CONTROLLER PLACEMENT MECHANISM IN SOFTWARE DEFINED NETWORK USING K-MEDIAN ALGORITHM

NOOR SAAD FAHAD

MASTER OF SCIENCE INFORMATION TECHNOLOGY
UNIVERSITI UTARA MALAYSIA
2016

Permission to Use

Abstrak

SDN memisahkan satah kawalan dengan sata data melalui pemindahan satah kawalan ke entity lain. Pemisahan ini menimbulkan beberapa masalah, antaranya penempatan pengawal dalam rangkaian. Kajian ini bertujuan untuk mengkaji penempatan node kawalan dalam SDN. Kadeah k-median digunakan untuk menentukan kudududukan nod pengawal, dan nod pengawal dengan purata kependaman terendah akan dipilih. Penentu kedudukan ini akan membandingkan algoritma greedy yang mengira kombinasi berdasarkan kedudukan nod dan mengira nilai terbaik untuk setiap turutan. Kajian ini turut menbandingkan kombinasi keputusan melalui kedudukan nod tertentu, dan keputusan menunjukkan kaedah k-median memberikan nilai yang lebih tinggi. Tiga nod pengawal dipilih sebagai bilangan nod minima and dinilai dari segi kelewatan dan beban, dan keputusan menunjukkan tiga nod memadai sekiranya tiada kelewatan atau bebenan dalam rangkaian.

Kata kunci: SDN; Pengawal; Penempatan; Purata Kependaman; K-median

Abstract

Software Defined Network (SDN) decouples the control plane and the data plane, and moves the control plane to an external entity. The decoupling raises many challenges, and one of these is the placement of the controller in the network. This study aims to address controller placement problem in SDN. k-median is used to determine the placement of the controllers, and the placement with the lowest value of average propagation latency will be chosen. The placement compares two resulted placements. First, comparing to greedy algorithm that computes the combinations according to the order of the nodes and calculates the best values at each step, and the results were identical. The second comparison was with the combinations results from considering the placement from specific nodes, and the results showed that it gives higher results than depending on the lowest values resulted from the k-median. Finally, three controllers are chosen as the minimum number of controllers, they were evaluated in terms of delay and load, and as results it was found that three controllers are suitable number of controllers as long as there is no delay or load in the network. Combining the two algorithms for finding the placement and the number results in Controller Placement Mechanism (CPM)

Keywords: SDN; Controllers; Placement; Average propagation latency; K-median

Acknowledgement

In the name of ALLAH, Most Gracious, Most merciful

All thanks and praises to Allah (SWT) for the blessings of life and for guiding me through studies and life.

My sincere appreciation goes to my supervisors, Dr. Adib Habbal and Mr. Suwannit Chareen Chit, for your patient guidance and encouragement through this research, without your valuable support, this research will not be possible, Thank you.

To my parents, without your love and support I would not be able to continue my studying, I'm so grateful for always believing in me, encouraging me, and never let me doubt myself, I love you, may ALLAH continue to bestow his blessings on you.

To the examiners committee, I'm grateful for your guidance and remarks to make this research better.

To UNIVERSITI UTARA MALAYSIA for giving the opportunity to further my study, and to make my master journey enjoyable and memorable, you have my sincere gratitude.

To my family and friends, thank you for all your support and praying.

Dedication

To my family.

Table of Content

List of Figures

List of Tables

List of Abbreviations

SDN	- Software Defined Network
CPM	- Controller Placement Mechanism
NOS	- Network Operating System
FD	- Forwarding Devices
DP	- Data Plane
SI	- Southbound Interface
CP	- Control Plane
NI	- Northbound Interface
MP	- Management Plane
API	- Application Program Interface
CPP	- Controller Placement Problem
CCPP	- Capacitated Controller Placement Problem
RCP	- Reliability aware Controller Placement
SA	- Simulated Annealing
POCO	- Pareto-based Optimal COntroller placement
PSA	- Pareto Simulated Annealing
GreCo	- GREEN CENTRALIZED CONTROLLER
BIP	- Binary Integer Program
MC	- Main Controllers
SC	- Slave Controllers
AVL	- Average Propagation latency
MyREN	- Malaysian Research & Education Network
MoE	- Ministry of Education
MDeC	- Multimedia Development Coperation
UITM	- Universiti Teknologi MARA

UTP	- Universiti Teknologi Petronas
UUM	- Universiti Utara Malaysia
UM	- University of Malaya
UNIMAS	- Universiti Malaysia Sarawak
UMS	- Universiti Malaysia Sabah
IUM	- International Islamic University Malaysia
UPSI	- Sultan Idris Education University
UPM	- Universiti Putra Malaysia
UTHM	- Universiti Tun Hussein Onn Malaysia
UTM	- University of Technology, Malaysia
UMT	- Universiti Malaysia Terengganu
UMK	- Universiti Malaysia Kelantan
UDM	- Universiti Darul Iman Malaysia
UMP	- Universiti Malaysia Pahang
UPNIM	- National Defence University of Malaysia
UTEM	- Universiti Teknikal Malaysia Melaka
NOC	- Network Operation Center
UKM	- National University of Malaysia
MMU	- Multimedia University
USIM	- Universiti Sains Islam Malaysia
UNITEN	- Universiti Tenaga Nasional
MIMOS	- Malaysia's national R&D centre in ICT
TMRND	- Telekom Research & Development
MOHE	- Ministry of Higher Education
USM	- Universiti Sains Malaysia
UNIMAP	- Universiti Malaysia Perlis

CHAPTER ONE

INTRODUCTION

1.1 Overview

The current network schemes are complex and very difficult to manage. Predefined policies make the network difficult to be configured and also very hard to reconfigured so that it can respond to the load, faults and changes in the network (Open Networking Foundation, 2012). Current networks are integrated vertically where the control plane (that decides how to handle the traffic), and the data plane (that forwards the traffic based on the decision of the control plane) are coupled together which lead to the reduction of the flexibility as well as holding back the innovation and the network infrastructure evolution.

Software Defined Network (SDN) has gotten a lot of attention recently as a solution to overcome the limitations of the current network schemes. According to the Open Networking Foundation, " the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications" (Open Networking Foundation, 2012). Based on this definition (Sezer et al., (2013)) extracted four features which are: the control plane and the data plane are separated, interfaces are open between the data plane and the control plane, the controller is centralized, and the network programmability by external applications.

Kreutz et al., (2015) defined SDN as an architecture for the network that has four pillars: First, the separation of the control plane and the data plane. The controller operations are removed from the network devices, which will have only the function of forwarding the packets. Second, the decisions for forwarding are flow-based and not destination based. In SDN the flow is defined as the sequence of packets between the source and the destination. All of the flow packets receive identical polices of service at the forwarding devices, which will behave in a unified manner regardless their type. Third, the control logic is moved to an external entity, the one that is called the controller, or the Network Operating System (NOS). It is logically centralized and it has the responsibility of providing resources and the abstractions to simplify the programming of the network devices. Fourth, the network can be programmed by software applications, which run on the top of the NOS that interacts with the data plane that form the underlying layer. Figure 1 shows SDN architecture in comparison to the current network scheme.

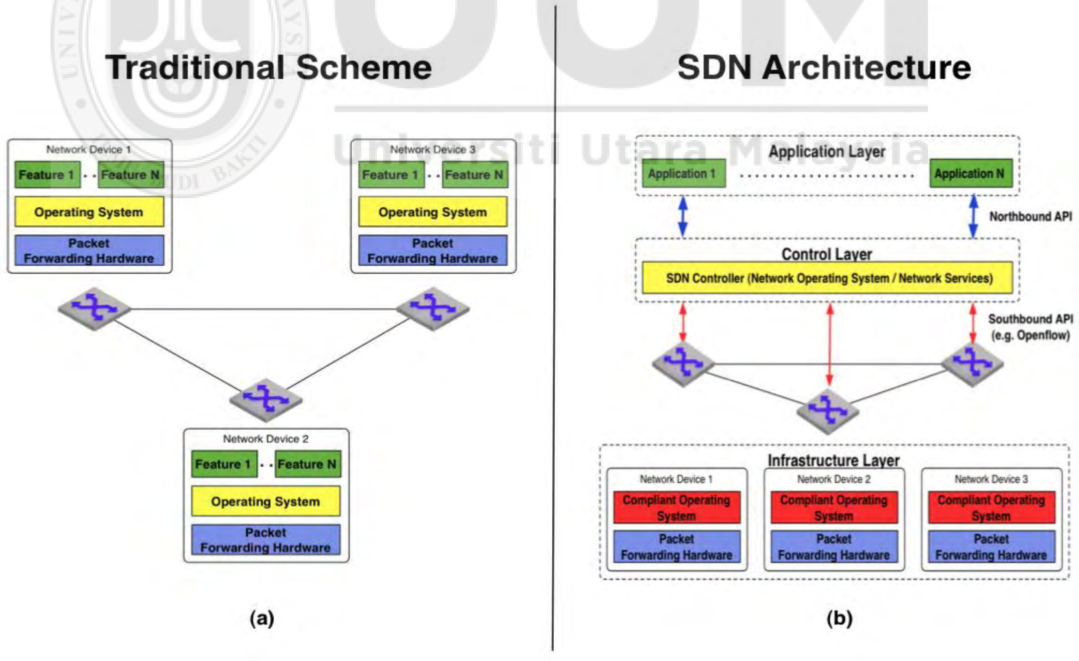


Figure 1: Traditional Scheme (a) SDN Architecture (b)

Due to the separation of the control plane and the data plane, many challenges such as the ability to handle high security, high touch and high performance of the packet processing that flows efficiently that must be addressed. Also, several other aspects such as the scalability of the network, which can be split into scalability of the controller and scalability of the network node, the security and protecting the network against attacks, interoperability which is the transition from traditional networks to SDN require great attention.

One of the key challenges is the controller placement, which is the main focus of this research. Many researchers have attempted to address this problem by proposing different solutions. Heller et al., (2012) considered the first to analyse the controller placement problem that becomes a motivation for a lot of researchers like Yao et al., (2014). This study is motivated by the work presented by Heller. k-median algorithm will be used to find the placement of the controllers. k-median is a well-known algorithm that is used for the purpose of finding k-center locations for instance (warehouses) through minimizing the sum of the distance of the desired points.

1.2 Problem Statement

The key problems in SDN is the controller placement problem (Yao, Bi, Li, & Guo, 2014). Every aspects of the decoupled control plane are affected by the controller placement problem like fault tolerance, state distribution and network performance. The location and the number of the controllers determine the performance of the network (Jiménez, Cervelló-Pastor, & García, 2014), as poor placement will affect the robustness of the network, which in return affect the operation of the network, for instance the long time to recover after failure.

Heller et al., (2012) pointed to the rising concerns about the availability, performance and scalability, also how the controller placement problem will affect the aspects of the control plane that decoupled from the data plane in the network. The load on the controllers should also be considered for designing the controller placement due three reasons: failure, the limitation of the server capacity and message processing latency.

1.3 Research Questions

1. How to determine the placement of the controllers across the network?
2. What is the impact of the proposed controller placement on SDN performance?

1.4 Research Objectives

The main aim of this study is to propose controller placement mechanism to achieve the following objectives:

1. To design a controller placement mechanism using the k - median algorithm as a solution for the controller placement problem in SDN.

2. To evaluate the proposed controller placement mechanism in terms of delay and the load on the controllers.

1.5 Significant of Research

SDN is a network architecture that is considered as the future of the Internet due to the limitation of the traditional networks scheme. It decouples the control and the data plane, which leads to simplified the management of the network and speed up the innovations (Yannan et al.,2014). The challenge is how to place the controllers across the network.

By proposing a solution for the controller placement, it will provide better performance of the network, which will have an impact on the user. The propagation latency will be reduced, which will provide faster response time. The network will be easy to monitor and managed by the network administrator.

1.6 Scope of the Research

This study focused on the controller placement in SDN at the control plane layer. K-median algorithm will be used to find the placement of the controllers. The proposed mechanism will be designed using Python programming language. The proposed mechanism will be evaluated by the metrics of delay, and the load on the controllers across the network.

1.7 Research Outcomes

The outcome of this research will be as follows:

1. Controller placement mechanism, which the location of the controllers will be known.

2. The optimal number of controllers will be chosen based on evaluating the placement using delay and load algorithm.

1.8 Organization of the Study

The study is organized as follows:

Chapter one is the introduction of this study including the problem statement, research questions, research objectives, significance, scope and the outcome. Chapter two is literature review of the work about SDN and the work related to the problem of this research, as well as summary of the related work. Chapter three explains the methodology. Chapter four describe the implementation of the mechanism, the tools that been used, and the formulas. Chapter five will include the contribution, future work, limitation and conclusion.

CHAPTER TWO

LITERATURE REVIEW

This chapter reviews the literature about SDN. The first section will be about the SDN in general; reviewing its layers and, its advantages, and challenges. The second section will be about the related work, which researches the controller placement problem, and the controllers' distribution. The third section will be presenting this research mechanism in terms of the work it based on. The final section will be summary of this chapter.

Current network faces the problem of being limited under the high network traffic, which in terms will have effect on the network performance. Other issues like the high demand for security, scalability, reliability, and the speed of the network can severely block the performance of the network devices, because of the increasing traffic of the network. The networks which are the backbone of the Internet must have the ability to adapt to the changes without causing huge labor intensive in terms of software and hardware modification. However, the traditional network cannot be re-tasked or reprogrammed easily.

One of the possible solutions to overcome the limitation is by implementing the rules of the data handling as software modules instead of including them in the hardware. This method will enable the administrators of the network to have more control over the traffic of the network, which will lead to great potential to improve the network's performance greatly in terms of using the resources of the network efficiently. Such solution is represented by the technology of SDN (Hu, Hao, & Bao, 2014).

2.1 Software Defined Network

SDN is a network architecture that separates the control and the data plane, and moved the control plane to external entity called the control layer. Kreutz et al., (2015) defined the SDN as the architecture that defined by four major features: the separation of the data plane and the control plane, the decisions are flow based not destination based, moving the control logic to an external entity, and finally the programmability of the network.

2.1.1 Software Defined Network Layers

SDN consist of three layers, namely the infrastructure layer (data plane), the control layer (control plane), and the application layer (management plane). These three layers as well as the interfaces can be defined as following (Kreutz et al., 2015)

Forwarding Devices (FD): are data plane software or hardware based devices that can perform a group of elementary operations. They have an instruction sets that used to take an actions on the incoming packets, the Southbound Interface (SI) define the instructions, the SDN controllers install the instructions in forwarding devices and also implementing the protocols for the southbound.

Data Plane (DP): the forwarding devices are interconnected by wireless radio channels or wired cables. The data plane is represented through the network infrastructure that is consisting of interconnected forwarding devices.

Southbound Interface (SI): consists of southbound (API) that defines the sets of the instruction of the forwarding devices, it also defines the protocol for communication between the control plane and forwarding devices elements. The protocol formalizes the interaction between the data plane and the control plane.

Control Plane (CP): it program the forwarding devices through the well-defined southbound interface. It can be known as the network brain and all the control logic reside in the controllers and applications forming the control plane.

Northbound Interface (NI): the Network Operation System (NOS) offers an Application Program Interface (API) to the developers of the application. The northbound interface is represented by this API, the northbound interface is common interface that use for developing applications and also it abstracts the sets of the low level instruction that are used by the southbound interfaces that used to program the forwarding devices.

