithm.

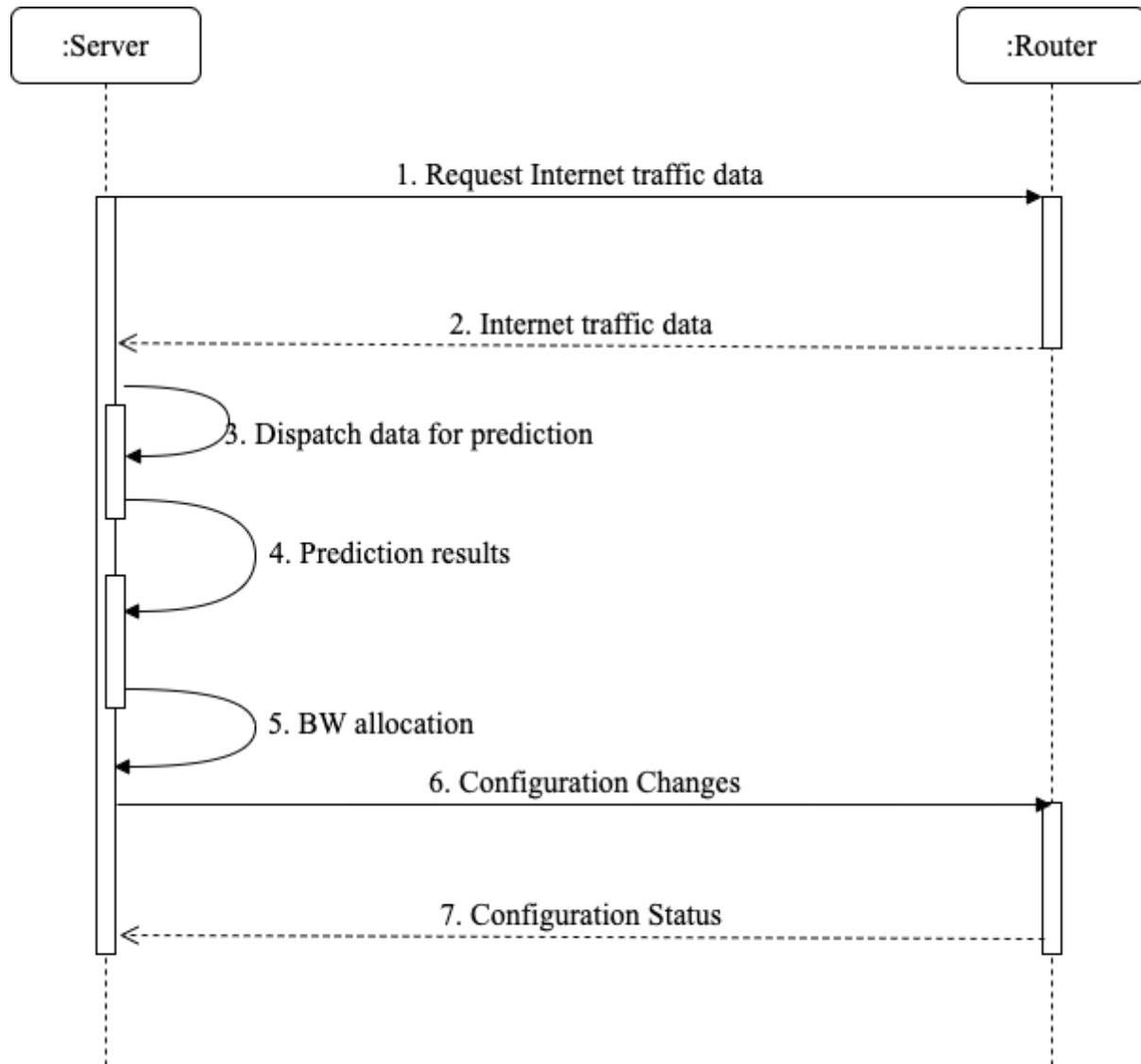


Figure 5.1: The Sequence Diagram for the operations

The Router and Server Operational Model

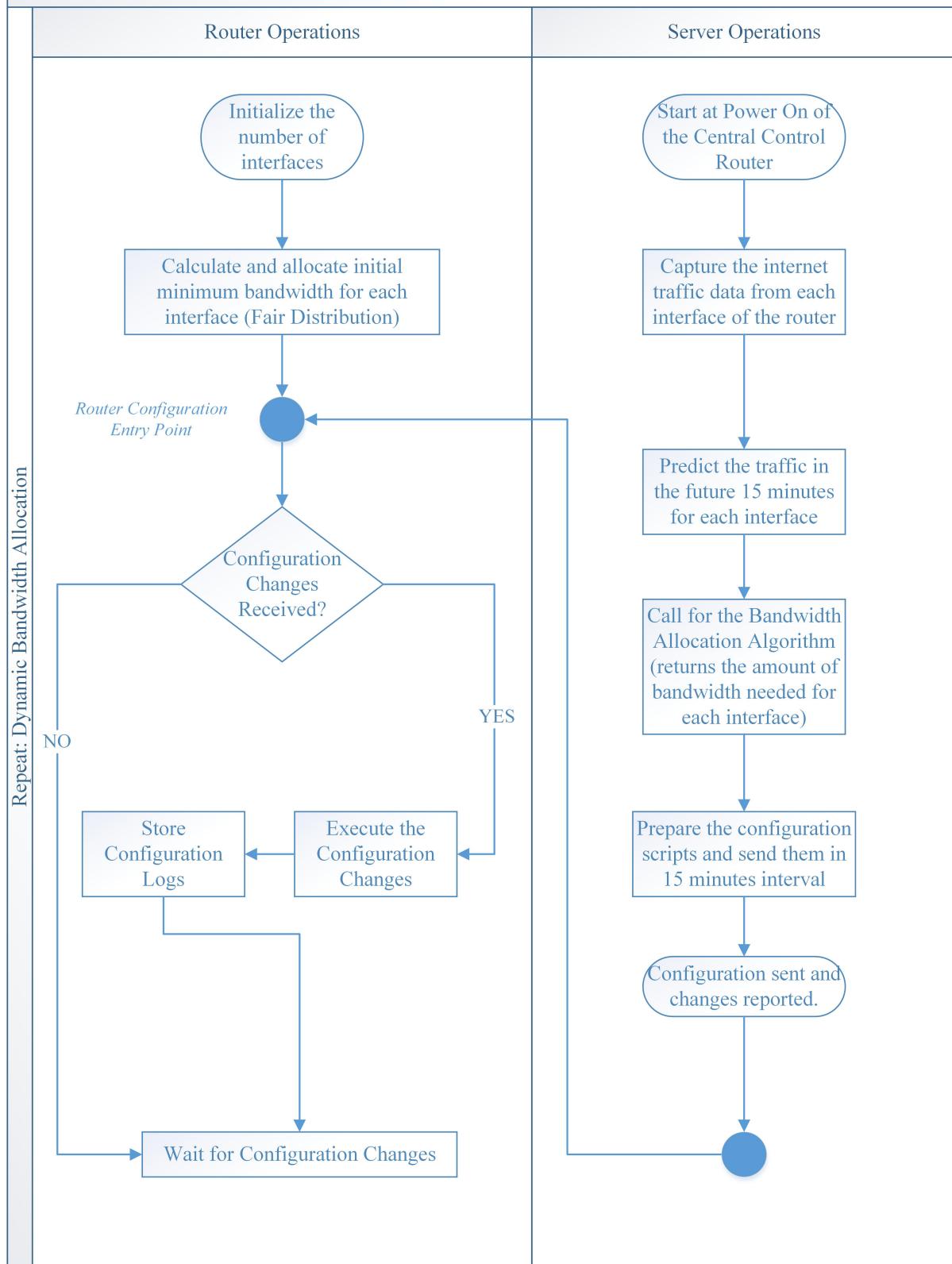


Figure 5.2: The Router and Server Operational Model

5.2 Architecture Design

The architecture of the system, Figure 5.3, consists of four major layers. The Internet traffic data generation layer is where all the devices sharing the data over the Internet are found. After the devices generate the data, packet initiation and delivery layer that consists of switches that are responsible for packet delivery comes in, then the routing services access layer follows which is responsible for routing the data over the IP layer. The fourth layer is where this work was focused at. It is where the network operations, traffic data analysis and bandwidth allocation are to be done.

With this configuration, the application of SDN would be realized in case there is more than a single router at the core of the network. This architecture ensures that the network core operations are focused on the data that is generated by at the interfaces at the core (Layer 4).

