ANALYSIS ON LOAD BALANCING MECHANISMS FOR BETTER NETWORK PERFORMANCE USING OPEN FLOW

MOHAMAD FIQHREEL EIZIQ BIN ISMAIL

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

ANALYSIS ON LOAD BALANCING MECHANISMS FOR BETTER NETWORK PERFORMANCE USING OPEN FLOW

MOHAMAD FIQHREEL EIZIQ BIN ISMAIL

This report is submitted in partial fulfilment of the requirements for the

Bachelor of Computer Science (Computer Networking)

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2019

DEDICATION

To my beloved parents, thank you for always support me through my up and down.

To all my friends who always support me to complete this project,

Thanks a lot.

Last but not least, special thanks to my lovely and supportive supervisor, thanks for all motivation, encouraging and believe me to complete this project.

May all bless us. Amin

ACKNOWLEDGEMENT

First of all, thanks to ALLAH S.W.T for this mercy and guidance in giving me full strength to complete this project. I would like to express my gratitude and appreciation to all those who gave me the possibility too complete this project. A special thanks to our final year project supervisor, Dr. Zaheera, whose help, stimulating suggestions and encouragements, helped me to coordinate my project especially in writing this report.

Then, I would like thanks to my parents, for supporting me mentally and physically not just finishing this project but also during my whole studies. Not to forget, I would like to thanks Universiti Teknikal Malaysia Melaka (UTeM) for giving me the opportunity to produce this project. Last but not least, grateful acknowledgement to all my friends who never give up in giving their support to me in all aspects of life. Thanks you very much my friends. I will never forget all of your kindness.

ABSTRACT

The idea of Software-Defined Networking is rapidly growing. Providers have made use of the idea of Software-Defined Networking has a lot of advantages, despite the challenges they face. There is a possible scalability issue with relying on a single controller in future networks. In order to overcome these conventional network-based problems, a newly emerging technology Software Defined Networking (SDN) has been implemented which vastly simplifies the data plane and control plane and makes the network fully programmable. This project proposes an approach to a load balancer with the implementation of SDN. The algorithm using for SDN Load Balancing is Round Robin. The design topology consists of the SDN switch and the Open Day Light controller. The packet entries are stored in the flow table stored in the data plane. The project separates the control and data plane and regulates the controller using Open Day Light controller. Such isolation makes it easier to manage the load balancers. Using this method, the process is explicitly programmable and agile. Requests from different clients will be guided to different pre-defined servers in Round-Robin mode.

ABSTRAK

Idea ‘Software Defined Networking’ berkembang dengan pesat. Penyedia menyatakan idea untuk menggunakan ‘Software Defined Networking’ mempunyai banyak kelebihan, walaupun perdapat halangan yang perlu dihadapi. Terdapat beberapa skala kemungkinan isu dengan bergantung kepada satu pengawal pada masa akan datang. Untuk mengatasi masalah-masalah berasaskan konvansional ini, satu teknologi baru yang telah dilaksanakan iaitu ‘Software Defined Networking’ untuk menyerderhanakan ‘control plane’ dan ‘data plane’ dan menjadikan rangkaian itu dapat diprogramkan sepenuhnya. Project ini dicadang untuk menggunakan pendekatan ‘SDN Load Balancing’ Bersama ‘Round Robin’ algorithma. Rekaan topologi terdiri daripada ‘SDN Switch’ dan pengawal ‘Open Day Light’. Paket yang masuk disimpan di dalam jadual aliran dalam ‘Data Plane’. Projek ini memisahkan pengawal, ‘Data Plane’ dan mengawal kawalan menggunakan pengawal ‘Open Day Light’. Pengasingan sedemikian menjadikannya lebih mudah untuk menguruskan ‘Load Balancer’. Menggunakan kaedah ini, proses ini secara jelas dapat diprogramkan dengan mudah. Permintaan daripada klien yang berbeza akan di bimbing kepada beberapa pelayar yang telah ditentukan sebelumnya.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE TITLE PAGE

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Introduction

The outcome of this project is to demonstrate on how simulation various network scenario can be designed and can effected on network performance. Besides, this project assessed the efficiency of the load balancing actuators and compared both the latency and the response time for the design topologies. The goal of this project, is to design a network scenario of load balancing using network controller software tools. At the end at this project, we can see that users can implement load balancing to their network using software based not just cheaper solution but also give a stratification network performance.

Therefore, the growth of the Internet and ICT (information-communication technology) developments, including internet, cloud, social networking, IoT (Internet of Things), multimedia and the trend towards a virtual world, the global demand for IP traffic has increased enormously in recent years. With an increasing order of internet traffic, the operation and installation of networking equipment has become complicated, difficult and time-consuming for service providers.

So that, every network need an implementation of load balancing. Load balancing refers to efficiently distributing incoming network traffic across a group of backend servers, also known as a server farm or server pool. Load balancer functions as a traffic policeman sitting in front of your servers and routing client requests across all servers capable of responding to those requests in a way that maximizes speed and capacity utilization and guarantees that no server is overworked, which could degrade efficiency. If a single server comes down, the load balancing device redirects traffic to the remaining online servers. When a fresh server is added to the server group, the load balancer will automatically send requests to the server group.

This project proposes an approach that implement a load balancer using network controller. Network controller is an emerging technology in the field of networking. Customers now use big volumes of data that the network generates a lot of traffic. It becomes difficult to manage the entire load for a single server. Using several servers is the answer for this. The load balancer will receive the applications. The client requests the information then forwarded to the servers depending on the load balancing strategy used. Additional use of load balancing is not required for dedicated hardware. In this document, we are applying and evaluating network performance using the Floodlight Controller, Mininet, and testing our outcomes using the Wireshark Network Analysis Tool.