Management Plane (MP): is the set of the applications that affect the functions that are offered by the northbound interface for the implementation of the operation logic and the network control. The applications are monitoring, firewalls, load balancers, routing, and so forth. The policies that will be translated later to the southbound instructions that are used to program the forwarding devices behavior, these policies are defined by the management application. Figure 2 shows SDN architecture and the open interfaces of between the layers.

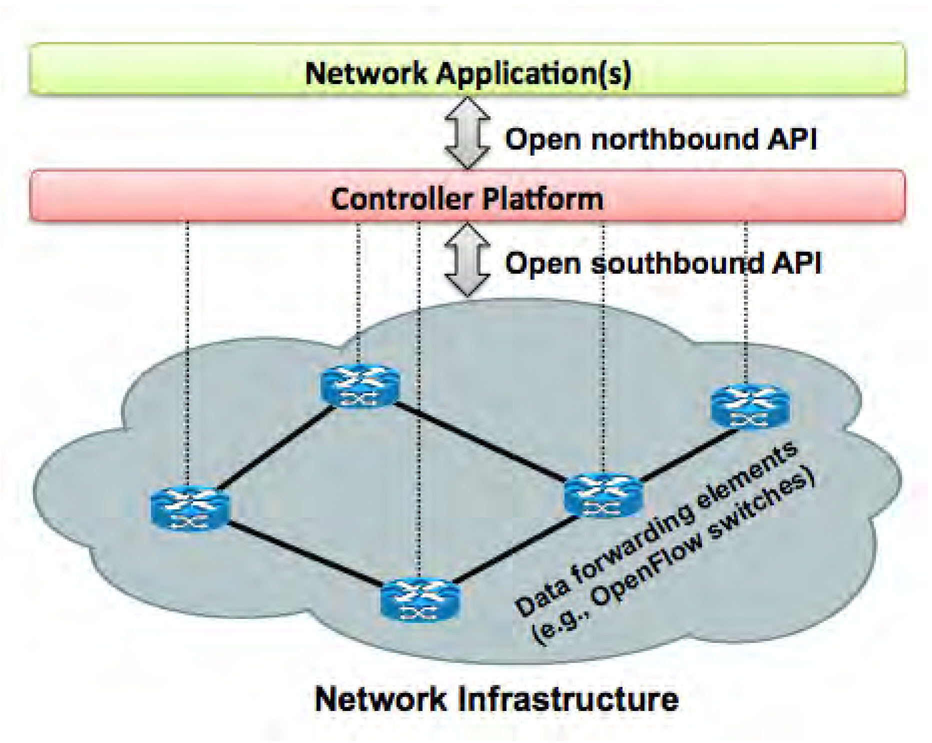


Figure 2 SDN Architecture (Kreutz et al.,2015)

2.1.2 Software Defined Network Advantages

SDN has four major advantages (Hu et al., 2014). Firstly, SDN possesses speed and intelligence: SDN has an ability of optimizing the distribution of workload through the control panel, which leads to transmit in high speed and use the resources in most efficient way. Secondly, SDN allows easy management of the network: the network is controlled remotely by the administrators and also changes the network's characteristics like the services and the connectivity that is based on the patterns of the workload. This will lead to enable the administrators to access the configuration modifications efficiently and instantly.

Thirdly, multi tenancy: the SDN has the ability to be expanded over the multiple partitions of the network like data clouds and data centers. Forthly virtual application networks: it uses the network resources' virtualization to keep the low level physical details hidden from the applications of the users.

2.1.3 Software Defined Network Challenges

As SDN still in its infancy it has so many challenges that need to be addressed and discussed. Sezer et al., 2013 presented four challenges in SDN in terms of questions that discuss performance, scalability, security, and interoperability.

Performance vs. flexibility: "How can the programmable switch be achieved?" One of SDN challenges is the ability to handle high security, high touch and high performance for the packet processing that flows efficiently. The two main elements that should be considered are the performance and the programmability or flexibility. The performance means network node's processing speed putting into consideration the throughput and latency. Programmability refers to the ability of changing and / or accepting a new instructions set and that to change the functional behavior. Flexibility means the capability to adapt systems in order to support new unexpected features.

Scalability: "How to enable the controller to provide a global network view?" Another issue of SDN is scalability; it can be split into scalability of the controller and the scalability of the network node. As the scalability of the controller is the main focus three challenges can be specified. First the latency that occurs due to the exchange of network information between a single controller and multiple nodes. Second is how the communication among controllers carried out using APIs of the east and westbound. Third is the controller back end database size and operation.

Security “How can the SDN be protected from malicious attack?” Fundamental challenge in SDN is the security challenge and protecting the network against attacks. At the level of the controller application many questions about authorization and authentication mechanisms have been raised to give multiple organizations the ability to have an access to the resources of the network and provide a protection for the resources. The same network privilege is not required for all the applications and the model of the security must be in a place to support the protection of the network and to isolate the applications.

Interoperability “How can SDN solutions be integrated into existing network?” This means the transition from traditional networks to SDN. For this all the devices and all the elements needed to be SDN enabled. The simplest transition to a new network is not possible but it is suited for campus network and data centers.

2.2 Related Work

This section presents the related work in three categories, each category will be summarized in a table represents all the work under that category. The first category (A) examines the research papers that used k-center problem as the solution to finds the placement of the controllers. The second category (B) examines the researchers that used different solutions for the controller placement. Finally, (C) is about the work that researched the controllers’ distribution in SDN.

2.2.1 Related Work Based on k-center problem

Heller et al., (2012) is considered as the first to search the controller placement problem in SDN. They focused in their research on finding answers to two main questions as a solution to the controller placement problem which are how many controllers are needed in the network? They also attempted to answer where to place these controllers? To find the answers they chose the propagation latency as the metric divided into average case latency and worst case latency. Finding the optimal minimum latency was not their main aim, but analyzing the problem of the controller placement. As for the topology they chose Internet2 OS3E topology and over 100 topologies for WAN from Internet topology zoo.

They found that for Internet2 OS3E topology the quality of placement is varying. Some are bad, most are mediocre and only a small percentage is optimal. As for the number of the controllers needed it differs depending on whether to optimize the average case latency or the worst case latency since one must be traded for the other. The Internet2 operators suggest that the number of the controllers should be plus one as the extra controller would be for fault tolerance. As for the other topologies they found that the larger the topology becomes more controllers will be needed to reduce the same amount of latency in the small ones. In most topologies one metric should be traded off for the other, while a quarter of the topologies have one solution to optimize both of the metrics. They conclude that the number of the controllers and where to place them depend on the metrics chosen, the topology of the network, and the desired reaction bound.

Yao et al., (2014) introduced the Capacitated Controller Placement Problem (CCPP) that corresponds to the capacitated k-center problem to reduce the load on the controllers. k-center is also used to reduce the radius that they defined as the maximum distance or latency between the controller and the switches related to it. They provided three reasons on why the load on the controllers should be considered which are failures, the limitation of the server capacity and message processing latency. They also defined the four components that form the load of the controllers in SDN. First the processing of PACKET_IN events and the delivery of these events to the applications. Second the view maintaining of the partition of the local network. Third, forming the global view by communication with other controllers, and finally, install the flows entry that are generated by the applications.

The authors pointed that the first component is the most significant part when it comes to the total load. When there is large quantity of messages that will arrive at the controller, there is possibility of bottlenecks for the controller memory, bandwidth and processor. The processing of events plays a major role in determining the availability and the efficiency of the SDN. As a result, they found that capacitated k-center strategy will reduce the number of the controllers that are required in the network to avoid the overload on the controllers, reduce the load on the heaviest load controller and finally reduce the radius of the network.

Jimenez et al., (2014) defined the principles of designing scalable control layer for SDN. They showed the characteristics of the controller that can optimize the network's management. The principles were addressed in term of the controller placement problem. They considered the control layer to be virtual overlay network above of the underlying physical network. The nodes are connected to the controller associated with them in tree form where the controller is the root of the tree.

The authors pointed that to design the control layer, the placement of the controllers and their number must be taken into consideration because good controller placement will lead to balancing the load among the controllers and minimize the communication time. At first they used two algorithms; the first is k-median that was used for the purpose of minimizing the average propagation latency between the controllers and the nodes, and then they used the k-center problem to minimize the furthest distance of the nodes to the closet controller to them. For the controller placement problem, they used k-critical that can find controllers' location and minimum number in order to create robust control topology that can deal with the failure robustly and would be balancing the load among the controllers.

After analyzing, evaluating and comparing to other solutions for the controller placement problem they found that the k-critical gave the best result and achieved its purpose. Also the performance of the network is determined by the number of the controllers as well as their location. If the controllers were more than the optimal number it will be unfit and costly, and bad placement will affect the performance of the network.

Table 1 Related Work of K-center

Author	Metrics	Algorithms	Contribution	Challenges
Heller et al., (2012)	Propagation latency (average case latency, worst case latency)	k-center problem	They found that the number of the controllers and their placement depend on the metrics that have been chosen, the topologies and the	Their main goal was not to minimize the latency or to find the optimal placement but to analyze the problem. In this article they found most of the time one

Author	Metrics	Algorithms	Contribution	Challenges
			desired reaction bound.	of the metrics should be traded off for the other.
Yao et al., (2014)	The load of the controllers and the radius of the network	Capacitated k-center problem	They were able to reduce the number of controllers, the load on the controllers and the radius of the network.	
Jimenez et al., (2014)	Latency, distance and failure	k-median, k-center and k-critical algorithms	Create robust control layer that deal with failure robustly and balance the load	Building tree topology from selected controllers considered different performance metrics and defining load migration mechanism among the controllers

2.2.2 Related Work Based on Different Algorithms

Yannan et al., (2014) proposed Reliability aware Controller Placement (RCP) problem that will decide the placement of the controllers as well as the switches assigned to the controller. Their purpose is to maximize the reliability of the network by minimizing the novel matric that they proposed, which is the expected percentage of the control path loss. They defined the control paths to be the route set that are used as communication mean between the controllers and the switches associated with them and among the controllers themselves. The control path loss was defined as the number of control paths that are broken because of the network failures.

The authors run simulation on real topologies which are Internet2 OS3E topology and Rocketfuel topologies. They evaluated the work using two algorithms, namely the l-w-greedy and the Simulated Annealing (SA). They compared both algorithms with the random placement algorithm. They also examined how the reliability is affected by the number of the controllers and the tradeoff between the latency and the reliability.

The authors found that the SA algorithm gives the best performance while the random placement algorithm was the worst. As for the number of the controllers they found that it is depend on the topology of the network but all the topologies showed that too many controllers or too less will lead to reduction of the reliability of the network. Finally, they found that they were able to improve the reliability without causing unacceptable latencies.

Sallahi &St-Hilaire (2015) proposed mathematical model that can determine the optimal number of controllers as well as their location, the controllers' type and the interconnection between all the elements of the network. The main goal of the model is to minimize the network's cost taking into consideration the various constraints like the controller capacity and the path setup latency.

The authors made an assumption that the following information are known: the switches' location in the network and the amount of traffic that goes from the switch to the controller. The bandwidth length that available for different links type that connect the controllers and the switches. The cost of the controllers and the number of the available physical ports. The maximum number of requests that the controller can handle per second. The number of the controllers that available from each type. Finally, the maximum latency of the link setup that is allowed for the communication between the controller and the switch.

As a result, they found that the model is suitable for planning small scale SDN. For larger problem instances, the solver will take much time and the memory will run out. They set the time limit to 30 hours and after the simulation ended, they found only the settings with small size can be optimized in reasonable time amount and 10% of the problems cannot be solved within the 30 hours.

Lange et al., (2015) presented POCO, a framework based on MATLAB that has the ability to compute the resiliency of Pareto-based Optimal COntroller placement. While POCO is workable and appropriate for small scale and medium scale networks they wanted to evaluated it in large scale network, the metrics that they chose for this purpose are the latency between the nodes and the controller they assigned to as well as the latency among the controllers themselves, balancing the load among the controllers, and the resiliency against the failure of the links and the nodes. They also analyzed the tradeoff between accuracy and time.

For evaluating the framework on large scale network they extended POCO toolset by adding heuristic approach called Pareto Simulated Annealing (PSA) for its ability to be implemented in MATLAB and providing set of solutions at any time. They ran an exhaustive evaluation on number of topologies from Internet zoo topology and they examined different number of solutions for the controller placement problem.

After the evaluation they found that the heuristic approach that had been added made POCO suitable for large scale networks and provided different solutions for the controller placement problem with respect to the four metrics. They left the choice to the decision makers to choose the solution that suitable for their desired requirement and to determine the metrics they want to tradeoff since some of these metrics compete with each other. They also found that the (PSA) is less accurate but it has fast computational time and the accuracy is acceptable. Finally, they found that this approach is suitable for evaluating large problem instances that due to the massive memory requirements cannot be computed.

Ruiz-Rivera et al., (2015) introduced (GreCo) the GREEN CENTRALIZED CONTROLLER algorithm that aim to reduce the consumption of energy in SDN by switching off the maximum number of links in terms of latency, controller load and link utilization. They also developed the Binary Integer Program (BIP) for the purpose of deriving the optimal solution for the problem.

The authors considered the possibility of shutting the links with the shortest path between the controller and the switches which might make the controller search for alternative path. They did not consider the possibility of shutting the switches off, due to the reason of to respond to the network’s events that the controller might need to access the switches related to it. GreCo makes sure that all of the controllers have similar number of the switches which is defined as the optimal number. If the number was more than the optimal number, the algorithm will check the possibility of moving the switches to another controller that has the lowest delay. If there were not such controller then the switches remain with their original controller, then they will examine the link utilization and the load balancing among the controllers. They used different kinds of topologies as well as Yen's algorithm for the purpose of finding the demand's path.

After the evaluation and comparing GreCo to BIP they found that they were able to save up to 55% of the energy during the off peak times and it used 20% more links in comparison with the optimal solution. They also found that shutting off too many links will cause higher load on the remaining links and it will lead to higher consumption of energy in comparing to the time when the links were active.

Table 2 Related Work of Different Algorithms

Author	Metrics	Algorithms	Contribution	Challenges
Yannan et al., (2014)	The expected percentage of control path loss	l-w greedy, Simulated Annealing (SA) and random placement	Minimizing the expected percentage of control path loss and improve the reliability without unacceptable latencies.	Improving the reliability of the control network itself
Sallahi & St-	The cost of network	Mathematical model	Reduce the cost and find the optimal	Suitable for small scale SDN, for

Author	Metrics	Algorithms	Contribution	Challenges
Hilaire , (2015)			number of controllers, their location, type and interconnection between network elements	larger it will consume time and memory
Lange et al., (2015)	Latency between the controller and between the controllers and switches, balancing the load on the controllers, resiliency against failure	Pareto Simulated Annealing (PSA)	They presented POCO a framework that has the ability to fined Pareto optimal placement considering the different performance metrics	The framework has acceptable accuracy but fast computational time also for the optimal placement some metrics needed to be treaded off depend on the decision makers
Ruiz-Rivera et al., (2015)	To reduce the energy consumption, delay, controllers	GreCo , BIP and Yen algorithm	They were able save the energy up to 55% with respect to the metrics	Considering other model for energy consumption where the rate of the energy

Author	Metrics	Algorithms	Contribution	Challenges
	load and link utilization			consumption of links is proportional to the links utilization

2.2.3 Related Work of Distributed Controllers

Dixit et al., (2014) proposed ElastiCon which is elastic distributed controller architecture that can dynamically make the controller pool shrunk or grown depending on the conditions of the traffic. It can also balance the load among the controllers which lead to better performance at all times regardless of the dynamics of the traffic. They proposed 4-phase protocol for switch migration from one controller to the other to balance the load. They designed the protocol to ensure liveness where at least one controller will be active for a switch at all times. Safety where the switch's asynchronous messages processed by exactly one controller, and finally serializability where the events that are transmitted by the switch processed by the controller in the same order.

