ADDIS ABABA UNIVERSITY
ADDIS ABABA INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Performance Analysis of POCO Framework Under Failure Scenario in SDN-Enabled Controller Placement

by

Sefinew Getnet

Advisor: Yalemzewd Negash(PHD)

A Thesis Submitted to the School of Electrical and Computer Engineering, Addis Ababa Institute of Technology, Addis Ababa University, in Partial Fulfillment of the Requirements for the Degree of Masters of Science in Computer Engineering.

February 2023

Addis Ababa, Ethiopia

Declaration

I, the undersigned, declare that this thesis titled "Performance Analysis of POCO Framework Under Failure Scenario in SDN-Enabled Controller Placement" and the work presented in it are my own.

I affirm that this work was done wholly or mainly while in contention for a master's degree at this university only.

The source is at all times specified where I have quoted from the work of others. With the exception of such quotations, this thesis is entirely my own work; and I have acknowledged all main sources of my thesis work for help.

Student Name:_____Sefinew Getnet_____

Signature:_____

Date_____

ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

The undersigned have examined the thesis titled:

Performance Analysis of POCO Framework Under Failure Scenario in SDN-Enabled Controller Placement

Presented by Sefinew Getnet, a candidate for the degree of Master of Science and hereby certify that it is worthy of acceptance.

Approval by Board of Examiners:

Dr. Bisrat Derebssa	_____	_____
Dean, SECE, AAiT	Signature	Date
Dr. Yalemzewd Nagash	_____	_____
Advisor	Signature	Date
Dr. Sosina Mengestu	_____	_____
Internal Examiner	Signature	Date
Dr. Surafel Lemma	_____	_____
External Examiner	Signature	Date

February 2023

Addis Ababa, Ethiopia

Abstract

One of the main causes of the demise of deep-rooted networks has been the explosive expansion of internet service providers and current data traffic. Nowadays, the most recent advancements in networking is the introduction of software-defined networking (SDN), which emerged for rebuilding and modifiable features. The core idea of SDN is the separation of the control and data planes. When controlled by an SDN controller, SDN offers advantages in terms of flexibility, manageability, and efficiency in contrast to traditional networks. This thesis work offered the Performance analysis of POCO Framework under Failure Scenario to address the challenges like rigidity, uncontrollability, and inadequately used network resources for the adoption of SDN networks. Furthermore, the network dataset of AAU was encoded, examined and the selected platform for determining the optimal number of controllers and their ideal location in the Metropolitan Area Network (MAN) were applied with the goal of minimizing latency and cost of controllers.

The optimization is done in two scenarios, (I) controller placement using failure free and (II) when either nodes or controllers fail, using the Pareto Optimal Controller Optimization Placement (POCO) framework. First, the controller's placement in the aforementioned scenarios for different numbers of K controllers was found and investigated. Following this, results of altered metrics like latencies of controller-to-controller(C2C) and node-to-controller(N2C). fault tolerance (for either node or controller failures), and controller imbalance for both the aforesaid scenarios were conducted accordingly. From the results, it is recommended that the minimum number of controllers required to adopt SDN in AAU-MAN be two controllers, considering the resiliency of controllers placed at Sidist kilo and Arat kilo. Also, we conclude that the POCO performance analysis under the failure-free scenario-based experiment showed minimum N2C latency compared to the worst-case scenario (node failure and up to K-1 controller failure). But showed minimum node failure C2C latency compared to failure free scenario.

Key-words: Software Defined Network, Controller Placement, Latency, load imbalance, resiliency, POCO

Acknowledgements

I want to start by expressing my gratitude and glorification to the Almighty God, who is the magnificent source of my strength. He gave me wonderful health, sharp insight, and direction in all aspects of my life.

Next, I want to express my sincerest thanks to my advisor, Dr. Yalemzewd Negash, who helped me with this senior thesis.

My thankfulness also goes out to my family and my outstanding friends for their valuable input, brilliant ideas, and unwavering support throughout the writing of my thesis.

Last but not least, I want to express my appreciation to everyone who helped me, directly or indirectly, finish the research study.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Symbols

AAU	Addis Ababa University
AvgL	Average Latency
C2C	Controller to Controller
CGLCPP	Capacitated Global Latency Control Placement Problem
CLI	Command Line Interface
CPP	Controller Placement Problem
DBCP	Density Based Controller Placement
GA	Genetic Algorithm
ILP	Integer linear programming algorithm
IP	Internet Protocol
KDE	kernel density estimation
MAN	Metropolitan Area Network
MIP	Mixed Integer Program
MOGA	Multi-Objective Genetic Algorithm
N2C	Node to Controller
PAM	Partitioning Around Medoids
PITS	Pareto Integrated Tabu Search
POCO	Pareto Optimal Controller Placement
POM	Parameter optimization Model

PSA	Pareto Simulated Annealing
PSO	Particle Swarm Optimization
QoS	Quality of Service requirements
RP	Random Placement
SANReN	South African Research Network
SCCPPM	Set Covering Controller Placement Problem Model
SDN	Software Defined Networking
SDNCPP	Software Defined Networking Controller Placement Problem
SDWAN	Software Defined Wide Area Network
WAN	Wide Area Network
ZB	Zeta Byte

CHAPTER 1

Introduction

1.1 Background

The concept of Software-defined networking (SDN) was first introduced in 2010 as a new networking paradigm that aims to simplify the control and management of a computer network environment [6][7][8]. It is a new networking trend that has enormous potential to promote modernization using configurable networks. This unique evolutionary paradigm for network design separates the control plane and data plane, and a logically centralized controller decides on routing on behalf of forwarding elements.

The controling role of individual nodes will be replaced by centralized controllers. In this regard the network oprating system (NOS) performes key functionalites which include gathering of information, creation of an abstract model for the topology and fasilitation of the controllers which hosts the application. Such information is used by the controller to manage in flow of packets and dynamic allocation of network resources [8]. Open Flow protocol was introduced ten years back to serve as a communication protocol between the control plane and the data plane. After its inrtoduction enormous improvements has been done for the protocol.

In the prevailing era, SDN is at the level of maturity in order to be deployed in any netwok which is serving operational network systems. However, the deployment and usage of SDN on existing netwok systems triggers enormous questions which are related to scalability, reliablity and security isssues.

The one and the first common problems associated with Software-defined networking (SDN) is (I) determining the number of SDN controllers, and the second is (II) the placement of the controllers. Therefore, this thesis focuses on the adoption of Software

defined networking enabled controller placement problems (SDN-CPP) in Performance analysis of POCO Framework under Failure Scenario using Addis Ababa University network dataset.

1.2 Motivation

Currently, Addis Ababa University networks' ICT infrastructure and service leaders are responsible for the visual sense of a network configuration throughout the whole wide range of network applications as per the network administrator of AAU networks' coordinator's genuine evidence. There are so many unfair factors to achieve the expected quality application services. The motivations behind why we used network dataset of AAU in this thesis work therefore, are the unique features within AAU campus networks like it has many branch campuses around fifteen, which are locating at different geographical landscapes. The other features are enormous users, high network traffic, massive infrastructures (buildings which are growing) and many complex networking devices. There are also many services hosted by AAU networks like research management systems, GIS, SIMS, DNS, Proxy, PRTG, UTM and WoredaNet... etc. (as per ICT infrastructure and service leaders countered to us).

Indeed, the aforementioned factors are difficult to achieve the due quality of service (QoS) because; (I) the complexity and the static nature of current network devices do not permit detailed control-plane configuration, and both the control and data planes are embedded within the same network node (hardware) that provided by the manufacturer and cannot be customized. (II) With the continuous evolution of IT, the internet has acquired large-scale infrastructure which affects the overall people's working and living style. (III) The huge amount of delays and latency which are mostly time-consuming and fault driven by the traditional networks, contribute negatively to the overall network performance [6].

On the other hand, the control plane in conventional networks is in charge of configuring and programming the channels for data flows [7]. Data forwarding then makes advantage of this control information. This technique is not dynamic, though, because once the control information is defined, it cannot be updated to add or delete networking devices without redesigning the system. The traditional networking strategy is less responsive to shifting traffic dynamics as a result of this limitation.

Consequently, one of the objectives of emerging networks is to offer operators tools that make network management simpler, shorten the time required to configure devices, and reduce configuration errors.

To accommodate the anticipated demand, the networks of the future must be adaptable and dynamic.

For instance, the overall IP traffic was increase at a compound annual growth rate (CAGR) of 24% from 2016 to 2021, according to the Cisco visual networking index [7].

Between 2016 and 2021, peak Internet traffic increased by a factor of 4.6, while average Internet traffic increased by a factor of 3.2. By 2021, smartphone traffic is dominating PC traffic where the data load might be as high as 278 EB each month, or 3.3 ZB per year (ZB; 1000 Exabyte's [EB]). It demonstrates unambiguously the requirement for networks to offer enough capacity to carry such massive amounts of data. However, in order for data transfers to start when necessary, they also need to be dynamic and adaptable.

1.3 Problem Statement

Traditionally, in the global world, the data and the control planes in the Ethernet networking devices (most of the communication devices) have been tied together. This means, the fundamental operating system and its features with the provided hardware are implemented in a single device. Therefore, network devices, such as switches, routers, firewalls, etc., are built with the intelligence of handling traffic relative to the neighboring devices. This makes the intelligence distributed and scattered in the network. In addition, most of the network devices are Command Line Interface (CLI) based and configuration is done separately per device, making configuration slow and prone to errors. This prevents the networking industry of responding quickly to feature requests or newly innovative management capabilities [9]. As the number of users and buildings increases, servers must quickly handle a huge volume of access requests. Data center networks are the main focus of traditional SDN controller deployment techniques. However, in the context of geographically dispersed heterogeneous multi-domain networks, the issue of virtualized controller placement for multi-domain transport networks needs to be addressed in this situation, edge data centers have given network operators the ability to position virtualized controller instances closer to users in addition to offering more potential places to install controllers. Indeed, it is challenging to manage and offer all necessary applications in accordance with the classical networking pattern given the aforementioned difficulties. Allocating a single SDN controller to the networking pattern provides certain advantages, such as allowing the controller to see the entire network, but there are also many drawbacks. Redundancy, load balancing, and network traffics are among the drawbacks of employing a single SDN controller in the given networking topology; a controller may get overloaded and may not know where to place it. This location issue typically arises when applying to

distinct geographic regions that are distant from one another.

As a result, managing network performance optimally is challenging when it is not assisted by automatic configuration. Even though using numerous controllers to address the single controller issue can reduce latency, boost scalability, and improve fault tolerance, it is not without its own challenges as understood from others work. Placing numerous SDN controllers and the minimum number are two of the difficulties In addition, although previous literatures conducted their research works on POCO framework, no one can conducted a research on the network effects and clear comparative analysis in the presence of failure scenario. Therefore, we go to investigate the performance analysis of POCO framework under failure situations in the controller placement and number determination by using available dataset's of AAU.

Consequently, the inquiry for the research is:

RQ1. In terms of controller placement, how many SDN controllers are needed for a given network dataset?

RQ2. Where SDN controllers should be ideally positioned within the the given network dataset?

RQ3. What incidents are observed in the performance of POCO framework under normal and failure scenarios?

The network properties in the network architecture and the metrics picked to measure when selecting where to deploy the controllers determine the solution to this research topic.

1.4 Objectives

1.4.1 General Objective

The general aim of this research is to conduct performance analysis of POCO framework under failure situations in the SDN-enabled controller placement and number determination for the effective usage of quality of service in the given network dataset.

1.4.2 Specific Objective

The specific objectives of this research are:

- To prepare the network connectivity for the creation of network dataset
- Encode the prepared topology in simulator format

- Test and compare the consequence of the network by applying performance metrics to the selected simulation platform
- To assess the POCO performance under failure and normal scenarios
- To determine the optimum number, location, and latency (N2C, C2C),
- To examine the load imbalance of the assigned controllers

1.5 Research Methodology

To conduct this research on the controller placement with regard to the network topology, we have mainly relied on basic research methods like literature reviews, algorithm and simulation platform identification, and we also figured out the possible state of affairs of controller placement scenarios. The following technical steps are outlined while conducting this research:

- ***Literature review***: From the very beginning, we have been reviewing numerous articles, and sites and journals related to SDN-CPP. The approaches used by previous authors that are provided on SDN networking are crucial for carrying out this thesis work.
- ***Data collection***: input datasets have been collected to test the effect or performance analysis of POCO simulation platform under different performance measures and scenarios in to the adoption of SDN controller placement.
  - Some sort of relevant information/data about the network dataset for this research is well collected so far. We had an evident from the IT infrastructure and networking team leader about the existing nature of network behavior on AAU campus networks. And the ICT office gave to us uncourageous ideas on the shifting of the existing network of the institution to software defined networking to have programmable and scalable feature of the network infrastructure. We have collected the existing architecture of the AAU network structure in advance. As we understand from the documentation, the majority of the networking nodes are connected via VPN from Ethio-Telecom, and the remaining seven nodes are directly connected from the main campus (Sidist Kilo). In chapter four, table 4.4 presents a short-term summary of the dataset metaphors.
- ***Design and Implementation***: In addition to the current metrics produced by [7] [9], our proposed model architecture is based on the POCO framework, which supports an exhaustive search algorithm. And then the proposed work is tested by

POCO − GUI considering both the failure and failure free of either nodes or SDN controllers.

- ***Result and Evaluation***: In order to clearly highlight the performance measures of the proposed model, we employ our evaluation metrics, including latency, load imbalance, and fault tolerance. The experiment data set was obtained from the datacenter of AAU by network administrator team leader.

1.6 Scope of the study

As stated in problem statement 1.2, there are several possible ways to be considered while optimizing SDN-CPP using appropriate platform. However, the scope of this research is limited to determining the number and placement of controllers, taking into account latency, load balancing and resiliency. Considering failure and failure-free scenarios of POCO.

1.7 Contributions

The present research work has many contributions. Followings are among the benefits of SDN-enabled controller placement for any organization to consider, are the least not the last.

- More Effective Network Administration. Real-time visibility into network performance is provided via SDN
- Cost-Savings. The total cost of necessary controllers and the maintenance cost of the entire network architecture can both be reduced as long as SDN controllers are placed optimally and in the proper numbers.
- Accelerated scalability. When necessary, it can install or remove new network infrastructures. Promising pattern for the next-generation networking paradigm
- To have an insight into how companies can adopt SDN for their network topology
- Opens good situations for further research and makes it available for academic motivations
- Recognize and find new test beds for a suitable algorithm that can optimize the SDN-CPP.
- Insight into what experiments can be conducted using POCO network simulator platform under failure and normal scenarios used in the adoption of SDN-enabled controller placement regards.

1.8 Paper Organization

The rest of this thesis work is organized as follows: Firstly, related works of software-defined networking, which use different algorithms and platforms in SDN implementation, are presented in chapter 2. Next to this, Chapter 3 explores the theoretical concepts of SDN and its fundamental elements. Chapter 4 presents an overview of the proposed architecture, system model, and framework. The simulation results and complete solution are described and discussed in Chapter 5. Finally, concluding remarks, recommendations, and possible future works are presented in Chapter 6.

