

Optimization Placement for SDN Controller: Bell Canada as a Case Study

by

Ibrahim Elabani

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2017

© Ibrahim Elabani 2017

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The tremendous proliferation of data traffic has been a key motivator for the upgrading of traditional IP networks. One new conceptual model that has been developed for redesigning and managing communication networks is software-defined networking (SDN). The main premise behind SDN is the decoupling of the control and data planes, which enables the centralization of the control plane and the programmability of the data plane. Despite these advantages, the use of SDN remains challenging with respect to a number of aspects, such as finding optimal locations for SDN controllers in a wide area network (WAN) and determining the effective number of controllers. The work presented in this thesis addresses these challenges through two proposed strategies for dealing with the SDN controller placement problem. The Bell Canada WAN was considered as a case study: the network was examined, and the modeled procedures for determining the best location for SDN controllers were applied with the goal of enhancing the quality of service (QoS) and minimizing global latency. The simulations conducted as a means of validating and comparing the performance of the two models produced consistent results.

Acknowledgements

All praises to Allah for the strengths in completing this work. I would like first to express my sincere appreciation to my supervisor, Professor Pin-Han Ho, for his constant support and help.

I am grateful to the Libyan Ministry of Higher Education for providing financial support during my research.

I also would like to express my warmest and deepest appreciation to my beloved family.

Finally, my thanks go to all the people who have supported me to complete the research work directly or indirectly.

Dedication

I would like to dedicate this thesis to my mother and to my brothers and sisters for their support and encouragement all the time.

Table of Contents

List of Tables	viii
List of Figures.....	ix
1 Introduction.....	1
1.1. Motivation.....	1
1.2. Scope of the Research.....	3
1.3. Thesis Organization	4
2 SDN background and Literature Review	5
2.1. Introduction.....	5
2.2. How Traditional Networks Work	6
2.3. Software Defined Networking (SDN)	10
2.3.1. SDN components	10
2.3.2. SDN Network Architecture.....	12
2.4. OpenFlow Protocols.....	14
2.5. Related Work	16
3 System Model	19
3.1. Introduction.....	19
3.2. Proposed algorithms.....	19
3.2.1. <i>k-Median</i> Algorithm.....	20
3.2.2. Johnson's Algorithm.....	21
3.3. Experimental Work.....	22

3.3.1.	Introduction.....	22
3.3.2.	Simulation Tools and Technologies.....	23
4	Case study results and analysis.....	31
4.1.	Case study results and analysis for a mathematical model.....	31
4.2.	Case study results and analysis for a Network analysis procedure.....	38
4.3.	Simulation Source Codes.....	42
4.3.1.	Simulation codes on GitHub.....	42
4.3.2.	Examples of Simulation codes.....	42
5	Conclusions and Future Work.....	44
5.1.	Conclusions.....	44
5.2.	Future Work.....	45
	References.....	46

List of Tables

Table 1.1: Cisco Projection for Internet Traffic [1].....	2
Table 2.1: OpenFlow protocol Messages.....	15
Table 2.2: OpenFlow entry in SDN [4]	15
Table 4.1: Average latency for SDN controllers	34
Table 4.2: The average time delay at Thunder Bay location (S22)	38
Table 4.3: The average time delay at St John location (S1)	39
Table 4.4: The average time delay at San Francisco location (S44).....	39
Table 4.5: The average time delay at San Francisco location (S30).....	40

List of Figures

Figure 1.1: The global internet projection based on devices and application categories [1]	2
Figure 2.1: OSI Model [4].....	7
Figure 2.2: Traditional IP network device [4]	8
Figure 2.3: SDN system overview [4].....	9
Figure 2.4: SDN controller functions	11
Figure 2.5: SDN Architecture	12
Figure 2.6: OpenFlow Protocol Communication	14
Figure 3.1: Simulation tools	23
Figure 3.2: Bell Canada WAN networks.....	24
Figure 3.3: Flowchart: Model description	26
Figure 3.4: Simulation the Bell Canada with Mininet and Floodlight SDN controller	28
Figure 3.5: Shows the link description	29
Figure 3.6: Floodlight SDN controller platform.....	30
Figure 4.1: The optimal & worst placement for SDN network when number of controller is one (k=1).....	32
Figure 4.2: The optimal & worst placement for SDN network when number of controller is one (k=3).....	33
Figure 4.3: The optimal & worst average latency for SDN network with number of controllers (k)	35

Figure 4.4: The optimal latency for SDN network with number of controllers (k)	36
Figure 4.5: The optimal latency for SDN network with number of controllers (k)	37
Figure 4.6: The Total average delay for SDN controllers	41
Figure 4.7: The best and worst locations for SDN controllers	41
Figure 4.8: Matlab code for best and worst locations for SDN controllers	42
Figure 4.9: Python code for best and worst locations for SDN controllers	43

Chapter 1

Introduction

The software-defined network (SDN) has designed as a powerful platform to incorporate into the design of future networks. The key technical and operational benefits of SDN technology are plane separation, centralized control, network automation, and virtualization. The Internet of Things (IoT) is driving the proliferation of smart devices which maintain a nearly constant connection to the Internet. This proliferation of connected devices is also driving the demand for network capacity and because of this organizations are forced to install new classical IP network devices. Because the data and control planes coexist together, traditional networks today are complex and challenging to manage. In the case of any communication link failures, a traditional network is difficult to configure according to predefined policies [1]. Another impact that a traditionally architected IP network faces is that those networks are not well suited for the proliferation of cloud computing services. To conclude, networks that utilize SDN technology are leading the architectural transition from static to dynamic based network architectures.

1.1. Motivation

The global IP data traffic will nearly triple from 2016 to 2021, the recent forecast project done by Cisco shows the overall internet traffic is predicted to increase to 278 Exabytes (EB) per month,

up from 96 EB per month [1]. The Internet traffic forecast in Table 1.1 and depicted in Fig 1.1. These projections show the tremendous growth in IP traffic in coming 5 years.

Table 1.1: Cisco Projection for Internet Traffic [1]

Year	Universal Internet Traffic
1992	100 GB per day
1997	100 GB per hour
2002	100 GB per second
2007	2000 GB per second
2016	26,600 GB per second
2021	105,800 GB per second

The figure below shows the global internet projection based on devices and application categories.

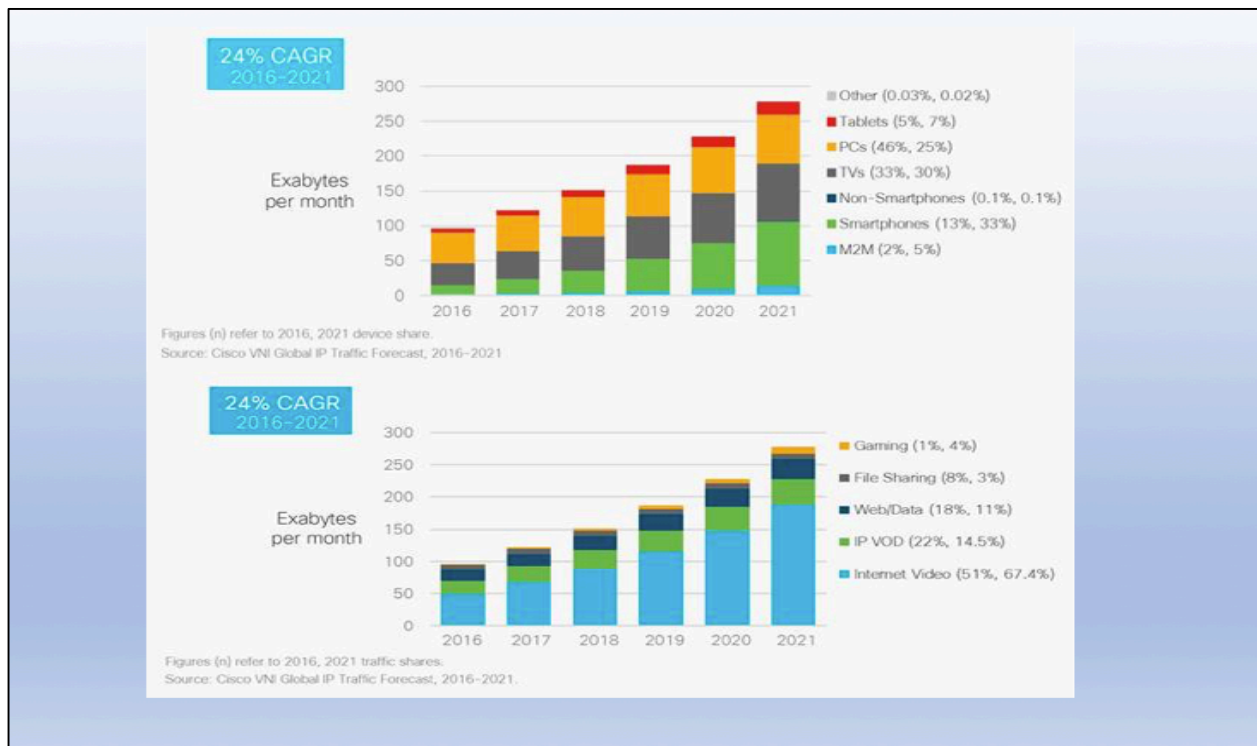


Figure 1.1: The global internet projection based on devices and application categories [1]

The primary motivation for the work presented in this thesis was a desire to improve the quality of internet networks. Due to the forecast high demand for increased IP traffic rates in the future, SDNs are receiving considerable attention with respect to transforming traditional IP networks so that they become more efficient, manageable, and automated. The research presented here was directed at finding an effective solution that would facilitate the transition to SDN by an internet service provider (ISP).

1.2. Scope of the Research

The main contributions of this thesis are:

1. This research presents a new case study of SDN network topology, our experiment was based on a mathematical and complex network analysis solutions to find the optimum location for SDN controller. This model analyzes the SDN network in terms of propagation delay.
2. An implementation of the proposed WAN network is conducted for the case of Bell Canada WAN network. The outcomes from this study are a cornerstone for any ISP's wants to move toward SDN network.
3. A web-based interface for SDN controllers is implemented by using Floodlight SDN controller to manage all Openflow controller.

1.3. Thesis Organization

This thesis is organized into four more chapters. Chapter 2 introduces the history of computer network and a summary of related work about SDN controller placement problem is presented at the of Chapter 2. Chapter 3 is dedicated for the proposed algorithms to tackle the SDN controller placement problem and the second part of this chapter presents the experimental work. Chapter 4 describes the case study work and results obtained to assess Bell Canada case study to find an optimal SDN controllers locations and following the methodology described in Chapter 3. And finally, Chapter 5 summarize the thesis results, conclusion, and future work are also provided.

Chapter 2

SDN background and Literature Review

2.1. Introduction

Advances in the computer industry over the last several decades have resulted in much greater changes in computing paradigms than in computer network architecture and design, which have remained relatively unaltered since the 1990s. The vast majority of multivendor legacy systems that are still operational around the world lack reliable remote identification, fault resolution, and troubleshooting capabilities. The absence of these features, which is considered a major issue facing international providers [2], makes computer networks difficult to manage, automate, customize, and optimize. The development of software-defined networking (SDN) in the first decade of this century by a group of researchers at Stanford University was predicated on the tackling of problems associated with legacy network designs. The main concept underlying SDN technology is the decoupling of the control plane from the data plane, which provides the benefit of being able to centralize a programmable control plane. The advantage of using SDN technology is the associated reduction in operational expenditures (OPEX). In [3] Kirsh Prabu, CTO and president of AT&T Labs, stated that, upon completion of the company's move to an SDN architected network in 2020, he expects to save 40 % to 50 % in OPEX. Prabu attributed these cost savings to the replacement of manual operations with automated scripts and procedures.

An SDN thus not only makes it easier and faster to create a new service but is also a key component in reducing service providers' OPEX [3]. SDN technology also offers dynamic management, initialization, and control of network behavior.

2.2. How Traditional Networks Work

What follows is a brief explanation of how traditional networks operate today and the history of how SDN was developed to address the operational and technical deficiencies in those traditional networks. A traditional network mainly is a mix of routers and switches that facilitate the transfer of packets from one part of the network to another. The routers use routing protocols to move these packets across the network efficiently. Packets contain sets of data transferred over a physical link (the network cables). At the transceiver side, data is divided into chunks based on which data link technique has been used to transfer the packets. On the destination side, the receiver reassembles the data, and the control plane in each router and switch determines the final destination of each packet based on the routing protocols deployed on the network. As shown in Fig (4.1), The Open Systems Interconnection model (OSI model) is a standard for defining the communication functions in telecommunications or computing systems.