1.2 Problem Statement (PS)

The biggest problem for that traditional load balancing is performance, ability and economy. The traditional load balancer is vendor implement on specific hardware

are very costly, inflexible and non-programmable to use, so administrators cannot create their own algorithm. So to overcome this problem, this project emulates scenario of load balancing with various network controller technologies. In today's network, every network need to handle huge number of clients are connected with the network and need to handle huge of request and has become a tedious job to a server. It is impossible for single server.

Table 1.1: Summary of Problem Statement

PS	Problem Statement
PS_1	The traditional load balancer are vendor implement on specific hardware are very costly, inflexible and non-programmable to use
PS_2	Current network need to handle huge number of client request and packet data transfer
PS_3	Difficult to configure a lot of network hardware in one time

1.3 Project Question (PQ)

Table 1.2: Summary of Problem Question

PS	PQ	Problem Statement
PS_1	PQ_1	What is the best way to overcome the costly and non-programmable load balancing hardware equipment?
PS_2	PQ_2	How to overcome the problem of huge number of client and packet data request and transfer?
PS_3	PQ_3	How to configure all the network hardware in one time?

1.4 Project Objective (PO)

Table 1.3: Summary of Project Objective

PS	PQ	PO	Project Objective
PS_1	PQ_1	PO_1	To emulate a network scenario of load balancing with emerging technologies like OpenFlow and using tools Mininet and Open Daylight Controller for cost reduction and flexible settings.
PS_2	PQ_2	PO_2	To implement dynamic load balancing using several technique for achieving better results and higher performance
PS_3	PQ_3	PO_3	To configure the Round Robin technique into the load balancer.

1.5 Project Scope

The scope for this study is to use a several network tools. Network controller tools technology is an approach to network management that enables dynamic, programmatically efficient network configuration in order to improve network performance and monitoring. to monitor performance network while using the load balancing mechanism.

1.6 Project Contribution (PC)

Table 1.4: Summary of Project Contribution

PS	PQ	PO		Project Objective
PS_1	PQ_1	PO_1	PC_1	A new implementation of network scenarios with and without load balancing for better network performance.
PS_2	PQ_2	PO_2	PC_2	Round Robin technique setting at network controller
PS_3	PQ_3	PO_3	PC_3	Load Balancing with Round Robin

1.7 Thesis Organization

Chapter 1 Introduction

This chapter discuss about introducing the readers to the general background of the project, the objective of the project, scope of the project, and the problem statement.

Chapter 2 Literature Review

This chapter consists of the all the study about the project by referring from the existing study which is thesis, paper, journal and etc.

Chapter 3 Project Methodology

In this chapter focussed on the all the material and data gathered for this study. It will show how all the data and information are collected from the new result or from the previous study. All the data will show in graphical such as graph and table.

Chapter 4 Design

This chapter will show how the data and information collected from the previous chapter will be detailed design to come out with the new result also with the metrical calculation.

Chapter 5 Implementation

This chapter show how data collected in this project will implement using a several setup and scenario and show how the hardware to be setup, software installation and configuration.

Chapter 6 Testing And Analysis

This chapter consists of the testing and analysis for this project. All the data and information are collected and tested in this chapter.

Chapter 7 Project Conclusion

This chapter will show the conclusion about the study and the recommendation for the future improvements.

1.8 Conclusion

Expected outcome for this project is this project can demonstrate on how simulation various network scenario can be designed and can effected on network performance. Besides, this project assessed the efficiency of the load balancing actuators and compared both the latency and the response time for the design topologies. The goal of this project, is to design a network scenario of load balancing using network controller software tools.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This section discussed on methods and technique for solving using load balancing. Furthermore, the purpose of this chapter is to research about difference performance of network with implementation of network balancing and how the implementation of network controller can affect the network performance in term of internet speed performance, packet lost and so on. This chapter also discussed about the method, techniques and parameters used in previous project.

2.2 Network Architecture

All the data from the previous paper and journal will be collected in this chapter and will be compared to become a good reference for this study henceforth will make the project become successful. The domain of this project is network performance and analysis. This project will analyse network performance with implementation of

network controller and capture every packet in and out. This project will create a several scenarios which create a basic simulation network using a software Openflow.

2.2.1 Issues in Network Architecture

One of the major issues in network architecture is delay in delivering an end-users response. Normally lengthening happens when the network device gets overwhelmed and causes a bottleneck (N Joshi,2019) situation in the network. In order to distribute the load among the servers and to prevent a bottleneck state of affairs, we need load balancing appliances. In the existing network load balancing machine, different types of load balancing algorithms are used, so it takes too much time to process and make the system inefficient to transfer a large amount of user traffic to multiple servers.

From the case study based on several past research paper, issues that have been founded related to domain problem is, traditional load balancer are non-programmable, complex in management, highly expensive in nature, and have tightly coupled control plane with data plane (N Joshi and D Gupta, 2019). Based on researcher the best technique to solved this problem are emerging technology network controller has been introduced which decouples the data plane and control plane and makes the network fully programmable (N Joshi and D Gupta, 2019). Therefore, every network application needs a hardware applied which means that hardware's are very expensive and vendor specific. Traditional hardware load balancer is based on manufactured and can be only use a specific algorithm that allow users to customised according to users' feasibility or requirements.