CHAPTER 2

Literature Review

In this section, enormous previously published research works regarding SDN-enabled controller placement have been reviewed.

2.1 Related Works

So far, many researchers have conducted dissimilar research on controller optimization and placement in software-defined networks at different times with differing metrics. On this thesis, the controller placement problem (CPP) in SDN can be reviewed or discussed under the following categories.

2.1.1 Controller Placement Based on Network Reliability and Scalability

In Software Defined Networking (SDN), many challenges remain to be investigated. In [10] and [11], the problems of scalability and robustness for the control plane, virtualization communication consistency, programming language, controller placement and so on are crossed to be solved. The authors focused on the issue of controller placement problems by considering shortest path and multipath network reliability cases. They also proposed the clustering based global optimization algorithm and the local optimization algorithm based on the greedy algorithm. Finally, even if it has verified the performance of their proposed algorithms based on the Internet2 OS3E and Internet Topology Zoo, the paper did not consider multiple objectives for optimization of controller placement.

Mamushiane, L. et al. [12] have examined the fundamental issues of controller placement in SDN. In order to investigate the number and locations of controllers in software defined network, the authors have used mathematical models including the silhouette, PAM, and Gap statistics approaches. The two unsupervised machine learning algorithms, silhouette, and Gap statistics, were applied in the given topology to optimize the number of controllers, whereas the authors used the classical algorithm PAM to determine the

optimal placement to install the controllers. The authors' result explored that a large soft idle timeout and polling frequency reduce the overall control plane overhead, and they argued that the solution to controller placement is dependent on the topology, so the findings presented in this work only apply to the SANREN topology. Although their work takes in to account the resiliency and control plane overhead metrics as well as balancing the switch to controller placement, there are still limitations, like using constant bandwidth for all connection links, the security aspects of SDN controllers to prevent the single point of failure, and not having a dynamic control traffic load balancing application based on current switch resources (RAM, CPU, and storage). Their work is extended and solved in [13] and [1] by introducing the use of an interdependence graph and cascading failure analysis to analyse of the impact of controller placement on network resilience, and proposing a controller placement strategy that minimizes the worst case latencies, respectively.

The research work of Xiaolan, H. and Muqing, W. [14] proposed a density clustering algorithm for controller placement problem in SDN and annoyed to examine the propagation delay and communication aspects between inter-switches and switch-controller. According to the authors thought, the delay compactness is intended by bring together the k-adjacent neighbours and the nearest neighbour with lesser density. General models or simulations are conducted on Internet2 *OS*3*E* network topology, and the number of controllers is obtained automatically through the number of side parts of γ density curve by the kernel density estimation (KDE). Other research work in Hailan Kuang et al. [15] have been conducted and expanded their research to solve and optimize controller placement problem by proposing an algorithm based on hierarchical K-means algorithm. The later authors also applied to clustering the large network into several single controller SDN domains, and do the experiments in the Internet2 OS3E topology to evaluate the effectiveness of their algorithm. As their investigational outcomes indicated, the partition by their algorithm is more balanced than the baseline optimized K-means algorithm. But their work has larger propagation delay than the optimized K-means algorithm proposed by G Wang et al. [16].

The problem in [16] is farther expanded in paper [17] by considering the SDN control plane consists of local controllers controlled by global controller, and by designing an offline application that aimed to the placement of controllers optimally. The performance analysis of this work achieved a minimum number of overhead and a minimum number of propagation delay in terms of network stability, minimizing the number of controllers overloads and minimizing the probability of overloads occurrences.

2.1.2 Controller Placement Based on Network Parameters

In [18], the research work proposed a particle swarm optimization algorithm (Global Latency Control Placement Problem with Capacitated Controllers (CGLCPP)) to solve the problem of controller placement problems in SDN scheme. And also the experiment was executed by C++ in VS2010 and CPLEX is used to solve integer programming. This author's paper addressed and investigated better performance compared with Greedy algorithm, Random algorithm and Integer linear programming algorithm (ILP) in propagation latency, computation time and convergence by considering both the latency between controllers and the capacities as well.

Obadiah M. et al. [19] have been attempted the issue of minimizing communication overhead in the control plane of distributed SDN networks by carefully placing the controllers including a building minimal spanning tree between controllers. They developed a MIP based linear formulation in order to find the optimal solution on topologies with a moderate size, and also presented heuristic based greedy algorithm to find a near optimal controller placement and its associated spanning tree topology. They urged that heuristic based greedy algorithm gives results very close the optimal in a reasonable time on real world as well as random topologies of different sizes.

There was a research work that proposes a Set Covering Controller Placement Problem Model (SCCPPM) [16] to discover the optimum controllers with respect to latency requirement. The proposed model in [20] uses IBM ILOG CPLEX Optimization package and performs the experiment on 124 graphs from the Internet Topology Zoo. Perhaps, the results indicate that topology and network size are the dependent factors to the number of required controllers for high resilient network. Furthermore, they argue, 86% of topologies must have more than one controller.

Hidenobu Aoki and Norihiko Shinomiya were presented solutions of controller placement problem in multi-domain SDN networks [21] by raising two concerns of Network partitioning and Controller placement. Optical multi-domain transport networks are extended to studied in Rahman et al. [22] often controlled by a hierarchical distributed architecture of controllers. The authors also proposed a machine-learning framework that helps the controller placement algorithm with proactive prediction (instead of traditional reactive threshold-based approach). Simulation studies, considering practical scenarios and temporal variation of load, show significant cost savings compared to traditional placement approaches.

J Liao et al. [23] proposed a general algorithm called Density Based Controller Placement (DBCP) to fragment the network into some sub set of networks. The authors focused on the optimum number of controllers to compatible with the capacity limitations of switching devices. Their experiment was conducted on Internet2 OS2E network with 34 nodes and 41 edges in Internet Topology Zoo, and tried to compare with other works (POCO, Capacity K-center algorithm and Min-cut) regarding to propagation delay, latency and fault tolerance. The experimental results showed that DBCP provides better performance than the state-of-the art approaches in terms of the aforementioned metrics.

The research work in [24] designed resource reallocation algorithm for dynamic placement of controllers. In the framework of the author, the communication protocol was took place between topology server and scheduler application server according with TCP/IP. Proposed system of [24] evaluated the experiment in two platforms named Mininet topology and POX controllers, and the author argued that the experiment successfully flexed the controller and prevented packets miss based on real time statistics. But this work did not examine load balancing as better as possible. Therefore, [25] further explored again and studied the load balancing problems of controller placement in SDN scheme to the aim of having less loads on the controllers. The author focused on dynamic controller reassigning approach by using central element (one supper controller) to gather information requests from all switches.

Lange, S et al. [26] conducted a research on controller placement to positioning a limited number of resources within a network to achieve several requirements like latency, fault tolerance and load balancing. The authors proposed an exhaustive fashion framework called Matlab based Pareto Optimal Controller placement (POCO) which is performed on real world network topologies from Internet Topology Zoo on different performance metrics regards. And also the authors evaluated their works on heuristic approach called Pareto Simulated Annealing (PSA) to determine accuracy and reliability.

2.1.3 Controller Placement Solutions Based on Clustering and Optimization

Authors in Sahoo et al. [27] considered multi objective combinational optimization problem to overcome controller placements in SDN, and they used two population based meta-heuristics algorithms named as (I) Particle Swarm Optimization (PSO) and (II) Firefly for placements of controllers optimally within the given network topology. The authors used controllers to switch and inter-switch latency as objective function or as performance measures, and conclude that Firefly method outperforms than PSO method in case of latency metrics. In [24] and [28], dynamic controller placement in SDN was studied. In [24] SDN controllers have been placed dynamically by using resource reallocation algorithm whereas in [28] extended research was conducted by proposed Traffic Engineering mechanism to predict number of controllers, and using the

K-means++ algorithm which is written in python code to identified optimal locations of SDN controllers. In the authors work, the proposed (Traffic Engineering mechanism) result was simulated on Mininet emulator, and the controller used for the experiment is the POX controller. Other parameters were taken into account in their research work like average latency and network throughput, and also compare the proposed result with other existing approaches like Pareto Integrated Taboo Search (PITS), Genetic Algorithm (GA), Standard K-means and Random Placement (RP). The author in [28] therefore, urged that the proposed method (Traffic Engineering mechanism) performs better than the other baseline works.

Liao, L. and Leung [29] proposed a hybrid system of a Multi-Objective Genetic Algorithm (MOGA) and Particle Swarm Optimization to manage distributed large scale SDN controllers, and investigate minimized propagation latency. The authors tried to solve the resulted outcomes by using MATLAB based GA multi objective solver. The authors then argued that the inclusion of heuristics allows their proposed MOGA and PSO algorithm to converge a Pareto frontier with significant diversity. Li, Y, and colleagues [30] also looked at a study that utilized a parameter optimization model (POM) to the CPP in SDN. The authors choose multiple algorithms to solve their suggested model, including the firefly algorithm, the bat algorithm, and the Varna-based optimization algorithm. They then utilize the PSO approach to optimize the parameters of the three aforementioned methods. Finally, they argue that parameter optimization model outperforms better over other algorithms in terms of the delay between controllers- switches and delay between switches.

Mohanty, S. et al. in [31] applied a mathematical model for reliable capacitated controller placement to satisfy constraints like controller failure, dis-connection of switches, propagation latency. Their work have been evaluated under several network topologies available in the Internet Topology Zoo, includes SPRINT (11 nodes), NSFNET (13 nodes), AGIS (25 nodes), JANET Backbone (29 nodes), SANET (43 nodes) and IRIS (51 nodes). To calculate the propagation latency, they use the graph markup language (GML), which is used to carry out the latitude and longitude information and run the program on MATLAB 2015. The following table 2.1 describes the summary of the aforementioned literature's.

Table 2.1: Summary of Related Works

Ref. No	Method used	Topology (s)	Implementations	Placement Metric	Remark
[7]	PAM	SANREN	determining the locations and number of SDN controllers	Switch to controller latency	Don't consider worst case latencies
[10]	Density clustering algorithm	Internet2 OS3E	Controller placemen and number	Controller overhead	Don't consider worst latencies
[14]	particle swarm optimization (PSO)	Internet too	propagation delay	inter-switches and switch-controller	Don't consider node failure scenarios
[19]	Density Based Controller Placement (DBCP)	Internet2 OS2E	Placements of controllers	Latency between controllers and the capacities as well.	Not insight of Worst case latencies
[22]	POCO	Internet Topology Zoo	fragment the network into some sub set of network s	Propagation delay, latency and fault tolerance.	Not queuing time and normal vs. worst latencies

CHAPTER 3

Theoretical Concepts of Software Defined Networks

3.1 Introduction

Nowadays, the time is the development and modernization of networking infrastructures due to the emerging of software defined networks (SDN). The main importance of SDN for networking is the promising pattern of decoupling the control plane from the data plane [29] [31]. Unlike traditional networks which are vendor specific problems of networking devices, manually configured and error prone, SDN networks have many positive aspects in the network communication to improve the problems of traditional networks.

Software Defined Networking presents a significant automation of network management and supervision to be programmed on physical network elements interfacing with customary servers. Openness, flexibility, programmability, maintainability, reliability and other features are the factual uniqueness of SDN over traditional networking paradigms.

3.2 Traditional vs. Software Defined Networks

TA traditional network is referred to as a conventional communication approach that employs fixed hardware devices, such as a switch or router to manage traffic on the network. Another networking strategy that makes use of software programs or applications to provide control and oversight of the network system is known as "software-defined networks." Network virtualization is utilized to enhance the network's performance in SDN. As depicts in the following figure 3.1, control plane is detached from data plane in the case of SDN networks. Unlike traditional networks, SDN networks have programmable switches, and capable to automate the flexibility of the whole networks via control plane.

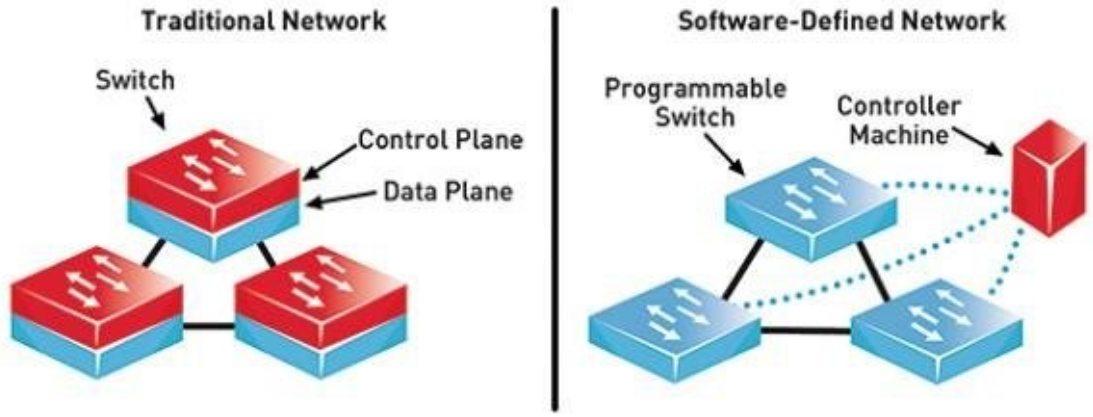


Figure 3.1: Traditional networks vs. Software Defined Networks [1]

The following table 3.1 shows the alterations between traditional and SDN networks.

Table 3.1: Comparison Summary of Traditional vs. Software Defined Networks

Traditional Networks	**Software Defined Networks**
It is lack of programming capabilities.	It uses a virtual networking strategy.
The interfaces on conventional networks are closed .	It is a method of centralized control.
Data and control planes are housed in the same module.	SDNs can be programmed.
Consists of manual and static configurations.	The interface on SDNs is open.
There is no prioritizing in the network.	The program is used to disconnect the two planes.
It is challenging to reconfigure the existing system.	It is automatic and requires little setup time.
It costs more than SDNs, in comparison.	Any packet inside the network block can be prioritized.
They have a complicated structural design.	Simple to modify as needed by the user.
The price of maintenance is extremely high.	SDNs don't come at a particularly high price.
Extensibility in conventional networks is relatively low.	SDNs have less intricate architecture.
	Low maintenance costs and easy to maintain.
	Extensibility in SDNs is fairly high.

3.2.1 Traditional Networking

Traditional networks are a networking paradigm in which control plane and data plane are coupled or mounted together [1][10 – –14] [16] and [2][24 – –31]. Traditional network architectures are hardware based, too old, rigid and more expensive to scale up their networking functionalities [2], and due to this reason these network architectures are out of the recent modern networking pattern which are highly programmable, agility, software based, easily manageable and so on.

The fundamental design principle of traditional network architecture is to combine the data plane and control plane in a single device [29]. The same processor is used in this architecture to forward the packets. The conventional networking architecture also prevents any network device from getting a global perspective of bigger networks. As shown in figure 3.2, traditional networking architecture has three components named as the management/ application plane, control plane and data/ infrastructure plane.