The authors also proposed three algorithms for the ElastiCon. First the load adaptation algorithm is used to determine whether the current controller pool can handle the current load of the network or not. Second, the rebalancing algorithm that tries to balance the controllers' average utilization. Third, the resizing algorithm that tries to keep the controller utilization between the two presets high and low thresholds. The load was measured by reporting the CPU utilization and the rates of I/O at the controller. After implementing and evaluating the ElastiCon they found that the design achieve it purpose of balancing the load automatically and the controller pool can shrink or grown dynamically depending on the conditions of the traffic.

Santos et al., (2015) proposed framework called D-SDN which is a decentralization SDN framework that enables the controllers to be distributed physically as well as logically in the network in hierarchy form with security as integral part of the framework. The controllers are divided into master or Main Controllers (MCs) and slave or Secondary Controllers (SCs). The slave controllers do not receive messages from the switches, the authors envision the roles of the slave controllers as the one who responsible of managing the switches in sub domain that placed within the master controllers' domain. The slave controllers can be changed into master controllers upon sending request to the master controllers who have the final word in this matter, when the master controllers agree they delegate the slave controllers to act like masters, the communication between the master and the slave controllers is conducted within the same administrative domain. The communication among the slave controllers themselves in designed to be fault tolerance.

In the experiment four controllers were used, one switch with only one node that acted as the master. They all were configured in a wireless ad hoc network and Paxos election protocol was integrated in the framework. After evaluating they found that the framework achieved it purpose of distributing the controllers and it was able to detect failure in optimal time.

Table 3 Related Work of Distributed Controllers

Author	Metrics	Algorithms	Contribution	Challenges
Dixit et al., (2014)	Balancing the load among the controllers	4-phase migration protocol, load adaption algorithm, rebalancing algorithm and resizing algorithm	ElastiCon was able to balance the load automatically and the controller pool shrunk or grown dynamically depending on the condition of the traffic.	They did not consider the factor of controller placement and controller performance in a multi-tenant data centers
Santos et al., (2015)	Fault tolerance	D-SDN	Distribute the controllers not only physically but also logically with the ability to detect failure in optimal time	Balancing the load among the controllers and inter domain routing

The controller placement mechanism proposed in this research is based on the work proposed by Heller et al., 2012, and Dixit et al., 2014. The mechanism is consisting of two algorithms. The first algorithm is k-median to find the placement of the controllers.

This part of the mechanism is based on Heller et al., 2012, where they used k-median to find the placement of the controllers with lowest average propagation latencies. They also used k-center to find the placement of the controllers with lowest worst case latencies. The number of the controllers were chosen solely on propagation latencies, they decided that the number of the controllers should be where the reduction of both propagation latencies reached half.

The topology they used for their research is OS3E topology which is USA based topology, and some other topologies which were also based on the USA. This research topology is MyREN which is Malaysian topology that will explained in more details in chapter 3. In addition, k-median is used only to find the placement of the controllers, after finding the placement the second algorithm will be used to find the number of the controllers.

The second algorithm is delay and load algorithm to evaluate the performance of the network as well as decide the optimal number of controllers. This part of the mechanism is based on Dixit et al., 2014, where the refer that the most direct way to sample if there is load in the network is by sampling the response time. This way was used in this research, also the threshold was set to three seconds which the same as Dixit. They proposed an architecture for controllers' distribution without considering the placement of the controllers.

2.3 Summary

This chapter presented the literature related to SDN. The first section was look through SDN in general, the definition of the network as well as the main features, the layers, the advantages, and the challenges. The second section was about the work related to controller placement, which is divided into three categories based on the algorithm chosen for that work, each category is summarized in table presenting the work belong to that category.

CHAPTER THREE

METHODOLOGY

3.1 Introduction

The process of this research will be conducted as the following (Vaishnavi, V. K., & Kuechler, W. (2015)): first, the awareness of the problem that focus on the problem conducted in this research. Second, the suggestion step that examines the design. Third step is the development which includes the approaches that will be used to find the controller placement using k-median algorithm. Then the evaluation step that will evaluate the proposed placement in terms of the delay and the load on the controllers, which also determine the number of controllers. The final step is the conclusion that will include the final results which is the Controller Placement Mechanism (CPM). Figure 3 shows the methodology framework of this research.

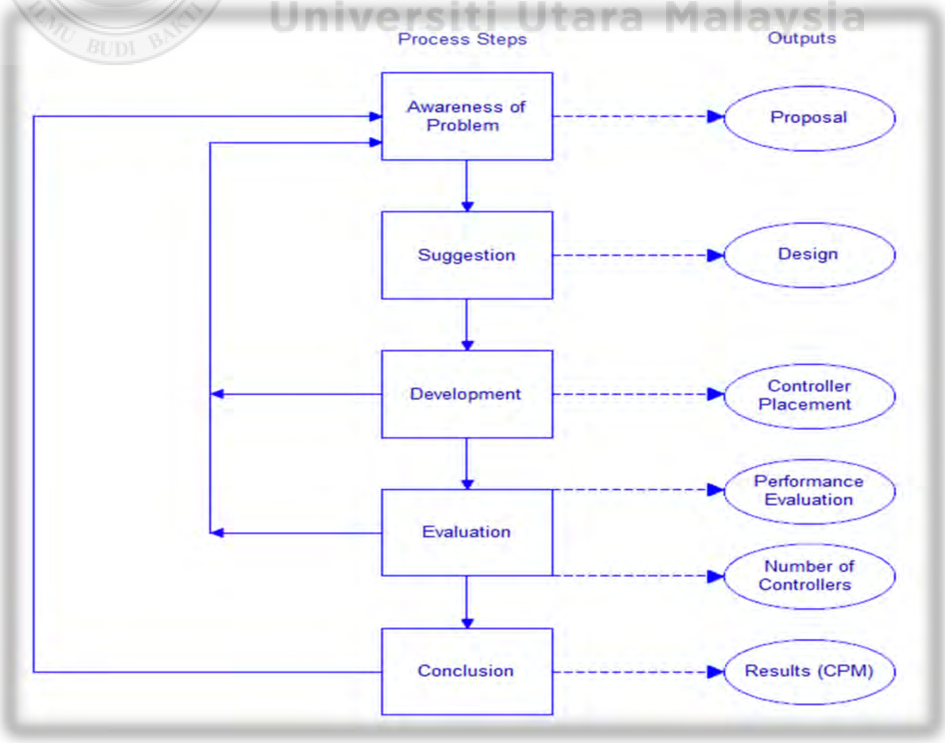


Figure 3 The Research Methodology Framework

3.2 Awareness of Problem

The controller placement problem is one of the main issues in SDN. The aspects of the network will be affected by the controller placement. Through good placement, the network's performance, scalability, and reliability can be improved, depending on the metrics chosen for the placement. The placement is also determined by other two factors, which are the topology and the reaction bound (Heller, Sherwood, & McKeown, 2012).

This study aims to find solution for the controller placement problem in SDN and evaluate the proposed solution in terms of delay and load on the controllers. This work is based on the work proposed by Heller et al., (2012) to find the placement of the controllers. The delay and load algorithm is used to find the optimal number of controller by setting the threshold of response time to three seconds based on Dixit et al., (2014).

3.3 The Proposed Controller Placement Mechanism Design

Heller et al., (2012) is recognized as the first paper to search the controller placement problem with propagation latency as their main metric. Their main objective was to analyse the problem and present their analysis for further studying, which many researchers did at later time where they considered this paper as their main reference or one of the main for instance Yao et al., (2014).

This controller placement mechanism is designed as following: first setting the topology (MyREN), followed by creating graphs of the network. k-median algorithm will be used to calculate average propagation latency for all the possible placements in the network. The resulted combinations of the placement will be compared to greedy algorithm, then to placement from specific node. After the comparison, the performance will be evaluated in terms of delay and load by following delay and load algorithm, and as a result the number of controllers suitable for the network will be decided. Figure 4 shows the design of this work.

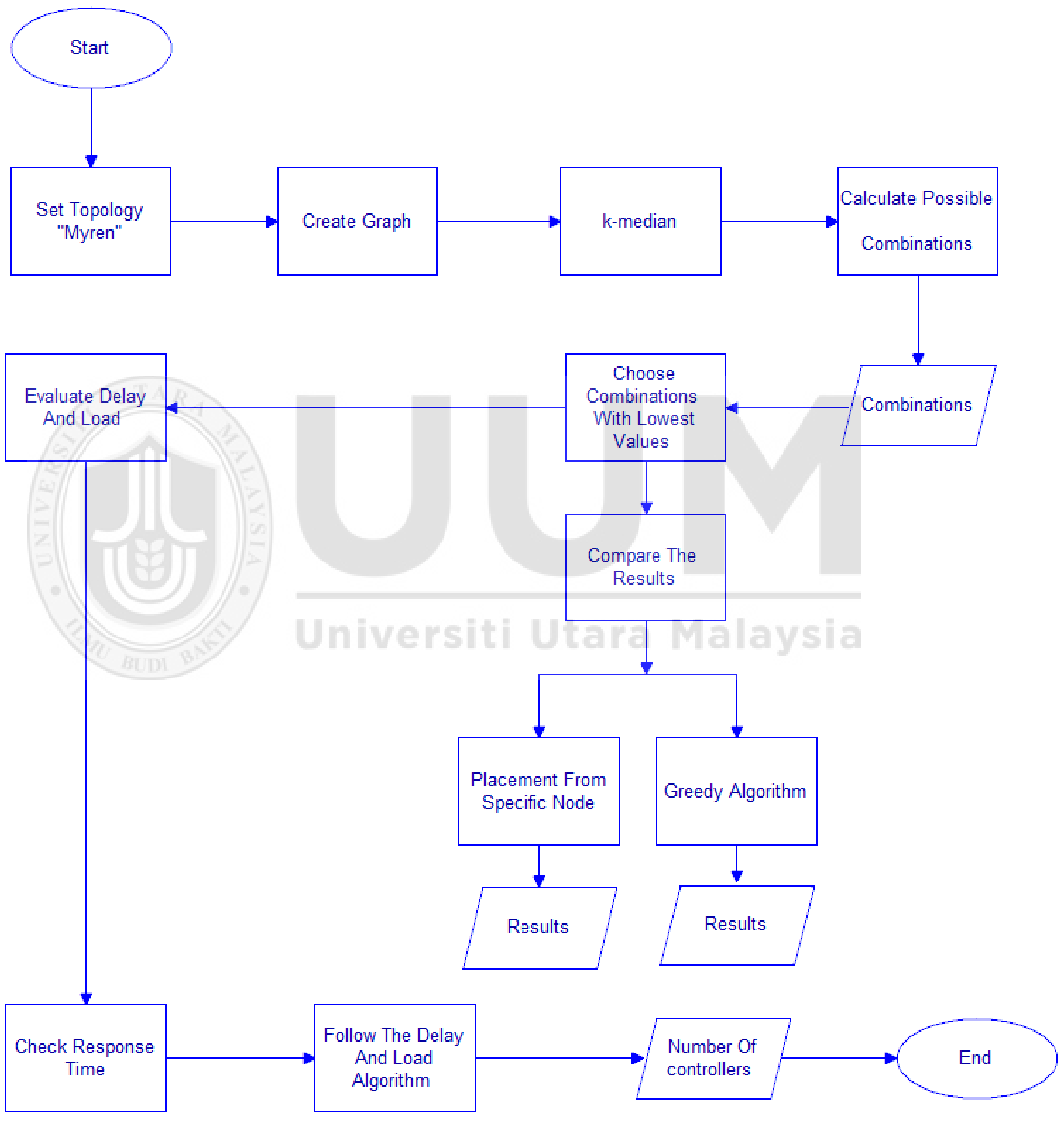


Figure 4: The Design of Controller Placement Mechanism

3.4 Development of Proposed Mechanism

The work is developed using Python Programming Language. The work is based on the work proposed be Heller et al., (2012), who found the placement according to k-median algorithm which finds the average propagation latency, and k-center which finds the worst case latency. This work uses only k-median algorithm that explained in the section 3.4.1. Section 3.4.2 is about simulation and development.

3.4.1 k-median

k-median is an algorithm that used to find k places based on the distances among the points, it will calculate the total length of the shortest distance between two given points, and divided on the total number of the points. In this research, k-median is used to find the average propagation latency according to the formula below.

For network graph $G\ (V, E)$

V Presents the nodes, E is the edge weights that present propagation latency

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

$d(v, s)$ Is the shortest path from the node $v \in V$ to $s \in V$

The number of the nodes $n = |V|$

S´ is the placement from all the possible set of placement S, $|S'| = k$ the number of controllers

$Lavg$ (S´) is minimum, which only the placement with lowest average propagation latency will be chosen.

3.4.2 Simulation and Development Tools

Anaconda 2, Spyder, and Python are used for the purpose of developing and evaluating the controller placement mechanism. Number of Python packages are used for this purpose, some of them will be explained in the next two chapters.

1. Anaconda 2

   It is a free distribution for Python programming language that developed by Continuum analytics "Anaconda @ continuum.io," n.d.). It contains over 400 python packages, and provide a system for package management called Conda, which through any package needed can be installed simply by writing:

```
Conda install 'package name'
```

2. Spyder

   It is a scientific environment for developing python ("Spyder @ pythonhosted.org," n.d.). It is can be used through Anaconda, and it supports the use of multiple consoles for python and Ipython and its include number of the important libraries for python, like numby, matplotlib, scipy.

3. Python

   Python is an open source, high level and structured programming language ("Python @ python.org," n.d.). It is created in the early 1990s by Guido Van Rossum after the comedy program of Monty Python's Flying Circus. It has been growing steadily over the years, as the interest in the language rise for its ability to perform large variety of programming tasks. Python can be used on any operating system; it also has large number of libraries that provides different services. It is powerful ,fast, friendly and easy to learn.

3.5 Performance Evaluation

The topology used in this research is MyREN (Malaysian Research & Education Network) topology which is an inter-institution network that connects researchers, academicians, and scientists across Malaysia through high-speed backbone network ("MyREN @ Myren.net.my," n.d.). It was launched first at March 2005, under the governance of Ministry of Education (MoE) and management of Multimedia Development Corporation (MDeC). The network is telco-neutral and operates on dual stack (IPv4/IPv6) environment. It consists of 37 nodes and 39 edges. Figure 5 shows the network topology of the core network, and Figure 6 shows the border router of the network.