2.2.2 Terms

2.2.2.1 Network Simulation

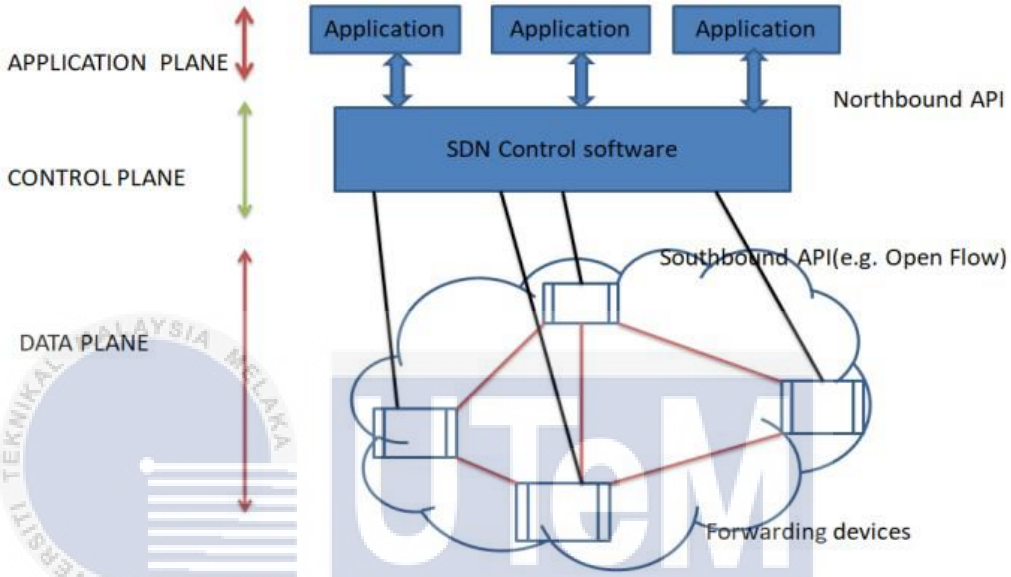
Therefore, Network simulation is a technique by which the software models the actions of the network by measuring the communication between the different network entities (routers, switches, nodes, access points, connections, etc.). Several simulators use discrete event visualization-system modelling in which state variables modify at discrete time points.

2.2.2.2 Software Defined Network

In implements network simulation, a software based network controller is applied. Software Defined Network is an evolving architecture that is flexible, scalable, cost-effective and adaptable, making it ideal for high-bandwidth, complex applications today. It architecture decouples network control and forwarding functions, allowing network control to become directly programmable and the underlying infrastructure to be decoupled for applications and network services. The

OpenFlow specification is a key element protocol for developing network controller solutions.

Figure 2.1 Software Defined Network Architecture (S Sarika, P Shiva, 2019)



2.2.2.3 Openflow Protocol

OpenFlow (OF) is considered to be one of the first network controller standards. It originally defined a communication protocol in network controller environments that allows network controllers to communicate directly with network devices such as switches and routers, both physical and virtual (hypervisor-based), so that they can better adapt to changing business requirements. We can initiate physical network and virtual devices using OpenFlow. OpenFlow is usually implemented

between the SDN controller and the OpenFlow enabled switches and uses flow tables to match traffic or flow through the network (S Mishra and M A Rahman 2017).

2.3 Critical Review Of Current Problem And Justification

2.3.1 Project Methodologies

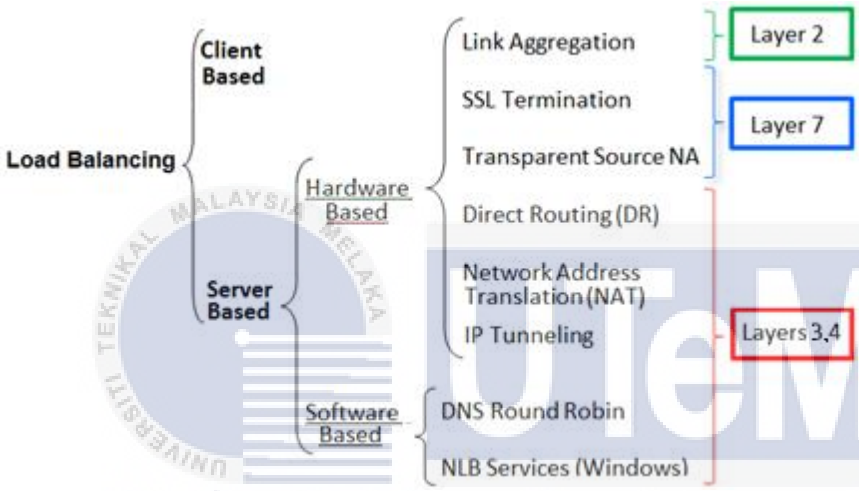
2.3.1.1 Software Defined Network

Software Defined Networking (SDN) is a way to approach computer networking through software abstractions instead of specific hardware. Through incorporating a few of the low-level features of the network to a software application, this helps network administrators to control complex networks more easily(P Sun, J Li, Z Guo, Y Xu, J Lan, 2019). SDN is a rapidly evolving networking technology that allows us to easily manage different network applications and services using the SDN architecture. One of main features of the network is load balancing so that we have implemented a load balancing mechanism over the SDN controller, and then the controller functions as a load balancing tool.

Load Balancing consist of a several type of mothed can be applied to the distributed system which is software based and hardware based. According to I

(Ivanisenko, 2015), Load balancing is represented on four levels of network model OSI: channel, network, transport, application.

Figure 2.2 Method of load balancing in OSI model (I Ivanisenko, 2015)



2.3.1.1.1 Software Based Load Balancing

Software based load balancing consists of a special software installed on the servers in the load balancing cluster. The software transmits requests from the user to the server due to different algorithms. For example, Software Defined Network (Network Controller) and Microsoft Network Load Balancing

2.3.1.1.2 Hardware Based Load Balancing

Load balancing hardware devices working on OSI layers 2-7 and used for splitting the network load among multiple servers in terms of factors such as utilization of processor CPU, number of connections, total server performance [Natario, 2011; Laviol, 2014].Hardware based load balancing consists of a special switch or router with programs that provide load balancing capabilities. This approach incorporates switching and load balancing into a single device, which reduces the amount of additional equipment required for load balancing. Current load balancing equipment is known as the network packet browser.

2.3.2 Load Balancing Techniques

2.3.2.1 Load Balancing Software Based

One of the technique can be implement to the network controller is load balancing. Load balancing technique can be applied without needed a load balancing hardware. Most of load balancing using a dedicated hardware which expensive and vendor specified which users can't program that hardware what their needed. One the

other hand special controller load balancing are programmable and allow users to design and implement own users load balancing strategy. Load balancing come with several coding technique such as Balance Flow, Hybrid Flow, Round Robin and etc.