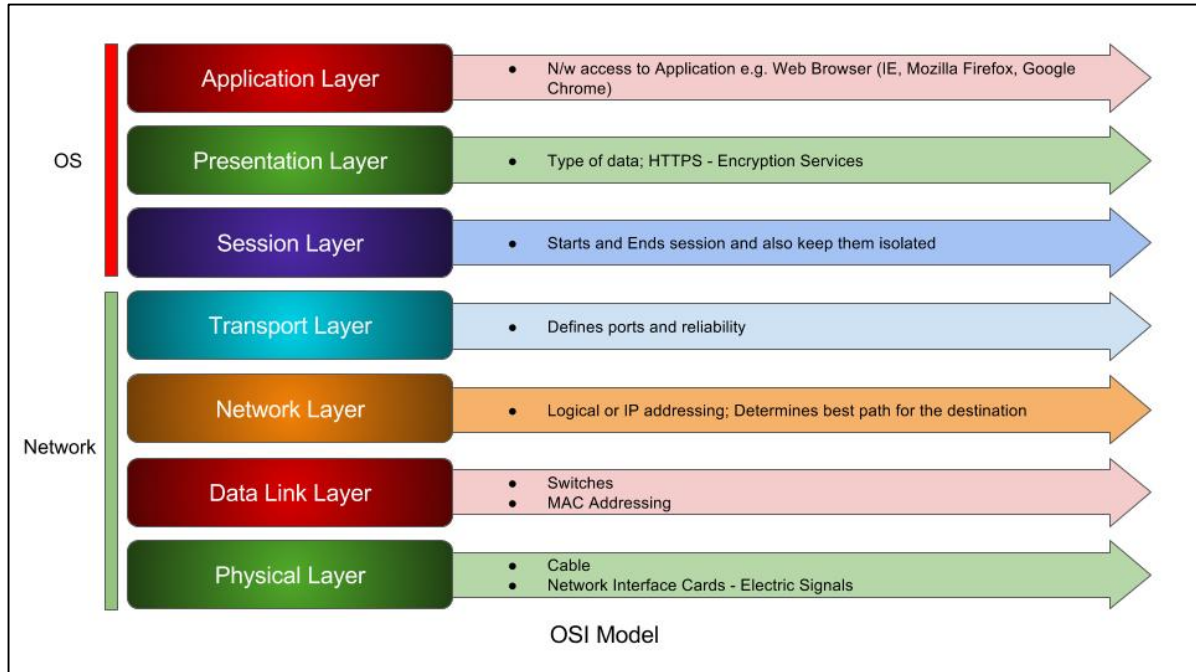


Figure 2.1: OSI Model [4]

IP network devices today have to decide what to do with packets received from another device in the network whether those devices are servers, a computer, or any other type of IP device connected to the network. In the extensive IP-based networks of today, that have thousands of devices connected to them, there is a need to move from local decisions made on the device itself, which may lead to some delays, to process the incoming packets utilizing a centralized programmable network. The method described above is the how the Internet works today where every IP device on the network is self-sufficient.

Traditional IP network devices like routers and switches have three logic planes as shown in the figure below.

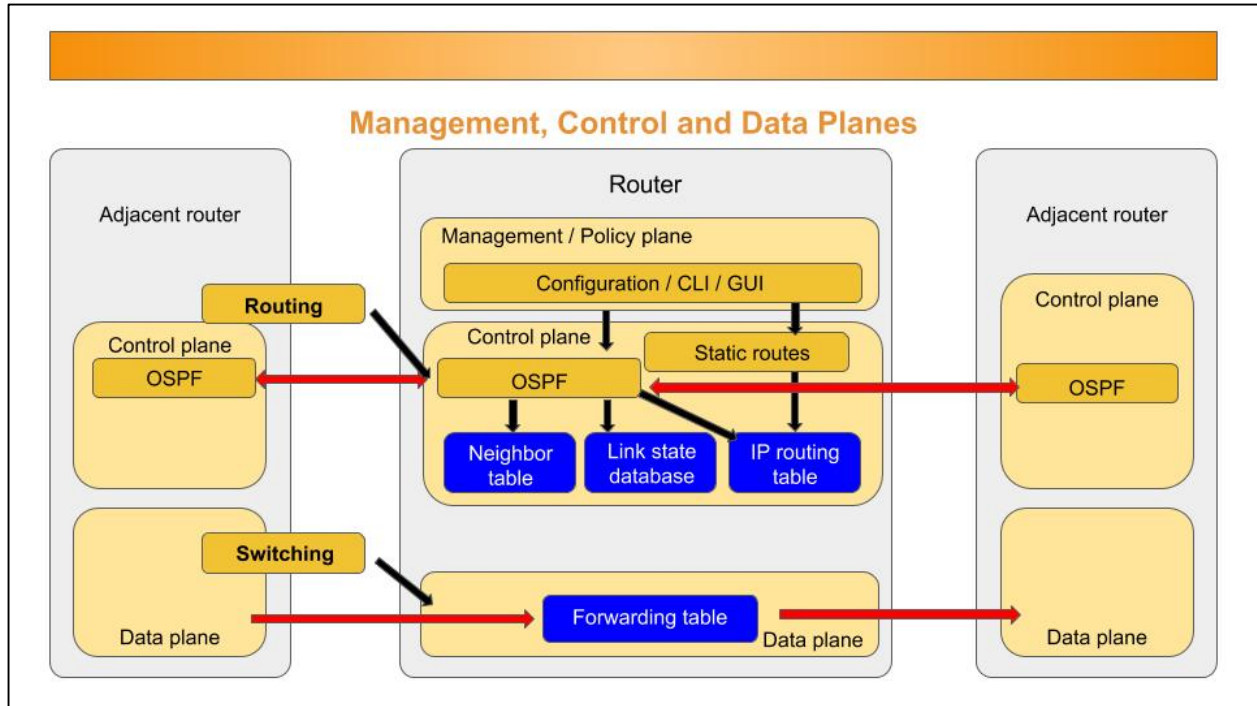


Figure 2.2: Traditional IP network device [4]

The management plane is the plane that stores the software services that are responsible for managing and configuring control plane functionality.

The control plane maintains a set of routing protocols such as Open Shortest Path First (OSPF) or the Border Gateway Protocol (BGP). These routing protocols are responsible for forwarding routing data to an appropriate destination router. Forwarding tables located on IP connected devices serve as the basis for these routing calculations.

The purpose of the data plane is to forward a packet from one network interface operational on a single device to one of the other operational interfaces on the same device.

In traditional networks, the introduction of network policies takes place in the management plane, the control plane executes these policies, and the data plane transfers data based on those policies [5]. Doherty describes the relationship between the control and forwarding planes. He states, "You can think of the control plane as the brain and the forwarding plane as the muscle." [6].

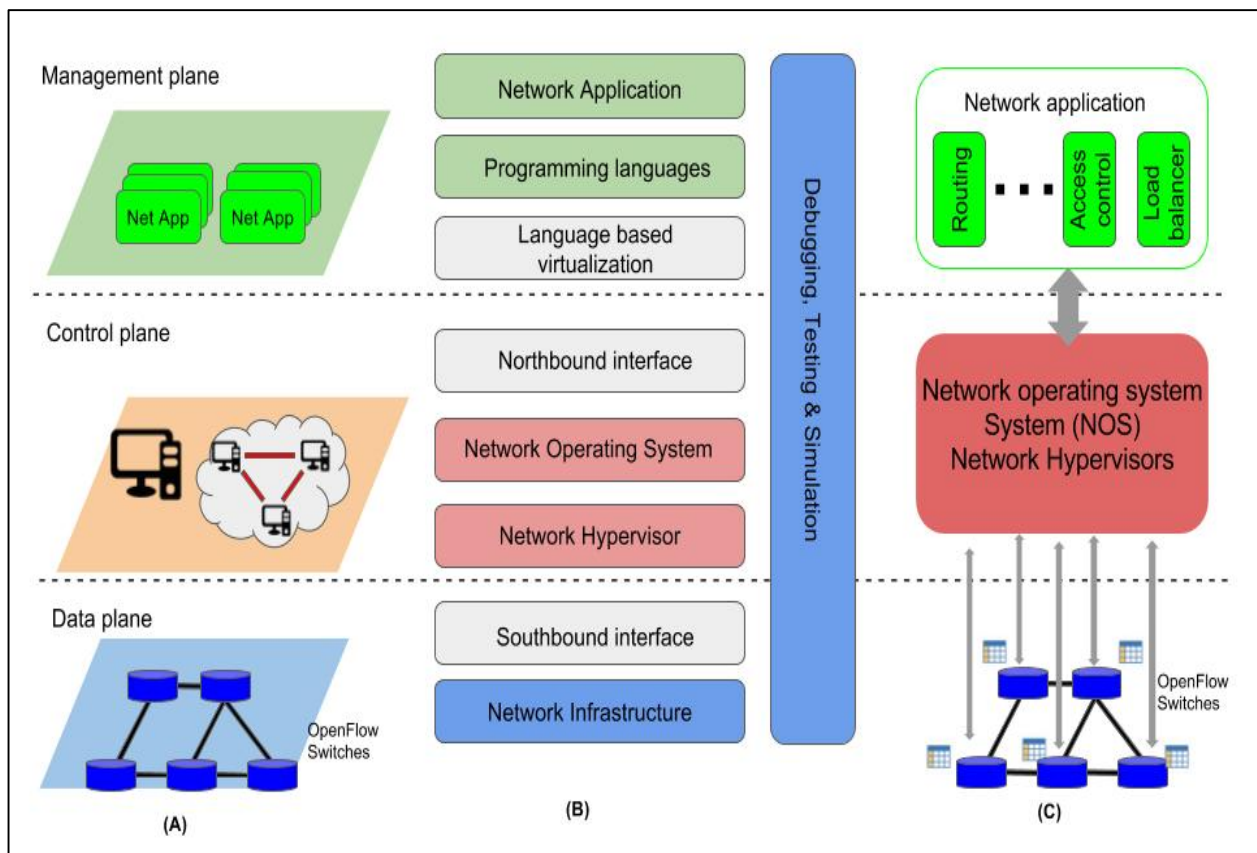


Figure 2.3: SDN system overview [4]

2.3. Software Defined Networking (SDN)

SDN is a new way to implement an Internet network that is manageable, dynamic, and cost-effective. The power of this new technology is driving big technology companies like Facebook, Microsoft, and Google to fund the Open Networking Foundation (ONF). The ONF is an organization that was established to accelerate and solve the issues that are faced by traditionally architected IP networks. ONF has defined SDN as a networking technology which allows for centralized, programmable control planes that permit network operations organizations to directly monitor and manage their own virtualized networks [7]. The basic premise of an SDN architecture is the separation of the two most important elements in IP network devices, the control plane and the data plane.

2.3.1. SDN components

SDN has two main operational components: the controllers and forwarding elements.

The SDN controller

As we know from SDN architecture, the SDN controller has visibility of all the network elements in the network; more precisely, it is functioning as a brain to the SDN system. In traditional IP networks, with a link failure, the network devices on the network with the link failure need to update their routing tables and swap the new routing tables with all IP devices on the network. Those devices then recalculate the best routes and share new routing information, which introduces network delay, or latency. Using SDN technology, the SDN controller can connect, interpret

routes and share alternative routes for all links, in the event of any failures. This approach for finding a new route, or path, is faster and more resilient than the standard Interior Routing Protocols like an Open Shortest Path First (OSPF) or Routing Information Protocol (RIP) found in traditional IP Networks. As a result, the SDN controller maintains an alternative shortest path already in the flow table, so there is no need to recalculate a new path in case of link failures; hence, there is no time required to update the routing tables or compute any routing algorithms. The SDN controller can manage and program the forwarding devices via southbound interfaces as shown in the figure below.