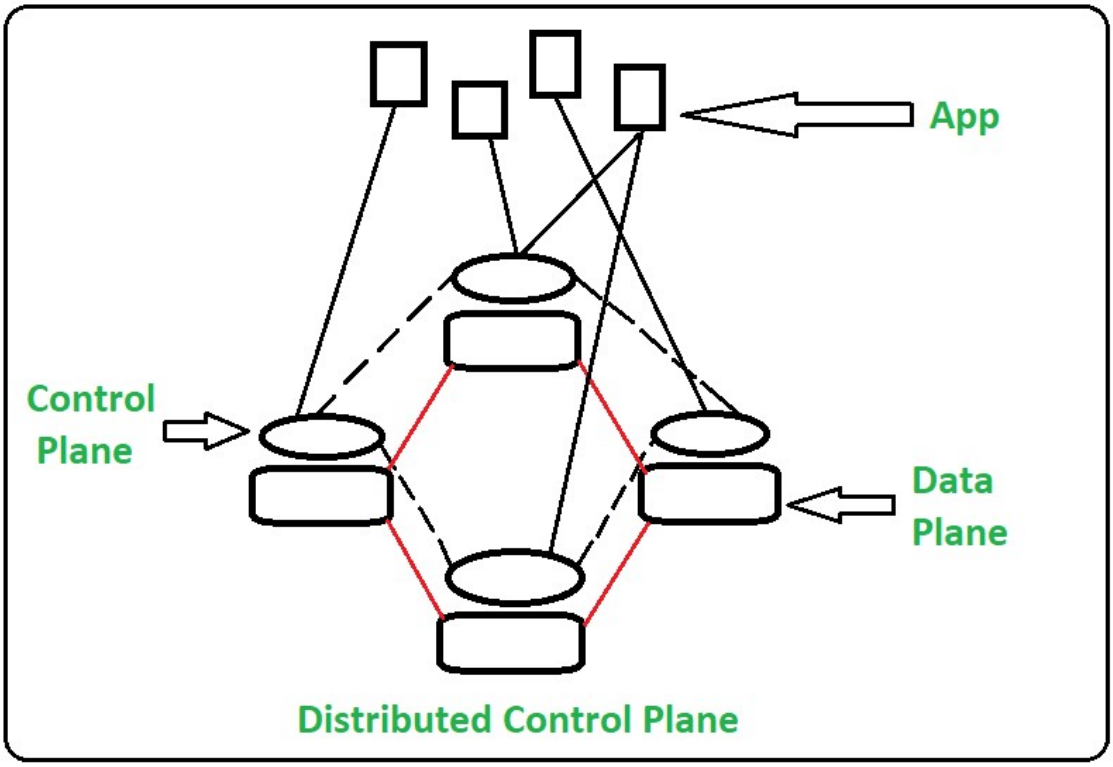


Figure 3.2: Architecture of traditional networking [2]

As shown in the aforementioned legacy or traditional network architecture, the control and data plane layers are attached together in one route. Due to this fact, it is difficult to automate the networking applications, and also inhibits maintenance service if some failures are occurred.

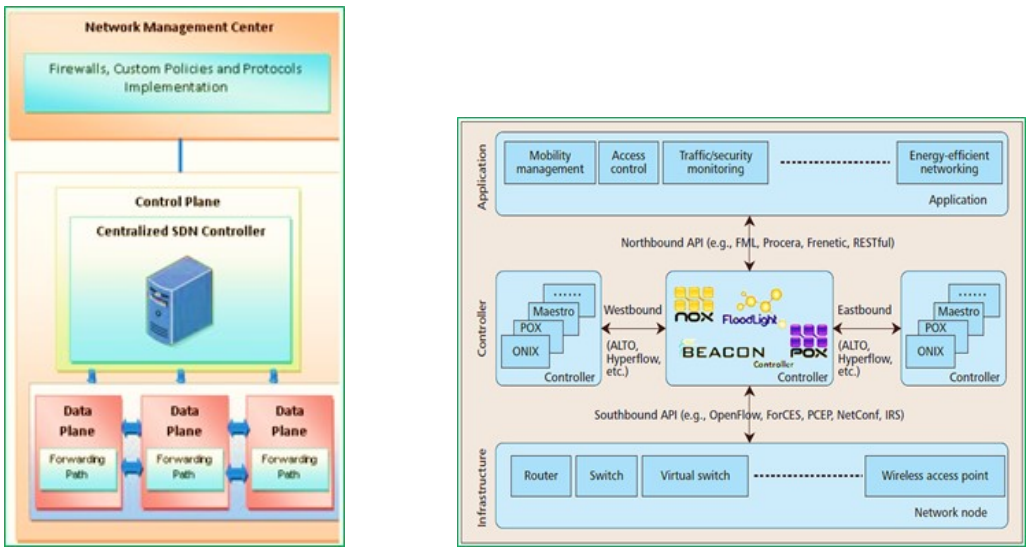
3.2.2 Software Defined Networking

Unlike traditional networking, software defined networking (SDN) is the networking pattern that control plane and infrastructure networking devices (data plane) are separated [29 − −31]. SDN supports the concept of open flow protocol which allows network virtualization (NV) programed by SDN controller. As per the separation of the two basic planes (control and data plane), SDN provide flexibility and good network management as well as operates the concept of open user controlled management via its centralized control plane brainpower. And also SDN provides scalability as the network administrator needs and it enables configurations of the network infrastructures easily from the controller directly.

SDN networking has the following features and characteristics:

- Dismounted or disintegration of control and forwarding (data) plane
- Logically centralized view of the whole network via the controller
- Openness of the networking protocol that interfaces between the networking devices
- The programmability and flexibility nature of the network
- The use of open interfaces via application interfaces between the three layers, application, control and data plane layers

The following figures illustrate the basic architecture and the functionalities of SDN.



(a) **(b)**

Figure 3.3: (a) SDN architecture[2], (b) SDN functionality architecture details [32]

As shown in Figure 3.3 (a) above, the network architecture of SDN has three basic elements named as management or application plane, control plane and data plane.

The management plane or Application Plane: this layer is served as the network abstraction view for the controller requirements. Network applications like quality of service (QoS), security and traffic engineering and virtualization functionalities are managed on this layer. As the control plane needs to share networking applications, there is an application interface which is called northbound interface.

The control plane : control plane contains one or more number of controllers and usually considered as the back bone of the network architecture. It supervises the whole networking elements and updates the table information of switches in the Open Flow manner through southbound API. It also considered as the abstract view of the whole network infrastructure which enables the network administrator practice networking policies and protocols across the networking devices.

More or less, the control plane does the following functionalities [32]:

- discovering and maintaining topologies
- Path failover techniques; selection and implementation of packet routes
- Install the forwarding rules on the forwarding tables in accordance with the network security policy and the performance requirements from the applications, and
- Gather status data on the forwarding devices.

On the other side, the control plane is the intermediate or the transitional plane in between the application and the data plane via the application programing interface (API). The interconnection interface between the control plane and application plane is called northbound whereas the interface between control plane and data plane is called the southbound interface.

The data plane: The data plane contains both actual and fictitious relaying components, which are reachable through the southbound interface. The south bound application programming interface (API) in this layer allows for communication between the controller and the transmitting or forwarding equipment. Routers and other switching devices are examples of delivering devices.

In addition to the aforementioned basic elements of SDN, the following three interfaces are demarcated [32].

The northbound application interface: as shown in Figure 3.3 (b), this interface represents the communication between the controller software platform and the SDN management planes in the application layer. These APIs are open sourced which is abstractly interfaced with the controller software view for use by the network applications. All in all, these APIs represent the universal network management data models and functionalities.

The east-west protocol: in the presence of more than two controllers (multiple controllers), this interface protocol take the mandate of the interactions and intercommunications between these multiple controllers.

The southbound application interface: obviously, the control plane needs to communicate with the data plane to have successful implementation and optimal network management. To doing so, there must be a certain interfacing protocol, and the interface between the data and control plane is called the southbound interface to manage several network infrastructures.

3.3 Communication Protocol in SDN Networks

In SDN operation, communication protocols are needed to address the forwarding plane of the network elements. One of the most common protocols in SDN enabled network is Open Flow [17] [18] [20]. It also considers the disintegration between data and control plane, and provides systematizes information exchange between them.

3.3.1 Open Flow Protocols

Open Flow is the means of communication protocol that enables to facilitate the interactions of control and data plane in the SDN architecture. This protocol used for managing the southbound interface by providing software based access to the networking devices including switches and routers.

Open Flow protocol provides a good management tools which can control all the topology changes and features of networking devices and the controller via the flow tables in which how to direct network traffic. The Open Flow specification is controlled by open network foundation (ONF) [2].

The forwarding tables that tell forwarding devices how to forward packets are accessible via software using Open Flow. For its operation, Open Flow specifies the following network elements: Open Flow controllers, open flow control channels, and open flow switches. The following figure 3.4 depicts the communication protocols of SDN controllers and its Open Flow switches.

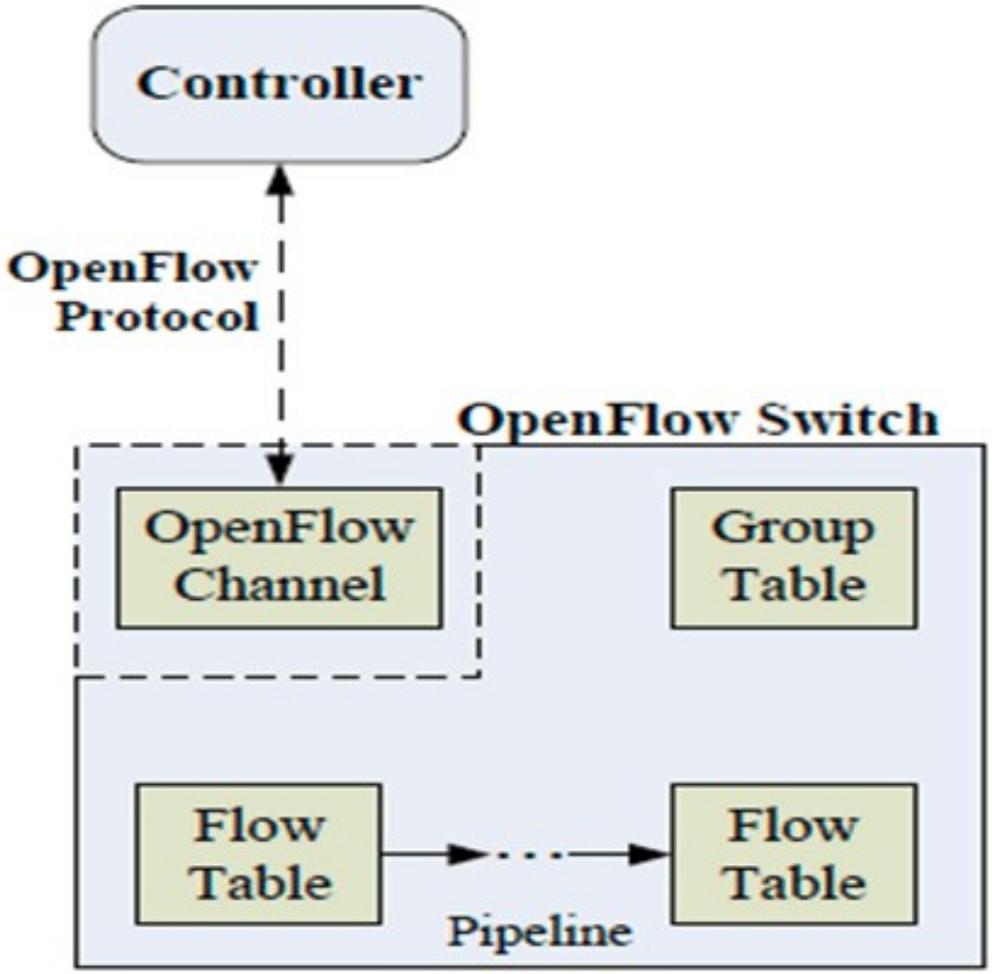


Figure 3.4: Open Flow Protocol Communications [3]

Open Flow switch

The Open Flow switch is a data center switch that supports Open Flow and realizes network connectivity via a centralized controller. Data packets are forwarded in the network using Open Flow switches, which utilize software-defined networking (SDN) technology to maximize network resources and automate traffic distribution as needed.

We then go into more detail about the Open Flow switch's specifications below.

Open Flow Switch Components: An Open Flow switch employs the Open Flow protocol to connect with the SDN controller and is made up of one or more flow tables and a group table. It can set up one or more Open Flow channels to external controllers in addition to its primary tasks of packet lookup and forwarding.

- ***Flow Tables***: Typically, packets are concatenated into flows for network transmission. Of course, packets belonging to the same VLAN or a pair of MAC addresses might

also be flows. The Open Flow protocol attempts to recognize and allocate these flows inside an Open Flow switch. As a result, the switch is now able to apply to a single port or to all ports, while performing additional action when a packet matches a flow. Each Open Flow switch has at least one flow table with a set of internal flow entries. The controller will create low entries when an end-to-end connection is required due to the switch position in the topology being identified. However, these flow entries also contain counters, and instructions that are applicable to matching packets, matching fields, etc. Table entries will be reviewed and compared depending on priority as packets start there first. The packet is forwarded to the directive's chosen table if the flow needs to continue to another table. There is a set of flow entries in these flow tables, and each controls how packets in a flow will be routed. Match fields, counters, and actions are all that make up flow entries as presented in figure 3.5.

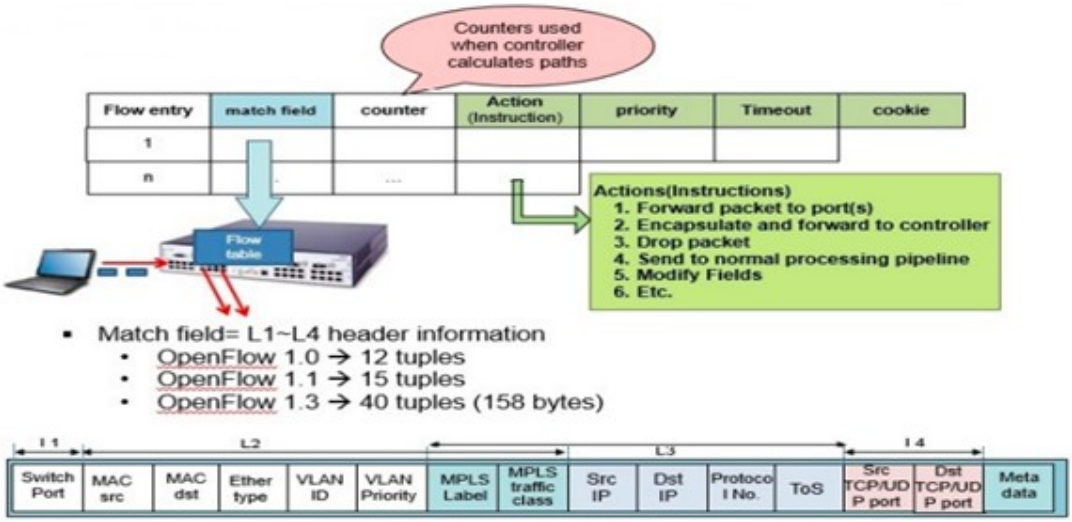


Figure 3.5: Open Flow Protocol Communications [3]

Open flow controller

An SDN controller that makes use of the open flow Protocol is known as an open flow controller [34]. A server program or other entity that communicates with an open flow switch utilizing the open flow protocol is referred to as also an open flow controller. An open flow Controller connects to and sets up network devices (routers, switches, etc.) using the open flow protocol to choose the optimum routing for application traffic.