2.3.2.1.1 Balance Flow

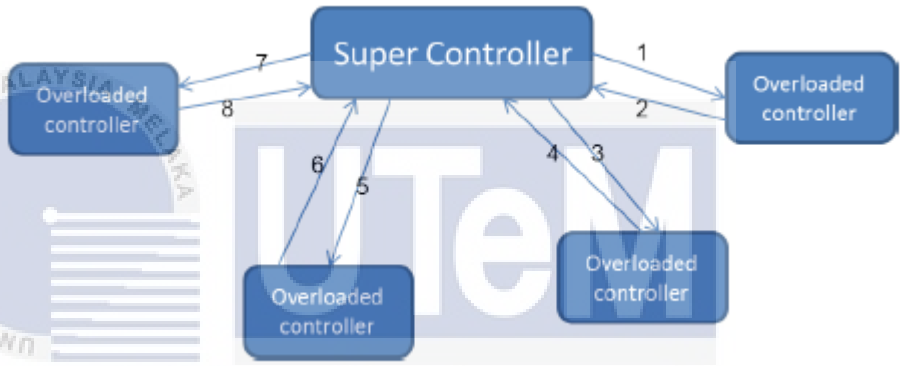
According to (H Sufiev and Y Haddad, 2016), This method called ”BalanceFlow” where the load balancing between the network controllers is performed by a SuperController (SC). "Balance flow” focuses on the load balancing of the system so that the flow-requests are dynamically distributed between the controllers in order to achieve rapid response. The load on the overloaded controller is switched dynamically to a suitable low-loaded controllers to enhance controller utilization. This technique requires each switch to allow a certain flow of service from some controllers. The accuracy of the algorithm is reached by dividing the switch load among some controllers by the source and destination of each current.

2.3.2.1.2 Hybrid Flow

(H Sufiev and Y Haddad, 2016) stated that the second technique that can be used in network controller are HybridFlow which consist in splitting the controllers

into clusters, such that in each cluster the controllers can help each other and perform load balancing within the same cluster. After all the controllers throughout the cluster are fully filled, a request will be sent to the SC to reduce the amount of switches to be operated in the cluster. This method of "local" load balancing helps to reduce the load on the SC without affecting the total load balancing.

Figure 2.3 : Load operation in HybridModel (H Sufiev and Y Haddad, 2016)



2.3.2.2 Network Virtualization

Another technique can be applied with SDN is Network Virtual. Network Virtualization (NV) refers to the transfer of network resources typically provided in hardware to software. NV can add multiple physical networks into one virtual software-based network, or split one physical network into different, independent virtual networks. Network Function Virtualization (NFV) decouples network

functions from the underlying hardware so that they run as software images on commercial off the shelf and purpose built hardware . (A YAZILIM, 2019). Consumer off-the-shelf or purpose-built hardware. This is achieved by using traditional virtualization technologies (networking, computation, and processing age) to virtualize network functions. The goal is to increasing the reliance on specific, specialized physical devices through the allocation and use of physical and virtual resources only when and where they are necessary.

2.3.2.3 Energy Efficient Networking

According to J Nyoupane, A Kumar, 2018 The shifting of focus of ICT towards energy-efficient and well-performed solutions (commonly known as green networking) in recent years has purposed numerous solutions. Most of these work can also be adapted in SDN concept. The goal is to increase the energy efficiency of the backbone network by dynamically changing the number of active links by network load. This incorporates the power of the origin routing, allows dynamic traffic engineering, and defines a forwarding route other than the usual shortest path that a specific packet is going through. The data plane used by SPRING follows the same principle of MPLS tag swapping, but its control plane has been completely redesigned.

2.3.3 Parameter of Load Balancing Based on Scenario

Table 2.1 : Comparison for parameter and scenario (N. Joshi and D. Gupta, 2019)

Parameters	Scenario (100 users)			Scenario (800 users)		
	Server load	Round Robin	Flow Stat	Server load	Round Robin	Flow Stat
Throughput	3 mbps	2.7 mbps	2.9 mbps	2.1 mbps	1.1 mbps	1.3 mbps
Response time	0.04 sec	0.05 sec	0.04 sec	3 sec	4.3 sec	2.2 sec
Transaction rate(trans per sec)	350 trans	280 trans	X	280 trans	180 trans	X

2.3.4 Project Software

2.3.4.1 Ubuntu

Operating system that will used to complete this project is Ubuntu. Ubuntu is an open source operating system (OS) based on the Debian GNU / Linux distribution. Ubuntu combines all the functionality of a Unix OS with an enhanced customizable GUI, making it popular with universities and research organizations. Ubuntu is primarily intended to be used on personal computers, although server versions are also available. Ubuntu consists of a variety of software programs operating under the GNU

General Public License. It allows users to copy change, create and redistribute their own version of the program.

2.3.4.2 Openflow

Many SDN Controller platform has come into existence in the past few years, like Beacon OpenFlow controller, NOX, POX,Nettle, OpenDayLight, FloodLight, Ryu (S Mishra and M A Rahman 2017). OpenFlow is a programmable network protocol designed to control and direct traffic between routers and switches from different vendors. Divides the software of the switches and routers from the underlying hardware. By using these platform researchers develop many applications such as load balancing, network virtualization, energy efficient networking, dynamic access control in enterprise network, Virtual machine mobility etc. Openflow technology is being considered one of the favourable technologies for isolation of control plane &data plane and logical placement of centralized control from SDN controller.

2.3.4.3 Mininet

Mininet is a network emulator which creates a network of virtual hosts, switches, controllers, and links. Mininet hosts run standard Linux network software,

and its switches support OpenFlow for highly flexible custom routing and Software-Defined Networking. Mininet supports research, development, learning, prototyping, testing, debugging, and any other tasks that could benefit from having a complete experimental network on a laptop or other PC.