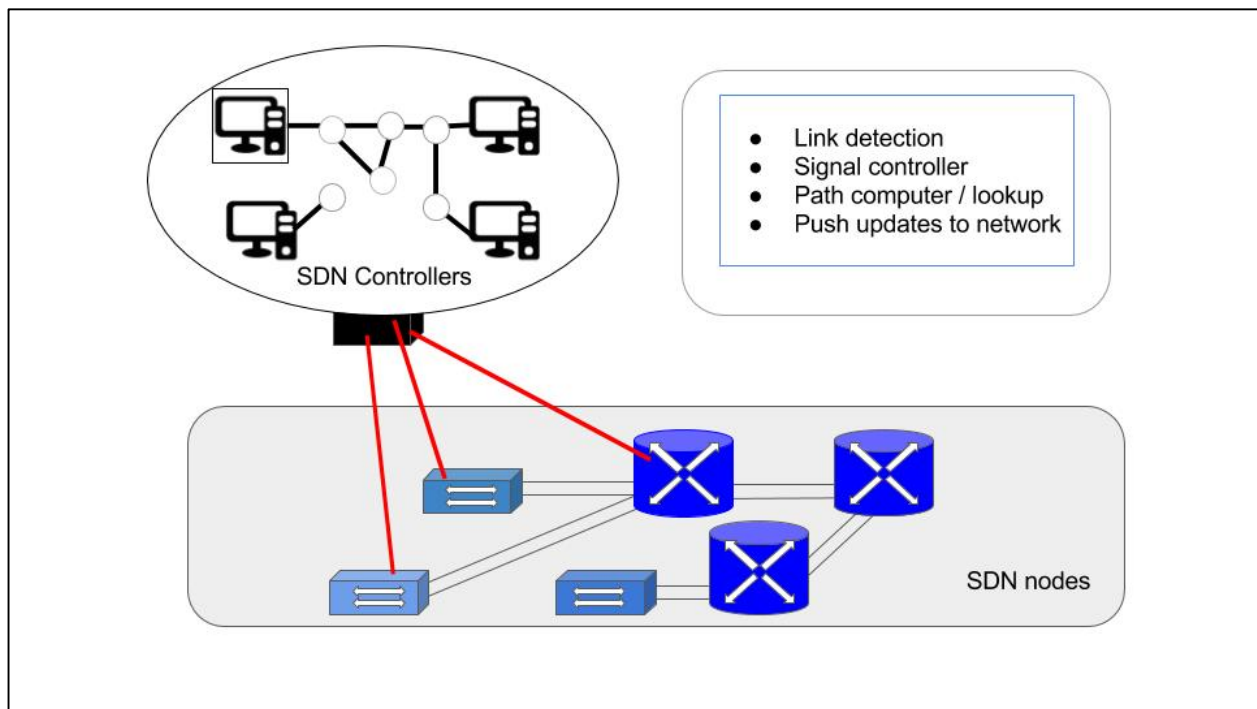


Figure 2.4: SDN controller functions

In the SDN market today, there are several commercial and open source controllers. Large networking companies such as Cisco, Juniper, VMWare, and HP sell deployable SDN controllers, some of which are architecturally proprietary and closed. Open source SDN controllers exist as well and are available such as OpenDaylight, Floodlight, and OpenContrail.

The Forwarding Elements

The forwarding elements are responsible for transporting user data among forwarding devices. The packets exchanged between forwarding elements do not have any IP source or destination addresses related to the forwarding elements (switches, routers). The forwarding packets only contain the addresses for endpoints [8]. Forwarding elements only connect to a controller via the southbound interface and only has data plane.

2.3.2. SDN Network Architecture

The general SDN architecture includes three different planes: The Control plane, the data plane, and the application plane. The figures below illustrate a typical SDN architecture.

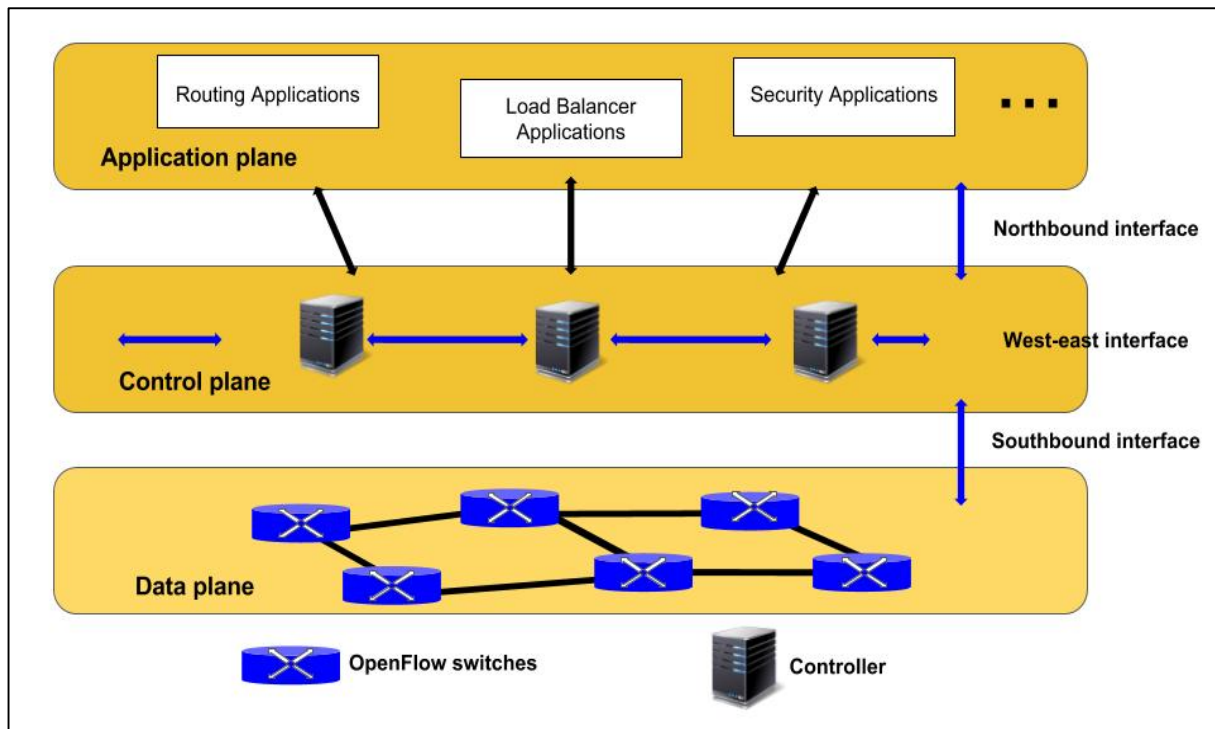


Figure 2.5: SDN Architecture

Control Plane

The control plane is responsible for creating all the network routes based on OpenFlow protocols. OpenFlow protocols are used to exchange information between the SDN controller and forwarding elements. The control plane also has responsibility for transferring flow policies to the forwarding table, and the data plane maintains those forwarding tables. OpenFlow protocols are responsible for flow measurement and analysis [9]. The tasks of the control plane are:

- Connection setup to forwarding devices.
- Proactive flow programming and the installation of forwarding rules to the data plane layer.
- Building a network view.
- Maintenance of route selection and data plane device availability on the network.

Data Plane

The forwarding plane includes physical and virtual devices. The main function of the forwarding plane is to forward the packets between these elements based on the route contained in flow tables for each device. The forwarding plane became simpler packet forwarding elements, there is no complex algorithm to be executed in this plane, the data plane is just a simple plane for forwarding data [10].

Application Plane

The SDN controller interacts with the application plane via an Application Program Interface (API) through the SDN controller's northbound interface (NBI). SDN applications are programs that directly, programmatically, and explicitly communicate the network demand and requests to

the SDN controller via NBIs [7]. This layer is responsible for handling network services like traffic engineering, quality of service (QoS), and security services.

2.4. OpenFlow Protocols

OpenFlow is a protocol that enables the communication between the data and control planes; the process also collects the implementation details of the network elements. OpenFlow uses a defined TCP port to establish a communication channel to the SDN controller, and then the OpenFlow protocol authorizes an SDN controller to proactively share the network policy to flow tables contained in forwarding devices.

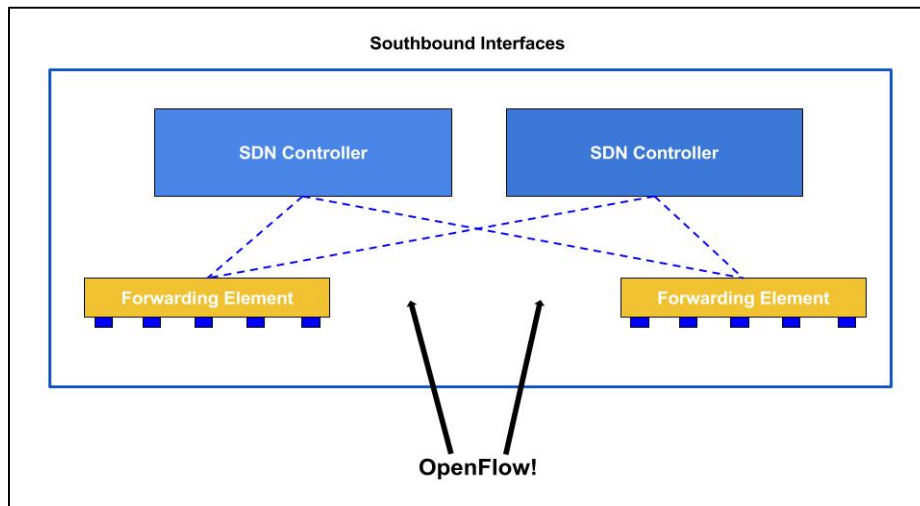


Figure 2.6: OpenFlow Protocol Communication

There are three types of OpenFlow protocol messages:

Table 2.1: OpenFlow protocol Messages

Message Types	Description	Examples
Controller-to-Switch	Initiated by a controller. OpenFlow switches may not need to respond.	Read State
		Modify State
		Packet Out
		Configuration
		Features
Asynchronous	Initiated by OpenFlow switch without solicitation from the controller	Error
		Packet In
		Flow Removed
		Port Status
Symmetric	Initiated in either direction without solicitation	Hello
		Echo
		Experimenter

Each Forwarding device consists of a group of flow tables that are responsible for forwarding a packet to the right destination. The figure below illustrates a flow table entry in SDN technology:

Table 2.2: OpenFlow entry in SDN [4]

Priority	Match	Action
2	tcp_dst:22	forward [1]
1	eth_dst: 0xababababab	forward []
1	eth_dst: 0xcdcdcdcdcdc	forward [2,3]
0	eth_dst: 0xefefefefefefe	set(eth_src=0) ; forward [1]

The flow table has a set of precedence rules to execute first. The matched field consists of an ingress port and a packet header. After successfully matching the packet in the flow table, the process then looks for an appropriate action to execute based on the action type, such as forward to a destination address or enforce a QoS rule. In cases where the packet does not match in the flow table, the forwarding table process returns that result to a controller. OpenFlow protocols have different versions that range from version 1.0 to version 1.5. These various versions introduce new or enhanced capabilities for the platforms that use OpenFlow.

2.5. Related Work

This section contains a brief discussion of the relevant studies related to the SDN controller placement problem. The primary emphasis of the work presented in this research was on the optimization of average latency in large-scale SDN networks.

SDN network deployments usually require several SDN controllers, whose placement in the infrastructure affects SDN operational characteristics. When the architecture of an SDN-based WAN is designed, the optimal placement of the SDN controllers is derived based on a non-deterministic polynomial (NP)-hard problem. Given this constraint, the selection of an effective controller placement algorithm is critical [11]. Most approaches for addressing this issue involve heuristic solutions for finding the optimal solution.

Since the first introduction of the SDN controller issue by Heller et al. in 2012 [12], many researchers have proposed different algorithms for dealing with one of the most difficult problems facing an engineer with respect to deploying an SDN network: the placement of controllers in the

network. A formidable challenge associated with solving SDN controller placement problems is that all of the algorithms proposed involve a tradeoff among scalability, resilience, and model expansion. With respect to investigations of the SDN controller problem, the technical paper published by Heller et al. [12] is one of the most cited. The authors proposed a heuristic approach for finding the optimum controller positions in large-scale SDN deployments. The main metric formulated in this study was average-case latency, which is deemed essential for determining latency values in large-scale SDN implementations. The approach is dependent primarily on propagation delay, with the location of a controller being based on the shortest path between switches and controllers that have been assigned in the network topology. This study offered the most accurate solution for addressing the problem. An interesting conclusion was that increasing the number of controllers does not necessarily decrease the average latency between switches and assigned controllers.

Aoki et al. [13] examined the fundamental issues related to SDN controller placement and presented a new way of addressing controller problems by first dividing the SDN network into different domains and then locating each controller in an appropriate domain. The authors proposed a greedy algorithm for linking each domain to a controller and then demonstrated the most commonly used metric for optimizing a controller in multiple domains based on a determination of the shortest path between a switch and a controller in each domain.

In [14], Bari et al. proposed a new framework for deploying multiple controllers, which they called the Dynamic Controller Provisioning Problem (DCPP). DCPP entails dynamically adapting the number of controllers and links that are active while maintaining consideration of the state of the network. The researchers formulated DCPP as an integer linear program (ILP). Because a greedy

knapsack algorithm can generate overhead for the existing configuration between switches and controllers, they defined two heuristic algorithms (DCP-GK and DCP-SA) to deal with the framework they developed in order to obtain the best possible performance and accuracy.

Hu et al. [11] studied the SDN controller placement problem by focusing on maximizing the reliability of the SDN control network. They introduced an integer programming formulation for Reliable Controller Placement (RCP). Their parameters specified a determination of both the shortest path between the controller and the forwarding elements, and the lowest probability of control path failure.

In conclusion, all of the controller placement approaches proposed in the literature reviewed were based predominantly on the use of just one or two input metrics for finding the optimal controller placement. Most of these solutions relied on mathematically based models. It is clear that a greedy approach that improves reliability also minimizes the probability of failure while maintaining the shortest distance between installed controllers and switches.