SDN controllers can make network management simpler by handling all connections between applications and devices, effectively managing network flows, and making changes to them to accommodate shifting requirements [4; 33; 34]. Administrators can more precisely and dynamically manage network traffic when the network

control plane is implemented in software as opposed to firmware. Information is relayed by an SDN controller to the switches/routers, applications, and business logic (through southbound APIs) and (via northbound APIs) respectively. To manage various Open Flow-enabled network components, Open Flow controllers in particular establish a central control point. By removing proprietary protocols from hardware suppliers, the Open Flow protocol is intended to promote flexibility.
Open Flow Controller Key Points: the following are some of the intents SDN Open Flow Controllers.

- An Open Flow Controller connects and sets up network devices using the Open Flow protocol to create the best possible path for traffic.
- Because SDN controllers manage all connections between applications and devices to control and alter traffic channels as traffic requirements change, they simplify network management.
- The Lumina SDN Controller, NOX, Beacon, Trema, NEC Programmable-Flow, and Big-Switch are SDN controllers that support Open Flow software.

The Open Flow channel

The interface that links each Open Flow switch to a controller is the control channel, or Open Flow channel [32]. The controllers manage and configure the switches over this interface, as well as receive events from the switches and send packets out to the switches. Usually, just one network connection is established while creating an Open Flow channel. Each Open Flow switch is connected to a controller via an interface called an Open Flow channel.

The following actions are carried out by the controller and switch through the exchange of control messages via the Open Flow channel:

- Manage and configure the switch
- Get switch events by receiving them
- Transmit packets through the switch

3.4 The Controller Placement

Although the decoupling of control functions increases network adaptability, it also poses a number of problems for the control plane in terms of data consistency, resilience, and scalability. In an SDN network, controller placements can be chosen to ensure that the control pathways and controllers have enough capacity to handle

the traffic flowing from the switches they govern, preventing them from becoming a bottleneck [3].

3.4.1 Single and Distributed Controllers

Single Controller Placement

Because the forwarding and control planes are separate, the forwarding plane can be made simple while the controller plane handles and manages the network intelligence. The performance of the system, for instance through lowering communication dependability, could be hampered by this split. Therefore, while designing a network, the placement of the controller or controllers should be taken into account in a way that maximizes performance and dependability [34].

The goal of single controller placement (SCP) is to position one controller in accordance with a predetermined purpose. To ensure a high level of resilience, or to prevent the controller from becoming detached from nodes, is a very typical goal. This is significant because maintaining a controller's ability to communicate with its network peers is a prerequisite for its proper operation.

It is preferable to deploy many distributed SDN controllers in order to get over the single SDN controller limitation. In this setup, the load is distributed evenly across the controllers, and node failures are handled by switching to another controller when the first one fails.

Multiple Controller Placements

Using a single controller causes the network's performance to decline and its dependability to decrease as the network's size grows. In order to improve network availability and management, multiple controllers are used. Multiple controller placements(MCP) are the designation for the CPP in this setting [34]. Consistency of the network state must be attained in the controllers' communications if several controllers are deployed. The decision on the metrics to be used and the actual network topology itself determines where to place the controllers. When the ideal solution is not possible, the CPP should have a close to optimal solution to put many controllers effectively.

3.4.2 Optimality of the Controller Placements

Finding the best position to install a single or multiple SDN controller/s in a given networking topology is/are difficult task on a network of switches (nodes) [1][2 − −4][32 − −34]. Moreover, this fragment discusses several objectives, strategies and metrics related to SDN controller locations that can be defined in the selection criteria.

Basic Intents of Controller Placements

The controller placement problem entails figuring out how many controllers are needed and where they should be placed in a network in order to achieve a particular goal that may be expressed as:

Quality of Service (QoS): Placements of controllers can be chosen in accordance with the service provider's established standards for service quality. Among others, these QoS criteria include controller utility [16], switch-to-controller delay [23], and inter-controller delay [26] [28]. Additionally, based on certain QoS metrics that ensure the flow setup time and request response time, the position of the controllers inside the network can be chosen [31]. For instance, in order for crucial services like video or voice-over IP to function properly, they must adhere or obey to certain QoS standards. The following figure 3.6 depicts the objectives of CPP is SDN networking.

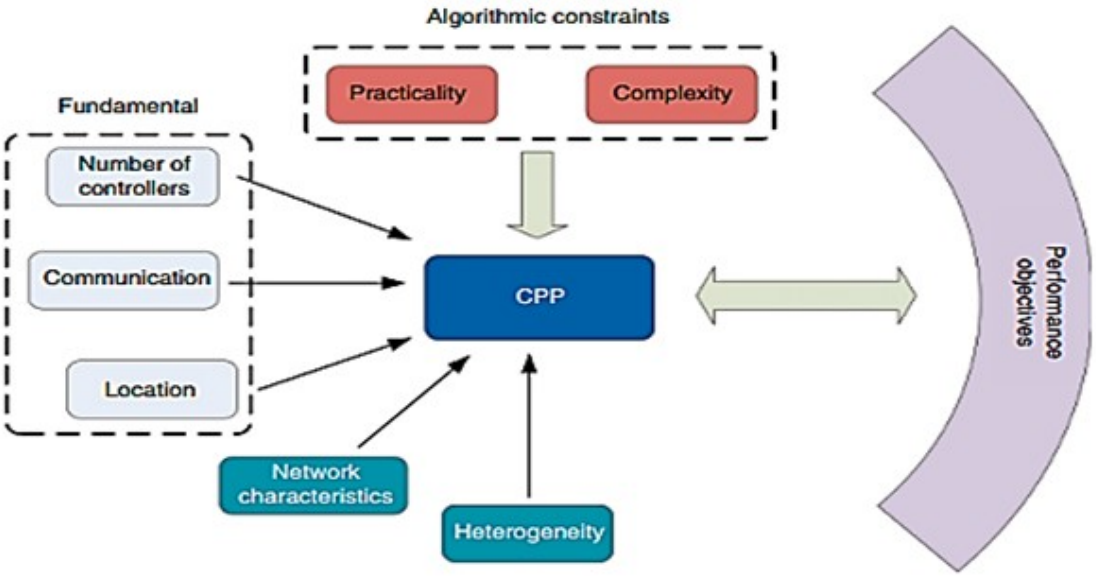


Figure 3.6: CPP parameters and performance objectives [4]

Economic profit of the controller placement: The benefit of controller from economic point of view: From network providers angle, it will reduce the initial capital and operational costs. The objective of scheme is if smaller number of controllers are used, there will be efficacious uses of resources. A network's installed controller count has a direct bearing on both capital and operating costs (CAPEX and OPEX). Finding the ideal number of controllers for a network and the switch-to-controller assignment become important as a result. The largest number of switches (nodes) should be assigned to each controller while still meeting a number of network requirements in order to accomplish this goal, according to controller placement methodologies [18] [20].

For instance, in [22] the goal is to reduce the cost associated with assigning a switch to the controller, and in [26] the goal is to reduce a given cost function that aims to increase resilience/robustness.

Resilience of the control plane: Robustness of the control plane: In the context of the control plane robustness is manifested in finding the controller allocation that performers the following two key features:(a) during network failures, it will minimize the data loss by reducing the network routes which are affected by the aforementioned failures. [2][31]; (b) by considering the previously defined robustness requirements, it will maximize the number of backup routes per each controller paths. [32][3][33].

Techniques of SDN Controller Placement problem

It is (Non-deterministic polynomial-time) NP-hard to place controllers [29]. As a result, it takes much longer to identify the ideal solution for huge networks. Controller placement approaches are proposed to (i) minimizing metrics of latency and cost as well as (ii) maximizing metrics related to resilience and reliability of controllers in the given network topology. The following figure 3.7 summarizes the arrangement of these approaches and the description is presented in view of that.

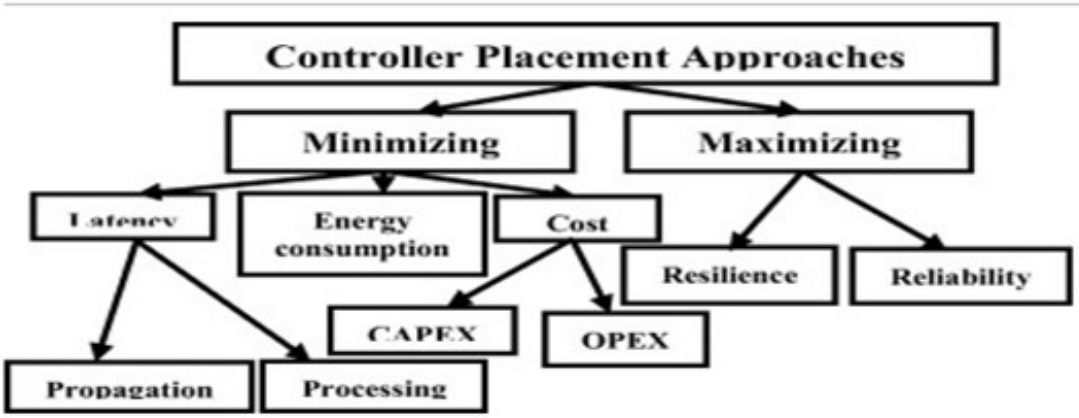


Figure 3.7: Categorizations of Controller Placement Techniques [5]

- ***Minimizing Network Latency***: The cumulative effect of both propagation delay and processing delay is latency. The propagation delay (latency) includes the average latency between switch/node to controller (S2C or N2C), the average latency between controller to controller (C2C) and the worst case latency between N2C. Since all SDN functions are carried out through regular message exchanges between controllers and switches, the latency between them is particularly important. Packet transmission latency, propagation latency, switch queuing latency, and controller processing latency make up the total latency. The ratio of the packet size to the link's transmission rate is the packet transmission latency. The distance between controllers and switches has a major role in determining the packet propagation latency.
- ***Minimizing Costs Related to Deployment and Usage of Energy***: It is necessary to take into consideration about the overall costs of a network while considering optimization of a network controller placement. These costs mainly involves cost of deployment and energy. On the one hand, the deployment cost includes network equipments and the expenditures used to operate these equipments. The energy consumption cost, on the other hand, is a cost which is related to the energy required to operate the network system at any operational time.
  - ***Minimizing Deployment Cost***: The better controller placement approach should be having a minimum setup and maintenance cost as much as possible. The cost of the controller may take into account CAPEX, CPEX, and energy consumptions accordingly.
  - ***Minimizing Energy Consumption***: Legacy devices on the Internet, such as switches and routers, typically run constantly regardless of the traffic load. After adopting SDN in a network, traffic can be monitored in real-time and rerouted quickly. Therefore, energy consumption could be reduced by minimizing the number of active links during off-peak periods.

- ***Maximizing Resiliency and Reliability***: In the context of SDN, since network failures could cause disconnections between the control plane and data plane, and further disable some of the switches, it is of extreme importance to enhance the reliability and resilience in SDN. Reliability is defined as a performance metric that is inversely proportional to the "expected percentage of control path loss" [9], and the optimization target is to minimize the expected percentage of control path loss. In contrast, resilience is defined to reflect the capacity of controller placement strategies to sustain loss of connectivity upon controller or link disruptions [12]. Although these two terms are often used interchangeably in other fields, they are kept different in the existing works on SDN.
  - ***Maximizing Resilience***: The probability that a system will carry out its intended functions without error, within design limits, under certain operating conditions, and for a specific amount of time is what IEEE defines as reliability [35]. In SDN-based networks, some switches will become inoperable if the connection between the controller and the forwarding planes is severed as a result of network failures. In order to guarantee the dependability of SDN, network availability must be guaranteed. Therefore, increasing dependability is crucial to avoiding disconnections between controllers or between controllers and the switch. When a network has numerous controllers, consistency should also be guaranteed. Each control path makes use of the switches' current connections. A proper communication between switches and their controllers must be made possible if a control path is represented as a logical link. This is necessary for control paths to be valid. When control routes fail, the connection between a switch and its controller or between controllers is severed, and then it renders the control network inoperative.
  - ***Maximizing Reliability***: while proposing controller placement approaches, reliability or dependability should be maximized. To increase controller's reliability:
    - Multiple control path should be applied
    - Multiple controllers are recommended
    - Minimizing path (shortest path); path should be minimized

Indeed, the following figure 3.8 summarizes the CPP-approaches profitable overwhelming's.

Figure 3.8: Instantaneous of CPP Approach paybacks

In this instance, various methods have been employed to effectively resolve the controller placement problem issue.

Exact solutions : it is an optimization technique in which the problem formulation has to be done in the worth of Integer Linear Programming (ILP) to find solutions of an optimal controller placement.
Network infrastructures like controllers, switches and link resources can be limited when using exact solutions. Furthermore, basic requirements such as switch-controller delay, inter controller delay and resilience can be defined. Nevertheless, finding the controller placements in large networks can take a long time is the disadvantage of this strategy as revealed in [3].

Heuristic solutions: compared to ILP based exact solutions, heuristic strategy permits the controller placement problem to be solved in a more efficient time [33]. Conversely, this technique does not find an optimal solution rather it finds a solution near to the optimal, compromising optimality for short execution time.

Clustering solutions: An essential method that aims to discover homogenous groups of items based on measurements of their properties, clustering is utilized in many areas and applications. Clustering may be applied to the controller placement problem with heuristic and accurate solutions. By narrowing the search area where a controller placement is sought and specifying the set of viable controller placements, the clustering strategy in this situation is responsible for lowering the search space. The controller location can then be chosen using a heuristic or precise technique.

3.4.3 The Controller Placement Metrics

The positioning of the controller has been chosen after taking into account several metrics. These may be categorized based on the primary network goal for choosing the controller locations, as stated in Section 3.4.2

Metric Related to Network Performance

The switch-to-controller path and the switch allocation across controllers are two measures that assess the performance of the resultant control planes. Additionally, these data may be utilized to contrast various placements of controller strategies.

Stretch: With the shortest path from the switch to the controller, this measure compares the length of each switch's path to the controller location discovered (control path). Because several control pathways might share the switches' and connections' resources, a high stress may cause some of those paths to experience an extra delay. The significance of such side effects increases as more control pathways use the same switch or connection. For instance, if many control routes share a switch or link, all of the control paths that include that switch are severed from the controller when it fails.

Path length: The number of trips between a controller and each switch it manages is measured by the route or path length metric. Resources are consumed more efficiently and node reliance grows with increasing control route length. This indicates that the cost of the resultant control plane is directly influenced by the path length. Switches with longer control pathways to their controllers may have significant communication delays, which can impair the performance of the control plane. This depends on the connections' traffic. When a network failure occurs along a lengthy path, the controller disconnects all downstream nodes that are linked to the problem node.