2.3.5 Conclusion

Method for implementation load balancing can be done by two option which hardware based and software based. The software based load balancing platform using a software defined network technology. Software-defined networking (SDN) technology is an approach to network management that enables dynamic, programmatically efficient network configuration in order to improve network performance and monitoring making it more like cloud computing than traditional network management. Besides, with the implementation of load balancing can give an impact to the network performance. We can see form the table 2.1 the result from the different scenario done by the pass researcher.

2.4 Proposed Solution

Studies have shown that the best way to implement load balancing is using software based platform. Software defined network technology is the best way to implement load balancing technique due to software defined network is an open source software and its enables dynamic, programmatically efficient network configuration in order to improve network performance. According to (N. Joshi and D. Gupta, 2019), SDN controller, which is fully programmable and behaves like a Load Balancer after installing the load balancing algorithms over it and represents number of servers where load is distributed according to the load balancing algorithms. Next, the technique will be used for this project based from previous studied is Load Balancing. The proposed this project is to analyse network performance based on implementation of load balancing using network controller.

This project applied dynamic application load balancing method. This method works together with an external server load balancer, and calculates the weight parameter of round robin scheduling in the load balancer to distribute requests among nodes (K HIKICHI, T SOUMIYA, and A YAMADA, 2016). Lastly, the best parameter used based on previous journal is Round Robin. This parameter show the best result to be used together with load balancing technology. This method is defined as the requests are sent to each server that is present in the queue one by one in a circular manner. When a packet arrives, the next selected server is available on the list of all the servers present on the network system. So that all servers in the database are in the same order and perform the same number of loads, except for the load present on each server. (N. Joshi and D. Gupta, 2019).

2.5 Conclusion

As a conclusion, this chapter important to completed this project. A literature review is an overview of research on a given topic and answers to related research question. We can gather all important data and source from the previous paper. From this chapter we can know what is the best method, technique, attribute or parameters best for used to complete this project. The technique consists of identifying redundant computation both within single runs as well as across consecutive simulator.

The next chapter is methodology. This chapter focussed on the all the material and data gathered for this study. It will show how all the data and information are collected from the new result or from the previous study. All the data will show in graphical such as graph and table.

CHAPTER 3: METHODOLOGY

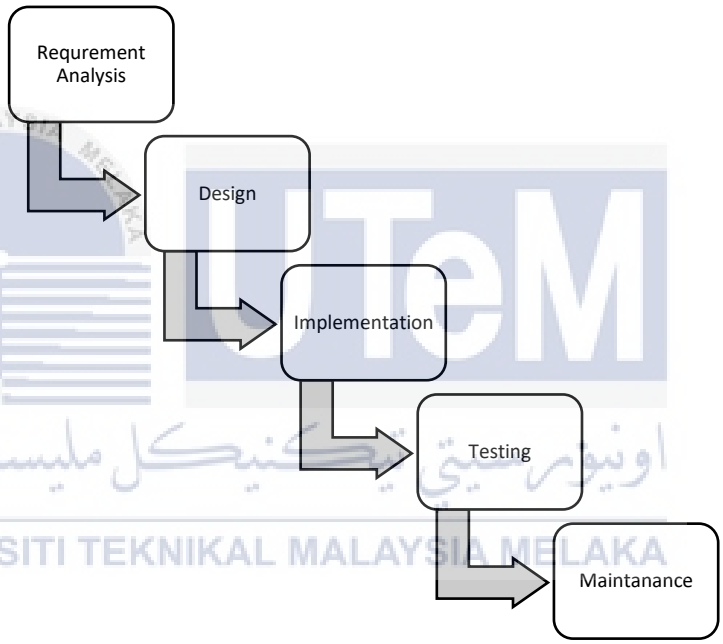
3.1 Introduction

The chapter will clarify the method used in this project. This chapter will list all components of data, population structure and sampling methods used in the interviews involved in carrying out this study. All the data gathered will be showed in step by step also showed every stages included in this chapter. Finally, this section gives a detailed description of the chosen method of research used and the process of data collection. Moreover, in this chapter also will show the Gantt chat for this project. The Gantt chart shows how the work is planned and whether the project is behind or ahead of schedule. The role of the Gantt chart is to guide the course of the project plan.

3.2 Methodology

There are several type of method can be used to developed this project such as Scrum, Kanban, Lean, Waterfall and Six Sigma. For this project waterfall method was selected to developed this project. It is also referred to as a linear-sequential life cycle model. Waterfall are knows as one of the more traditional project management methodologies. The Waterfall model was introduced in 1970 by Winston W. Royce. It's very easy to understand and use. In a waterfall design, each step must be completed until the next stage can begin and there's no overlapping throughout the phases. The Waterfall model is the first SDLC approach used to develop software. The following illustration is a representation of the different phases of the waterfall model.

Figure 3.1 : Waterfall model illustration (W W Royse 1970)



3.2.1 Requirement Software and Hardware Analysis

In this stage 1, all applicable requirements of the project to be developed are taken into this phase. All the data required for completed this project are gathered in this stage which for example. Operating system that being used in this project is Ubuntu. Software for create the network simulation is Mininet and Openflow. Also

several network performance analysis are used in this project. All the data from the previous studies about this software are collected and analyse for making this project complete. Beside all data are analysed so we can know what is the weaknesses and problem existing to the software and the solution to be as a main consideration data for this stage.