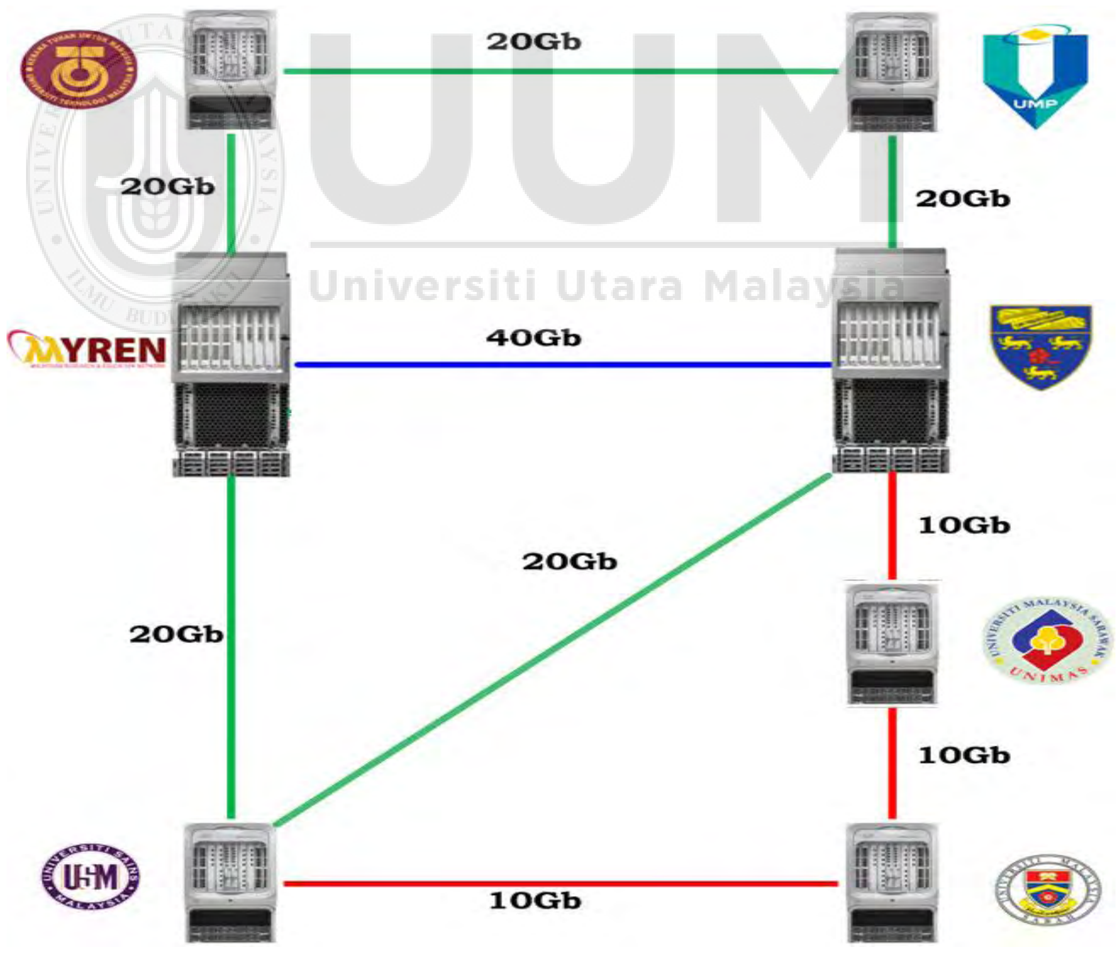


Figure 5 The Network Topology of Core Network

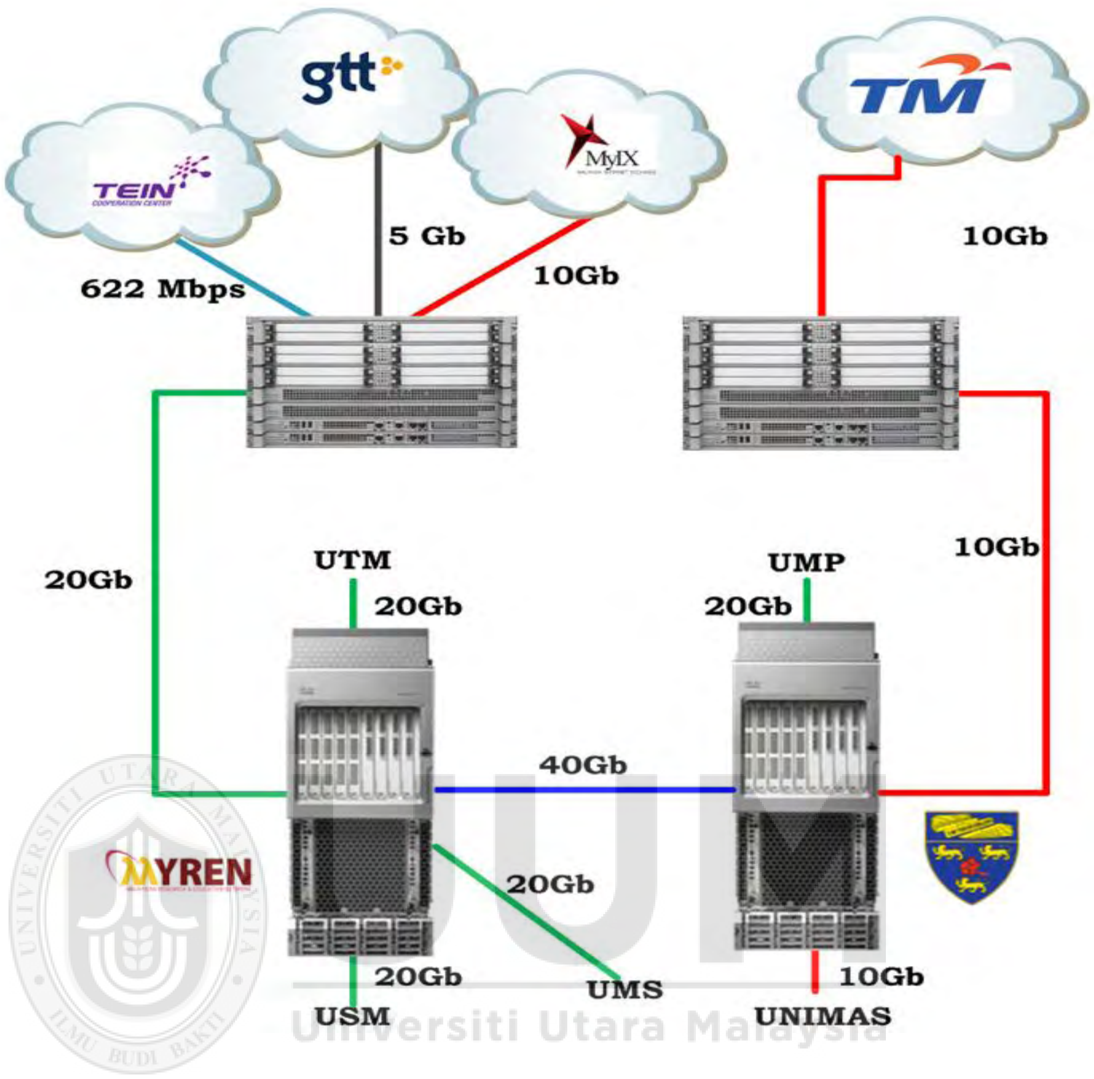


Figure 6 Network Border Router

The performance is evaluated in terms of delay, and load which will be examined according to response time from the controllers. Dixit et al., (2014) pointed that the most direct way to examine the load is by sampling the response time from the controllers, if there is delay in the response, then it is mean there is load on the controller. They set the threshold to sampling the response time to 3 seconds, if it passes this threshold then there is delay and in return there is load.

This work adapts this idea and examined the response time from the controllers, to evaluate the performance of the network, as well as to determine the number of the controllers in the network. The thresholds are high threshold and lower threshold, the high threshold is 3 seconds, if the response time pass it then a new controller will be added to the network. The lower threshold is set to 1.5 seconds, if the response time is less than that, then there is no need to an extra controller as long as the minimum number of the controllers is three controllers. If it passes 1.5 seconds threshold then there is delay and possibility of load, in this case no changes will added to the network.

3.6 Summary

This chapter presented the methodology of this research in details. First section was about the steps of the methodology and the results of each step. The second section was about the problem that this research focus on. The third section presented the design of this research. The fourth section about the development, which presented the algorithm, and the tools. The tools are used for implementation and evaluation. The fifth section explained the topology used in this research, and the evaluation of the performance of the network and how the optimal number of controllers will be chosen.

CHAPTER FOUR

CONTROLLER PLACEMENT MECHANISM

This chapter explains the controller placement mechanism conducted in this research. First, it will explain the mechanism as whole, then the tolls used, after that implementation steps to find the placement. The resulted placement will be explained after that, followed by comparison of the placements, then evaluation of the performance that will decide the number of the controllers. Finally, the summary of this chapter.

4.1 The Proposed Controller Placement Mechanism

The controller placement mechanism proposed in this research consists of two algorithms. The first one is k-median to find all possible placements of controllers in the network. The placements will be decided by calculating the average propagation latency for any placement, then the placements with the lowest values will be chosen. This step will be explained in details through this chapter.

The second algorithm is the delay and load algorithm that sample the response time to determine if there is a delay and load in the network. This algorithm will be used for evaluating the performance as well as determined the optimal number of the controllers, by adding and removing controllers following the algorithm. Figure 7 shows the controller placement mechanism and its two algorithms.

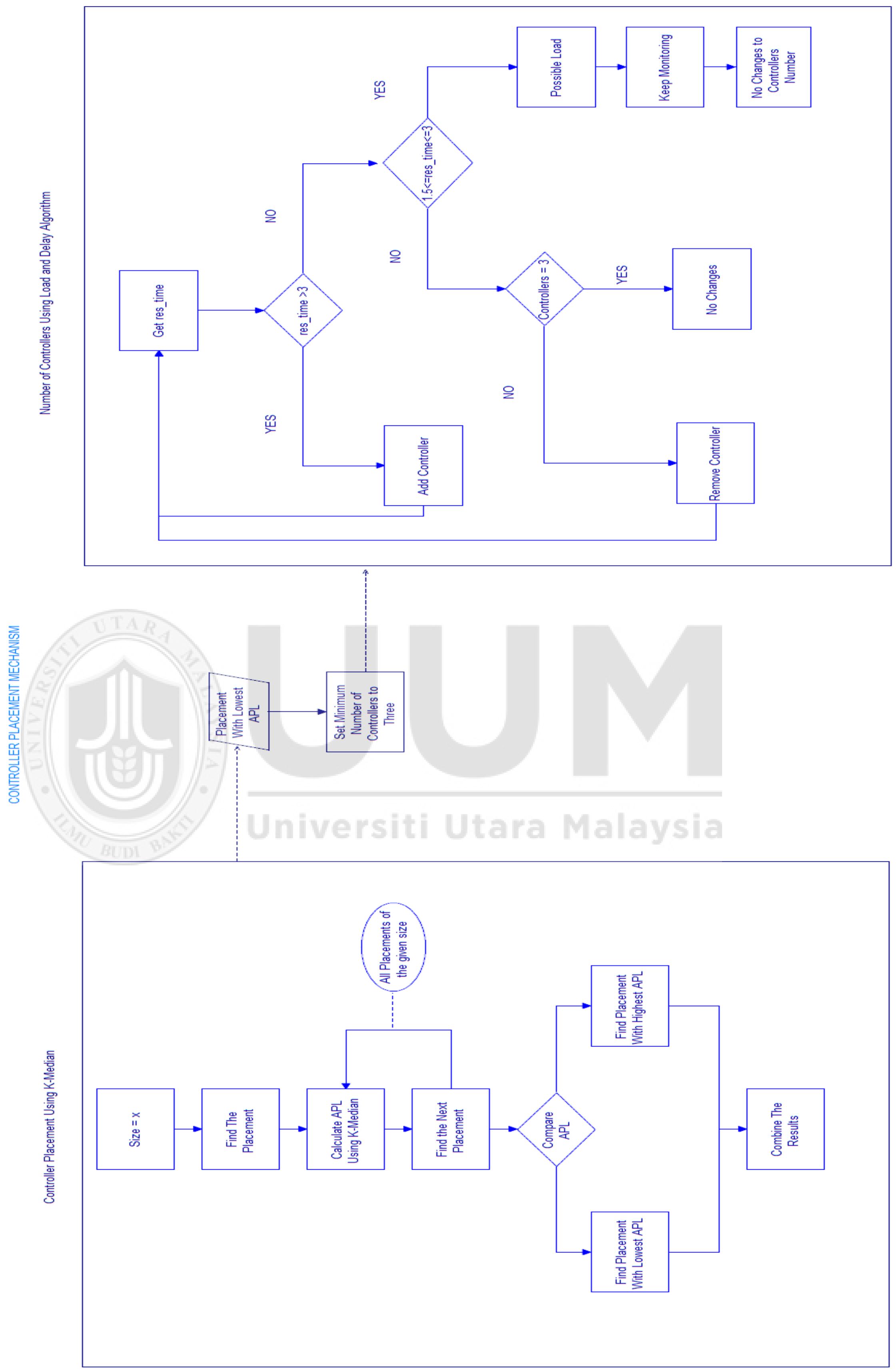

CONTROLLER PLACEMENT MECHANISM
Controller Placement Using K-Median
Size = x
Find The Placement
Calculate APL Using K-Median
All Placements of the given size
Find the Next Placement
Compare APL
Find Placement With Highest APL
Find Placement With Lowest APL
Combine The Results
Placement With Lowest APL
Set Minimum Number of Controllers to Three
Number of Controllers Using Load and Delay Algorithm
Get res_time
res_time >3
YES
NO
Add Controller
1.5<=res_time<=3
YES
NO
Possible Load
Keep Monitoring
No Changes to Controllers Number
Controllers = 3
YES
NO
No Changes
Remove Controller

Figure 7 Controller Placement Mechanism

4.2 Implementation Tools

The implementation was carried on Linux mint 17.3 Rosa Cinnamon 32-bit operating system. The controller placement approach is coded using Python 2.7 through Spyder program as part of Anaconda 2 distribution, below are some of the main packages used for developing the mechanism.

a. Networkx

   Is one of Python packages that used for creating, studying, and manipulating the structure, functions, and dynamics of complex networks(“networkx @ networkx.github.io,” n.d.). In this study, it is used to create the graph of the topology, and to find the nodes and the edges of the topology, and the total number of each.

b. Multiprocessing

   It is Python packages that are used for generating a number of processes, its offers concurrency both locally and remotely (“multiprocessing @ docs.python.org,” n.d.). in this study, this package is used for the purpose of finding all the possible combination in the network.

c. Time

   This package provides all the functions that are related to time(“time @ docs.python.org” n.d.). In this study, the package has been used to calculate the duration for creating all the possible combination of the controller placement. It also used for calculating the duration of sending and receiving from the controller to examine the delay and the load on the controller, and in this case, it was used with socket package.

d. Geo

This package works with the function for geographic coordination. It was included as part of the code and it was used to read the longitude and latitude of each node in the network for the purpose of finding the distance between them.

4.3 Implementation Steps

This section explains the implementation steps that had been taken in order to find the placement of the controllers in SDN. The first step is to choose the topology and coding it. Second is to find the distances between the nodes. In third step k-median is used to find the all possible combinations of the controllers.