In the next chapter, we discuss the optimal placement for an SDN controller on a new network utilizing the research conducted previously on the SDN Controller placement problem. To better illustrate these complex concepts a network case study from Bell Canada is presented.

Chapter 3

System Model

3.1. Introduction

This chapter presents the methodologies that were used for finding the optimal placement of an SDN controller and for ascertaining the number of controllers needed in the Bell Canada WAN in order to achieve the best quality of service (QoS). The determination of the best controller locations for a large-scale SDN is still a significant target in SDN research. Our proposed solution for minimizing global latency and ensuring the best QoS is to use a clustering approach based on a k-median algorithm, a method widely used for addressing a facility location problem through consideration of the minimum distances between a variety of network locations [14].

3.2. Proposed algorithms

The research presented in this thesis involved the development of two algorithms for finding all pairs of the shortest paths in the Bell Canada WAN and for determining the optimal controller placement in their network. A brief introduction to these algorithms follows.

3.2.1. *k*-Median Algorithm

As mentioned previously, our approach for addressing the SDN controller placement problem was to employ a *k*-median algorithm, which utilizes a clustering approach for determining the optimum location of SDN controllers so that the global average delay between a controller and a switch is minimized [15] [16]. The average latency can be found using the following equation:

$$Lavg (S') = \frac{1}{N} \sum_{v \in V} \min d(v, S') \quad [12] \quad (3.1)$$

where

- $Lavg (S')$: Average latency between switch and controller
- N : Number of SDN forwarding elements
- $d(v, S')$: Shortest path from node v to node S

The first step in computing the average latency, $G (V, E)$, is to derive a mathematical formula to represent the Bell Canada network graph: V signifies the number nodes, and E denotes the number of edges (the fiber links). The shortest paths connecting each pair of all of the forwarding elements must also be determined. Johnson's algorithm provides a means of finding the shortest paths between network pairs and is a well-known method of addressing network optimization [17] in an SDN environment [17]. A *k*-median algorithm is a cluster analysis process that finds the center of a cluster in order to minimize the distances between all nodes [18]. in a network. Table I shows the flowchart for the *k*-median algorithm used in this study.

3.2.2. Johnson's Algorithm

Donald Johnson, from Pennsylvania State University, introduced his algorithm in 1977. Johnson's algorithm, as it is known, is used to find the shortest path to all nodes in the network [17]. We used this algorithm in our work to find all the shortest paths in Bell Canada's networks. The two most important steps in Johnson's algorithms are:

- 1- Adding an artificial source vertex S and a new edge (s, v) with length of zero to the input graph $G = (V, E)$, we will get a new graph called G'
- 2- Run Bellman-Ford algorithm on graph G' with source vertex S to find all shortest paths in the network.

3.3. Experimental Work

3.3.1. Introduction

This section explains our approach to finding an optimization solution for a controller problem. Our solution is based on the results of two experiments conducted in order to verify the optimal SDN controller placement. The goal was to establish how many controllers were needed for the achievement of the minimum delay and to determine the optimal locations for these controllers in the Bell Canada network. A mathematical formulation was used in the first experiment, which was conducted using MATLAB 2015b and MATLAB coding for the minimization of the global average latency in the Bell Canada WAN. We focused only on the real Bell Canada network topology. The purpose of the second experiment was to verify the mathematical formula, for which we used complex network analysis to evaluate the optimal SDN controller placement. The Bell Canada network was implemented using an OpenFlow network emulator called Mininet, along with the Python 2.7 programming language.

3.3.2. Simulation Tools and Technologies

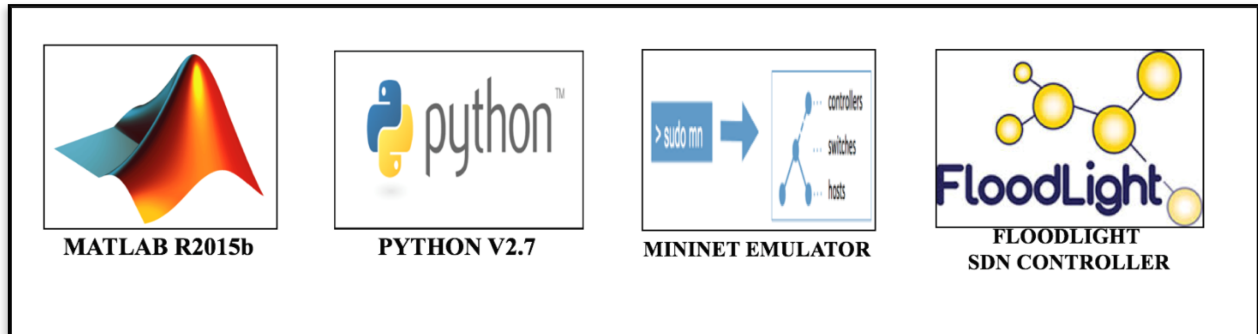


Figure 3.1: Simulation tools

5.3.2.1 Mathematical model description

To maintain realism, our proposed work was applied to a real-world WAN network operated by Bell Canada. The development of the mathematical model was based on the assumption that the following information is known at the beginning of the problem formulation:

- The bandwidth for all fiber links is constant.
- Documentation exists for all WAN network switch locations.

As mentioned above, we use the Bell Canada network topology to integrate our experiment to optimize the global average latency. Shown in the diagram below, we have 48 nodes across Canada and the US. In addition, we have 65 fiber links that connect each node together. The key factor in our mathematical model is the distance while the bandwidth is constant across all sites.

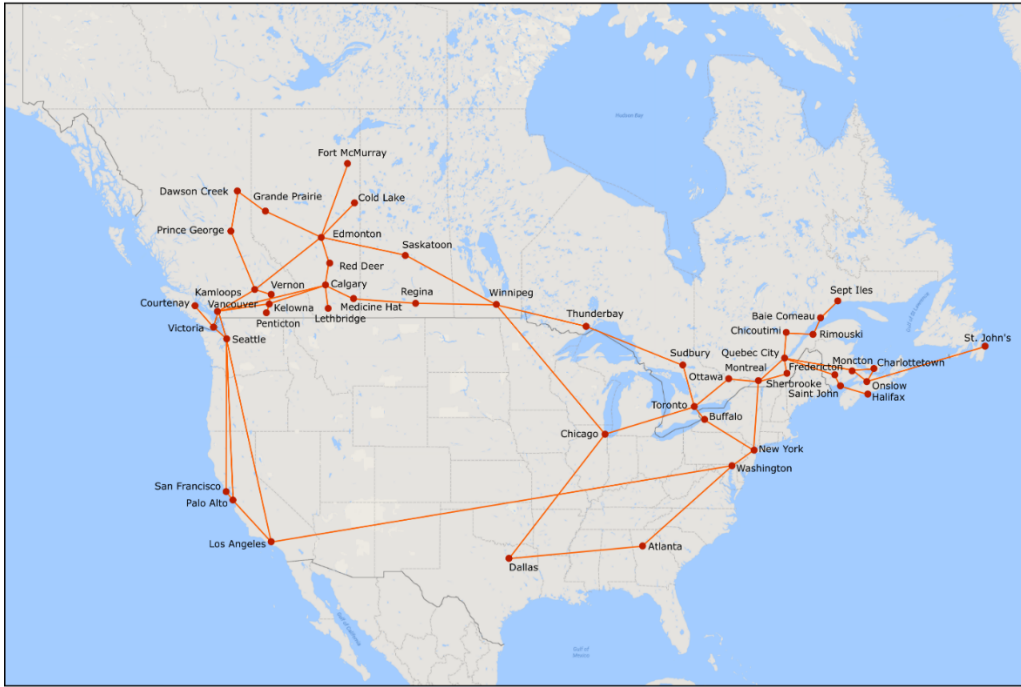


Figure 3.2: Bell Canada WAN networks

In our model, the network topology is defined as an undirected graph $G = N, M$: N refers to network switches, and M represents edges (fiber links). Candidate SDN controllers are placed at switch locations. These controller locations guarantee a minimum delay for all communications between the switches and the SDN controllers. For this study, the minimum average delay and the worst average delay were identified for use in the k-median clustering approach. The determination of these metrics ensures the best QoS based on the number of SDN controllers that manage all network nodes. The flowchart shown in Figure 3.3 summarize the steps in the models. The Geography Markup Language (GML) was used for generating the network topology. The identification of the best controller locations first required a determination of weights for all edges (M). These values were calculated by implementing the adjacency matrix (A) between all connected nodes. Once the weight matrix for all edges was identified, Johnson's algorithm was applied in order to find the shortest path for all nodes N (S_{dist}). At this point, the average

propagation delay ($Lavg$) was computed. The objective is to establish placement S' from the group of candidate controller placements S , with a minimum value for $Lavg (S')$.

$$Lavg (S') = \frac{1}{N} \sum_{v \in V} \min_{(s \in S')} d(v, S') \quad (3.2)$$

where

- $Lavg (S')$: Average latency between switch and controller
- N : Number of SDN forwarding elements
- $d(v, S')$: Shortest path from node v to node S

The second placement metric in the model is the establishment of the worst-case delay, the calculation of which reveals the maximum delay in the network.

$$Lwc (S') = \max_{(v \in V)} \min_{(s \in S')} d(v, s) \quad (3.3)$$

where

- $Lwc (S')$: Worst-case latency between switch and controller
- N : Number of SDN forwarding elements
- $d(v, s)$: Shortest path from node v to node s

For the validation of our optimization solution based on the Bell Canada WAN, the average-case latency and worst-case latency metrics are more important than any other constraint.

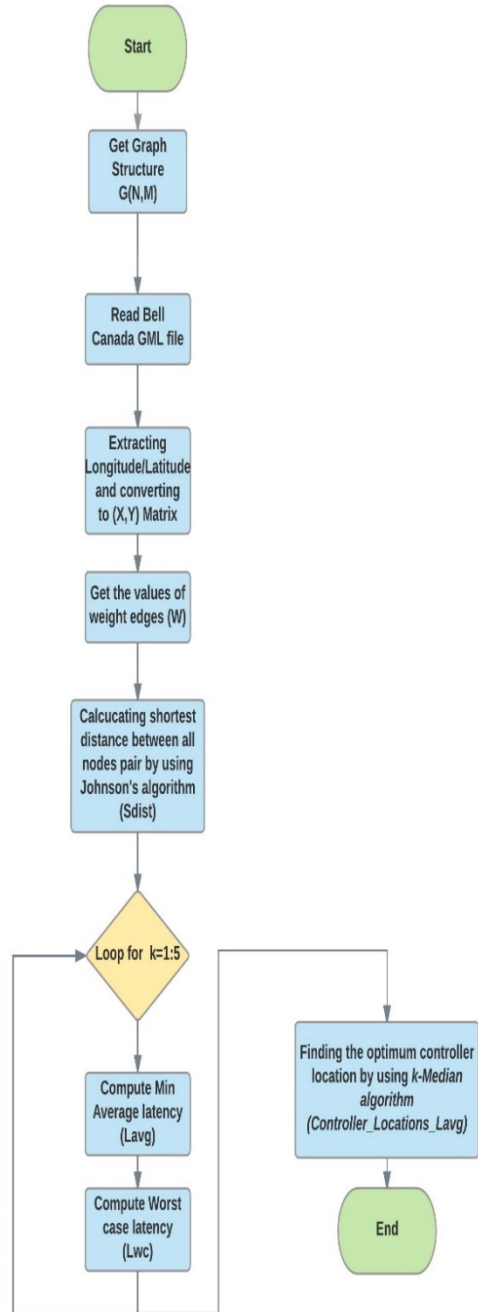


Figure 3.3: Flowchart: Model description

5.3.2.2 Complex network analysis method

This section describes a method for finding optimal and worst-case SDN controller locations based on a well-known application for computer networks: an Internet Control Message Protocol (ICMP). ICMP packets are used for measuring the latency between network layer nodes. An ICMP generates multiple packets from a source to the destination[20] [21], with the outcome being the average latency from one node to another.

In the test topology shown in Figure 3.4. OpenFlow switches were implemented using Mininet, the SDN network emulator employed because it facilitated the development of OpenFlow switches, which were connected with external SDN controllers [22]. In our test environment, a Floodlight controller was used for managing all of the SDN switches in the Bell Canada WAN. Floodlight controllers are Java-based open source software and are among the controllers most widely used in an SDN environment [23].