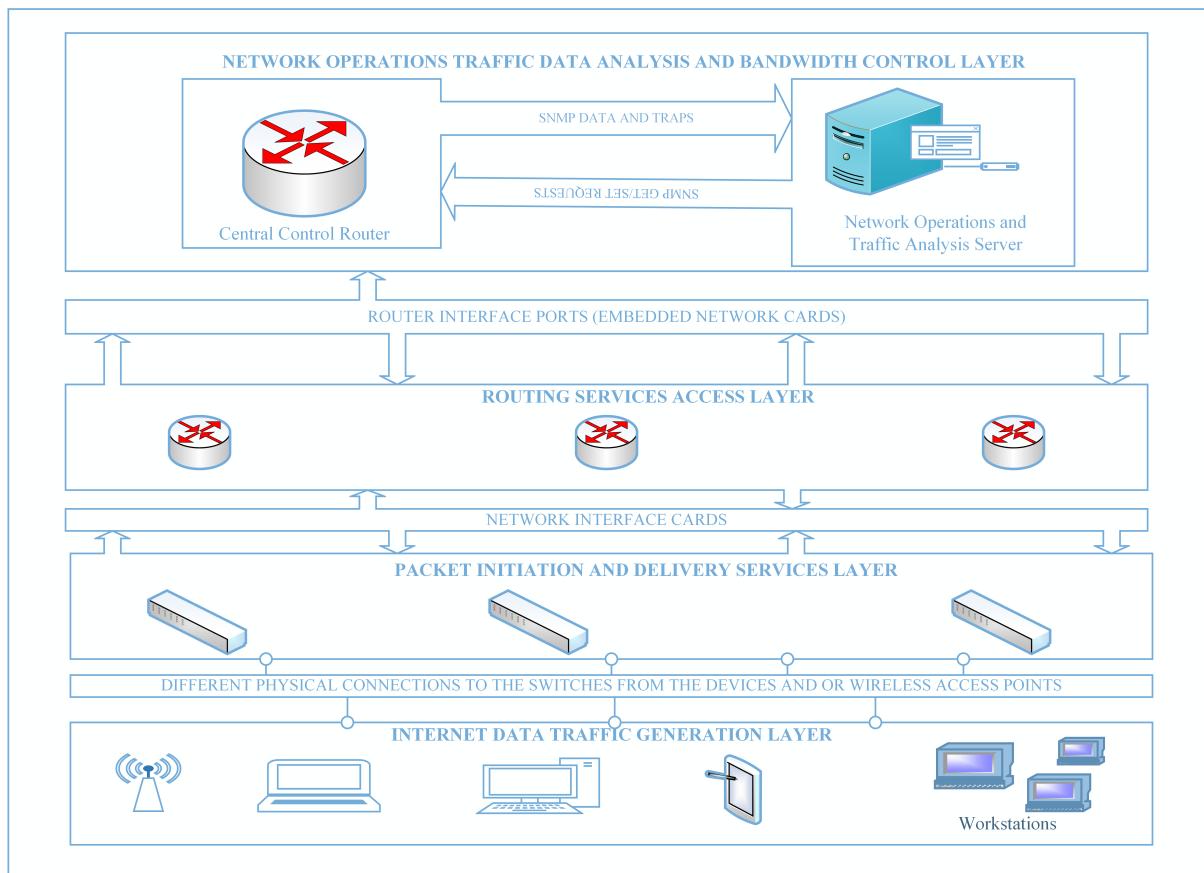


Figure 5.3: The Architectural Design of the Operational Environment

Chapter 6

Conclusion

In conclusion, the contributions of this work are summarized and some important observations are noted. Some areas that can extend this work beyond the stated limitations are also suggested.

6.1 Contributions

The contributions of this work are summarized as follows:

1. The design and implementation of traffic learning model

The internet traffic data learning model described herein considers the stochastic nature of the internet traffic data. Therefore, in its uniqueness it is not limited to working with the data from the case study only. It can—with very few modifications—be implemented in other networks and be expected to work fine.

2. The development of the data driven bandwidth allocation algorithm

The bandwidth allocation algorithm suggested and developed in this work relied on the results obtained from learning the traffic data, and forecasting some of the data against what was produced. This led to a the considerations of setting the mean the predictions at one subnetwork against others as the measure of the amount of bandwidth to be allocated.