1. The Topology

MyREN is the topology chosen for this research, it is Malaysian-based topology that consists of 37 nodes and 39 edges. Two of the nodes were removed from the calculation because they were external nodes that connect the nodes in Malaysia to other countries. The topology had been coded in a class called zoo_myren using the data set of the topology from Internet Topology Zoo, some of the coordinations of the nodes were missing, but were able to obtain from google map, and by contacting the administer of the topology. They were added manually to create the graph. The nodes, the edges, as well as figure 8 showing the graph of the topology, are showing bellow:

Nodes: 35 nodes

[“uitm”, “utp”, "uum" ,"um", "unimas", "ums", "ium", "upsi", "nottingham malaysia", "upm", "uthm", "utm", "umt", "umk", "udm", "ump", "upnim", "utem", "noc", "ukm", "mmu", "usim", "uniten", "mimos", "monash malaysia", "tmrnd", "mohe", "usm", "unimap", "cyberjaya pop", "border_router", "south pop", "east coast pop", "north pop", "kl pop"]

Edges: 37 edges

[(“east coast pop”, “umt”), (“east coast pop”, “ump”), (“east coast pop”, “cyberjaya pop”), (“east coast pop”, “kl pop”), (“east coast pop”, “umk”), (“east coast pop”, “udm”), (“south pop”, “uthm”), (“south pop”, “kl pop”), (“south pop”, “utem”), (“south pop”, “cyberjaya pop”), (“south pop”, “utm”), (“unimas”, “kl pop”), (“unimap”, “north pop”), (“mohe”, “cyberjaya pop”), (“north pop”, “uum”), (“north pop”, “usm”), (“north pop”, “kl pop”), (“north pop”, “cyberjaya pop”), (“upsi”, “kl pop”), (“ums”, “kl pop”), (“uniten”, “cyberjaya pop”), (“uitm”, “kl pop”), (“usim”, “cyberjaya pop”), (“monash malaysia”, “cyberjaya pop”), (“ukm”, “cyberjaya pop”), (“ium”, “kl pop”), (“tmrnd”, “cyberjaya pop”), (“mm”, “cyberjaya pop”), (“noc”, “cyberjaya pop”), (“kl pop”, “upnim”), (“kl pop”, “cyberjaya pop”), (“kl pop”, “um”), (“kl pop”, “utp”), (“border_router”, “cyberjaya pop”), (“upm”, “cyberjaya pop”), (“cyberjaya pop”, “mimos”), (“cyberjaya pop”, “nottingham malaysia”)]

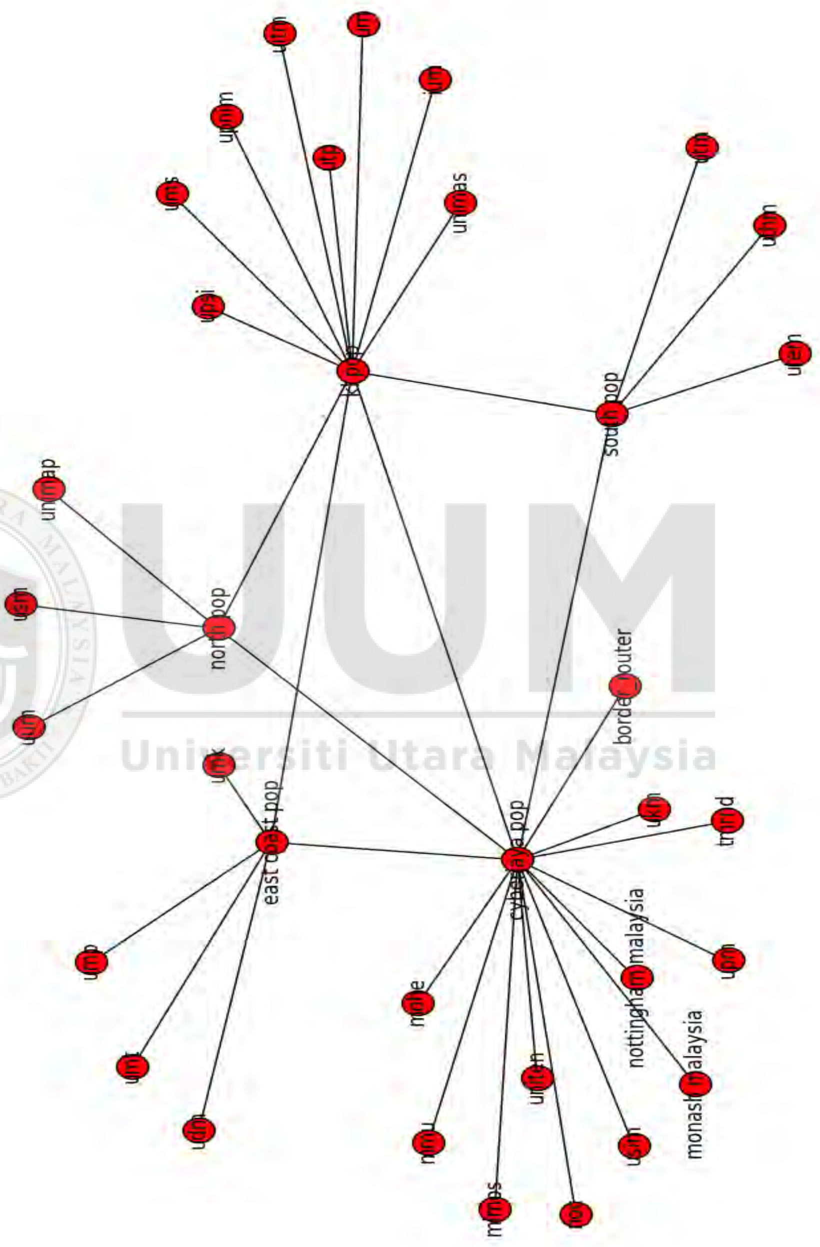
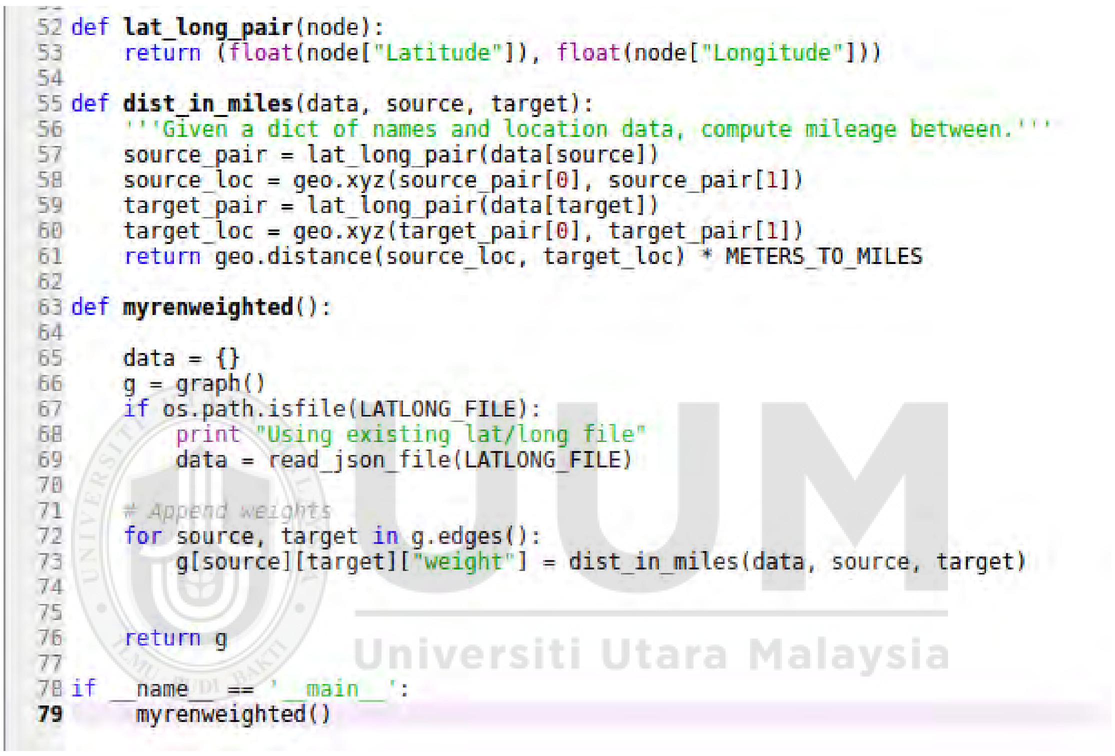


Figure 8 MyREN Topology

2. Calculating distances

For this purpose, the longitude and the latitude of each node must be used, so (.json) file was created containing each node and its longitude and latitude. Figure 9 shows the code for calculating the distances and creating a weighted graph.



```
def lat_long_pair(node):
    return (float(node["Latitude"]), float(node["Longitude"]))

def dist_in_miles(data, source, target):
    '''Given a dict of names and location data, compute mileage between.'''
    source_pair = lat_long_pair(data[source])
    source_loc = geo.xyz(source_pair[0], source_pair[1])
    target_pair = lat_long_pair(data[target])
    target_loc = geo.xyz(target_pair[0], target_pair[1])
    return geo.distance(source_loc, target_loc) * METERS_TO_MILES

def myrenweighted():

    data = {}
    g = graph()
    if os.path.isfile(LATLONG_FILE):
        print "Using existing lat/long file"
        data = read_json_file(LATLONG_FILE)

    # Append weights
    for source, target in g.edges():
        g[source][target]["weight"] = dist_in_miles(data, source, target)


    return g

if __name__ == '__main__':
     myrenweighted()
```

Figure 9 The Code for Creating Weighted graph

The function MyRENweighted() is the main function, which will read the file that contains all the node coordination of latitude and longitude. Then it will call dist_in_miles()function that calculates the distance from one node to another in the edge in order to create a weighted graph, it calls lat_long_pair() function which returns the latitude and longitude of each node. The function uses geo package to read the coordination of the source, and target nodes in the edges, then return the distances in miles. The graph used in MyRENweighted is the topology graph created in zoo_myren class.

3. Finding the Placement

For this purpose, two classes were created where each contains a number of functions to find all the possible combinations of the placement. Each combination is formed by considering each node as a possible placement for the controller. The total path length of that node will be calculated, and the average propagation latency will be calculated using k-median, which divided the total path length on the number of the nodes. Then all combinations resulted of the same combo size will be compared to each other to get the highest combination with the highest average propagation latency, lowest combination with the lowest propagation latency, the mean which presents the average, the sum of all the average propagation latencies calculated, and the number of how many combinations was calculated for that specific combo size.

The first class metrics_MR is the main class that contains get_controllers() function, it decides how many possible placement should be calculated. This function is called through other function called do_metrics(), which can be considered as the main function. It calculates apsp which is the shortest path length in the graph that calculated using networkx package, and apsp_paths is the same as apsp but here it calculated for weighted graph. After that it call run_all_combos() function.

run_all_combos() is included in another class called metrics_mrlib, which can be considered as the place where all the calculations done. The class contain number of functions, that run_all_combos call to calculate. This function use multiprocessing package to calculate all the possible combinations of controllers' placement for any given size. It first calls handle_combo_all() function. This function in return calls three important functions. First init_metric_data(), which contains the intial values for the metric.

The second one is get_latency() which calculates the average propagation latency according to k-median algorithm. It gets the total path length from another function, which in returns examining the closest nodes to the one that considered to place the contollers at. After the calculations are finished it will return the values calculated to handle_all_combo.

The returned values will be handled by process_results() function which is responsible for comparing the lowest and highest values of average propagation latency, as well as the placements. It will also calculate the summation of all propagation latencies in that given combo size, and how many combinations of placements were calculated. All calculations will be returned to run_all_combos()

run_all_combos() will call function called merge_data_metric. It is responsible of merging the results of calculations and create a list of all possible placements of any given size, and returned the values to run_all_combos(). run_all_combos() will calculates the average of each placements and ruturns all the calculation as well as the average to do_metrics in the first class.

Upon receiving all the calculations from run_all_combos(), do_metrics will first print the reults showing all the information of each placements size. The information is: combo size, lowest latency, lowest combos, highest latency, highest combos, mean, sum, and the number. Then all of these information will be stored in (.json) file. Figure 10 shows the placement of controllers of any given size. Figure 11 shows the functions used and their sequence to find the placement.

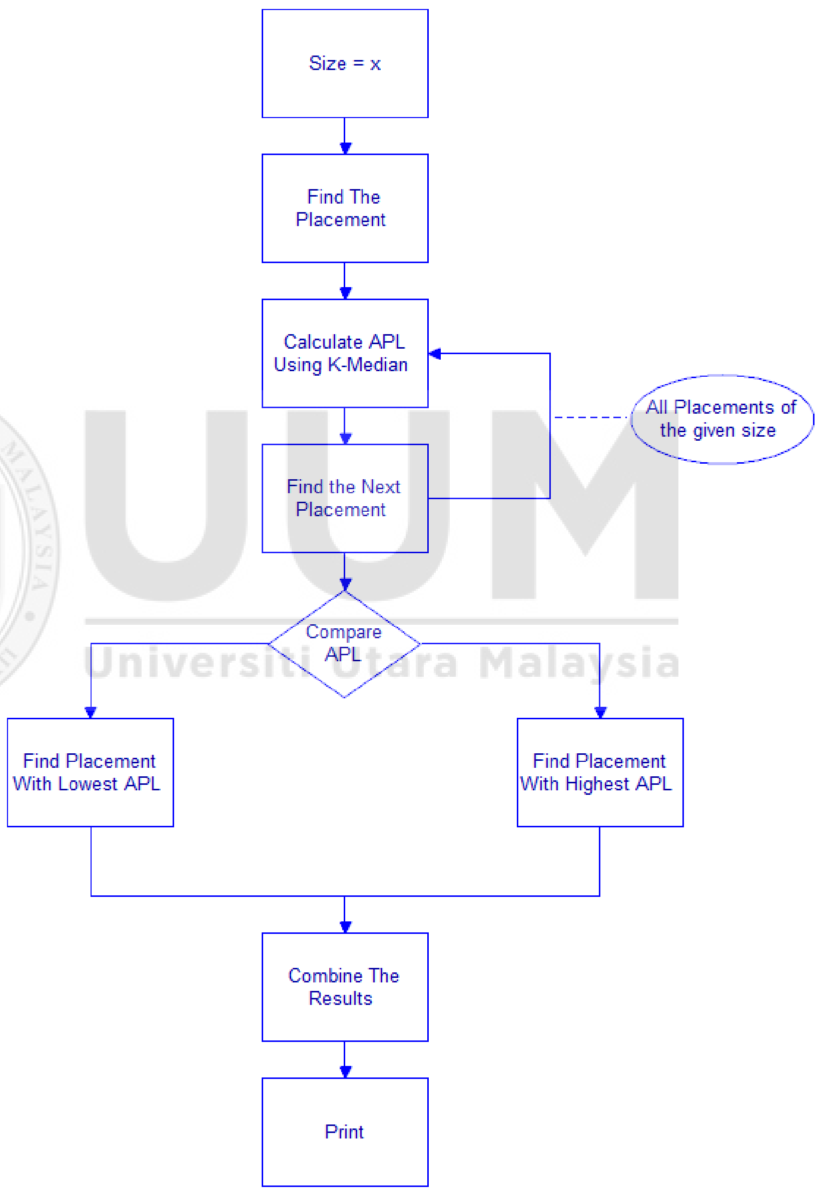

Size = x
Find The Placement
Calculate APL Using K-Median
All Placements of the given size
Find the Next Placement
Compare APL
Find Placement With Lowest APL
Find Placement With Highest APL
Combine The Results
Print