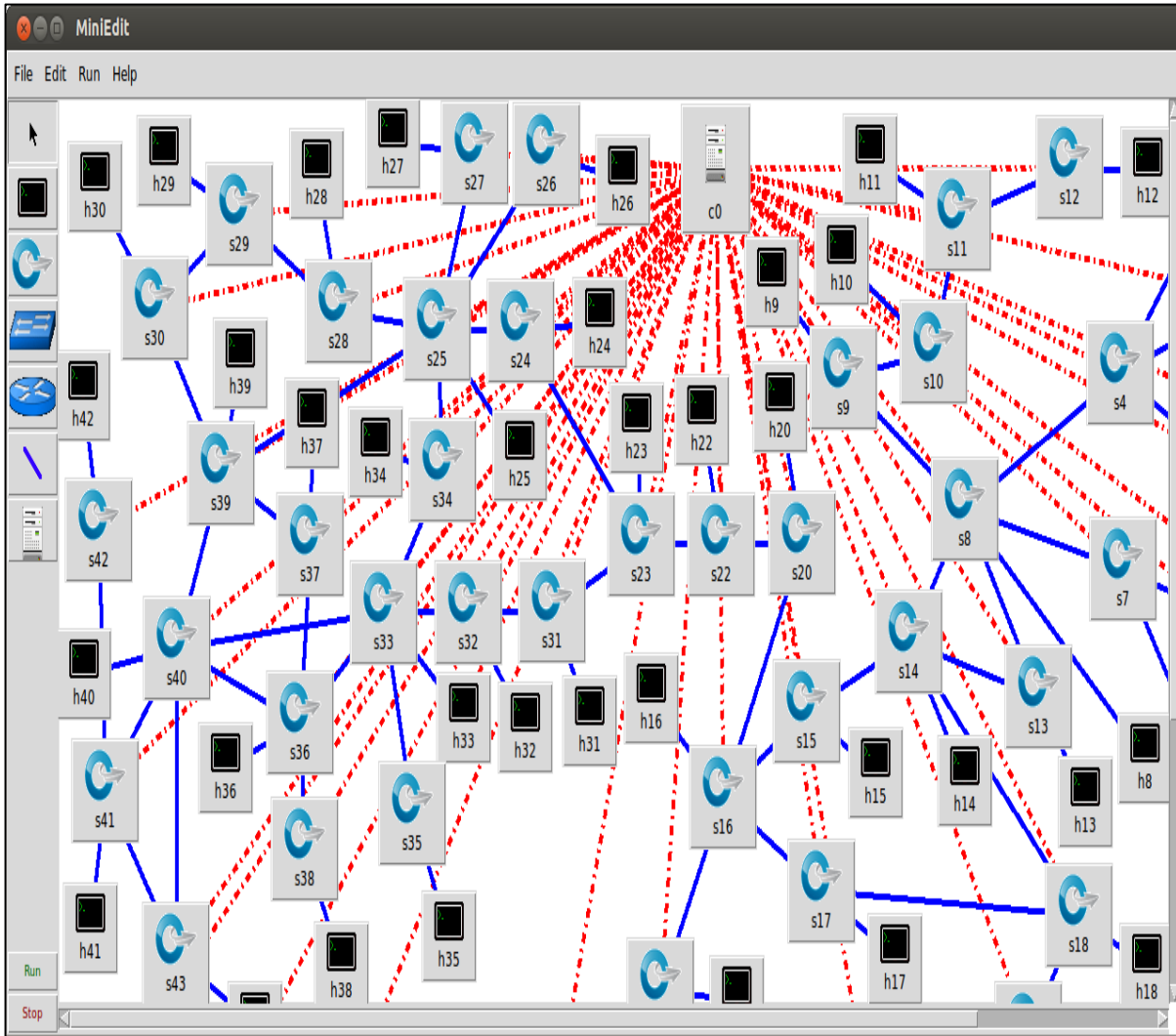


Figure 3.4: Simulation the Bell Canada with Mininet and Floodlight SDN controller

For implementation purposes, three main components were included in the network topology shown in Figure 3.4.

Link description: With respect to the links between the OpenFlow switches, we used a latency formula to compute the solutions because the physical links are known to be fiber optics, for which the propagation delay is almost the speed of light $\left(2 * 10^8 \frac{m}{sec}\right)$. Since all network distances are

known, to maintain authenticity, the propagation delay was calculated and added to our SDN topology:

$$\left(\text{Propagation time} = \frac{\text{Distance (m)}}{\text{propagation speed} \left(\frac{\text{m}}{\text{Sec}} \right)} \text{ (sec)} \right) \quad (3.4)$$

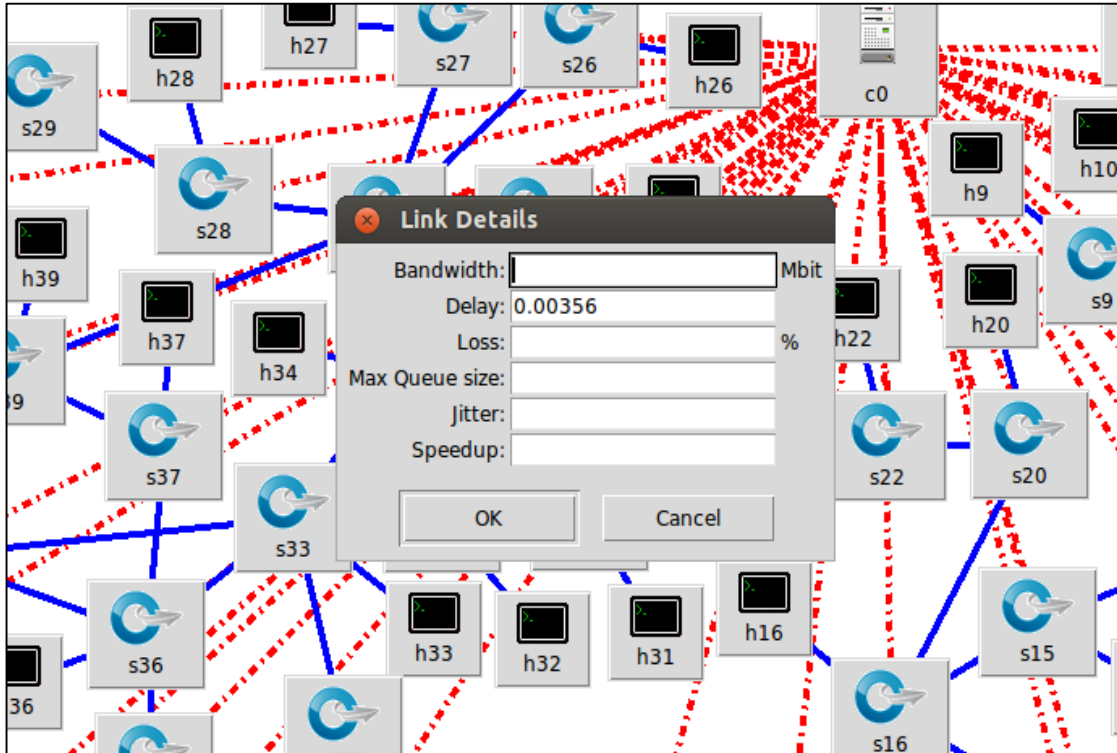


Figure 3.5: Shows the link description

In our simulation, all OpenFlow switches are managed by a Floodlight SDN controller. Figure 3.6. illustrates the Floodlight SDN controller platform, showing all OpenFlow switches, with each switch having a unique datapath ID (DPID).

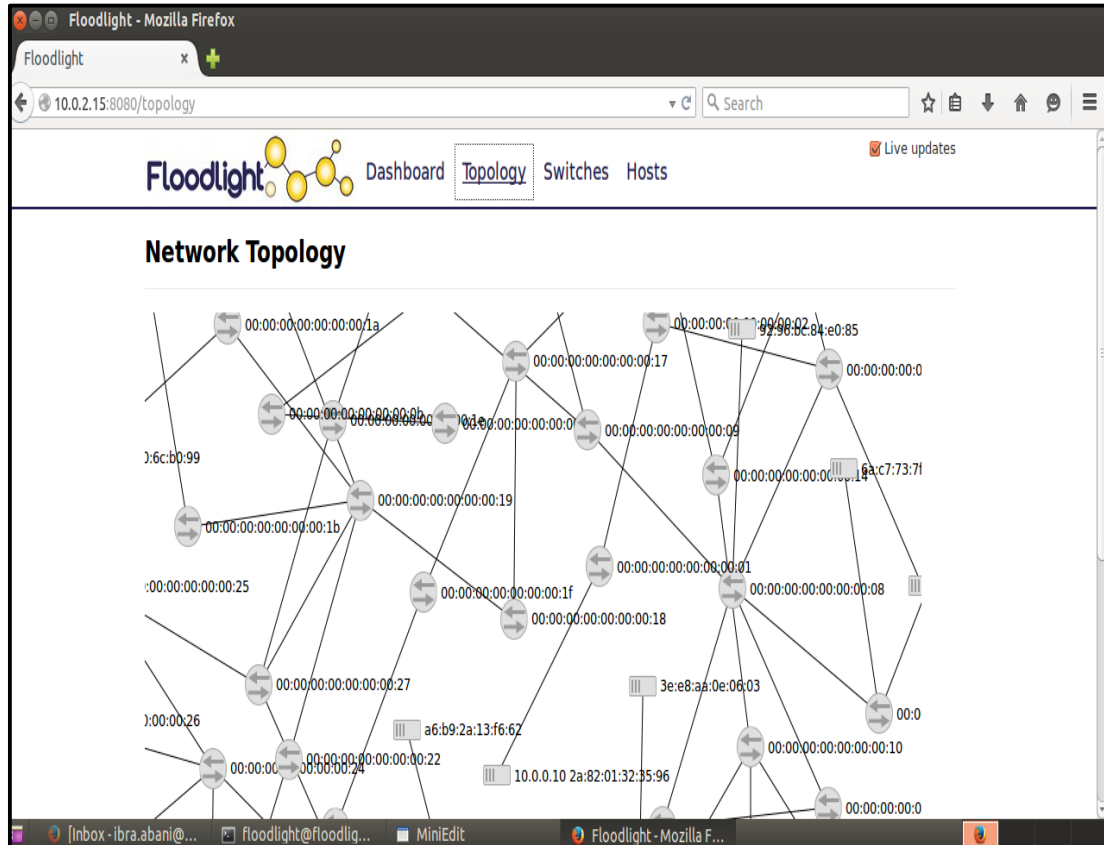


Figure 3.6: Floodlight SDN controller platform

The Bell Canada network, which has 48 nodes and 68 links, was used as a case study for the experiments. The main simulation and development environment for replicating the Bell Canada network involved the use of Python, Mininet, and Floodlight controller simulation tools. Hypothetically, we considered that, for representing the SDN network using these tools and taking into account only the propagation delay, 48 OpenFlow nodes should be the optimal size. To prove our hypothesis, extensive data were collected from the simulation. As mentioned with respect to ICMP pinging, the first step is to install the SDN controller in the first OpenFlow switch node, followed by the computation of the ICMP pinging procedure from this node to all nodes in the Bell network. These steps are then repeated for all of the nodes in the Bell network.

Chapter 4

Case study results and analysis

This section presents and discusses the case study conducted for validation purposes.

4.1. Case study results and analysis for a mathematical model

The models implemented were developed with the goal of assisting Internet service providers (ISPs) who wish to move to SDN so that they can compute the optimum locations for SDN controllers according to the overall network delay. Our assessment of network performance was based on the amount of delay. This section details the model results and the overall outcomes of this study. Fig. 5 shows that the optimal SDN placement when the number of controllers is one ($k = 1$) is the City of Thunder Bay, which is the best location for the controller that has the minimum average latency ($L_{avg} = 0.4459$). This result indicates that Thunder Bay represents a balance point between the western and eastern nodes. The selection of this location ensures the best network performance in our network topology with respect to the communication between the controller and the forwarding devices. In contrast, the worst-case latency with one controller would result from placement in San Francisco, with the worst-case delay being ($L_{wc} = 1.0412$).