3.2.2 Design

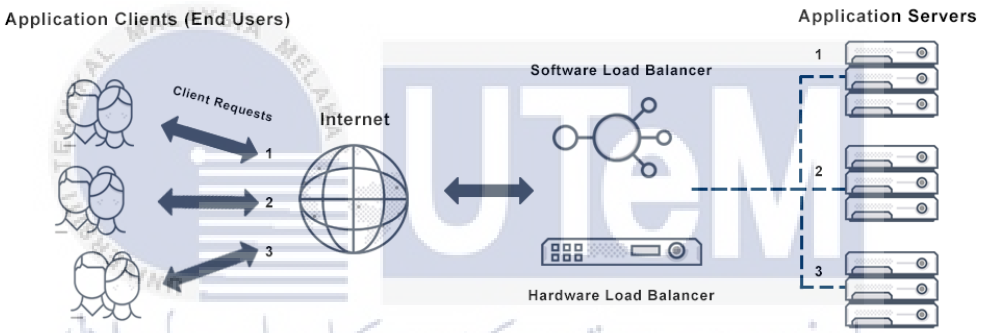
In this stage 2, the proposed of this stage is design all the project diagram. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture. The design that need to considered in this project is the designing network structure. The network structure designed using a simulation method using a Mininet software. Network will be designing with several scenario and with the implementation of load balancing strategies.

3.2.3 Implementation

In this stage, With the system design inputs, the system is first built in small programs called units, which are implemented in the next step. Each system shall be designed and tested for its functionality referred to as System Testing. After the

network has designed. It will be implemented with the network controller and load balancing technique. Data that collected from the chapter two was used to be implementation in this stage. After a several studies from the previous paper this project decided to using a Round Robin algorithm to be implement in load balancing technique. Using this method, client requests are routed to available servers on a cyclical basis. Round robin server load balancing works best when servers have roughly identical computing capabilities and storage capacity. The implementation will be testing in the testing stage.

Figure 3.2 Load balancing architecture (source: https://avinetworks.com/glossary/round-robin-load-balancing/)



3.2.4 Testing

In this stage 4, All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures. All the network scenario will be testing the performances and the results will be collected for documentation. The network testing,

the network controller testing, the load balancing testing and the parameter testing The network tested with several parameter which is throughput, transfer rate, delay and response time.

3.2.5 Maintenance

In this stage 5, while doing the testing, if testing is failed. Some maintenance will be doing to find the problem and do some fix to the error. Every coding to the network controller will be checked to find the error. After that the testing stage will be doing again with for a few times until get the better result.

3.3 Project Milestones

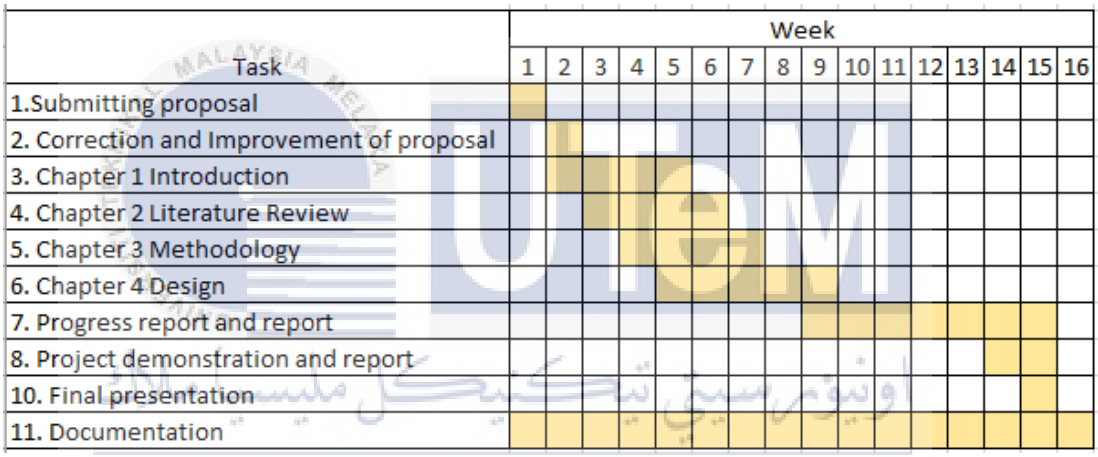
Project milestones has been assign to this project. A milestone is a particular period in time within the life cycle of the plan was using to assess the advancement of the project towards its ultimate goal. Project Milestones are being used as markers of: the start and end date of the project, the need for external approval or feedback, the need for budget limits, the submission of significant deadlines, and much more. Milestones have a fixed date but no length of time which is from the first week until the last week of presentation.

Table 3.1 Project Milestones PSM 1

Week	Activity	Output
W1 10 – 15 September	Decided for project title and proposal	Assign a supervisor Title was chosen Developed a proposal
W2 16 – 22 September	Submit proposal to supervisor to approval	Proposal submitted
W3 23 – 29 September	Proposal modification	Proposal approved
W4 30 – 6 October	Project begins	Chapter 1 Introduction
W5 7 – 13 October	Submit Chapter 1 to supervisor	Progress report
W6 15 – 20 October	Discussion with supervisor for Chapter 2	Chapter 2 started
W7 21 – 27 October	Started on studies for related article related with project	Chapter 2 Literature review
W8 4 – 10 November	Chapter 2 completed	Submitted Chapter 2 for supervisor evaluation
W9 11 – 17 November	Started for chapter 3	Chapter 3 Methodology
W10 18 – 24 November	Begin previous research for methodology	Submitted progress for chapter 3
W11 25 – 1 December	Network design for chapter 4	Chapter 4 Design
W12 2 – 8 December	Design for network environment for chapter 4 PSM 1 report preparation	Submitted chapter 4 to supervisor for evaluation Create slide presentation

		Demonstration for supervisor and submitted full report for PSM 1
W13 9 – 15 December	Final presentation PSM 1	Final evaluation from supervisor and evaluator

Figure 3.3 Gantt Chart



3.4 Conclusion

As a conclusion, the project methodology which involve all 5 phase which are Requirement Analysis, Design, Implementation, Testing and maintenance in the project a identified. This chapter very importance to this project to make sure this project follows the right method and to make this project process smoothly while develop this project. Also, Gantt charts and project milestones are also very important to be sure this project follow the schedule. The next chapter to be developed is design. This chapter will show how the overall design for this project.