6.2 Challenges and Limitations

For this work to be useful, the following need to be addressed:

- (i) The challenge in proving that the throughput of the network was improved by employing the algorithms developed in this work. To prove that throughput would improve, this work required feeding the network implemented in NS3 with the data with which the models were developed from. Having spent little time with NS3 and NS3 being the discrete event simulation software, it was not possible to feed in the data and compare between the outputs. However, the network topology was implemented in NS3 for anyone in need of investigating this to do just that.

- (ii) Vendor specific operations: There are different network equipment vendors in the market. This limits the development of tools that address configurations for each vendor. Therefore, this work being one of those whose end product is associated with interactions with the configurations in the network equipment, it becomes difficult to implement in environments that would contain the equipment from different vendors.
- (iii) The use of traffic shapers and traffic schedulers: In addressing problems associated with the internet resources allocation the role of traffic shapers and schedulers has not been left behind. The writers in [11] went further to explain the roles of each. Also the writers in [12] proved the roles of the two with graphs that show the mathematical correlation between the service curves and the arrival curves. It is with this consideration that the integrity of internet traffic data obtained from the core could be misleading, hence limiting the performance of the algorithm.

6.3 Conclusion and Recommendation

6.3.1 Conclusion

The implementation of any kind of network should consider the Quality of Experience with which the users will be provided. One way to look at it from the engineering point of view is to analyze the QoS of the network by considering throughput, packet delays and delay jitters in a timely manner. Employing the network administrators is still what is being done to optimize the QoS. Hence, any changes made to the network—despite being necessary—aren't done in a timely manner.

Therefore, throughout this report, the potential of employing machines (servers)—ML—for the dynamic allocation of bandwidth has been discussed in details while proposing the need for having a one stop point for all the network elements configurations—SDN. Hence, "IP Networks QoS Optimization Using ML Powered SDN" remains an important work that will solve most of the associated problems in network resources allocation.

6.3.2 Recommendations

Throughout this work, there have been a number of technologies that have been discussed. These technologies are not available for a student reach if one focuses on only what is being taught in class. Therefore, it is my advice that:

1. Machine Learning should be taught in the early days of the TE course (2^{nd} year should be fine)
2. Algorithms and Algorithmic Complexities should be made as one of the optional courses and taught with respect to real computing problems, and
3. The students should be exposed to computer networking research area in the early days of computer networking courses.

It is also necessary to note that this work sets the foundation on which other projects could come from. With respect to the limitations addressed, the following would make better project ideas:

1. The investigation of throughput of the network in a data driven bandwidth allocation should be of utmost importance.
2. The development of the middleware to address the associated complexity with the configuration of equipment from different vendors.
3. The investigation and analysis of traffic shapers and traffic schedulers limitations in data driven bandwidth allocation algorithm performance.

References

- [1] Mohamad Awad, Mohammed El-Shafei, Tassos Dimitriou, Yousef Rafique, Mohammed Baidas, and Ammar Alhusaini. Power-efficient routing for sdn with discrete link rates and size-limited flow tables: A tree-based particle swarm optimization approach: A pso-based and power-efficient routing (psopr) algorithm. *International Journal of Network Management*, page e1972, 01 2017.
- [2] Raphael Cohen-Almagor. Internet history. *International Journal of Technoethics*, Vol. 2:45–64, 04 2011.
- [3] Internet Society. Bandwidth management: Internet society technology roundtable series. pages 1–2, 2012.
- [4] Purna Chandra Sethi and Prafulla Behera. Network traffic management using dynamic bandwidth on demand. *International Journal of Computer Science and Information Security*, 15:369–375, 06 2017.
- [5] Joberto Martins. *Quality of Service in IP Networks*. 01 2005.
- [6] Chengzhi Li, Almut Burchard, and Jörg Liebeherr. A network calculus with effective bandwidth. *IEEE/ACM Trans. Netw.*, 15(6):1442–1453, December 2007.
- [7] Sumit Badotra. A review paper on software defined networking. *International Journal of Advanced Computer Research*, 8, 03 2017.
- [8] Purna Chandra Sethi and Prafulla Behera. Network traffic management using dynamic bandwidth on demand. *International Journal of Computer Science and Information Security*, 15:369–375, 06 2017.
- [9] Emerging Technologies. Bandwidth Scaling. <http://etinc.com/169/Bandwidth-Scaling>, 2018. [Online; accessed 10-Jan-2020].
- [10] History of network traffic models. https://en.wikipedia.org/wiki/History_of_network_traffic_models. Accessed June 20, 2020.
- [11] Victor Firoiu, Jean-Yves Le Boudec, Don Towsley, and Zhi-Li Zhang. Theories and models for internet quality of service. *Proceedings of the IEEE*, 90:1565 – 1591, 10 2002.