Controlling-path delay: This is the amount of time it takes for a switch and its controller to communicate with one another. As a result, this delay affects (i) how quickly time switches notify their controller of events and (ii) how quickly controllers setup their switches. Given this metric's significance to network performance, the majority of techniques consider it when choosing where to position controllers.

Controller Load Balancing: An increase in the number of switches controlled by a controller increases the controller's load as well. If a load of a controller exceeds its processing ability, it could result in queuing delays or new request not served from the switches since it has only the capacity to manage a limited number of switches' request at the same time [11] [12] [25]. That is, the controller fails to process requests, and the communication overhead of switch to controller (S-C) and controller to controller (C-C) increases. Therefore, it is important that S-C assignment is well-balanced. However, finding the best placement with minimum controllers and switches assignment is a difficult task when load balancing is considered [5].

Fault Tolerance: Each switch in the SDN is assigned a controller, and if a controller fails, no requests will be sent to or handled by the controller from the switch, which affects the link between the controller and switch. As a result, anytime the switch to controller (S-C) connection is lost, link failure has a detrimental effect on the controlled switches and limits various control plane operations. No new routing instructions are sent to the switch, and all packets are lost [5]. Therefore, it's crucial to choose the placement that minimizes the number of controllers while maintaining reliability.

Inter-controller Communication: To establish global consistency in the SDN, C-C or inter-controller communication is maintained via state synchronization.
When controllers in this scenario want to transmit a message to switches that are controlled by others, they communicate. As a result, such communication affects the efficiency of end-to-end communication between various switches controlled by various controllers [31] and is crucial from the standpoint of CPP.

Performance indicators [10] [14] [16] are typically the most important factors to take into account during CPP when determining the necessary number of controllers and their locations.

The measures are essential to the network's performance, QoS, efficiency, scalability, and reliability and are used to assess the quality of the various controller placements in the system [10] [14] [16]

Metric Related to the Control Plane Scalability

In fact, one controller would not be able to manage the entire network and might as a result constitute a bottleneck in terms of processing power, memory, or input output bandwidth is one of the most crucial justifications for distributing network management [10][11]. According to [25], A centrally managed and reactive SDN network may experience scalability problems due to flow initiation costs or failure resilience. These scalability issues can also occur in big networks with a distributed control plane because controllers must handle requests submitted by other controllers in addition to demands from switches they are in charge of.

The control plane architecture can thereby restrict the network's availability, reaction time, and convergence time in large-scale WANs [25]. The rationale is that in order to achieve network goals in SDN, control applications need to be able to reprogram data plane state at incredibly precise timeframes. Therefore, choosing the controller locations to minimize the flow setup time is essential for a successful network operation.

Link capacity: In SDNs, network resources are shared by the data and control pathways. Accordingly, it is important to choose the controller placements so that all the switches are handled, the switch-to-controller control links have quite enough capacity, and the controllers have enough resources that are available.

One or more control pathways using a congested link may cause a switch to controller communication delay that impacts how quickly a request is answered [4].

Controller Utilization: The amount of flows that each controller manages serves as a gauge or measure of its controller utilization. There is a maximum amount of control messages and flows that each controller can manage at any one moment. Therefore, a controller can only manage a certain amount of switches. In [12], it was demonstrated that, in terms of controller capacity, a single controller is capable of handling an entire network. However, the performance of the application may be impacted by a very high switch to controller latency. One controller is insufficient in big SDN networks, data center networks, or business networks that can handle a high traffic load. While each controller should not be overloaded, the location of the controllers should aim to reduce propagation delay.

Number of controllers: In addition to the traffic that the network manages, additional criteria (such as control route latency and inter-controller delay) also affect the number of controllers in a control plane. Furthermore, it has been confirmed by [10] [31] that the topology has a greater impact on the needed number of controllers than does network size.

As demonstrated in [21], adding more controllers to the network can greatly increase resilience. In [24] and [29], authors verified that possibly the topology of the network determines the optimal number of controllers to increase network resilience. Additionally, it has been validated that adding extra controllers does not enhance performance of the network; rather, it raises the cost of the solution and it may inhibit controller communication.

3.4.4 Simulators for Software Defined Networking

The following are the most popular and possible simulators for software defined based networks

1. MATLAB and Simulink
2. Pareto Optimal Controller Placement (POCO) Matlab-based tool
3. OpenDaylight SDN Controller with the Mininet Network Emulator
4. Mininet Network Emulator
5. NS-3 Network Simulator
6. Opensource Openflow network emulation (OFNet)
7. EstiNet 9.0 OpenFlow Network Simulator

We then prefer the popular software defined network simulator of Pareto Optimal Controller Placement (POCO) Matlab-based tool which is an appropriate simulator format for a given network topology dataset. The selection criterion of using POCO framework is the appropriateness to the SDN-enabled controller placements by using a valid network dataset related to other possible platforms. The framework is also able to examine multi-objective functions considering both failure existence and failure free scenarios.

CHAPTER 4

Methodology and System Design

4.1 Proposed Approach

This chapter outlines the methods used to determine the ideal location for an SDN controller and the number of controllers required in order to achieve the highest level of service quality (QoS). Still a major goal in SDN research is identifying the appropriate controller locations for a large-scale SDN. In this research, we aim to identify the appropriate location of controllers in small-scale SDN. We present an exhaustive search which is a commonly used methodology for handling a facility location problem by taking into account looks at all possible sets of allowed values for all attributes to finding a solution to a problem that requires searching among combinatorial objects like permutations, combinations, or subsets of a set for an element with a specific feature. Figure 4.1 depicts the block diagram of the general approach and technical procedures to conduct this research work.

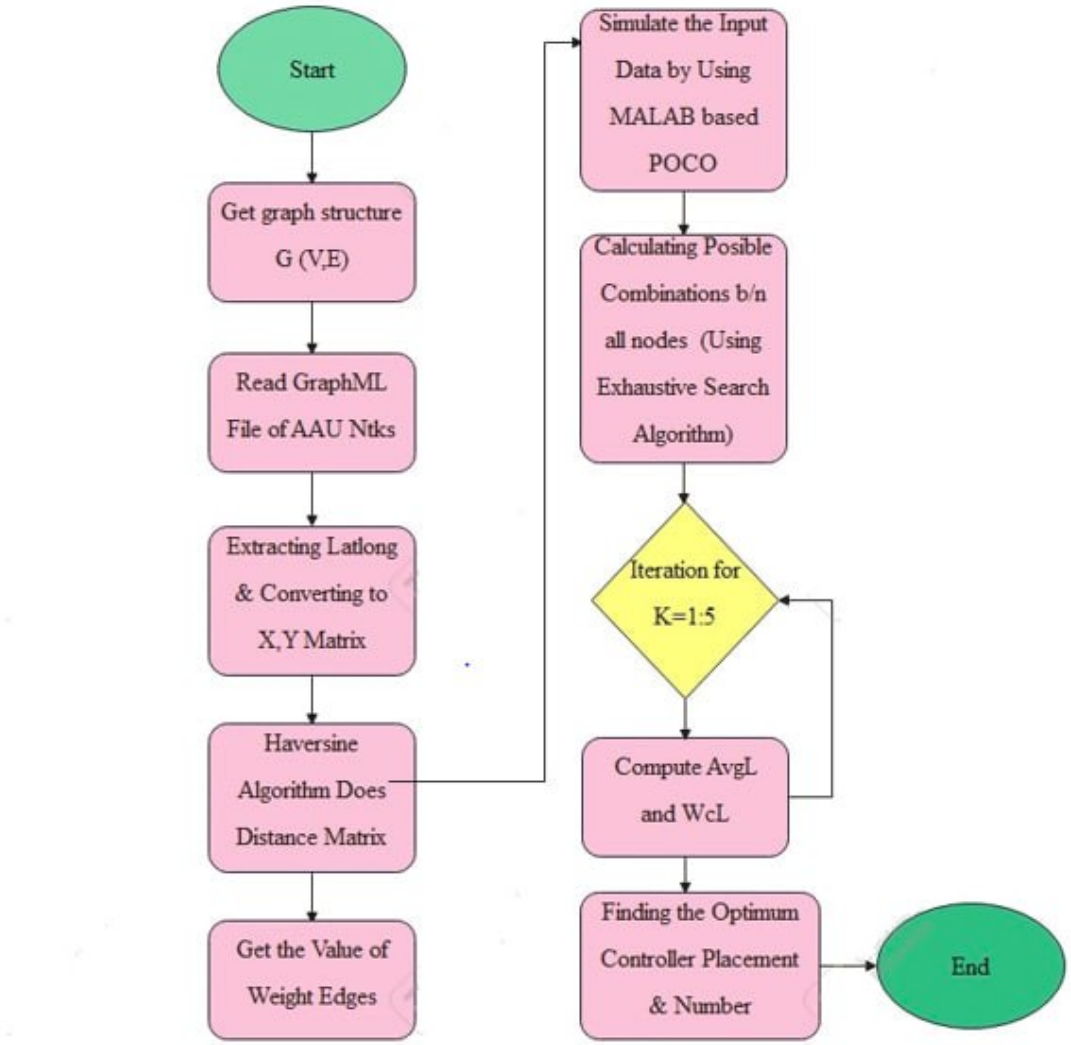


Figure 4.1: Proposed System Architecture

The actions listed below are carried out, as depicted in the block diagram of Figure 4.1:

- We prepare the graph structure networks depending on node connectivity.
- We then prepare these data and read them in comfortable format using GRAPHML (geographical markup language).
- The Haversine formula is used to calculate the distance matrix from the latitude and longitude locations of each node.
- Followed by, implementing the adjacency matrix between all pairs of nodes to generate the weights of the edges of the graph

- In the third, an exhaustive search algorithm was used to examine every combination of permitted levels of all attributes for network graphics.
- And finally, the POCO algorithm framework was used to determine the ideal placements that minimize average latency and worst case latency.

4.2 Formulations in the Placement of Controllers in SDN

SDN promotes centralized network management with a broad perspective of the topology [3]. Multiple controllers in one or more controller domains may make up large-scale SDN networks if the network management is shared among them. Each controller sees the network from a conceptually centralized but physically scattered perspective. The main challenge of the SDN providers is managing of the load distributions, on the one hand, between controllers and on the other hand, between the controllers and switches. The reason for this is the delay between controllers or between controllers and switches results will boost the response time of controllers. The impact of this elongated response time is affecting the reliability of the entire system by affecting the response time of network events.

In order to attain the highest level of service quality, the Addis Ababa University MAN needs to determine the appropriate location for an SDN controller and the necessary number of controllers.

Table 4.1: Mathematical Character with Explanation

Character	**Explanation**
G(V, E)	Undirected Network Graph
V	Set of Nodes in the Graph
E	Set of Edges in the Graph
$d_{v,p}$	Distance matrix between node and controller.
K	Number of controllers that distribute across the network
P	Number of Placements

The following table 4.2depicts the metrics related to controller placement and their descriptions.

Table 4.2: Placement Metrics and Representations

Notations	**Representations**
$\pi^{averageN2C-Latency}$	Average Node to Controller Latency
$\pi^{maximumN2C-Latency}$	Maximum Node to Controller Latency
$\pi^{averageC2C-Latency}$	Average Controller to Controller Latency
$\pi^{maximumC2C-Latency}$	Maximum Controller to Controller Latency
$\pi^{load}imbalance$	Load Imbalance

Average latency between nodes and its controller, maximum latency between nodes and controllers, average latency between controllers, maximum latency between controllers, and load balancing are the metrics taken into account for controller placement. In the following part, we've articulated about how to formulate issues for the aforementioned metrics presented in section 4.2.

4.3 System Design of the Research

In this chapter, we go over the suggested technique for placing controllers, which involves input datasets, experiment setup, modeling a solution to the issue, tuning the exhaustive based POCO algorithm to share the network, and optimizing controller placement. Additionally, we described how CPPs are formulated and figured out using the performance metrics, the system configuration and the topology encoding.

4.3.1 Dataset

Dataset is the crucial element to study a valuable research. We have used AAU's network dataset for our simulation and evaluation of the proposed methodology. We collected the existing datasets of AAU which has seven connected nodes (campuses). We then conducted two sets of investigations to determine the location and minimum number of controllers of the network topology of AAU campus based SDN-CPP metrics. In the first set of trials, we constructed and assessed the controller placement models using the experimental metrics and the outcomes of these tests are referred to as the simulator performance under the failure free cases. In the second series of trials, we constructed and assessed the SDN-CPP topology models using the simulator performance under failure existence dataset (AAU's dataset). We illustrated the seven campuses of AAU in the following table 4.3.

Table 4.3: Addis Ababa University Campus Names with Corresponding Node IDs

Assigned Node ID	Campus Name	Remark
#1	Sidist Kilo Campus	
#2	Amist Kilo Campus	All Campus
#3	Arat Kilo Campus	Nodes are
#4	School of Fine Arts	Directly Connected
#5	FBE	From Sidist Kilo
#6	Yared School of Music	
#7	Dental Medical School	

Although the primary intention is conducting our research on all available campuses of Addis Ababa University, as presented in the above table 4.3, we only consider seven campuses. This is because there is no any evident to have the connectivity of all those campuses which are not listed in the aforementioned table; rather the seven campuses have connected from sadist kilo.

4.3.2 Experiment Setup

To prepare the network dataset and implement the intended solution, the POCO MATLAB algorithm and framework was used. The POCO framework is a MATLAB-based tool made to evaluate where controllers should be placed in various network settings. The framework offers a scenario that may determine how many controllers (K) are required in a given network to ensure that, in the event of node failures, all nodes can still communicate with a controller.

We then employ exhaustive search algorithm which is supported by MATLAB based POCO to carry out an empirical evaluation of our suggested approach. Below is a quick description of how each of the aforementioned algorithms' hyper parameter configurations is set up.

4.3.3 Experimental Structures and Dataset Encoding

We used seven campus nodes(switches in this case) of AAU in order to prepare and encode network dataset graph represented by undirected graph G (V, E), where V represents the vertices or the nodes and E represents the links or edges of the topology graph. For this matter, the topology has seven nodes and six edges.

Utilizing the Geography Mark-up Language (GML), the network topology file for GRAPHML and GML is created. After that, we added it to the MATLAB-based

POCO simulator. The GML and GRAPHML-based AAU topology is prepared and structured to make it convenient for controller placement. On the other hand, the network topology could not be shown because the discovery by the POCO frontier was not convenient for visualization due to the proximity of the nodes geographic location. Therefore, we intentionally excluded the topology figure from this document. Few samples of our investigation procedures are included in Appendix A.1 to A.5.

4.3.4 Experiment Environment

Software, hardware requirements and programming language are discussed in the afterward sections.

Software Requirement: the following lists are the software requirements while conducting this research.

- ***Operating System***: Windows operating system.
- ***MATLAB Version***: MATLAB R20120a.
- ***Matlab Editor***: Matlab programming editor is invented by David Garrison and provides a new way to create, edit and run MATLAB® code. It is used to evaluate the results.
- ***Notepad Editor***: It is used for preparing the .gml and .graphml file.
- ***yEd graph editor***: to export .GML file.

Hardware Requirement: the following lists are the hardware requirements while conducting this research.