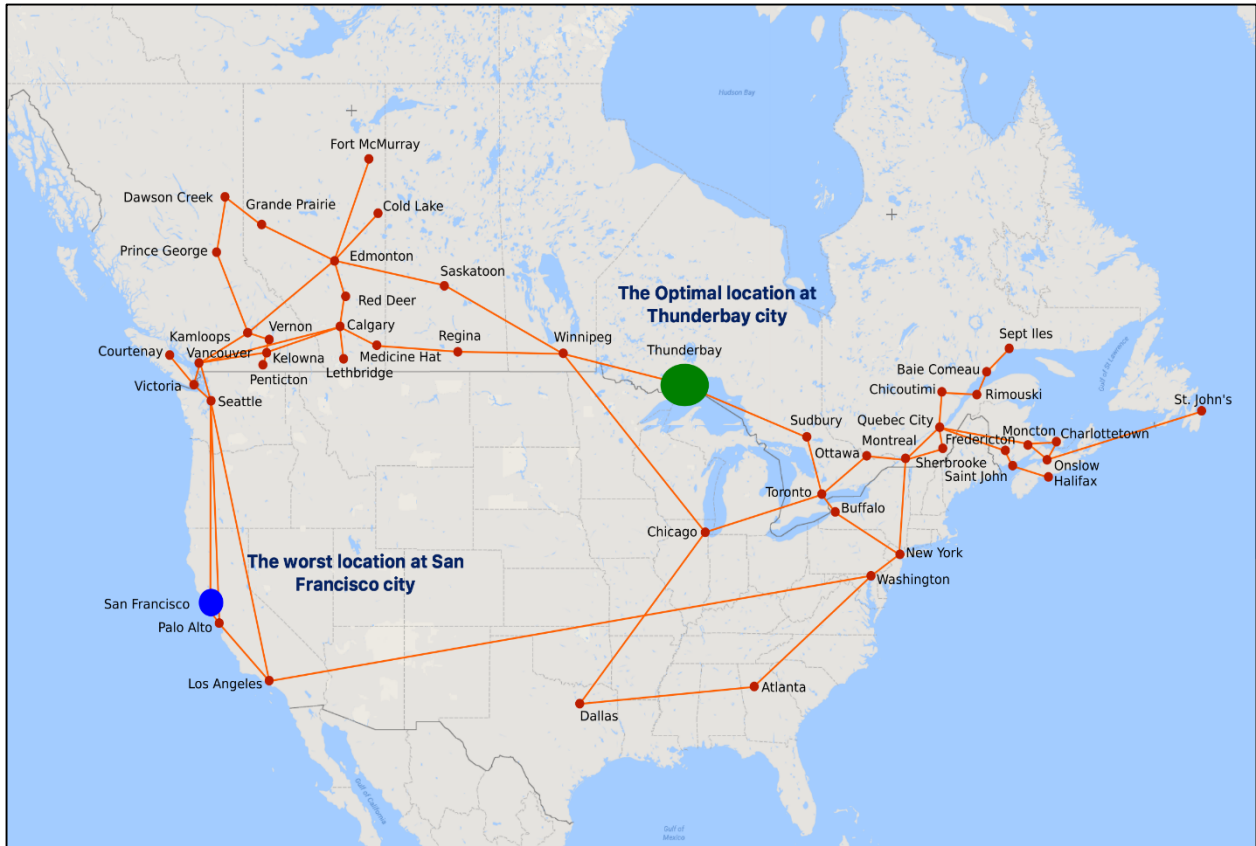


Figure 4.1: The optimal & worst placement for SDN network when number of controller is one ($k=1$)

The number of controllers (k) input to the model is important because the latency outcomes are dependent on these values. The results demonstrate that the average latency is reliant on the number of controllers.

The figure below shows the optimal placement when we used three SDN controllers.

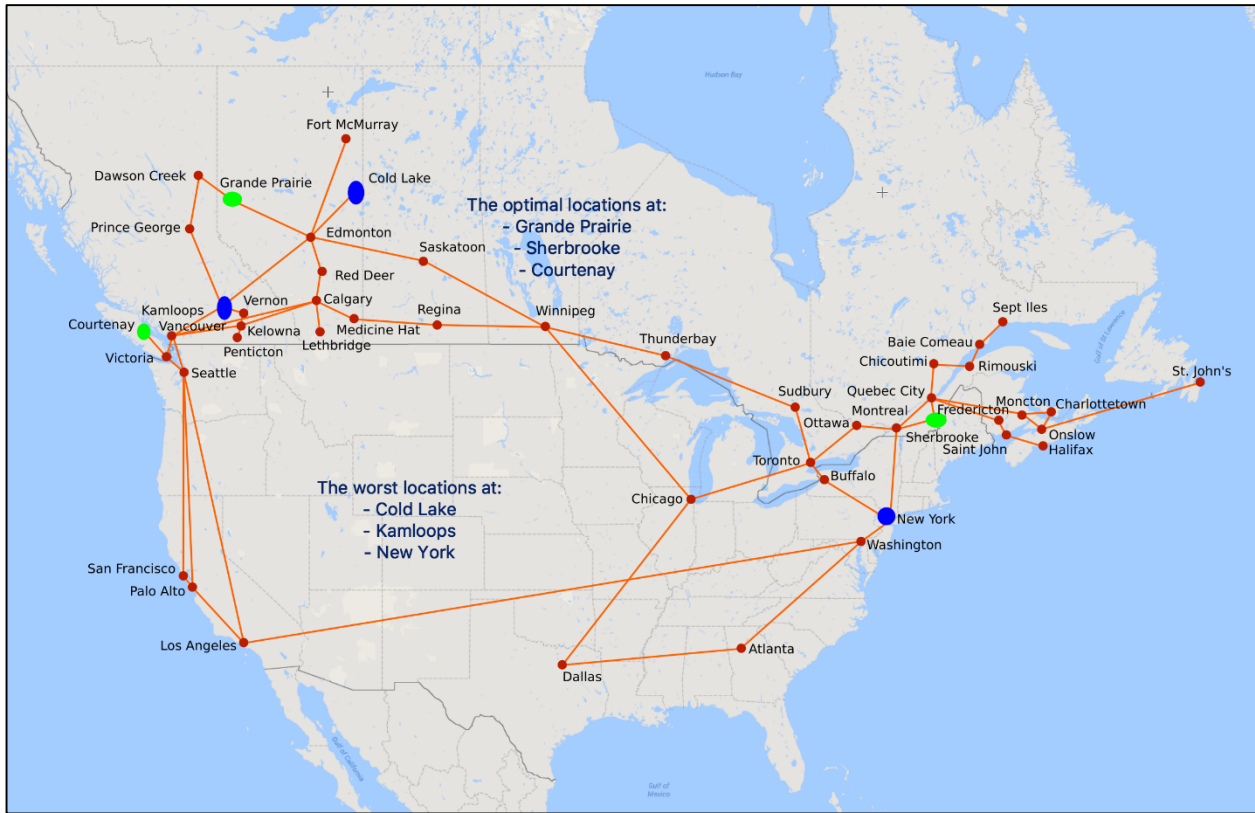


Figure 4.2: The optimal & worst placement for SDN network when number of controller is one ($k=3$)

The finding presented in Figure 4.2 show a reduction in the total average latency to ($L_{avg} = 0.1360$). In this case, network performance was improved, with a 75 % reduction in the delay: from 0.4459 to 0.1360. The worst-case latency also decreased to ($L_{wc} = 0.4173$). Table II summarizes the results for ($k = 1$) to ($k = 5$). Using more than five controllers led to no significant change in the overall latency; five controllers can therefore be considered the optimal number for this network topology.

Table 4.1: Average latency for SDN controllers

	K=1	K=2	K=3	K=4	K=5
Locations names for <i>Lavg</i>	Thunder bay	Sherbrook Lethbridge	Grande-Prairie Sherbrook Courtenay	Grande-Prairie Ottawa St-John's Courtenay	Grande-Prairie Kamloops Ottawa St-John's Courtenay
Locations names for <i>Lwc</i>	San Francisco	Prince-George Penticton	Cold-Lake Kamloops New-York	Fort-McMurray Kamloops Sherbrook Vancouver	Fort-McMurray Kamloops Baie-Comeau Atlanta Sudbury
<i>Lavg</i>	0.4459	0.1665	0.1360	0.1166	0.0985
<i>Lwc</i>	1.0412	0.8043	0.4173	0.3300	0.2683

Based on the results listed in Table 4.1, it can be observed that a threshold exists with respect to the number of controllers in each SDN topology. When the threshold is reached, there is no marked improvement in the total average delay metric.

As can be seen in Figure 4.3, no significant change in network latency is evident above a specific number of SDN controllers. Figure 4.3. shows that changing the number of SDN controllers from one to two results in a reduction of up to 75 % overall; further increases have a much less significant effect on latency than this first change in the number of SDN controllers. The advantage of our approach is that the optimum number of SDN controllers that should be deployed on a network to achieve the best QoS can now be established for any WAN topology. The new approach is based on a comparison of the average latency (*Lavg*) and the worst-case latency (*Lwc*) in order to determine the maximum overall delay. However, the ultimate decision about how many

controllers to deploy is dependent on the unique needs and constraints of each service provider. For this study, we concluded that using five SDN controllers is the most efficient way to achieve the best QoS outcomes.

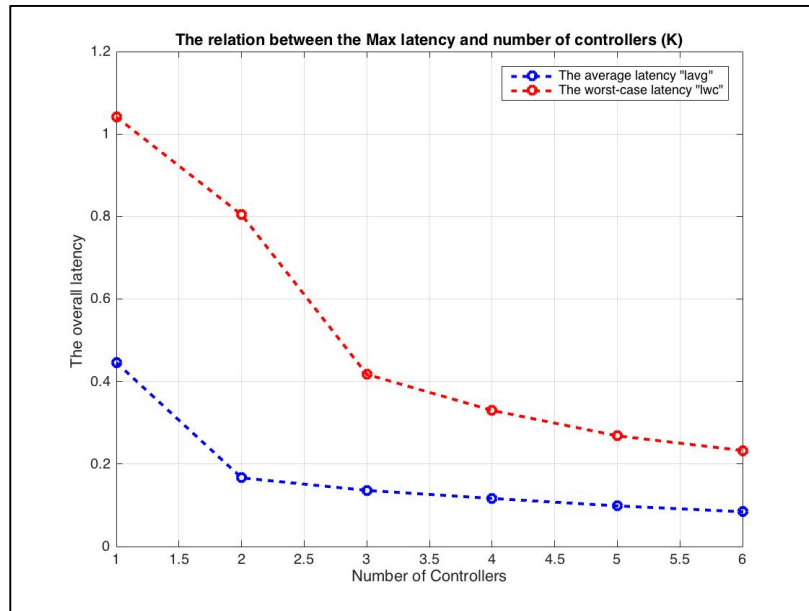


Figure 4.3: The optimal & worst average latency for SDN network with number of controllers (k)

From figure 4.4. We compute cumulative distribution function (CDF) for the mean latency compared with the number of controllers. Indeed, we observed drastic differences when we used one controller versus when we used five controllers.

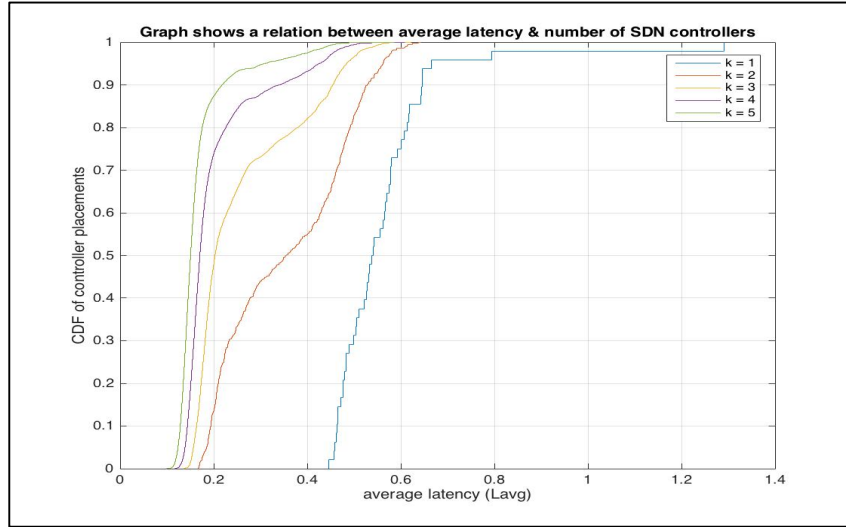


Figure 4.4: The optimal latency for SDN network with number of controllers (k)

Another important metric for determining the optimal SDN controller location is the cost of installing new SDN controllers. In previous work, we examined the optimal location by computing the propagation delay between OpenFlow switches. In this section, new decision based on the cost of adding a new controller by dollars (C_k).

$$\text{The cost benefit} = \text{The average delay (Lavg)} + \text{Number of controllers (k)} * C_k \quad (4.1)$$

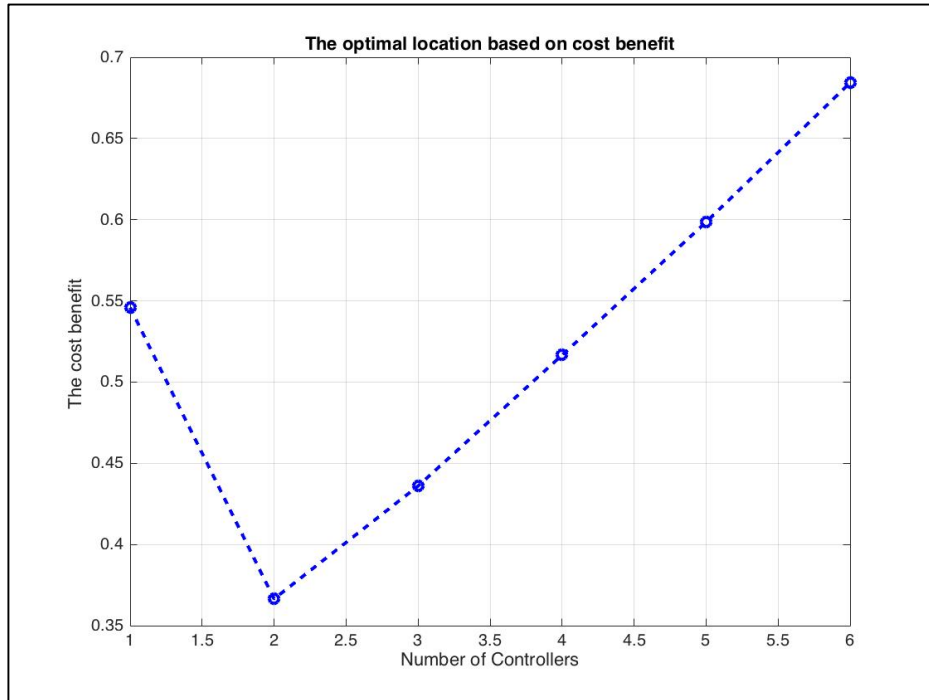


Figure 4.5: The optimal latency for SDN network with number of controllers (k)

The figure above shows the cost benefit related to the number of controllers. By visual observation, we achieved the optimal placement for SDN controller at ($k=2$). Past that point, the cost of an SDN controller implementation increases. As a result, the number of controllers at the end based on an Internet Service Provider's (ISP) budget to upgrade their network.