- [12] El-Bahlul Fgee, William Phillips, William Robertson, A. Elhounie, and Adel Smeda. A scalable mathematical qos model for ip networks. 04 2008.

Appendices

Appendix A

UDSM LAN Topology Implementation Code

```
1 // C++ specific libraries:  
2 #include <string>  
3  
4 #include "ns3/core-module.h"  
5 #include "ns3/network-module.h"  
6 #include "ns3/internet-module.h"  
7 #include "ns3/point-to-point-module.h"  
8  
9 using namespace ns3;  
10  
11 int main (int argc, char *argv[]) {  
12  
13     // Declare some constants  
14     const int ROUTERS = 3;  
15     const int DISTRIBUTION_SWITCHES = 5;  
16     const int ACCESS_SWITCHES = 6;  
17  
18     // Create the Network nodes  
19     NodeContainer routers, distribution_switches, access_switches;  
20  
21     routers.Create(ROUTERS);  
22     distribution_switches.Create(DISTRIBUTION_SWITCHES);  
23     access_switches.Create(ACCESS_SWITCHES);  
24  
25     // Install the internet stack on each network node above  
26     InternetStackHelper stack;  
27     stack.Install(routers);  
28     stack.Install(distribution_switches);  
29     stack.Install(access_switches);  
30  
31     // Create a point-to-point helper
```

```

32 PointToPointHelper p2p;
33
34 // Create an address helper
35 Ipv4AddressHelper address;
36
37 // Create the node containers for the subnets
38 NodeContainer subnet1;
39 NodeContainer subnet2, subnet3, subnet4;
40 NodeContainer subnet5, subnet6, subnet7;
41 NodeContainer subnet20, subnet30, subnet40, subnet50, subnet60,
42     subnet70;
43
44 // Subnet 1 Configuration
45 subnet1.Add(routers.Get(0));
46 subnet1.Add(distribution_switches.Get(0));
47
48 NetDeviceContainer subnet1Devices = p2p.Install(subnet1);
49
50 address.SetBase("10.1.1.0", "255.255.255.0");
51
52
53 // Subnet 2 Configuration
54 subnet2.Add(distribution_switches.Get(0));
55 subnet2.Add(distribution_switches.Get(1));
56 NetDeviceContainer subnet2Devices = p2p.Install(subnet2);
57 address.SetBase("10.1.2.0", "255.255.255.0");
58 Ipv4InterfaceContainer subnet2Interfaces = address.Assign(
59     subnet2Devices);
60
61 // Subnet 20 Configuration
62 subnet20.Add(distribution_switches.Get(1));

```

```

63    subnet20.Add(access_switches.Get(0));
64    NetDeviceContainer subnet20Devices = p2p.Install(subnet20);
65    address.SetBase("10.1.20.0", "255.255.255.0");
66    Ipv4InterfaceContainer subnet20Interfaces = address.Assign(
67        subnet20Devices);
68
69 // Subnet 3 Configuration
70 subnet3.Add(distribution_switches.Get(0));
71 subnet3.Add(distribution_switches.Get(2));
72 NetDeviceContainer subnet3Devices = p2p.Install(subnet3);
73 address.SetBase("10.1.3.0", "255.255.255.0");
74 Ipv4InterfaceContainer subnet3Interfaces = address.Assign(
75        subnet3Devices);
76
76 // Subnet 30 Configuration
77 subnet30.Add(distribution_switches.Get(2));
78 subnet30.Add(access_switches.Get(1));
79 NetDeviceContainer subnet30Devices = p2p.Install(subnet30);
80 address.SetBase("10.1.30.0", "255.255.255.0");
81 Ipv4InterfaceContainer subnet30Interfaces = address.Assign(
82        subnet30Devices);
83
83 // Subnet 4 Configuration
84 subnet4.Add(distribution_switches.Get(0));
85 subnet4.Add(distribution_switches.Get(3));
86 NetDeviceContainer subnet4Devices = p2p.Install(subnet4);
87 address.SetBase("10.1.4.0", "255.255.255.0");
88 Ipv4InterfaceContainer subnet4Interfaces = address.Assign(
89        subnet4Devices);
90
90 // Subnet 40 Configuration
91 subnet40.Add(distribution_switches.Get(3));
92 subnet40.Add(access_switches.Get(2));