CHAPTER 4 DESIGN

4.1 Introduction

In this chapter, it will describe all the network design for this project also defines the results of the analysis of the preliminary design and the result of the detailed design.. In this project there are consists with several network scenario to be analyses using a simulation software based. Then, it will discussed the detailed of each network scenario and also the parameter of each network design.

4.2 Network System Architecture

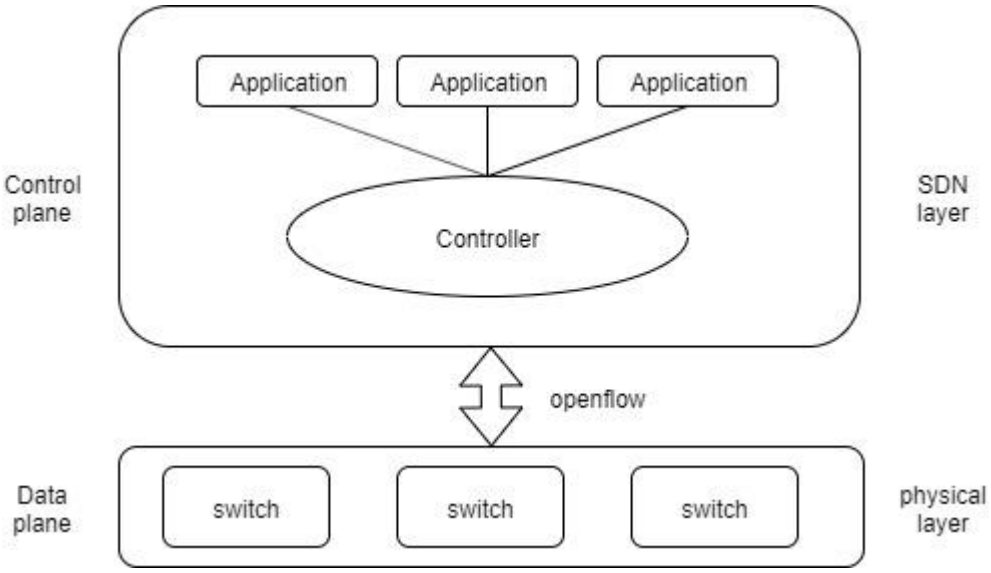
4.2.1 SDN Application

SDN applications are systems that interact behaviours and resources with the SDN controller via application programming interfaces (APIs). In addition, applications may build an abstracted view of the network by gathering data from the controller for decision-making purposes. These applications may include networking, analytics, or business applications used it to support large data centres.

4.2.2 SDN Controllers

SDN controller is a logical organization that obtains instructions or requirements from the SDN application layer and transmits them to the networking components. The controller also extracts network information from hardware and forwards an abstract view of the network to SDN applications, including statistics and events about what's going on.

Figure 4.1 : Network architecture for Software defined networking

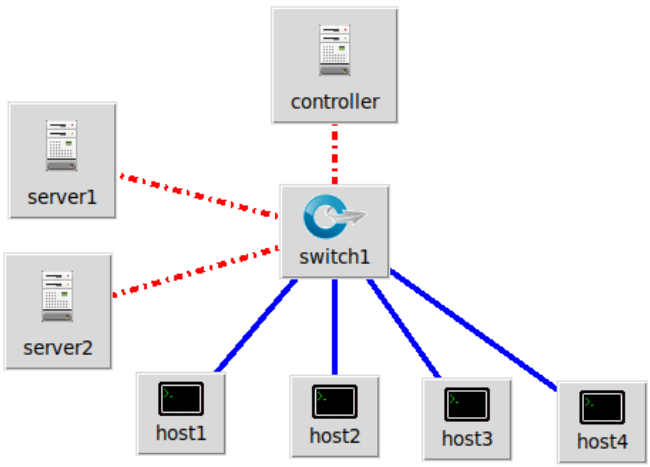


4.3 Possible Scenarios

This project consists with a several network scenario which is Network A, Network B and Network C. All this design will be design using simulation software which is Mininet and this software will be based for this project. This project will running fully using simulation environment. Every network design for this project consists of router, switches and different amount client connected which is important to be testing in this project.

4.3.1 Scenario A

Figure 4.2: Scenario A



The first design topology is simple design in which four are equal number of clients and two servers, here there are clients and two servers in the server pool. Figure

4.2 depicts the design topology one in which two servers acting as server pool connected to SDN switch. The communication between these servers happens in a systematic manner where the servers in the pool receive these requests in a Round Robin fashion.