4.2. Case study results and analysis for a Network analysis procedure

A Python 2.7 programming languages was used for implementing an SDN network with a Floodlight SDN controller. The goal form this experiment to match the outcome with the results from a mathematical model in previous section. The results, which are presented in Tables below, reveal that the optimum controller location is node 22 (Thunder Bay) and that the worst location is node 44 (San Francisco). To quantify the impact of the propagation delay, we generated 30 ICMP packets between the nodes, then calculated the total average delay. The ICMP ping message rate (packets per second) is the main parameter for measuring the round-trip time (RTT) from one node to another. The final step was to compute the overall delay for each Bell Canada node. Table III shows the experimental results when the controller was placed at the Thunder Bay location (OpenFlow switch S22). The ICMP packets from this node were generated to randomly selected nodes.

Table 4.2: The average time delay at Thunder Bay location (S22)

Source location	Destination Location	Average time per (ms)
S22	S1	0.033
S22	S48	0.041
S22	S26	0.067
S22	S44	0.039
S22	S30	0.049

The total average for this location is (0.048 ms)

Table 4.3 shows the result for the St John SDN controller location (S1):

Table 4.3: The average time delay at St John location (S1)

Source location	Destination Location	Average time per (ms)
S1	S12	0.155
S1	S48	0.198
S1	S26	0.072
S1	S44	0.073
S1	S30	0.082

The total average for this location is (0.116 ms)

The table below shows the result of San Francisco SDN controller location (S44):

Table 4.4: The average time delay at San Francisco location (S44)

Source location	Destination Location	Average time per (ms)
S44	S1	0.078
S44	S5	0.202
S44	S26	0.149
S44	S30	0.086
S44	S48	0.090

The total average for this location is (0.121 ms)

The table below shows the result of San Francisco SDN controller location (S30):

Table 4.5: The average time delay at San Francisco location (S30)

Source location	Destination Location	Average time per (ms)
S30	S1	0.103
S30	S48	0.098
S30	S26	0.090
S30	S44	0.110
S30	S35	0.074

The total average for this location is (0.0956 ms)

As can be seen in the bar chart shown in Figure 4.6. and figure 4.7, the best location for the SDN controller is at Thunder Bay (Node 22), since this node has the lowest propagation delay. The worst location would be San Francisco (Node 44). In our Bell Canada case study, the best possible SDN controller locations are thus close to the center of the network.

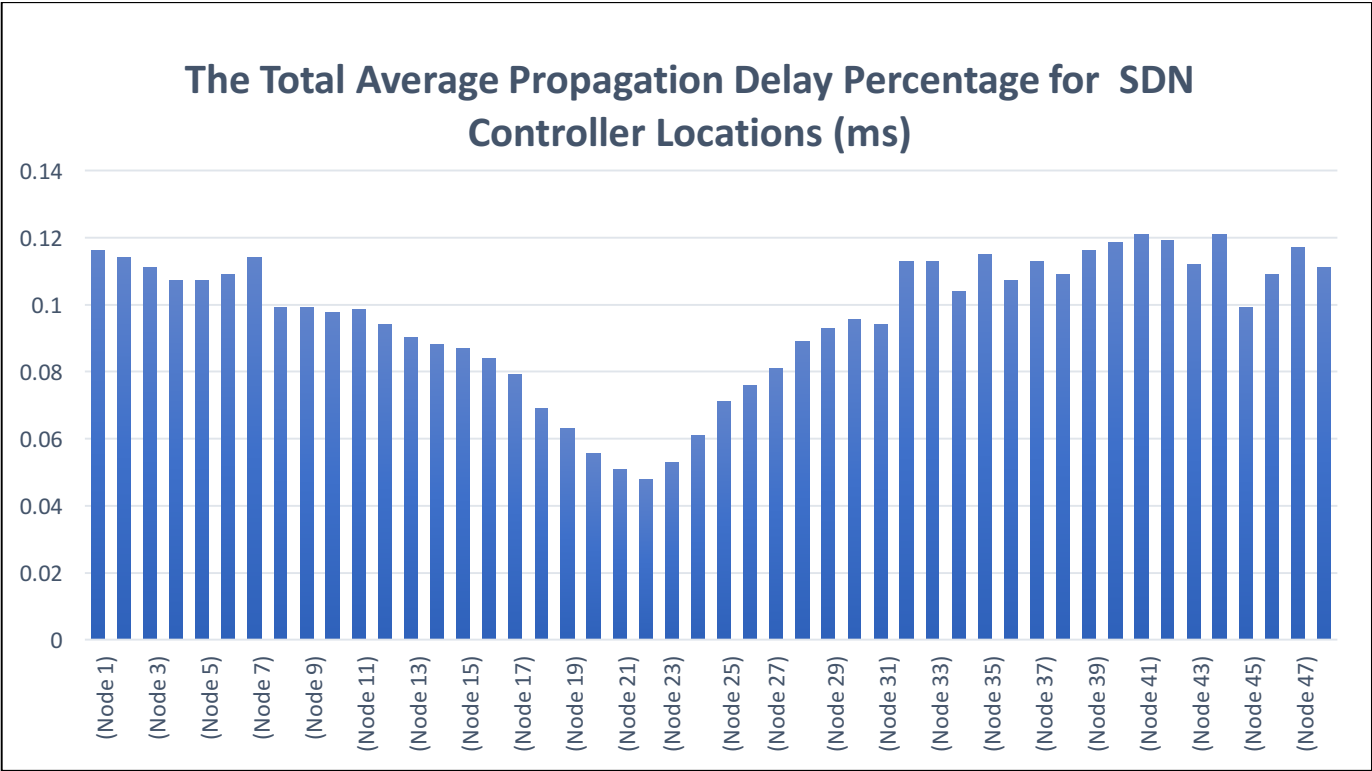


Figure 4.6: The Total average delay for SDN controllers

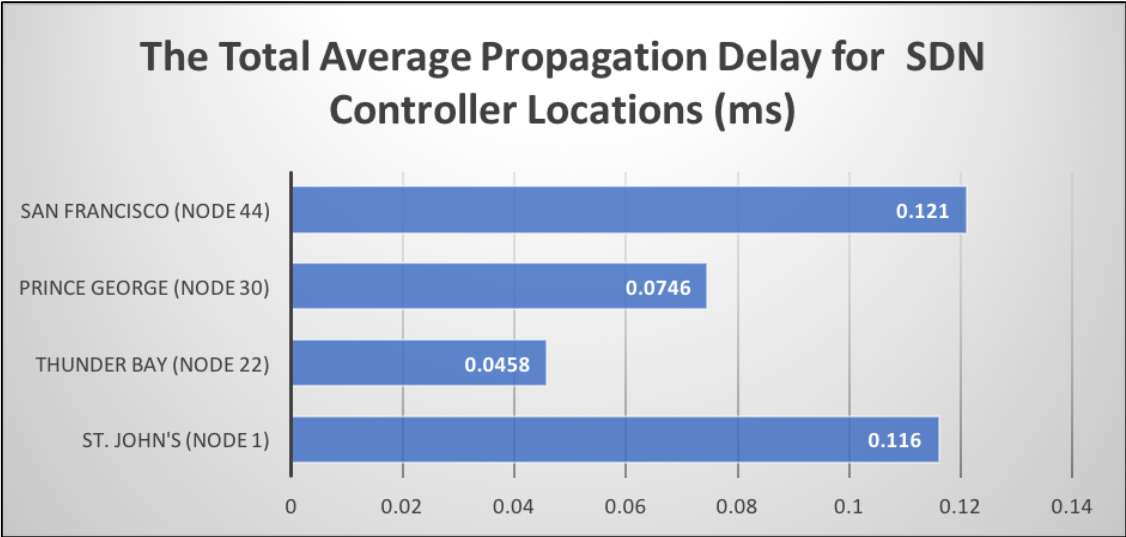


Figure 4.7: The best and worst locations for SDN controllers

4.3. Simulation Source Codes

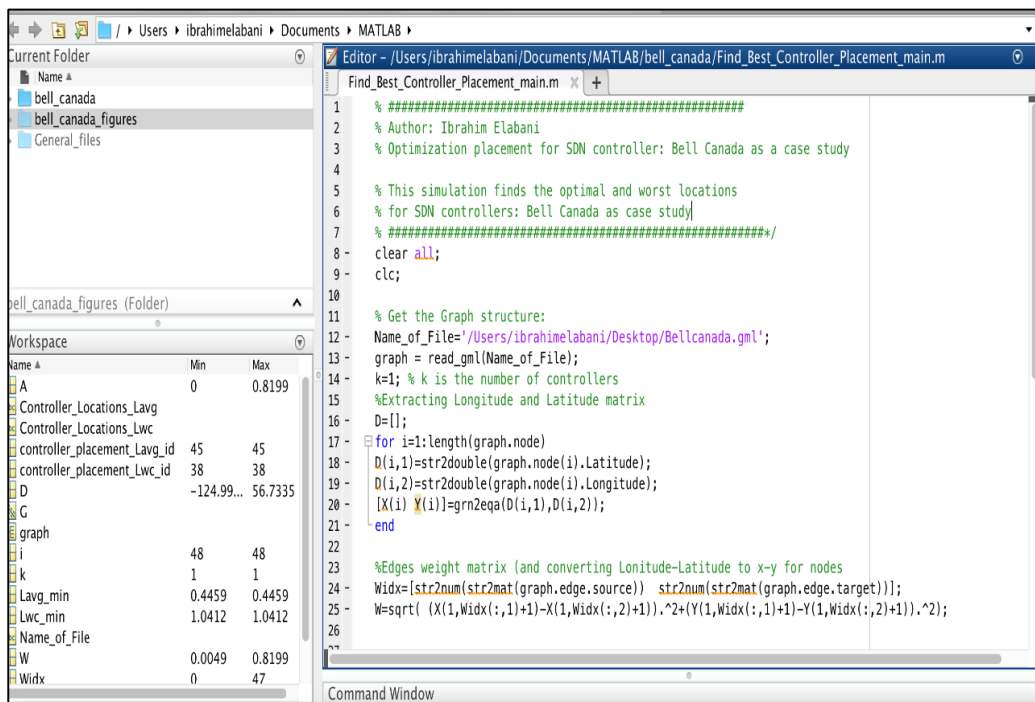
4.3.1. Simulation codes on GitHub

The experimental codes for the proposed study have been publicly posted on GitHub, it's the world leading development software.