- ***Processor***: Intel(R) Core(TM) i3-4000M CPU @ 2.40GHz 2.39 GHz
- ***Installed Memory (RAM)***: 4.00GB
- ***System Type***: 64 –bit operating system, x64-based processor.
- ***Hard Disk***:1TB

Programming Languages: the following lists are the programing languages while conducting this research.

- ***Matlab***: We used Matlab programming for creating interfaces for network topology upload, calculate the shortest distance, for clustering of the given topology file, for controller placement, to calculate latency metrics, for load balancing, and also for evaluation of the results.

– ***C Programming***: We have used c programming for algorithm optimization by integrating with Matlab.

Table 4.4: Summary of system configuration

Software Specifications	Operating System	Windows 10 Pro (64 bits)
	Development kit	Mat lab 2020a
	Language	C programming
	Network simulator	POCO
Hardware Specifications	Hard Disk	1 T
	CPU	Intel(R) Core(TM) i3-4000M CPU @ 2.40GHz 2.39 GHz
	RAM	4 GB

4.4 System Model

The network was optimized for the adoption of SDN in this thesis by figuring out how many controllers to put where. In order to provide application-driven network optimization and determine the number and location of controllers, POCO was created [36]. In order to do this, it first uses the import graph technique to import the WAN's network topology and gather network information.

The POCO framework is made to assess where controllers should be placed in a network to achieve maximum efficiency, specifically in one of the following situations:

1. Failure free: all nodes of the network, including controllers, function normally.
2. Node failure: failures of nodes, which are those nodes that were not chosen as one of the k controllers.
3. Up to k-1 controller failure: failures of up to k-1 controllers are tested to see how nodes react when forced to fall back on backup controllers.

   Additionally, the POCO framework has a scenario that can determine the bare minimum of k controllers required in a given network to ensure that, in the event of any node failures, all

nodes may still communicate with a controller. Network model evaluation for a range of issue scenarios is a capability of POCO.

For a specific network topology, the application includes some common problem profiles to load:

- Determine the optimal k=1-to-5 controller placements with the possibility of up to node failures present. Finding a minimum "resilient" k number is another approach; by resilient, we mean that there is no chance that a node will lose its controller in the event of any node failures.
- Determine the best k=1-to-5 controller placements with the possibility of no controller failure.
- Given the possibility of up to k-1 controller failures, determine the optimum k=1-to-5 controller locations.

Additionally, the user has the option of directly choosing which nodes are controllers, which controllers fail, and which nodes fail by id number for each of these profiles.

The research described in this thesis focused on the conception of POCO's algorithms for locating all pairs of the shortest pathways in AAU Metropolitan Network (MAN) and for figuring out the ideal location for controllers inside its network. Here is a quick overview of these algorithms.

Networks within a.*topo. Mat files are represented as a series of table. Some of these data points are provided by the input topology; some are generated during runtime:

Coordinates : n-by-2 matrix of the x, y coordinates of each node in the graph.

Distance Matrix: n-by-n matrix of the distance between every pair of nodes. Values are normalized based on the diameter of the graph. These distance values also represent the two-way 'latency' between nodes.

Topology: n-by-n matrix identifying links between nodes. An existing link between nodes (i, j) contains the same distance in the corresponding node of the distance Matrix table. Lack of a link between nodes (x, y) is represented by 'Inf' for infinity.

Node names: n-by-1 matrix of string values identifying each node.

nkx,y: a runtime generated table consisting of every possible combination of k chosen controllers out of n available nodes, by node id #. This can quickly become

computationally prohibitive for medium-to-large sized networks and is only used for exhaustive search. Each of these data structures are manipulated by POCO to generate all objective functions. The following figure 4.2 depicts the experimental system model.

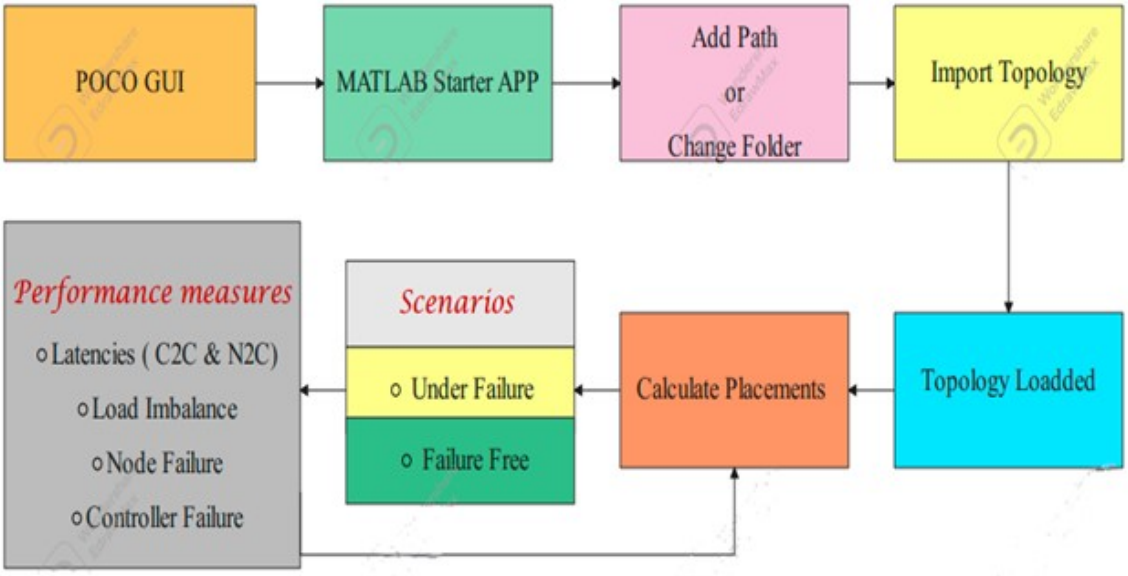


Figure 4.2: Experimental Setup

From the very beginning, in the principal application of using the aforementioned experimental setup, we open it in a MATLAB starter application. Perhaps, to import the prepared network dataset (AAU in our case) into the POCO framework, we should apply the change folder or add the path where the prepared network dataset is collectively saved as a *.topo.mat file (including node names, longitude and latitude coordinates, the topology itself, and the distance matrix of AAU).

4.5 Experiment Metrics

We employ the conventional assessment criteria, which are generated from the typical objective function of different metrics depicted in Table4.2, to assess how well our approach performs. All possible outcomes of the calculation findings are listed in Table 4.2. Evaluation metrics like latency, load imbalance and fault tolerance are taken into account while conducting this research.

4.5.1 Latency as Evaluation Metrics

One of the most important and commonly employed measures for assessing the functioning of the network in the given topology is latency [9] [19]. It falls under the switch to controller latency and controllers to controllers' latency

categories. The average switch to controller latency, the maximum (worst case) switch to controller latency, the average controllers to controllers' (C2C) latency, and the maximum (worst case) (C2C) latency were all taken into consideration.

Algorithms for controller placement are created to either maximize or minimize certain latency-based characteristics or measures' and mathematical modeling, which are as follows:

Switch/node to Controller (N2C) Latency

Information on the node's and its associated controller's connectivity is provided by switch (node) to controller latency. Average and maximum latency cases are incorporated

- ***Average node to controller latency (N2C-AvgL)***: The average transmission delay value between a node and the controller that is assigned to it is known as average N2C latency [3][4][33][34]. Equation 4.1 displays the average N2C's mathematical modeling.

$$\pi^{\text{average } N2C-Latency=} \frac{1}{|\mathrm{v}|} \sum_{\mathrm{v}\varepsilon \mathrm{V}} \left(\min_{\mathrm{p}\in \mathrm{p}} \mathrm{d}_{\mathrm{v,p}} \right) \tag{4.1}$$

  Where,
  V: all the available set of nodes
  v: an individual node
  P: the number of placements
  p: the controller likely selected to the placements
  $d_{v.p}$: represents distance matrix between node and controller

- ***Worst case node to controller latency (N2C-WcL)***: Maximum N2C latency is the term used to describe the maximum packet transmission delay value between a node and the controller that is allocated to it. When there is a strict constraint and a high-performance environment, this latency is typically the ideal goal [5]. The maximum (worst case) N2C latency is depicted mathematically in Equation 4.2 as follows

$$\pi^{\text{maximum } N2C\text{-Latency} =} \max_{v\in V} \min_{p\in P} d_{v,p} \tag{4.2}$$

Controller to Controller Latency

To achieve a consistent view of the network state and improve network performance, controller communication is a key role. It is crucial for controllers to communicate with one another in order to keep a shared state [3]. The delay between controllers is measured by C2C latency.

– ***Average Controller to Controller Latency***: The average packet transmission time between controllers is known as the C2C latency [3]. Equation 4.3 displays the average C2C latency numerically.

$$\pi^{\text{average C2C-Latency}} = \frac{1}{|P|} \sum_{p1,p2 \in p} \left(d_{p1,p2}\right) \tag{4.3}$$

where,
P: all possible set of controller placements
p_1: placements of the first controller
p_2: placements of the second controller
$d_{p1.p2}$: the distance between p_1 and p_2

– ***Maximum Controller to Controller Latency***: The maximum packet transmission between controllers is the maximum C2C latency [3]. Equation 4.4 displays the mathematical maximum C2C delay.

$$\pi^{\text{maximum } C2C\text{-Latency} =} \max_{p1,p2 \in P} d_{p1,p2} \tag{4.4}$$

4.5.2 The Load Imbalance

The distinction is the number of switches that can be assigned to a controller between the maximum and minimum. Switches are distributed across the controllers in each cluster [1][13][14] to balance the traffic demand. The smallest load imbalance value is preferred to reduce delay and increase performance. The imbalanced load is shown in Equation 4.5.

$$\pi^{\text{load}} \text{ imbalance } = \max_{p \in P} n_{p-} \min_{p \in P} n_p \tag{4.5}$$

Where, np is number of node assigned to controller p.

– Distance to Haversine Algorithm: Calculations are performed without taking into account ellipsoidal effects, which is accurate enough for the majority of purposes given that the GPS coordinates (longitude and latitude) for the switches' locations are known. The Haversine formula is used to determine the distance between network switches. The distance of the great circle, as

opposed to the usual Euclidean distance, is the shortest separation between any two points on the surface of a sphere. The Cosine Law is another method for calculating distances in a given region.

While it may not perform well over long distances, this strategy excels at shorter ones. The Harvesine method [36] is employed in the equation below, which is what is used to calculate the great circle's distance.

$$distance = \ 2.\text{r. } \arcsin\left(\sqrt{\sin^2\frac{\varphi_2-\varphi_1}{2}\cos(\varphi_1)\cos(\varphi_2)\sin^2\frac{\lambda_2-\lambda_1}{2}}\right) \quad (4.6)$$

Where:

- $\varphi 1$ and $\varphi 2$ denote the latitudes of points 1 and 2 respectively.
- $\lambda 1$ and $\lambda 2$ also denote the lengths of points 1 and 2 respectively
- r represents the earth's radius, which is a constant equal to 6371 km. The optimization algorithms were used on the actual Addis Ababa University network topology.

This network topology dataset was taken from the network administrator of AAU datacenters.

CHAPTER 5

Result and Discussion

5.1 Results

The specifics of each experiment's findings are presented in this section. We give each set of tests along with the corresponding research question in an attempt to make the findings understandable and simple to grasp.

5.1.1 Controller Placement of POCO Results Under Failure-free Scenario

The following table 5.1 illustrates about the controller placements, latencies, and load imbalance in accordance with the number of controllers using POCO framework. In this case, we apply only to the normal situation (neither controller nor nodes at risk condition) as described in section 4.4.

Table 5.1: N2C Average Latencies in failure free scenario

#No of Controllers (K)	K=1	K=2	K=3	K=4	K=5
Location names for AvgL	Dental_S	Yared_S Dental_S	FBE Dental_S Yared_S	FBE Art_S Yared_S Dental_S	4 kilo FBE Yared_S Dental_S Art_S
AvgL (ms)	0.13	0.13	0.13	0.13	0.13
Load imbalance (%)	-	0.38	0.33	0.27	0.18

As shown in Table 5.1, experimental placement results of AAU Campus nodes are presented in accordance with their corresponding average case latencies (in

milliseconds) and assigned loads (in percent) in order to balancing among SDN controllers. As a result, the findings showed a direct association between the number of controllers and the loads they were allocated (i.e., as increased the number controllers, the load balancing between controllers also outperformed) assuming that the experiment is a normal scenario. The incremental performance of the load imbalance implies the shared effects among the number of controllers. Furthermore, the resulting value of average latency remains the same even if the controller's number is varied (i.e., 0.13ms is the minimum measured value for star nature topology of AAU nodes). Due to the inter-controller delay, we claimed that one SDN controller may be sufficient in terms of latency measurements (i.e., adding more controllers does not demonstrate a substantial change in average latency). But the incremental performance in load balancing is due to the shared load between controllers. On the other hand, the POCO resuls under normal scenario showed that the controller placements does not change their locations as shifting the number of controllers from one to five, rather adding extra controllers locations up to the controller iteration columns (i.e. controller number ranges from1:5). In this case, inter-controller (C2C) latency and load balancing issues resulted, as the same showed in the node-to-controller (N2C) latency in a failure-free scenario. And we intentionally excluded the measured results in order not to redundant.

5.1.2 POCO results under Worst Case Scenarios (up to K-1 Controller and Node Failures)

In this section, we computed the inter-controller (C2C) and node to controller (N2C) worst case latencies under the consideration of failure scenario. We considered the following evaluation metrics for both failure scenarios (node failure and controller failure). Worst case latencies of C2C (table 5.2) and N2C (table 5.3) with the associated controller placements and load imbalance among controllers for controller or node failure are specified.

Table 5.2: C2C Worst Case Latencies in the failure scenario

#No of Controllers (K)	K=1	K=2	K=3	K=4	K=5
Location Names for WcL	6 kilo	6 kilo 4 kilo	6 kilo 4 kilo Art_S	6 kilo 5 kilo 4 kilo Art_S	6 kilo 5 kilo 4 kilo Art_S Yared_S
WcL (ms)	-	0.07	0.29	0.49	0.56
Load imbalance (%)	-	0.07	0.07	0.06	0.05
Failure Node Id	-	2	2	3	3

In the presence of a failure scenario, especially the failure of nodes, as shown in above table 5.2, it is clearly shown that the controller placement, the corresponding inter-communication latencies between controllers, the controller load balancing, and the failure ID of nodes are simulated. In this case we consider controller placements of POCO framework under a single node failure scenario for each controller number. But the failed node is not as per our choice rather by the placement calculation in the POCO framework. From the point of view of the controller placement, the results under failure of node showed the locations of controllers have got changed compared to the failure-free scenario of the POCO toolset. As the results showed, we can realize that the worst case latency between controllers has increased as the number of controllers has increased from K = one to K = five. This is because of the communications between inter-controllers that the time taken to reassign of controllers to the failed nodes. Similarly, the number of controller has its own impact on the load balancing metrics and has a direct correlation with them (i.e., load balancing slightly increased as the controllers increased).