Figure 10 Controller Placement

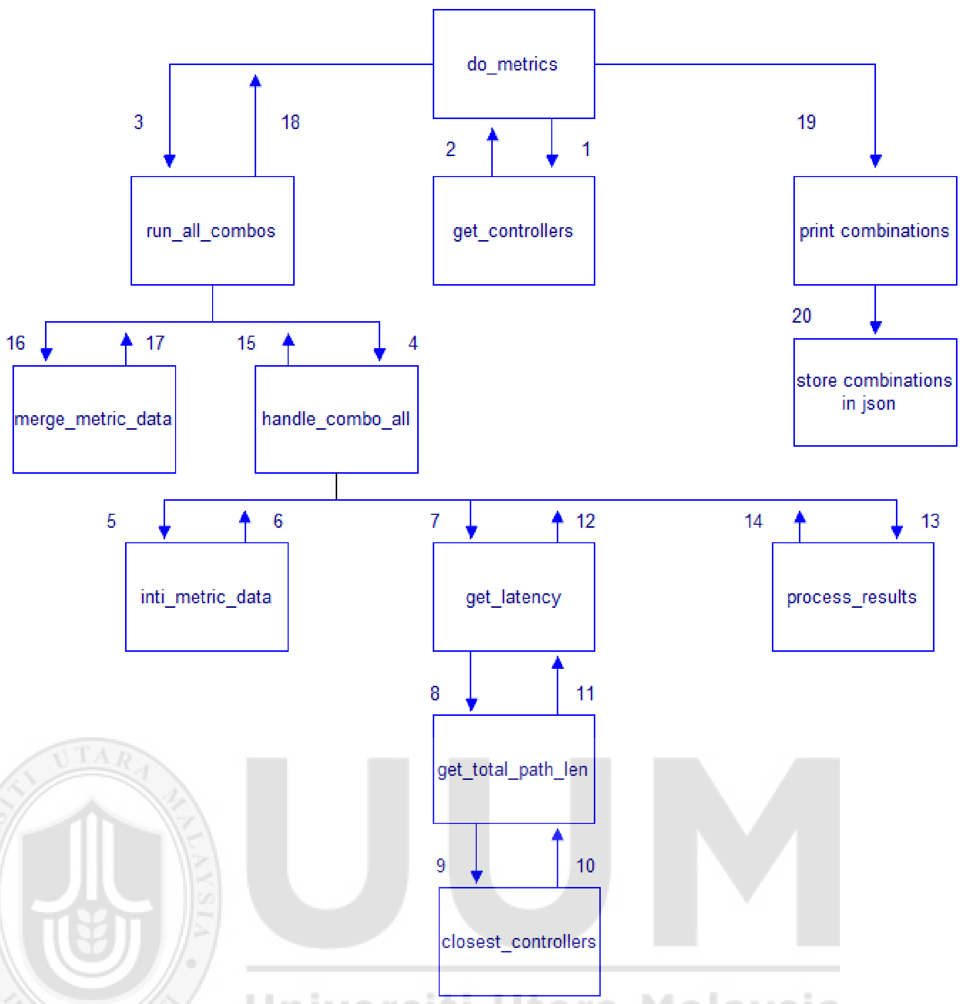


Figure 11 Placement Functions

4.4 The Placement Validation

This section shows the combinations of the controller placement resulted from using the k-median. Figure 12 shows the highest and lowest average propagation latencies up to eight controllers. As it is shown in the figure, the lowest average propagation latency decrease from (169.66714220977224) for one controller to (34.93987130148739) for eight controllers, which show decrement by 79.41%. The highest propagation latency decreases from (1121.4315981719635) for one controller to (253.61243087710884) for eight controllers, which show decrement by 77.38%. The highest propagation latency showed dramatic decrement up to four controllers, then it starts to decrease by small amount. As for the lowest propagation latency it showed steadily decrement all the time. The controllers' placement with the lowest average propagation latency is selected as the suitable placement for the controllers. Table 5 shows the details of the placements up to eight controllers. The number of the controllers will be chosen through evaluating the controllers in terms of examining the response time it takes to send and receive from the controller to see if there are delay and load on the controller.

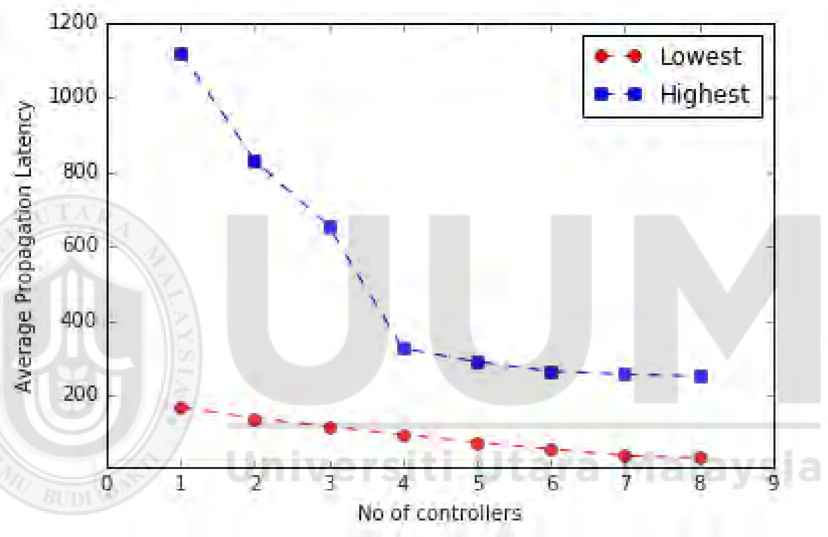


Figure 12 Highest, Lowest Average Propagation Latency

Table 4 The Placement of The Controllers

No. Cont	Highest		lowest	
	Placement	Average propagation latency	Placement	Average propagation latency
1	[“ums”]	1121.4315981719635	[“ mmu”]	169.66714220977224
2	[“ums”, “mimos”]	831.2908626063158	[“ums”, “mmu”]	140.39145153399411
3	[“unimas”, “ums”, “mimos”]	656.3246572463133	[“north pop”, “ums”, “mmu”]	118.15887780498913
4	[“unimas”, “ums”,	327.7288161485884	[“north pop”,	96.5682712089791

No. Cont	Highest		lowest	
	Placement	Average propagation latency	Placement	Average propagation latency
	"umk", "mimos"]		"ums", "mmu", "mimos"]	
5	["unimas", "umt", "umk", "udm", "mimos"]	291.83801473983556	["south pop", "north pop", "ums", "mmu", "mimos"]	76.39367758375272
6	["unimas", "uthm", "umt", "umk", "utem", "udm"]	266.7339794730244	["east coast pop", "south pop", "north pop", "ums", "mmu", "mimos"]	57.777374010703646
7	["unimap", "north pop", "umt", "usm", "umk", "uum", "udm"]	258.4214674790002	["east coast pop", "south pop", "unimas", "north pop", "ums", "mmu", "mimos"]	40.00358982434542
8	["south pop", "unimap", "uthm", "north pop", "usm", "utem", "uum", "utm"]	253.61243087710884	["east coast pop", "south pop", "unimas", "north pop", "ums", "umk", "mmu", "mimos"]	34.93987130148739

The table above showing the placement of the highest and lowest average propagation latency and the values of each placement. As it shows the placement of one controllers for the highest average propagation latency is at (“ums”) with value of (1121.4315981719635), while for the lowest average propagation latency is at (“mmu”) with value of (169.66714220977224). The values of both of highest and lowest average propagation latencies decreased as the number of controllers increased.

4.5 Evaluation of Performance

This research is based on the work proposed by Heller et al., (2012). It calculates both of average propagation latency using k-median algorithm, and worst case latency using k-center problem. In this research, the controller placement was found using k-median only. The resulted placement will be compared to two possible placements. First the placement resulted will be compared to greedy algorithm that consider the ordering of the nodes, the greedy algorithm calculates only the best values at each step. The other comparison will be with the placement resulted from using (“nx.closeness_centrality”), which is part of networkx package that had been explained in chapter four. It considers the placement from specific node at the center of the topology. The detail of each comparison will be shown below.

To confirm the results of the placement in MyREN topology, a greedy algorithm was used to compute the ordering of the nodes. It also calculates the average propagation latency using k-median and the total path length, but its only calculates the best values for the placement at each step. The results of the greedy algorithm were identical to the controller placement as shown in the table below. Figure 13 shows the lowest and greedy algorithm average propagation latency.

Table 5 The Placement Calculated and Greedy Calculations

No	Placement using k-median algorithm		Greedy algorithm	
	Combo	Latency	Combo	Latency
1	[“ mmu”]	169.66714220977224	[“ mmu”]	169.66714221
2	[“ums”, “mmu”]	140.39145153399411	[“mmu”, “ums”]	140.391451534
3	[“north pop”, “ums”, “mmu”]	118.15887780498913	[“mmu”, “ums”, “north pop”]	118.158877805
4	[“north pop”, “ums”, “mmu”, “mimos”]	96.5682712089791	[“mmu”, “ums”, “north popo”, “mimos”]	96.568271209
5	[“south pop”, “north pop”, “ums”, “mmu”, “mimos”]	76.39367758375272	[“mmu”, “usm”, “north pop”, “mimos”, “south pop”]	76.3936775838
6	["east coast pop", "south pop", "north pop", "ums", "mmu", "mimos"]	57.777374010703646	[“mmu”, “ums”, “north pop”, “mimos”, “south pop”, “east coast pop”]	57.7773740107
7	["east coast pop", "south pop", "unimas", "north pop", "ums", "mmu", "mimos"]	40.00358982434542	[“mmu”, “ums”, “north pop”, “mimos”, “south pop”, “east coast pop”, “unimas”]	40.0035898243
8	["east coast pop", "south pop", "unimas", "north pop", "ums", "umk", "mmu", "mimos"]	34.93987130148739	[“mmu”, “ums”, “north pop”, “mimos”, “south pop”, “east coast pop”, “unimas”, “umk”]	34.9398713015

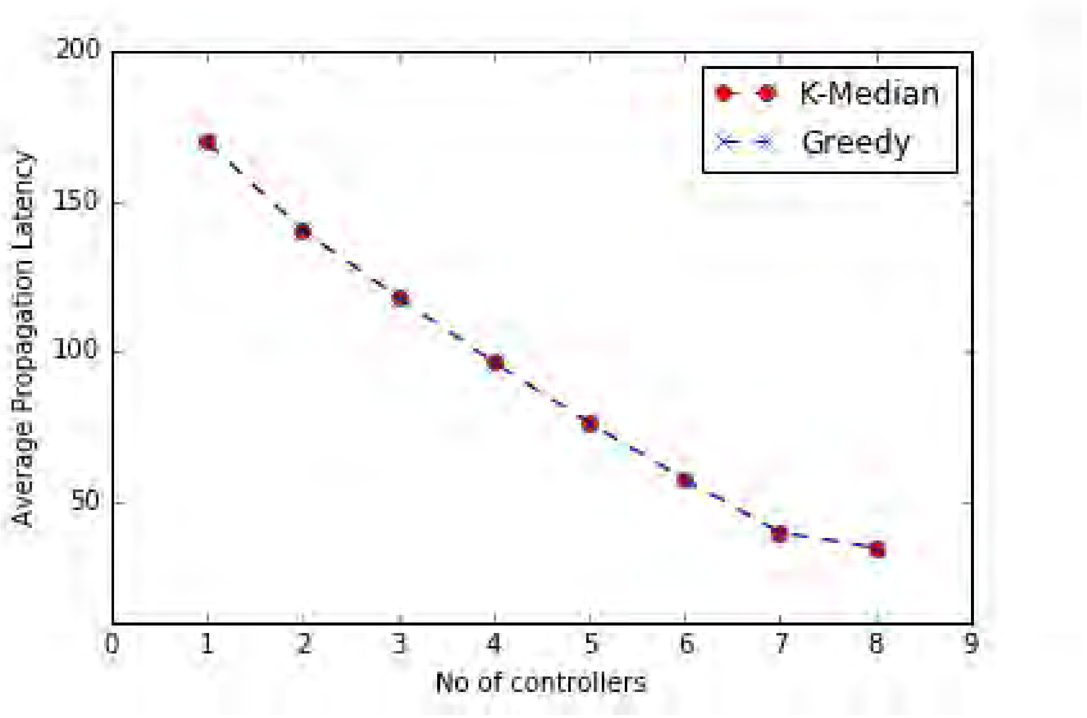


Figure 13 K-Median and Greedy Algorithm

The second comparison was to the placement resulted from using ("nx.closeness_centrality") which calculates the shortest distances to certain nodes, and the propagation latency once again was calculated by the k-median algorithm. The average propagation latency resulted from the algorithm is less than using this method by 65.15% up to eight controllers. Table 7 and Figure 14 show the comparison.