<https://github.com/ibraabani/optimization-sdn-controllers-bellcanada-casestudy.git>

4.3.2. Examples of Simulation codes

The figure below is an sample from the Matlab code, we used this code to implemented a mathematical model:

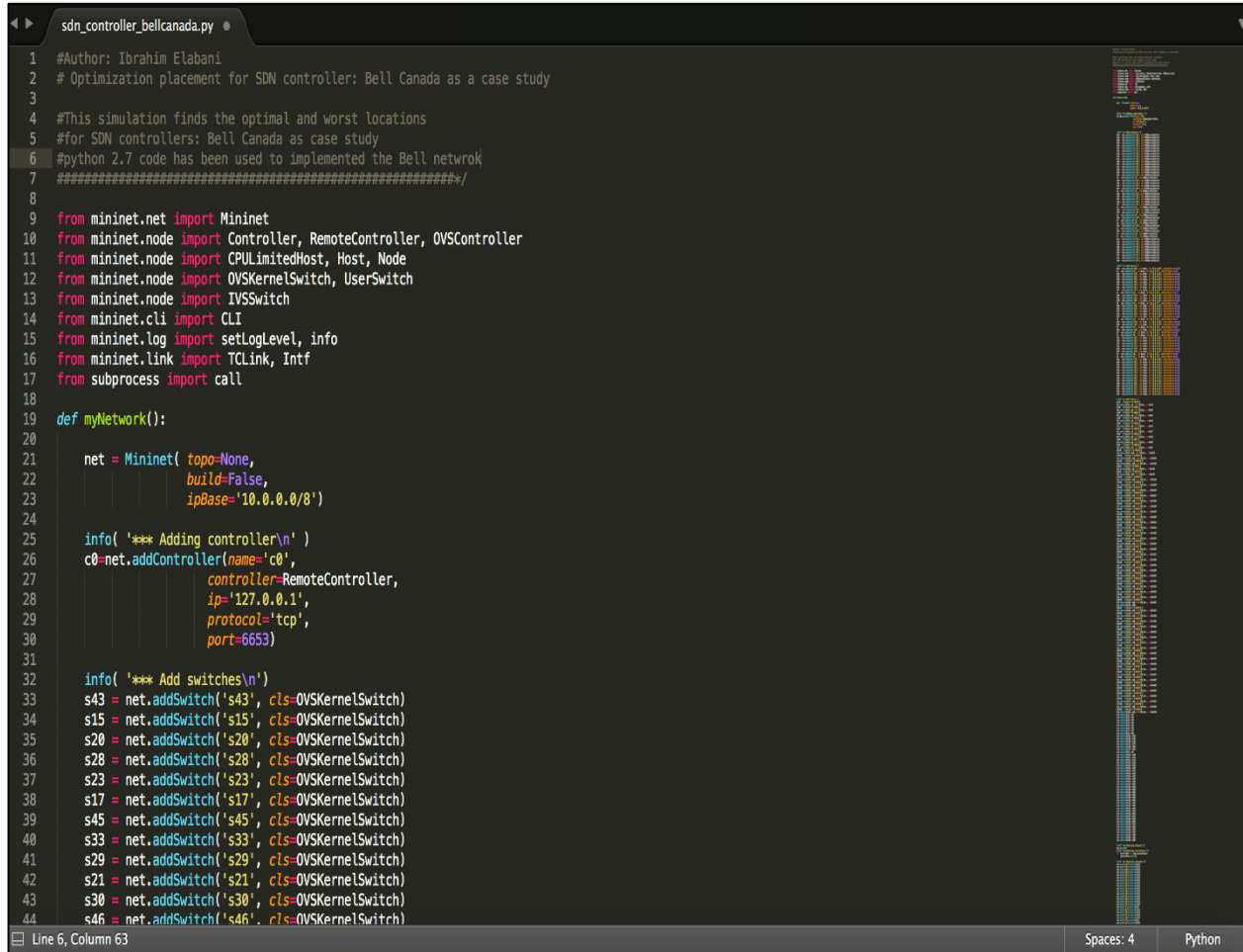


```
1 % #####
2 % Author: Ibrahim Elabani
3 % Optimization placement for SDN controller: Bell Canada as a case study
4
5 % This simulation finds the optimal and worst locations
6 % for SDN controllers: Bell Canada as case study
7 % #####/
8 - clear all;
9 - clc;
10
11 % Get the Graph structure:
12 - Name_of_File='/Users/ibrahimelabani/Desktop/Bellcanada.gml';
13 - graph = read_gml(Name_of_File);
14 - k=1; % k is the number of controllers
15 %Extracting Longitude and Latitude matrix
16 - D=[];
17 - for i=1:length(graph.node)
18 - D(i,1)=str2double(graph.node(i).Latitude);
19 - D(i,2)=str2double(graph.node(i).Longitude);
20 - [X(i) Y(i)]=grn2eqa(D(i,1),D(i,2));
21 - end
22
23 %Edges weight matrix (and converting Longitude-Latitude to x-y for nodes
24 - Widx=[str2num(str2mat(graph.edge.source)) str2num(str2mat(graph.edge.target))];
25 - W=sqrt( (X(1,Widx(:,1)+1)-X(1,Widx(:,2)+1)).^2+(Y(1,Widx(:,1)+1)-Y(1,Widx(:,2)+1)).^2);
26
27
```

Name	Min	Max
A	0	0.8199
Controller_Locations_Lavg		
Controller_Locations_Lwc		
controller_placement_Lavg_id	45	45
controller_placement_Lwc_id	38	38
D	-124.99...	56.7335
G		
graph		
i	48	48
k	1	1
Lavg_min	0.4459	0.4459
Lwc_min	1.0412	1.0412
Name_of_File		
W	0.0049	0.8199
Widx	0	47

Figure 4.8: Matlab code for best and worst locations for SDN controllers

The python programming language was used to implement the Bell Canada WAN network, sample from python code in figure below.



```
1 #Author: Ibrahim Elabani
2 # Optimization placement for SDN controller: Bell Canada as a case study
3
4 #This simulation finds the optimal and worst locations
5 #for SDN controllers: Bell Canada as case study
6 #python 2.7 code has been used to implemented the Bell netwrok
7 #####*/
8
9 from mininet.net import Mininet
10 from mininet.node import Controller, RemoteController, OVSController
11 from mininet.node import CPUimitedHost, Host, Node
12 from mininet.node import OVSKernelSwitch, UserSwitch
13 from mininet.node import IVSSwitch
14 from mininet.cli import CLI
15 from mininet.log import setLogLevel, info
16 from mininet.link import TCLink, Intf
17 from subprocess import call
18
19 def myNetwork():
20
21     net = Mininet( topo=None,
22                 build=False,
23                 ipBase='10.0.0.0/8' )
24
25     info( '*** Adding controller\n' )
26     c0=net.addController(name='c0',
27                       controller=RemoteController,
28                       ip='127.0.0.1',
29                       protocol='tcp',
30                       port=6653)
31
32     info( '*** Add switches\n' )
33     s43 = net.addSwitch('s43', cls=OVSKernelSwitch)
34     s15 = net.addSwitch('s15', cls=OVSKernelSwitch)
35     s20 = net.addSwitch('s20', cls=OVSKernelSwitch)
36     s28 = net.addSwitch('s28', cls=OVSKernelSwitch)
37     s23 = net.addSwitch('s23', cls=OVSKernelSwitch)
38     s17 = net.addSwitch('s17', cls=OVSKernelSwitch)
39     s45 = net.addSwitch('s45', cls=OVSKernelSwitch)
40     s33 = net.addSwitch('s33', cls=OVSKernelSwitch)
41     s29 = net.addSwitch('s29', cls=OVSKernelSwitch)
42     s21 = net.addSwitch('s21', cls=OVSKernelSwitch)
43     s30 = net.addSwitch('s30', cls=OVSKernelSwitch)
44     s46 = net.addSwitch('s46', cls=OVSKernelSwitch)
```

Figure 4.9: Python code for best and worst locations for SDN controllers

Chapter 5

Conclusions and Future Work

This chapter summarizes the thesis work. A conclusion, along with a discussion of future work, is provided.

5.1. Conclusions

This thesis has presented a mathematical model for finding the optimal location for an SDN controller in the Bell Canada WAN. Measurement of the overall latency of this network was based on the delay in propagation among the control and data planes. The proposed model was assessed with respect to answering an essential question associated with the planned implementation of an SDN network: how many SDN controllers are needed in the network topology. The simulation results were obtained using MATLAB_R2015b and Python programming languages. A k-median algorithm was employed to solve for an important factor in SDN network scalability, and an ICMP was used for verifying the mathematical results. This work has proposed a method that enables SDN operators to identify the number of SDN controllers, and to optimize the locations for their placement in a way that will achieve the best network performance.

5.2. Future Work

Due to time constraints, some metrics have been omitted from this paper. Future research can encompass the application of additional constraints in our mathematical model. The assumptions underlying our examination of the optimization of the Bell Canada WAN included only propagation delay. We have not included queueing at the controllers, i.e., arrival and departure waiting times for packets before they are processed. Incorporating queueing delay into the model might improve the results by enhancing accuracy with respect to overall network latency. A further consideration is that because flow setup processing time increases the accuracy of a model so that it more closely reflects real-life scenarios, it should also be taken into account.

References

- [1] Cisco VNI, "The Zettabyte Era: Trends and Analysis," Cisco, 01 August 2017. [Online]. Available:http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.html#_Toc484556817. [Accessed 01 August 2017].
- [2] J.GUILLERMO,"Cengn.ca,"Fujitsu,1052016.[Online].Available: <https://www.cengn.ca/multilayer-data-connectivity-orchestration-exploring-the-cengn-proof-of-concept-javier-guillermo-fujitsu/>. [Accessed 12 4 2017].
- [3] K.Prabu,"www.sdxcentral.com,"AT&T,2june2016.[Online].Available: <https://www.sdxcentral.com/articles/news/att-cto-expects-sdn-reduce-opex-costs-40/2016/06/>. [Accessed 16 april 2017].
- [4] A. Voellmy, " Programmable and scalable software-defined networking controllers," 2014, New Haven, 2014.
- [5] Y. A. J.´. Agudelo, "Scalability and Robustness of the control plane in Software-Defined Networking (SDN)," Universitat Polité´cnica de Catalunya, Barcelona, 2016.
- [6] J. Doherty, SDN and NFV Simplified: A Visual Guide to Understanding Software Defined Networks and Network Function Virtualization, Kendallville, Indiana: Addison Wesley Professional , 2016.

- [7] opennetworking, "www.opennetworking.org," opennetworking, 12 may 2016. [Online]. Available: www.opennetworking.org.
- [8] N. Abouzakhar, ECCWS2015-Proceedings of the 14th European Conference on Cyber Warfare and Security 2015, Reading: Academic Conferences and Publishing international Limited, 2015.
- [9] L. D. Chou, C. W. Tseng, C. H. Tseng and Y. M. Chen, "The Novel SDN Testbed with Virtual Network Functions Placement," *2016 International Conference on Software Networking (ICSN)*, Jeju, 2016, pp. 1-5.
- [10] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," in *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236-262, Firstquarter 2016.
- [11] Yan-nan HU, Wen-dong WANG, Xiang-yang GONG, Xi-rong QUE, Shi-duan CHENG, On the placement of controllers in software-defined networks, *The Journal of China Universities of Posts and Telecommunications*, Volume 19, 2012, Pages 92-171, ISSN 1005-8885.
- [12] R. S. N. M. Brandon Heller, "The Controller Placement Problem," *Proceedings of the first workshop on Hot topics in software defined networks*, vol. 0, no. 0, pp. 7-12, 2012.
- [13] N. S. Hidenobu Aoki, "Controller Placement Problem to Enhance Performance in Multi-domain SDN Networks," in *The Fifteenth International Conference on Networks (includes SOFTNETWORKING 2016)*, Tokyo, 2016.
- [14] M. F. Bari *et al.*, "Dynamic Controller Provisioning in Software Defined Networks," *Proceedings of the 9th International Conference on Network and Service Management (CNSM 2013)*, Zurich, 2013, pp. 18-25.

- [15] Moses Charikar, Sudipto Guha, Éva Tardos, David B. Shmoys, A Constant-Factor Approximation Algorithm for the p -Median Problem, *Journal of Computer and System Sciences*, Volume 65, Issue 1, 2002, Pages 129-149, ISSN 0022-0000.
- [16] V. V. Kamal Jain, "Approximation algorithms for metric facility location and k-Median problems using the primal-dual schema and Lagrangian relaxation," *Journal of the ACM*, vol. 48, no. 2, pp. 274-296, 2001.
- [17] D. B. Johnson, "Efficient Algorithms for Shortest Paths in Sparse Networks," *Journal of the ACM*, vol. 24, no. 1, pp. 1-13, 1977.
- [18] P. Ghuli, "A Comprehensive Survey on Centroid Selection Strategies for Distributed K-means Clustering Algorithm," *International Journal of Computer Applications*, vol. 125, no. 5, pp. 36-42, 2015.
- [19] V. Arya, N. Garg, R. Khandekar et al, "Local search heuristic for k-median and facility location problems," *In Proceedings of the thirty-third annual ACM symposium on Theory of computing (STOC '01)*. ACM, New York, NY, USA, 21-29, 2001.
- [20] D. Erickson, "The Beacon OpenFlow Controller," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking - HotSDN 13*, USA, 2013.
- [21] L. Han, Z. Li, W. Liu, K. Dai and W. Qu, "Minimum Control Latency of SDN Controller Placement," *2016 IEEE Trustcom/BigDataSE/ISPA*, Tianjin, 2016, pp. 2175-2180.
- [22] Mininet, "Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet," Mininet, 24 July 2017. [Online]. Available: <http://mininet.org>. [Accessed 24 July 2017].

- [23] P. Floodlight, "Floodlight OpenFlow Controller," Floodlight, 24 July 2017. [Online]. Available: <http://www.projectfloodlight.org/floodlight/>. [Accessed 24 July 2017].
- [24] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," in *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14-76, Jan. 2015.