Table 5.3: N2C Worst Case Latencies in failure scenario

#No of Controllers (K)	K=1	K=2	K=3	K=4	K=5
Location Names for WcL	6 kilo	6 kilo 4 kilo	6 kilo 4 kilo Art_S	6 kilo 5 kilo 4 kilo Art_S	6 kilo 5 kilo 4 kilo Art_S FBE
WcL (ms)	0.53	0.47	0.29	0.27	0.23
Load imbalance (%)	-	0.07	0.07	0.06	0.05
Failure Node Id	2	2	3	2	2

Results of node-to-controller (N2C) latency in the worst case (node failure) scenario are examined in table 5.3 above. As a result, in the presence of node failures, the worst case latency between the node and controllers should a relatively better result as the number of controllers varied from one to five. The controller placement in this case is the same as shown from the failure of nodes in performing of C2C latency from the above table 5.2. except FBE in table 5.3 while using at the fifth controller. In this case, load balancing among controllers did not show significant effect but showed a minimal effect. The rational behind this effect may raised from the nature of the topology itself (limited link paths). As a result, the load balancing between controllers in the measure of node -to-controller has affected by the inter-controller communication latency.

Table 5.4: N2C Worst Case Latencies in failure scenario

#No of Controllers (K)	K=1	K=2	K=3	K=4	K=5
Location Names for WcL	6 kilo	6 kilo 4 kilo	6 kilo FBE Dental_S	6 kilo FBE Yared_S Dental_S	6 kilo 5 kilo FBE Yared_S Dental_S
WcL (ms)	0.53	0.60	0.75	0.80	0.82
Load imbalance (%)	-	0.05	0.05	0.05	0.05
Controller failure ID	-	1	1,7	1,5&7	1,5,6&7

As can be seen in table 5.4, the evaluation of node-to-controller latencies is conducted. The best controller placement in the worst case scenario has also been examined. For the case of controller placements using POCO performance under controller failure, we considered the placement results on the controlling effect of the overall network by one controller. In case of the controller placement calculations, we can observed that the locations of controllers have changed from K=3 up to five compared to the results shown in Table 5.3, where there are failure cases. As seen in the results, no significant change in load balancing is evident despite the increased number of SDN controllers. In addition, as the network topology is controlled by one controller only, the load balancing between SDN controllers has no significant effect on the network at all. Table 5.4 shows that varying the controller numbers from one to five results in degradation in network performance (i.e., node-to-controller latency increases with increased controllers). This indicates that the impact of controllers as a risk condition causes the increment of latency measures; and this also negatively influences load balancing. Thus, the overall network activity is supervised and communicated with one controller only as long as up to K-1 controller takes the risk of the network.

And also the reason of insignificant change of load imbalance is coming from this rationale.

5.1.3 Comparative Analysis of the Results

We compare the experimented results on several settings in this part.

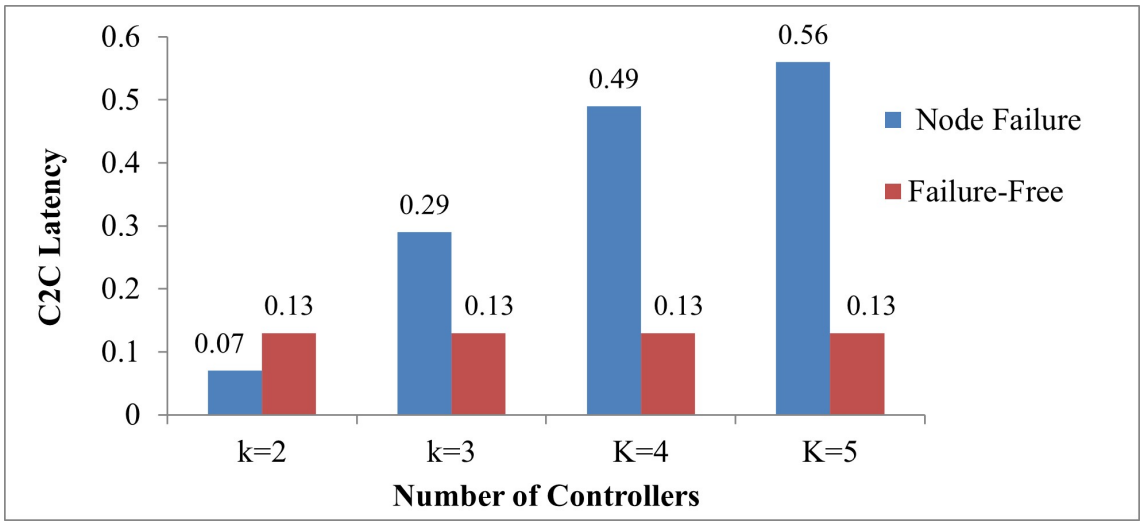


Figure 5.1: Comparison Results of C2C Latency (in Milliseconds)

As shown in figure 5.1 above, the controller-to-controller latency for the two scenarios, failure-free and node failure, is observed. The horizontal, X axis represents the number of controllers, whereas the vertical Y axis also represents the equivalent latencies in milliseconds. In reality, two controllers are needed to measure the latency of the inter-controller communication between them. So, we can compute the latency measures starting from using controller number two up to controller five. As results showed, the intercommunication between controllers has increased as the number of controllers has varied from one to five. But in the case of latencies in a failure free scenario, the average controller to controller latency remains the same. This is because, under the assumption of no- risk failures of controllers and nodes, this small-scale network topology can be optimized by one controller without considering resiliency and scalability. But in the case of failure situations, especially node failures, the controller to controller latency get increased. This is the fact that as once the node failures, the controller should re-assigned to the failure node, and as a result the communication between controllers leads to the larger the delay value.

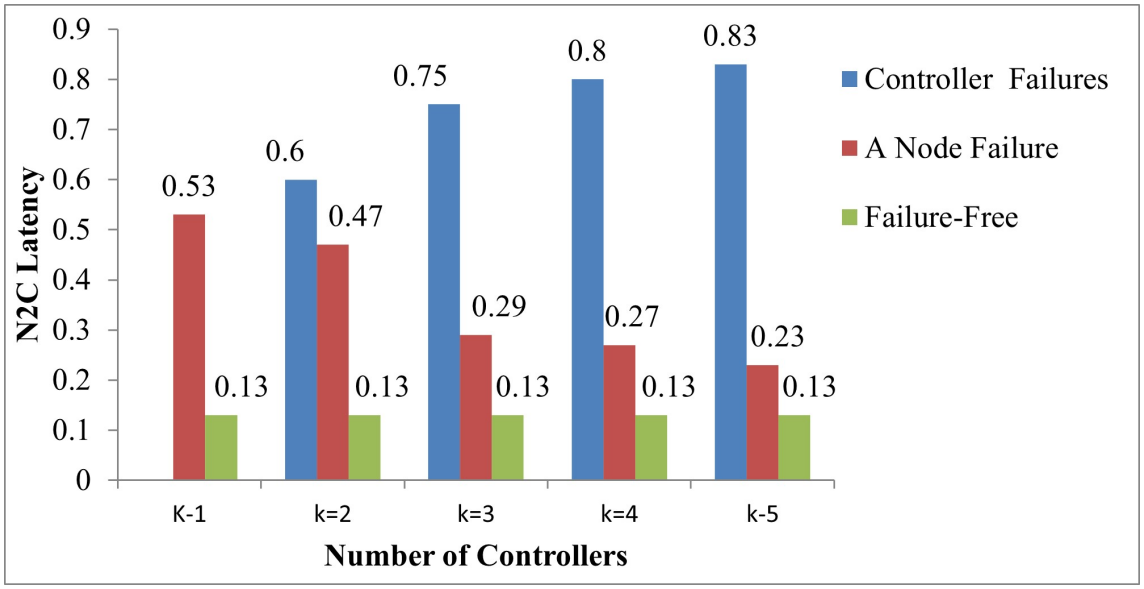


Figure 5.2: Comparison Results of N2C Latency (in Milliseconds)

The graph depicted in above Figure 5.2 points out the comparative analysis of the results of failure free, node-failure, and controller failure scenarios. To make it clear, the X-axis specifies the number of controllers, and the Y- axis also indicates the worst and average latencies in node to controller (N2C) communication. In this scheme, the N2C latencies drastically get increase where controller failure exists as the controller number increases. This is because of the controller less nodes (i.e., once nodes are in a failed state, they are not informed by any controllers unless they are reassigned to other controllers). Indeed, the latencies from node to controller when the node fails are still better than the controller failure scenarios. But the average node-to-controller latency in a failure free (free of either node or controller) scenario has remained the same. From this point of view, the normal scenario showed better node-to-controller latency related to both the failure scenarios (node and controller failures).

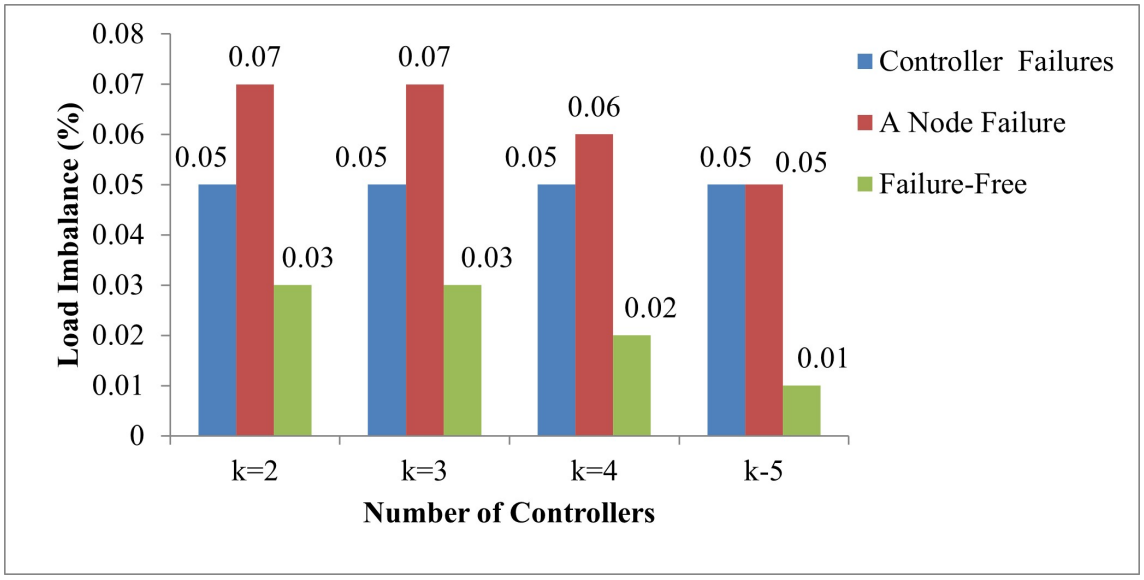


Figure 5.3: Comparison Results of Load imbalance (in Percent)

To be clear, the above graph indicates the load imbalance between node-assigned controllers in the given network topology (AAU in our case). From this end, the horizontal, X-axis represents the number of controllers, whereas the vertical, Y-axis is delegated to the ratio of load imbalance values in percentages. Using equation 4.5 from Chapter 4, the node assignment difference between the most loaded controller and the least loaded controller for each number of controller placements is shown in the bar graph for both scenarios. As is evident, adding more controllers from one to five has a growing impact on load balancing in both the failure-free and node failure scenarios. And also, load imbalance has decreased as a result of increasing the controller's number. In general, in the case of controller failure, there is no significant effect on the load imbalance between controllers. This is because of the intercommunication delay between controllers as well as the overall workload being bottlenecked by up to K_1 controller risk only.

5.2 Discussion

The evaluation outcomes of this research indicate that the MATLAB-based POCO algorithm can used to estimate the minimum number and placement of controllers for an AAU network topology. The three stated questions in the study are addressed based on the results and evaluation metrics that have been done. It is discussed as the following:

RQ1. In regard to controller placement, how many SDN controllers are required in a given network dataset network topology? Based on a comparison of the

POCO performance under failure-free (average latency (Lavg)) and the failure existence (worst-case latency (Lwc)) in order to determine the optimum number of controllers, the ultimate decision about how many controllers to deploy is dependent on the unique needs and constraints of each service provider. As a result, the number of controllers at the end is based on an Internet Service Provider's (ISP) budget to upgrade their network. For this study, we concluded that using two SDN controllers is the most efficient way to achieve the best quality of service outcomes and resilience. Obviously, using the minimum number of controllers saves on the costs of deploying extra controllers and helps to reduce the inter-communication between controllers. But for the sake of reliability and resiliency, fault tolerance in general, it is needed to have two controllers. The two scenarios of POCO under failure-free and failure existence are examined. In the former scenario, one SDN controller is recommended in the latency metrics. But in the latter metrics cases, resiliency and load balancing issues require two controllers for AAU networks, even if it adds another cost to the deployment and maintenance.

RQ2. Where in the topology should SDN controllers be placed optimally? For the sake of determining optimal positions of the topology, the optimum locations of controllers for the AAU network are at the Sidist kilo and Arat kilo campus nodes by looking at failure scenarios. In general, the average N2C latency, the worst N2C latency, the average C2C latency, the worst C2C latency, and the load imbalance are evaluated with their corresponding controller placement as shown in section 5.1, particularly from section 5.1.1 to section 5.1.3.

RQ3. What incidents are observed in the performance of POCO framework under normal and failure scenarios? The investigation showed that the POCO performance under failure-free scenario-based experiment showed minimum N2C latency compared to the worst-case scenario (node failure and up to K-1 controller failure).

CHAPTER 6

Conclusion and Recommendation

6.1 Conclusion

When creating centralized network control architecture, determining the number of controllers needed, where they should be placed, and the metric considerations within the network is essential. In this research, we tackled these issues while taking into consideration a number of crucial factors, such as resilience, quality, and load balancing. Network quality was measured in terms of the maximum/worst latencies that might exist between nodes and controllers.

The best values for the minimum latency and resilience of the metrics are frequently difficult to obtain at the same time, and suitable trade-offs must be made, as we highlighted in our discussion of various resilience mechanisms.

The optimization is done in two scenarios, (I) controller placement using failure free and (II) when either nodes or controllers fail, using the Pareto Optimal Controller Optimization Placement (POCO) framework. First, the controller's placement in the aforementioned scenarios for different numbers of K controllers was found and investigated. Following this, results of altered metrics like latencies (C2C and N2C), fault tolerance (for either node or controller failures), and controller imbalance for both the aforesaid scenarios were conducted accordingly.

The latency calculated using the POCO framework technique described in Section 5.1.1, particularly in Table 5.1 for a variable number of controllers, is the average latency. Our analysis of the trade-off between resiliency and network performance

yielded the findings in section 5.1.3. As predicted, the findings show that using just one controller is the best option for minimizing the trade-off between cost and network performance. However, we recommend utilizing two controllers to provide network scalability and failover. This is mostly due to the fact that our POCO based study results suggest that the ideal number of controllers to deploy on AAU is two, which is the second-best option that offers the least trade-off (considering worst case). From the results, it is recommended that the minimum number of controllers required to adopt SDN in AAU-MAN be two controllers, considering the resiliency of controllers placed at Sidist kilo and Arat kilo. Also, we conclude that the POCO performance under failure-free scenario-based experiment showed minimum N2C latency compared to the worst-case scenario (node failure and up to K-1 controller failure). But showed minimum node failure C2C latency compared to failure free scenario.