```

```

93 NetDeviceContainer subnet40Devices = p2p.Install(subnet40);
94 address.SetBase("10.1.40.0", "255.255.255.0");
95 Ipv4InterfaceContainer subnet40Interfaces = address.Assign(
96   subnet40Devices);
97
98 // Subnet 5 Configuration
99 subnet5.Add(distribution_switches.Get(0));
100 subnet5.Add(distribution_switches.Get(4));
101 NetDeviceContainer subnet5Devices = p2p.Install(subnet5);
102 address.SetBase("10.1.5.0", "255.255.255.0");
103 Ipv4InterfaceContainer subnet5Interfaces = address.Assign(
104   subnet5Devices);
105
106 // Subnet 50 Configuration
107 subnet50.Add(distribution_switches.Get(4));
108 subnet50.Add(access_switches.Get(3));
109 NetDeviceContainer subnet50Devices = p2p.Install(subnet50);
110 address.SetBase("10.1.50.0", "255.255.255.0");
111 Ipv4InterfaceContainer subnet50Interfaces = address.Assign(
112   subnet50Devices);
113
114 // Subnet 6 Configuration
115 subnet6.Add(distribution_switches.Get(0));
116 subnet6.Add(distribution_switches.Get(5));
117 NetDeviceContainer subnet6Devices = p2p.Install(subnet6);
118 address.SetBase("10.1.6.0", "255.255.255.0");
119 Ipv4InterfaceContainer subnet6Interfaces = address.Assign(
120   subnet6Devices);
121
122 // Subnet 60 Configuration
123 subnet60.Add(distribution_switches.Get(5));
124 subnet60.Add(access_switches.Get(4));
125 NetDeviceContainer subnet60Devices = p2p.Install(subnet60);
126 address.SetBase("10.1.60.0", "255.255.255.0");

```

```

123     Ipv4InterfaceContainer subnet60Interfaces = address.Assign(
124         subnet60Devices);
125
126     // Subnet 7 Configuration
127     subnet7.Add(distribution_switches.Get(0));
128     subnet7.Add(distribution_switches.Get(6));
129     NetDeviceContainer subnet7Devices = p2p.Install(subnet7);
130     address.SetBase("10.1.7.0", "255.255.255.0");
131     Ipv4InterfaceContainer subnet7Interfaces = address.Assign(
132         subnet7Devices);
133
134     // Subnet 70 Configuration
135     subnet70.Add(distribution_switches.Get(6));
136     subnet70.Add(access_switches.Get(5));
137     NetDeviceContainer subnet70Devices = p2p.Install(subnet70);
138     address.SetBase("10.1.70.0", "255.255.255.0");
139     Ipv4InterfaceContainer subnet70Interfaces = address.Assign(
140         subnet70Devices);
141
142     // Run the simulator
143     Simulator::Run();
144     Simulator::Destroy();
145
146     return 0;
147 }
```

Appendix B

Traffic Data Generation and Cleaning Code

```
1 import numpy as np
2 import pandas as pd
3 from datetime import datetime
4 import time
5 import csv
6
7
8 def data_generator():
9     """
10         This function specifically generates some data points for
11         processing
12
13     Some generalized considerations:
14         Average delay = 350ms; hence a delay between 250 and 400
15         Average flows = 19.9k/s; hence 19k abd 25k
16
17     start = time.time()
18     start_time = []
19     transmission_duration = []
20     packets = []
21
22     i = 0
23     while (time.time() - start) < 300:
24         start_time.append(datetime.now())
25         packets.append(np.random.randint(19000, 25000))
26         transmission_time = 0.001*np.random.randint(250, 400)
27         time.sleep(transmission_time)
28         transmission_duration.append(transmission_time)
29         i += 1
30
```

```

31     print("Generated " + str(i) + " data points")
32
33     return zip(start_time, transmission_duration, packets)
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59