Table 6 Comparing the Results of Each Placement

No	Placement using k-median algorithm		nx.closeness_centrality	
	Combo	Latency	Combo	Latency
1	[“ mmu”]	169.66714220977224	[“cyberjaya pop”]	169.66714221
2	[“ums”, “mmu”]	140.39145153399411	[“cyberjaya pop”, “kl pop”]	163.183031736
3	[“north pop”, “ums”, “mmu”]	118.15887780498913	[“‘cyberjaya pop”, “kl pop”, “east coast pop”]	145.744941782
4	[“north pop”, “ums”, “mmu”, “mimos”]	96.5682712089791	[“‘cyberjaya pop”, “kl pop”, “east coast pop”, “south pop”]	125.570348157
5	[“south pop”, “north pop”, “ums”, “mmu”, “mimos”]	76.39367758375272	[“‘cyberjaya pop”, “kl pop”, “east coast pop”, “south pop”, “north pop”]	104.729396369
6	["east coast pop", "south pop", "north pop", "ums", "mmu", "mimos"]	57.777374010703646	[“cyberjaya pop”, “kl pop”, “east coast pop”, “south pop”, “north pop”, “mohe”]	104.673148327
7	["east coast pop", "south pop", "unimas", "north pop", "ums", "mmu", "mimos"]	40.00358982434542	[“cyberjaya pop”, “kl pop”, “east coast pop”, “south pop”, “north pop”, “mohe”, “uniten”]	100.949887613
8	["east coast pop", "south pop", "unimas", "north pop", "ums", "umk", "mmu", "mimos"]	34.93987130148739	[“cyberjaya pop”, “kl pop”, “east coast pop”, “south pop”, “north pop”, “mohe”, “uniten”, “usim”]	100.270240595

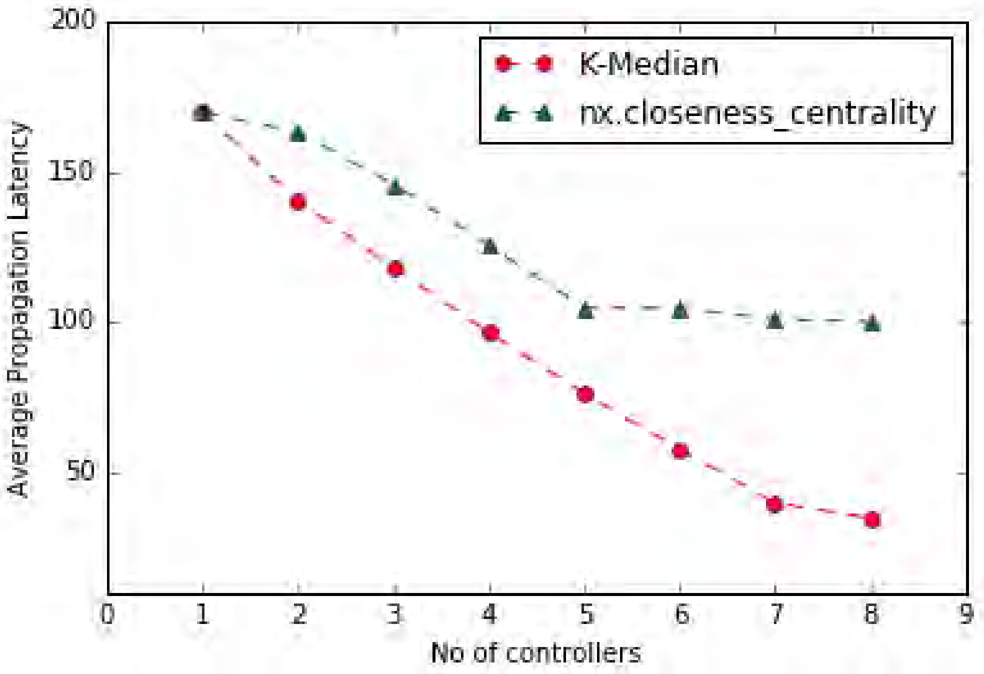


Figure 14 Shows Average Propagation Latency for Both Methods

The figure above shows that the average propagation latency resulted from the algorithm continue to decrease widely, while using ("nx.closeness_centrality") shows wide decreasing up to five controllers, then it starts to decrease in very small amounts that can be shown as straight line between (5-6) and (7-8). Figure 15 shows all of the comparisons.

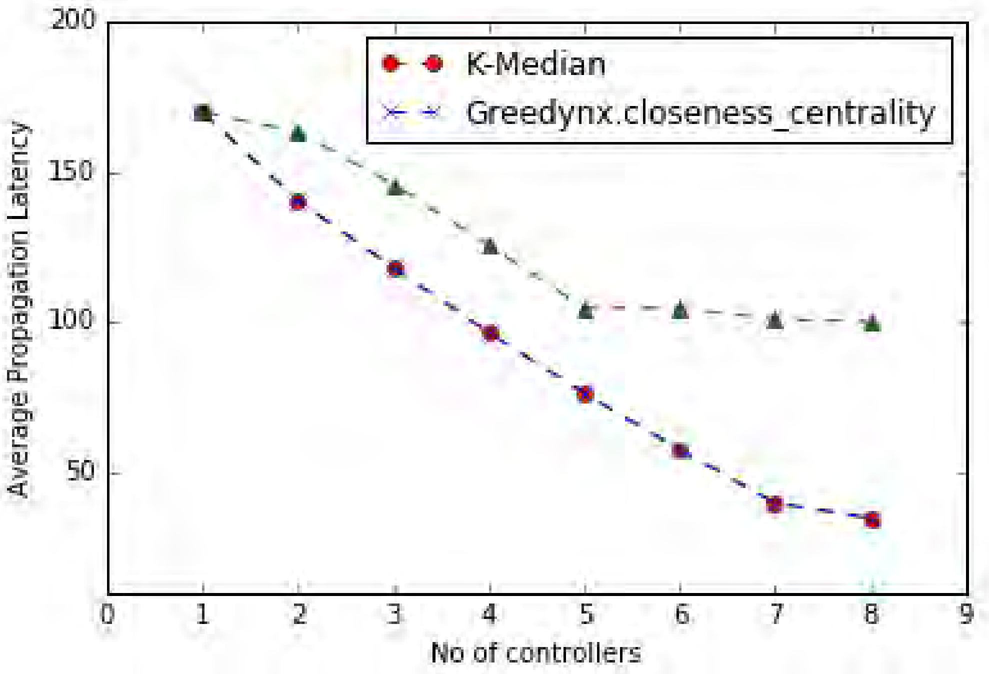


Figure 15 Comparing All Methods

4.6 Results and Discussion

To evaluate the placement of the controllers in MyREN topology, three controllers were chosen as start, because although one controller might be enough according to Heller et al., (2012), but in case of failure the whole network is down, and in case of two controllers and one of them was down the remaining one will have all the load, so three controllers were chosen for fault tolerance reasons.

The placement of the three controllers is (“north pop”, “mmu”, “ums”) as it shown in Figure 14. This research assuming that the switch assigned to the controller at “mmu” is “cyberjaya pop”. For the controller at “ums” the switch will be “kl pop”. As for the controller at “north pop” it connected to two nodes “cyberjaya pop” and “kl pop”, the switch will be “cyberjaya pop” because it has more nodes connected to it, so assuming that the nodes will be divided between the two controllers. The communication to calculate based on the response time between the controller and the switch assigned to it, regardless of which node send it.

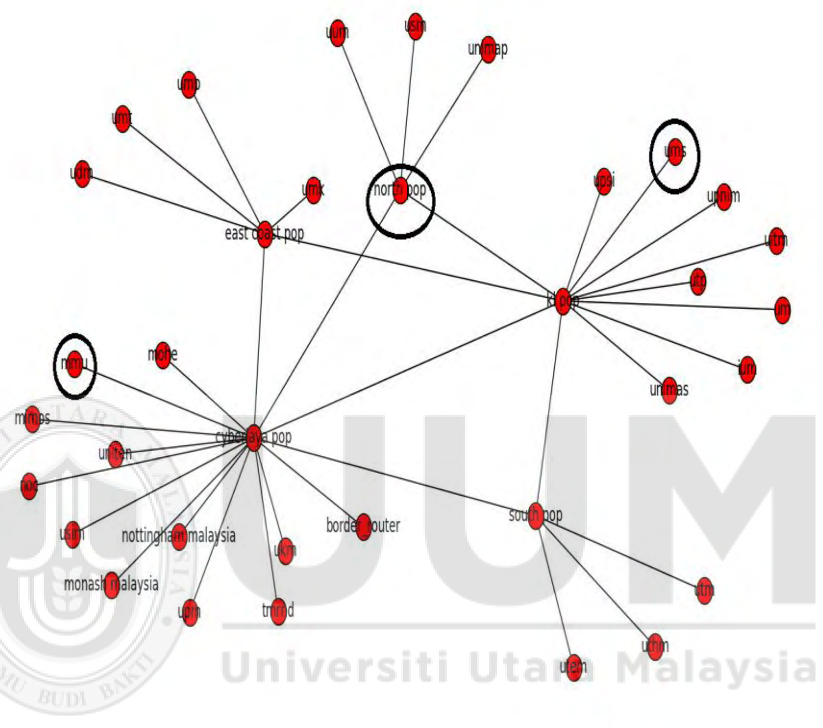


Figure 16 The Placement of Three Controllers

In this step socket package was used for the network communication over TCP/IP, at first, 100 messages were sent to each controller and the time was calculated using time package, then 1000 messages were sent, and finally 10000 messages were sent. Table 8 shows the maximum, minimum, and total response time results of each time.

Table 7 The Results of The Response Time

	100	1000	10000
C1	Maximum Delay:0.0278768539429 Minimum Delay:0.000993967056274 Total:0.248062849045	Maximum Delay:0.0278980731964 Minimum Delay:0.00096607208252 Total:4.29643654823	Maximum Delay:0.061882019043 Minimum Delay:0.000959873199463 Total:43.1862213612
C2	Maximum Delay:0.0258641242981 Minimum Delay:0.0012309551239 Total:0.567728281021	Maximum Delay:0.023885011673 Minimum Delay:0.00095009803772 Total:3.82639598846	Maximum Delay:0.0357580184937 Minimum Delay:0.000944137573242 Total:42.5973906517
C3	Maximum Delay:0.0195109844208 Minimum Delay:0.000967025756836 Total:0.272239685059	Maximum Delay:0.0301787853241 Minimum Delay:0.000960826873779 Total:4.25423502922	Maximum Delay:0.0385489463806 Minimum Delay:0.00092887878418 Total:42.2689399719

To examine if there is load on the controllers the compassion was set according to the algorithm showing in figure 17 If the response time was over 3 seconds according to Dixit et al., (2014) then there is a delay which equals to the response time and in turn, there is load and new controller must be added. If it is between 1.5 to 3 seconds then there is a possibility of loading, less than that remove a controller as along as the minimum controller's number is equal to three. Figure 17 shows the evaluation of the placement in terms of delay and load by sampling the response time to determine the suitable number of controllers.

Delay and Load Algorithm

```
while True do
    check res_time
    if res_time > 3 then
        add controller
    else
        if 1.5 <= res_time <= 3 then
            do nothing
        else
            if res_time > 1.5 then
                check controllers
                if controllers = 3 then
                    do nothing
                else
                    remove controller
                end if
            end if
        end if
    end if
end while
```

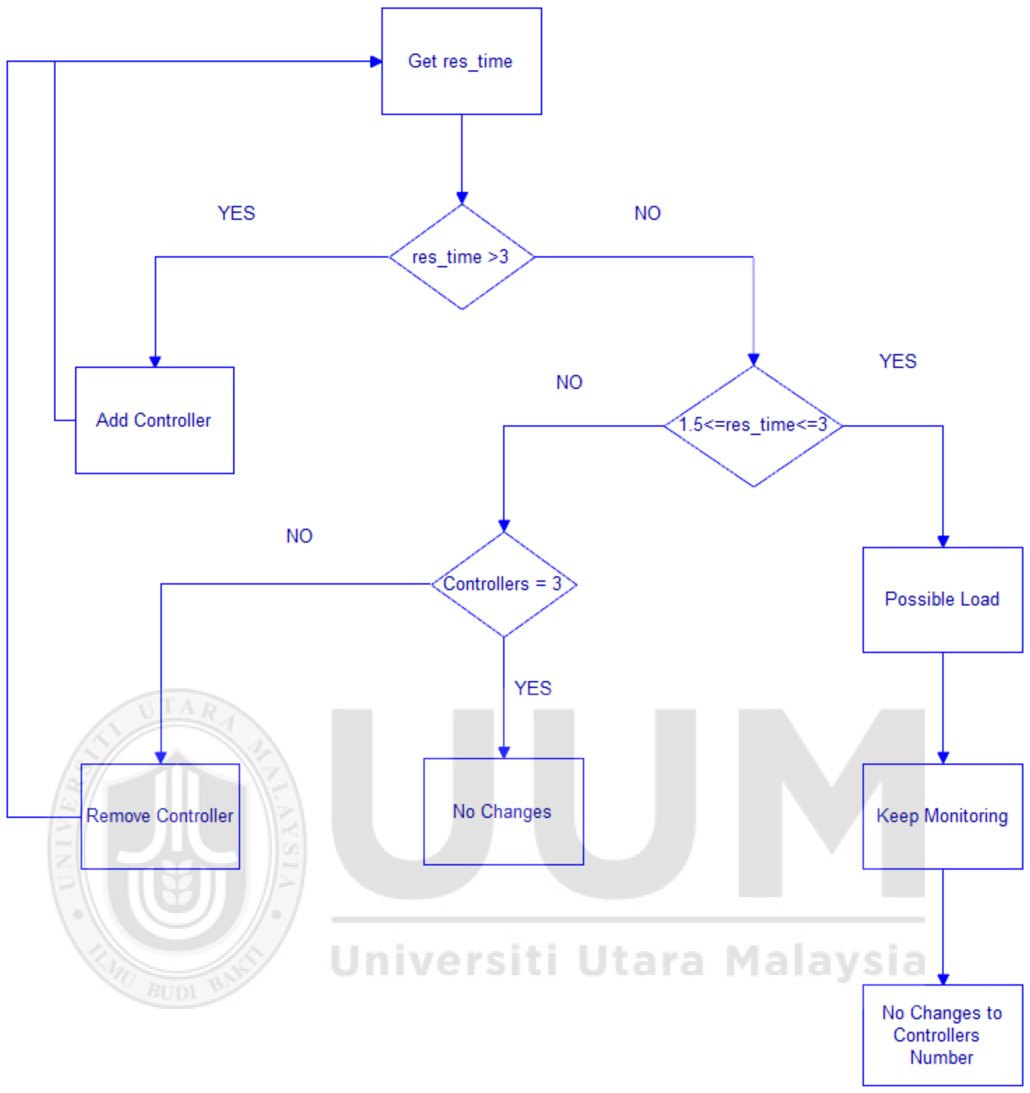
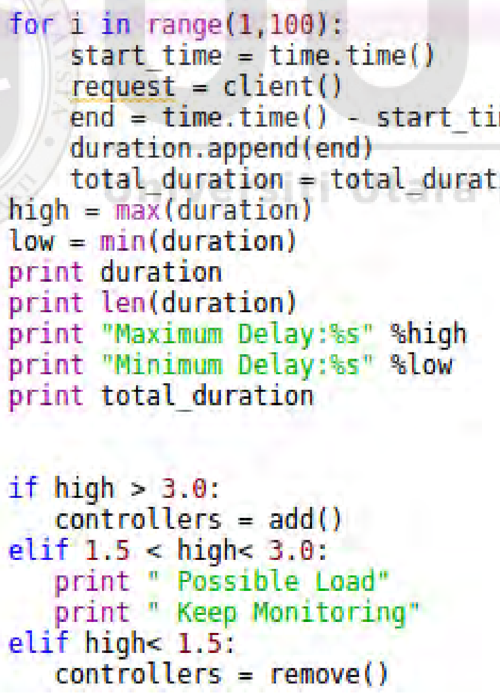


Figure 17 Checking the Response Time

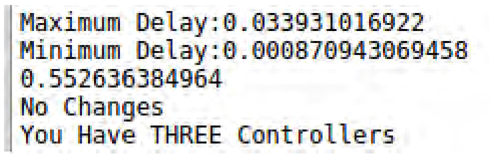
The evaluation shows that there is no delay more than 3 seconds, so the number of the controllers is set to three controllers. As long as there no delay more than 3 seconds or load on the controllers, there is no need to add controllers to get better performance of the network.

Then the algorithm was tested in case of delay and load by changing the threshold. At first the code ran the same as it is to find the maximum delay, then the higher threshold was changed from 3 seconds to the maximum delay of the pervious step. It was noticed that every time the code is run a new controller is added based on the placement that is already found. Then the high threshold was changed once again to 3 seconds, and every time the code is run the controllers that were added are removed until it reached three controllers, which is the minimum number and stopped the changing. Figure 18 shows the code without changing the high threshold. It provided the results showing in Figure 19. The change is showing in Figure 20, and the results are showing in Figure 21 for the first run, and 22 for the second. After changing the threshold back to 3 seconds, the results are showing in Figure 23,24,25 in sequence.