6.2 Recommendation

Our recommendation for Addis Ababa University is to change the network from a traditional IP-based network to the newly generated networking pattern which is software defined network with appropriate topology, in this case it is better if mesh or full mush network topology is used. Based on our investigation, we recommend the company deploy two controllers at the Sidist kilo and Arat kilo campuses, which reduce the cost of deployment and improve the performance of the network by considering worst case situations. Our recommendation also goes to future research directions enabled by POCO toolsets should be includes a brief outlook on extensions and features of POCO and its GUI which are currently explored but beyond the scope of this demonstration: controller placements under dynamic network conditions enabled by the POCO extension and the placement of controllers in meshed and multipath networks exemplarily visualized for the strongest network with maximum node numbers and large geographical proximity of nodes.

In addition, it is better if the $POCO-GUI$ take account of further routes especially in the failure scenarios; like the failure numbers of nodes and controllers in its GUI Menu. Also we recommend that small-scale networks of nearby distance nodes shall not be used by POCO framework because the network topology could not be shown for visualization due to the proximity of the node geographic location unless further investigations has to made to scale up the visibility issues.

6.3 Future Works

The results of our study show how multi-objective metrics for SDN-CPP can improve network resilience and performance. We believe that further investigation into novel network measurements and configurations may enhance the performance of an AAU network topology.

In light of this, the following research ventures are planned to be completed in the future:

- To examine the implications of controller placement metrics, we advise combining controller location data with other relevant measurements, such as flow setup time at the SDN controller metrics.
- To build a more trustworthy model, we suggest investigating the potential use of different test beds.
- We intend to investigate the complex network analysis of the SDN networking to come up with the performance metrics using best emulator formats like mininet.
- We also plan to investigate on the performance measures like cost benefits and others by considering average and worst scenarios using the comparison of POCO framework integrating with suitable algorithms.

References

[1] B. P. R. Killi and S. V. Rao, "Capacitated next controller placement in software defined networks," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 514–527, 2017.

[2] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Computer Networks*, vol. 72, pp. 74–98, 2014.

[3] I. Elabani, "Optimization placement for sdn controller: Bell canada as a case study," Master's thesis, University of Waterloo, 2017.

[4] H. Selvi, S. Güner, G. Gür, and F. Alagöz, "The controller placement problem in software defined mobile networks (sdmn)," *Software Defined Mobile Networks (SDMN) Beyond LTE Network Architecture*, pp. 129–147, 2015.

[5] A. A. Ateya, A. Muthanna, A. Vybornova, A. D. Algarni, A. Abuarqoub, Y. Koucheryavy, and A. Koucheryavy, "Chaotic salp swarm algorithm for sdn multi-controller networks," *Engineering Science and Technology, an International Journal*, vol. 22, no. 4, pp. 1001–1012, 2019.

[6] J. Lu, Z. Zhang, T. Hu, P. Yi, and J. Lan, "A survey of controller placement problem in software-defined networking," *IEEE Access*, vol. 7, pp. 24 290–24 307, 2019.

[7] C. V. N. Index, "Global mobile data traffic forecast update, 2016–2021 white paper," *Cisco: San Jose, CA, USA*, vol. 7, p. 180, 2017.

[8] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications surveys & tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[9] R. Jain, "Introduction to software defined networking (sdn)," *Washingt. Univ. Saint Louis*, vol. 11, no. 7, pp. 1–44, 2013.

[10] T. Koponen, M. Casado, N. Gude, and J. Stribling, "Distributed control platform for large-scale production networks," Sep. 9 2014, uS Patent 8,830,823.

[11] J. Liu, J. Liu, and R. Xie, "Reliability-based controller placement algorithm in software defined networking," *Computer Science and Information Systems*, vol. 13, no. 2, pp. 547–560, 2016.

[12] L. Mamushiane, J. Mwangama, and A. A. Lysko, "Controller placement optimization for software defined wide area networks (sdwan)," 2021.

[13] B. P. R. Killi and S. V. Rao, "Towards improving resilience of controller placement with minimum backup capacity in software defined networks," *Computer Networks*, vol. 149, pp. 102–114, 2019.

[14] H. Xiaolan and W. Muqing, "An effective clustering algorithm for controller deployment in sdn," in *2018 Global Wireless Summit (GWS)*. IEEE, 2018, pp. 338–342.

[15] H. Kuang, Y. Qiu, R. Li, and X. Liu, "A hierarchical k-means algorithm for controller placement in sdn-based wan architecture," in *2018 10th international conference on measuring technology and mechatronics automation (ICMTMA)*. IEEE, 2018, pp. 263–267.

[16] G. Wang, Y. Zhao, J. Huang, Q. Duan, and J. Li, "A k-means-based network partition algorithm for controller placement in software defined network," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–6.

[17] N. Mouawad, R. Naja, and S. Tohme, "Optimal and dynamic sdn controller placement," in *2018 International Conference on Computer and Applications (ICCA)*. IEEE, 2018, pp. 1–9.

[18] C. Gao, H. Wang, F. Zhu, L. Zhai, and S. Yi, "A particle swarm optimization algorithm for controller placement problem in software defined network," in *Algorithms and Architectures for Parallel Processing: 15th International Conference, ICA3PP 2015, Zhangjiajie, China, November 18-20, 2015, Proceedings, Part III 15*. Springer, 2015, pp. 44–54.

[19] M. Obadia, M. Bouet, J.-L. Rougier, and L. Iannone, "A greedy approach for minimizing sdn control overhead," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2015, pp. 1–5.

[20] A. Jalili, M. Keshtgari, and R. Akbari, "A new set covering controller placement problem model for large scale sdns," *Information Systems & Telecommunication*, vol. 25, 2018.

[21] H. Aoki and N. Shinomiya, "Controller placement problem to enhance performance in multi-domain sdn networks," in *Proc. ICN*, 2016, p. 120.

[22] S. Rahman, T. Ahmed, S. Ferdousi, P. Bhaumik, P. Chowdhury, M. Tornatore, G. Das, and B. Mukherjee, "Virtualized controller placement for multi-domain optical transport networks using machine learning," *Photonic Network Communications*, vol. 40, pp. 126–136, 2020.

[23] J. Liao, H. Sun, J. Wang, Q. Qi, K. Li, and T. Li, "Density cluster based approach for controller placement problem in large-scale software defined networkings," *Computer Networks*, vol. 112, pp. 24–35, 2017.

[24] T. Wang, F. Liu, J. Guo, and H. Xu, "Dynamic sdn controller assignment in data center networks: Stable matching with transfers," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[25] H. Sufiev, Y. Haddad, L. Barenboim, and J. Soler, "Dynamic sdn controller load balancing," *Future Internet*, vol. 11, no. 3, p. 75, 2019.

[26] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic approaches to the controller placement problem in large scale sdn networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4–17, 2015.

[27] K. S. Sahoo, A. Sarkar, S. K. Mishra, B. Sahoo, D. Puthal, M. S. Obaidat, and B. Sadun, "Metaheuristic solutions for solving controller placement problem in sdn-based wan architecture," in *ICETE 2017-Proceedings of the 14th International Joint Conference on e-Business and Telecommunications*, 2017.

[28] G. Ramya and R. Manoharan, "Prediction based dynamic controller placement in sdn," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 8, no. 32, 2021.

[29] L. Liao and V. C. Leung, "Genetic algorithms with particle swarm optimization based mutation for distributed controller placement in sdns," in *2017 IEEE conference on network function virtualization and software defined networks (NFV-SDN)*. IEEE, 2017, pp. 1–6.

[30] Y. Li, S. Guan, C. Zhang, and W. Sun, "Parameter optimization model of heuristic algorithms for controller placement problem in large-scale sdn," *IEEE Access*, vol. 8, pp. 151 668–151 680, 2020.

[31] S. Mohanty, P. Priyadarshini, B. Sahoo, and S. Sethi, "A reliable capacitated controller placement in software defined networks," in *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE, 2019, pp. 822–827.

[32] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao, "Are we ready for sdn? implementation challenges for software-defined networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36–43, 2013.

[33] V. Huang, G. Chen, Q. Fu, and E. Wen, "Optimizing controller placement for software-defined networks," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 224–232.

[34] A. Dvir, Y. Haddad, and A. Zilberman, "The controller placement problem for wireless sdn," *Wireless Networks*, vol. 25, pp. 4963–4978, 2019.

[35] H. Wang, F. R. Yu, L. Zhu, T. Tang, and B. Ning, "Finite-state markov modeling for wireless channels in tunnel communication-based train control systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 3, pp. 1083–1090, 2014.

[36] I. Ivanov, G. Hristov, and V. Stoykova, "Algorithms for optimizing packet propagation latency in software-defined networks," in *IOP Conference Series: Materials Science and Engineering*, vol. 1031, no. 1. IOP Publishing, 2021, p. 012072.

Appendix

A.1 Topology Matrix of AAU

Table A.1: matrix of topology table

Inf	**773.881998**	**1413.680485**	**1254.966893**	**600.9837953**	**708.015126**	**182.9691689**
773.882	Inf	Inf	Inf	Inf	Inf	Inf
1413.68	Inf	Inf	Inf	Inf	Inf	Inf
1254.967	Inf	Inf	Inf	Inf	Inf	Inf
600.9838	Inf	Inf	Inf	Inf	Inf	Inf
708.0151	Inf	Inf	Inf	Inf	Inf	Inf
182.9692	Inf	Inf	Inf	Inf	Inf	Inf

A.2 Distance Matrix of AAU

Table A.2: matrix of distance table

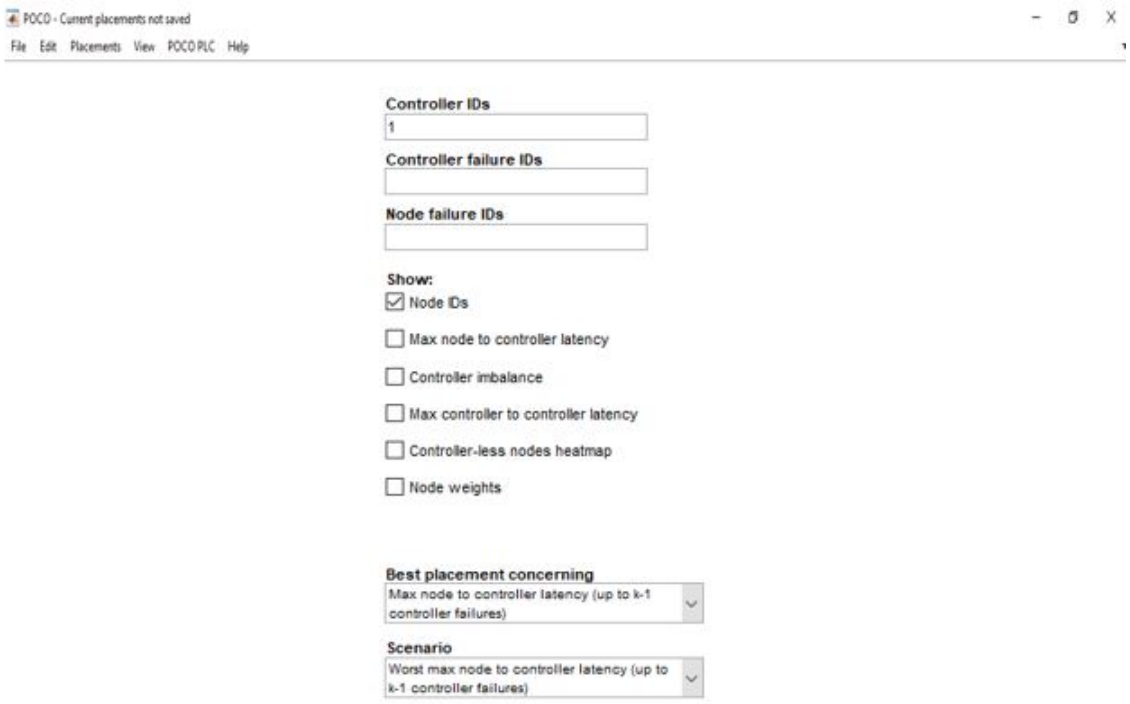
0	**773.881998**	**1413.680485**	**1254.966893**	**600.9837953**	**708.015126**	**182.9691689**
773.882	0	Inf	Inf	Inf	Inf	Inf
1413.68	Inf	0	Inf	Inf	Inf	Inf
1254.967	Inf	Inf	0	Inf	Inf	Inf
600.9838	Inf	Inf	Inf	0	Inf	Inf
708.0151	Inf	Inf	Inf	Inf	0	Inf
182.9692	Inf	Inf	Inf	Inf	Inf	0

Table A.3: Coordinate Matrix of AAU

Longitude	Latitude
38.7578781	9.04458048
38.7637079	9.04067733
38.7637328	9.03326829
38.7641964	9.04332907
38.7658859	9.03653033
38.7591028	9.04347349
38.7626955	9.04714128

A.3 Coordinate Matrix of AAU

A.4 Sample Controller Placement Metrics and Scenarios



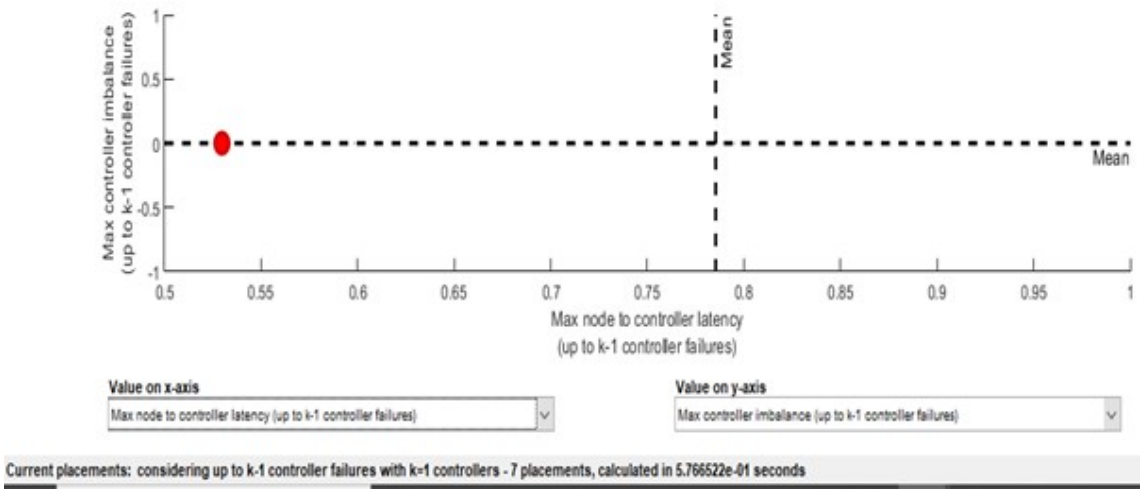


Figure A.1: Sample Controller Placement Metrics and Scenarios