```

```

print("Generated " + str(i) + " data points")

return zip(start_time, transmission_duration, packets)

def save_data(to_file_name, data_to_save):
    with open(to_file_name, "w", newline='') as csv_file:
        file_headers = ["Start_Time", "Transmission_Duration", "Packets"]
        writer = csv.DictWriter(csv_file, fieldnames=file_headers)
        writer.writeheader()

        for start_time, transmission_duration, packets in data_to_save:
            writer.writerow({"Start_Time": start_time,
                            "Transmission_Duration": transmission_duration,
                            "Packets": packets})

def clean_data(from_file_name, to_file_name):
    data_to_clean = pd.read_csv(from_file_name) ["Packets"]
    print(data_to_clean.head())

    with open(to_file_name, 'w') as csv_file:
        file_headers = ["packets_1", "packets_2", "packets_3", "packets_4"]
        writer = csv.DictWriter(csv_file, fieldnames=file_headers)
        writer.writeheader()

        for index in range(0, (len(data_to_clean)-3)):
            writer.writerow({"packets_1": data_to_clean[index], "packets_2": data_to_clean[index+1],
                            "packets_3": data_to_clean[index+2], "packets_4": data_to_clean[index+3]})

            index += 1

```

```
60  
61  
62 # Main Program Operations  
63 save_data(data_generator(), "generated_data.csv")  
64 clean_data("generated_data.csv", "generated_clean_data.csv")  
65  
66  
67
```

Appendix C

Traffic Data Learning, Prediction, and Visualization Code

```
1 import pandas as pd
2 from patsy import dmatrices
3 import numpy as np
4 import statsmodels.api as sm
5 import matplotlib.pyplot as plt
6 import time
7
8
9 # Create a pandas DataFrame for the counts data set.
10 df = pd.read_csv('clean_data.csv', header=0, infer_datetime_format=True)
11
12 print(df.head())
13 print(df.index)
14
15 time.sleep(30)
16
17
18 # Create the training and testing data sets.
19 mask = np.random.rand(len(df)) < 0.8
20 df_train = df[mask]
21 df_test = df[~mask]
22 print('Training data set length=' + str(len(df_train)))
23 print('Testing data set length=' + str(len(df_test)))
24
25 print(df_train.head())
26 print(df_test.head())
27
28
29 # Setup the regression expression in patsy notation. We are telling
   patsy that packets_4 is our dependent variable and
```

```

30 # it depends on the regression variables: packets_1, packets_2, and
31 expr = """packets_4 ~ packets_1 + packets_2 + packets_3"""
32
33 # Set up the X and y matrices
34 y_train, X_train = dmatrices(expr, df_train, return_type='dataframe',
35                               )
36
37 # Using the statsmodels GLM class, train the Poisson regression
38 # model on the training data set.
39
40 poisson_training_results = sm.GLM(y_train, X_train, family=sm.
41                                    families.Poisson()).fit()
42
43 # Print the training summary.
44 print(poisson_training_results.summary())
45
46 # .summary_frame() returns a pandas DataFrame
47 predictions_summary_frame = poisson_predictions.summary_frame()
48 print(predictions_summary_frame)
49
50 predicted_traffic= predictions_summary_frame['mean']
51 actual_traffic = y_test['packets_4']
52
53 # Plot the predicted traffic versus the actual traffic for the test
54 # data.
55 fig = plt.figure()
56 fig.suptitle('Predicted versus Actual traffic')
57 predicted, = plt.plot(X_test.index, predicted_traffic, 'go-', label=
58                      'Predicted Traffic')

```

```
57 actual, = plt.plot(X_test.index, actual_traffic, 'ro-', label='
    Actual Traffic')
58 plt.legend(handles=[predicted, actual])
59 plt.savefig("predict_actual.png")
60 plt.show()
61
62 # Show scatter plot of Actual versus Predicted Traffic
63 plt.clf()
64 fig = plt.figure()
65 fig.suptitle('Scatter plot of Actual versus Predicted counts')
66 plt.scatter(x=predicted_traffic, y=actual_traffic, marker='.')
67 plt.xlabel('Predicted counts')
68 plt.ylabel('Actual counts')
69 plt.savefig("predict_scatter.png")
70 plt.show()
71
72
```

Appendix D

Project Budget for Semester II

S/N	Item	Cost (TShs.)
1	Internet	150,000
2	Stationery	50,000
3	Emergency	20,000
	Total	220,000