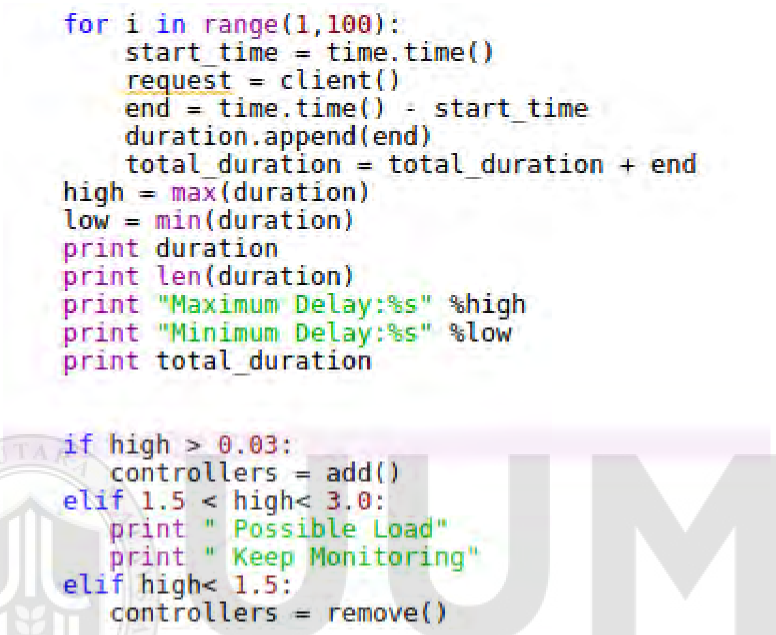
```
for i in range(1,100):
    start_time = time.time()
    request = client()
    end = time.time() - start_time
    duration.append(end)
    total_duration = total_duration + end
high = max(duration)
low = min(duration)
print duration
print len(duration)
print "Maximum Delay:%s" %high
print "Minimum Delay:%s" %low
print total_duration


if high > 3.0:
   controllers = add()
elif 1.5 < high< 3.0:
   print " Possible Load"
   print " Keep Monitoring"
elif high< 1.5:
   controllers = remove()
```

Figure 18 Before Changing the Threshold



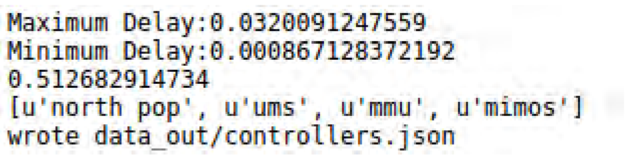
```
Maximum Delay:0.033931016922
Minimum Delay:0.000870943069458
0.552636384964
No Changes
You Have THREE Controllers
```

Figure 19 The Results Before The Change



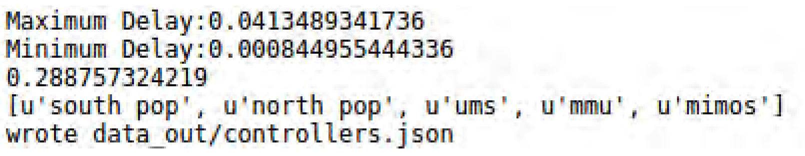
```
for i in range(1,100):
    start_time = time.time()
    request = client()
    end = time.time() - start_time
    duration.append(end)
    total_duration = total_duration + end
high = max(duration)
low = min(duration)
print duration
print len(duration)
print "Maximum Delay:%s" %high
print "Minimum Delay:%s" %low
print total_duration

if high > 0.03:
   controllers = add()
elif 1.5 < high< 3.0:
   print " Possible Load"
   print " Keep Monitoring"
elif high< 1.5:
   controllers = remove()
```

Figure 20 After The Change



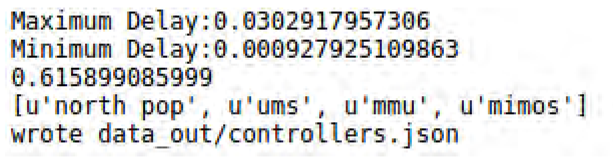
```
Maximum Delay:0.0320091247559
Minimum Delay:0.000867128372192
0.512682914734
[u'north pop', u'ums', u'mmu', u'mimos']
wrote data_out/controllers.json
```

Figure 21 The First Add After The Change



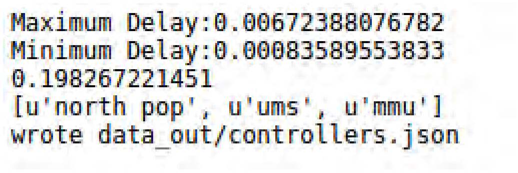
```
Maximum Delay:0.0413489341736
Minimum Delay:0.000844955444336
0.288757324219
[u'south pop', u'north pop', u'ums', u'mmu', u'mimos']
wrote data_out/controllers.json
```

Figure 22 The Second Add After The Change



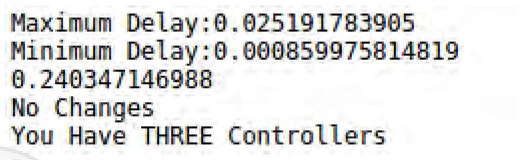
```
Maximum Delay:0.0302917957306
Minimum Delay:0.000927925109863
0.615899085999
[u'north pop', u'ums', u'mmu', u'mimos']
wrote data_out/controllers.json
```

Figure 23 The First Remove After Changing Back



```
Maximum Delay:0.00672388076782
Minimum Delay:0.00083589553833
0.198267221451
[u'north pop', u'ums', u'mmu']
wrote data_out/controllers.json
```

Figure 24 The Second Remove After Changing Back



```
Maximum Delay:0.025191783905
Minimum Delay:0.000859975814819
0.240347146988
No Changes
You Have THREE Controllers
```

Figure 25 The Final Run

4.7 Summary

This chapter presented the controller placement mechanism development. First section explained the mechanism as whole, and the algorithms combined to create the mechanism. The second section was about the tools. Third section was about the implementation steps to find the placement. The fourth section presented the placement resulted from using k-median. The fifth section compared the placement to another two possible placements. The sixth section was about evaluation of the placement to decide the optimal number of controllers.

CHAPTER FIVE

CONCLUSION

This chapter evaluates the proposed controller placement mechanism in terms of response time delay and load on the controllers. The first section will the contribution of this research. The second section will be about the limitation of this research. The third section will be future work, and finally the conclusion.

5.1 Contribution

SDN controller placement is one of the main challenges in SDN network for its ability to affect the performance of the network. By finding an appropriate placement for the controllers will ensure that the performance of the network will be in a good state. In this research MyREN topology was set, and the placement of the controllers were chosen through the use of k-median algorithm, which calculate the average propagation latency by dividing the total path length on the number of the nodes in the network.

The resulted placement were compared to greedy algorithm that consider the order of the nodes, and calculate the best values at each step, it was found that the results are identical, then the resulted placement were compared to ("nx.closeness_centrality") which consider the placement from specific node in the network, it was found that not only the placement results from k-median provides better results, but it also showed that as the number of the controllers increase the average propagation latency resulted from this method decrease in very small amounts that are hardly noticed.

Later, the resulted placements were examined in terms of delay and load, which is done through sampling the response time, and it was found that three controllers that were set as the minimum number of the controllers in the network, is suitable as long as there is no delay or load in the network, since in that case it will require more controllers.

5.2 Limitation

The load on the controllers in this research was just examined through the response time. If there is a delay means there is a load on the controllers, but the actual level of the load was not calculated. Also, the delay and the load were considered only to evaluate the placement of the controllers, and to determine the number of the controller suitable for the network, but they were not part of finding the placement of the controllers. The placement combinations were find using only k-median algorithm.

5.3 Future Work

For future work, the limitations of this research will be considered, as well as taking the placement of switches and how many nodes assign to each switch into consideration of the controller placement, through customizing the topology itself to achieve that. Also, other metrics should be taking into considerations like reliability, scalability, throughput, fault tolerance... etc.

5.4 Conclusion

This study searched the problem of the controller placement in SDN using k-median algorithm for searching the placement, and MyREN topology to find the placement for the controllers. The number of the controllers was decided be following the delay and load algorithm, which is three controllers. The number of the controllers should be suitable for the network, to ensure that the performance of the network is acceptable. If the number of the controllers were less than acceptable, then the performance of the network will be bad, and there is a great possibility of delay and load. In a case of too many controllers, the performance will be better, but it is on the expenses on the physical cost, and there will be a number of controllers that will not be needed since the performance can be good without the unnecessary numbers of the controllers. To find locations and the number of the controllers the topology chosen must be considered, the metrics chosen for achieving that because different metrics and different topologies give different results.

REFERENCES

1. Dixit, A. A., Hao, F., Mukherjee, S., Lakshman, T. V., & Kompella, R. (2014, October). ElastiCon: an elastic distributed sdn controller. In *Proceedings of the tenth ACM/IEEE symposium on Architectures for networking and communications systems* (pp. 17-28). ACM.
2. Heller, B., Sherwood, R., & McKeown, N. (2012). The controller placement problem. *ACM SIGCOMM Computer Communication Review*, *42*(4), 473. doi:10.1145/2377677.2377767
3. Hu, F., Hao, Q., & Bao, K. (2014). A Survey on SDNing (SDN) and OpenFlow: From Concept to Implementation. *IEEE Communications Surveys & Tutorials*, *16*(c), 1–1. doi:10.1109/COMST.2014.2326417
4. Hu, Y., Wendong, W., & Gong, X. (2013). Reliability-aware controller placement for Software-Defined Networks. *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on , vol., no., pp.672-675, 27-31*
5. Introduction-to-Mininet @ github.com. (n.d.). Retrieved from https://github.com/mininet/mininet/wiki/Introduction-to-Mininet
6. Jiménez, Y., Cervelló-Pastor, C., & García, A. J. (2014). On the controller placement for designing a distributed SDN control layer. *2014 IFIP Networking Conference, IFIP Networking 2014*. doi:10.1109/IFIPNetworking.2014.6857117
7. Kreutz, D., Rothenberg, C. E., Ieee, M., Azodolmolky, S., Ieee, S. M., Uhlig, S., & Ieee, M. (2015). Software-Defined Networking : A Comprehensive Survey, *103*(1). doi:10.1109/JPROC.2014.2371999
8. Lange, S., Gebert, S., Zinner, T., Tran-gia, P., Hock, D., Jarschel, M., & Hoffmann, M. (2015). Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks, *12*(1), 4–17.

9. Open Networking Foundation. (2012). Software-Defined Networking: The New Norm for Networks [white paper]. *ONF White Paper*, 1–12.
10. Pinheiro, R. S., Pinheirot, B. A., Jorge, A., & Abelem, G. (2014). Model of Organization and Distribution of Applications for SDNs : SDNrepo, 188–193.
11. Ruiz-rivera, A., Chin, K., & Soh, S. (2015). GreCo : An Energy Aware Controller Association Algorithm for SDNs. *IEEE Communication Letters*, *19*(4), 541–544.
12. Sallahi, A., & St-hilaire, M. (2015). Optimal Model for the Controller Placement Problem in SDNs, *19*(1), 30–33.
13. Santos, M. a S., Nunes, B. a a, Obraczka, K., & Turletti, T. (2014). Decentralizing SDN ' s Control Plane, 1–4.
14. Sezer, S., Scott-Hayward, S., Chouhan, P., Fraser, B., Lake, D., Finnegan, J., … Rao, N. (2013). Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine*, *51*(7), 36–43. doi:10.1109/MCOM.2013.6553676
15. Vazirani, V. V. (2001). Approximation Algorithms. *Approximation Algorithms*, *94*(2), xx+378. doi:10.1002/rsa.10038
16. Vaishnavi, V. K., & Kuechler, W. (2015). *Design science research methods and patterns: innovating information and communication technology*. Crc Press.
17. Wang, H. (2014). Authentic and Confidential Policy Distribution in Software Defined Wireless Network, 1167–1171.
18. Xia, W., Wen, Y., Member, S., Heng Foh, C., Niyato, D., & Xie, H. (2015). A Survey on Software-Defined Networking. *IEEE Communication Surveys & Tutorials*, *17*(1), 27–51. doi:10.1109/COMST.2014.2330903

19. Yannan, H. U., Wendong, W., Xiangyang, G., Xirong, Q. U. E., & Shiduan, C. (2014). On Reliability-optimized Controller Placement for Software-Defined Networks, (February), 38–54.

20. Yao, G., Bi, J., & Guo, L. (2013). On the cascading failures of multi-controllers in SDNs. *Proceedings - International Conference on Network Protocols, ICNP*, (1), 3–4. doi:10.1109/ICNP.2013.6733624

21. Yao, G., Bi, J., Li, Y., & Guo, L. (2014a). On the Capacitated Controller Placement Problem in SDNs. *IEEE Communications Letters*, *18*(August), 1339–1342. doi:10.1109/LCOMM.2014.2332341

22. Anaconda. https://www.continuum.io.

23. Spyder. https://pythonhosted.org.

24. Python. https://www.Python.org.

25. MyREN. https://www.myren.net.my.

26. Networkx. https://networkx.githup.io.

27. Multiprocessing. https://docs.python.org

28. Time. https://docs.python.org.

29. Google Maps. https://maps.google.com

30. Google Earth. https://earth.google.com

APPENDIX

The following code is the main page that call all the other class to find the placement of the controllers:

```
COMBOS_FILE = 'data_out/myren_combos.json'
import logging
import os
import time

import networkx as nx


from file_libs import write_json_file, read_json_file
import metrics_mrlib as metrics
from myren_weighted import myrenweighted
from lib.options import parse_args
import json

logging.basicConfig(level=logging.DEBUG)



def get_controllers(g, options):

    controllers = []
    if options.controllers:
        controllers = options.controllers

    else:
        # Controller numbers to compute data for.
        controllers = []

        # Eventually expand this to n.
        if options.compute_start:
            controllers += range(1, options.from_start + 1)
        if options.compute_end:
            controllers += (range(g.number_of_nodes() - options.from_end + 1, g.number_of_nodes() + 1))
    print controllers
    return controllers



def do_metrics(options, topo, g):

    print "computing metricss for topo: %s" % topo
    controllers = get_controllers(g, options)
    data = {}  # See top for data schema details.
    apsp = nx.all_pairs_dijkstra_path_length(g)
    apsp_paths = nx.all_pairs_dijkstra_path(g)
    start = time.time()
    weighted = True
    metrics.run_all_combos(options.metrics, g, controllers, data, apsp,
                           apsp_paths, weighted, options.write_dist,
                           options.write_combos, options.processes,
                           options.multiprocess, options.chunksize, options.median)
    total_duration = time.time() - start
    print "the total duration for all the combos:"
    print "%0.6f" % total_duration


    print "*******************************************************************"

    write_json_file('data_out/myren_combos' + '.json', data)
    return data

def opt_metric(options, topo, g):


    apsp = nx.all_pairs_dijkstra_path_length(g)
    apsp_paths = nx.all_pairs_dijkstra_path(g)
    with open(COMBOS_FILE) as data_file:
        data = json.load(data_file)
        metrics.run_greedy_informed(data, g, apsp, True)
        metrics.run_greedy_alg_dict(data, g, 'greed_cc', 'latency', nx.closeness_centrality(g), apsp, True)

if __name__ == '__main__':
    options = parse_args()
    topo = options.topo
    g = myrenweighted()
    do_metrics(options, topo, g)
    opt_metric(options, topo, g)
```