4.3.2 Scenario B

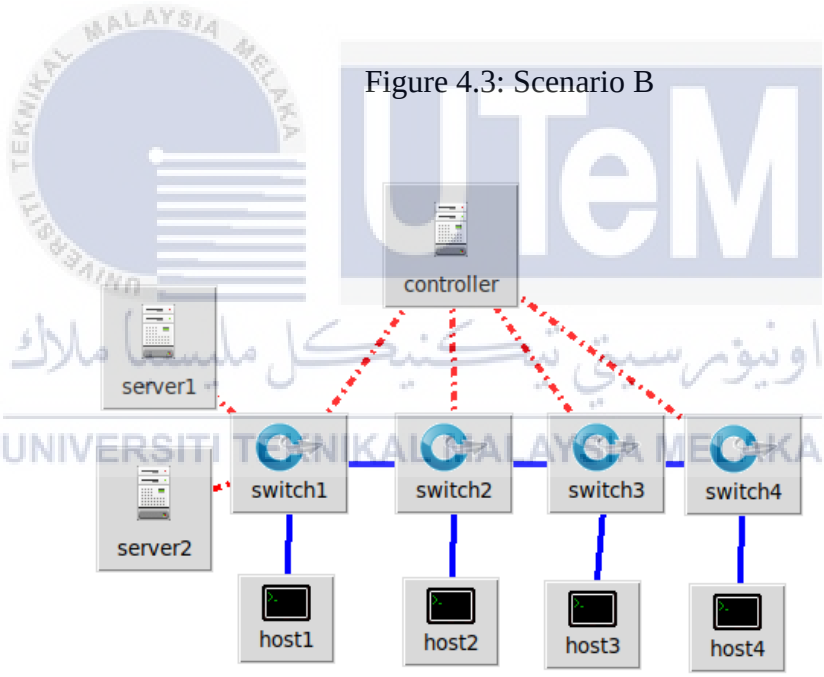
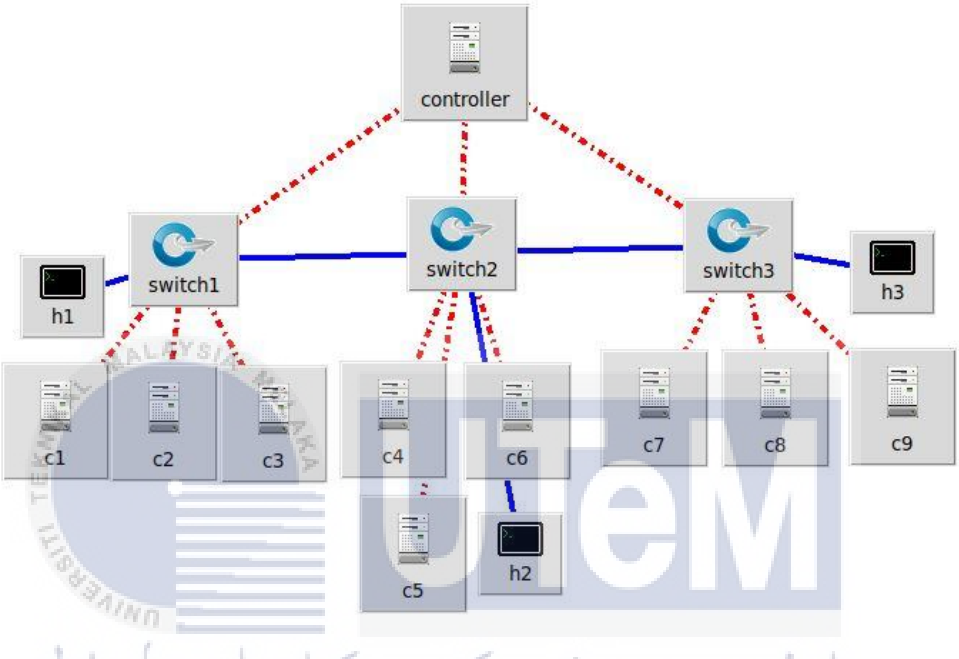


Figure 4.3: Scenario B

The second design topology is four clients. Figure 4.3 depicts design two, in which there is an inequality of servers and clients. There are four switch connected to the controller and two server pool.

4.3.3 Scenario C

Figure 4.4: Scenario C



The third design is an incredibly complex model in which there are twelve servers in the database pool, but there are only three clients. Figure 4.4 illustrates the complex design where there are only three switches connected to the server pool that support twelve clients. The design is similar to the previous two, but there is an increase in the number of clients. The traditional load balancers lag in these types of design topologies where the response time and latency will increase for the traditional load balancers.

4.4 Metric Measurement

For testing, we compare the load balancing algorithm with each other by the help of attributes like throughput, response time, and transaction rate. Mathematical calculation for the throughput can be calculated as:

$$throughput = \frac{number\ of\ bits}{seconds}$$

Response time is defined as total processing time of all users and is divided by the number of users. The response time is usually the total amount of time made by the request response process. It is also the amount of time that the servers process the request when the client receives the request. They measure the response time using the following calculation:

$$response\ time = \frac{total\ processing\ time}{total\ number\ of\ users}$$

The transaction rate can be measured as the amount of the http request-response pair is processed per unit time. Normally, the amount of information or the request-response pair is exchanged from the server in a given amount of time. Thus, the maximum transaction rate shows a faster and better response. It could be denoted as:

$$transaction\ rate = \frac{num\ request\ response\ pair}{second}$$

4.5 Conclusion

The main focus of this chapter is on the design of the network scenario for this project, which is an important step in the planning of the project. This chapter include all the important design requirement as part that need to be implementing and testing in the next chapter. It also explains the design of the architecture for this project that has been planned.

CHAPTER 5 IMPLEMENTATION

5.1 Introduction

This chapter discusses about the implementation project for the Analysis On Load Balancing Mechanisms For Better Network Performance Using Open Flow. In this phase, it focusses on how data collected in this project implements using a several setup. Network scenario setup, controller setup and network monitoring setup and show how the software installation and configuration management. Every detail on how the implementation and configuration are established in this chapter.

5.2 Environment Setup

In this project, network scenario that has been chooses are designed using simulation software which is Mininet. Mininet is an open source based software operated using Linux Operating System and can creates a realistic virtual network also

creates a switch and application code in a seconds and using a single command. Besides, the implementation of this project also uses the Openflow protocol which is controller protocol that control server where the sending packet need to transfer by tell the network switches where to send the packets. Network monitoring also involved in this project which is using a Wireshark software. Wireshark is used to analysis captured packet transfer. All the data capture gathered and analysed to get the network performance result which throughput, response time and transaction rate will be using in the next chapter.

5.2.1 Network Scenario Environment Setup

This topic discuss on how the method of network configuration for this project are implement. The step on how to implement are show step by step in this sub topic. For this project, it fully simulation based using Mininet software and illustrated in details.

5.2.1.1 Topology A Environment Setup

The configuration is done using the Mininet software. To create a network topology using Mininet software, it can be create using a single command. It has a several topologies type available in this software which is single topology, linear topology and tree topology (Idris Z B, Upena D. Dalal, 2016) also users can customs topology.

Refer to Figure 5.1, it shows that topology a created using a single topology. This topology contains six hosts and one switch. Two from six hosts will be change to the HTTP server pools connect to the single switch. It creates with a Openflow controller using port 6633.

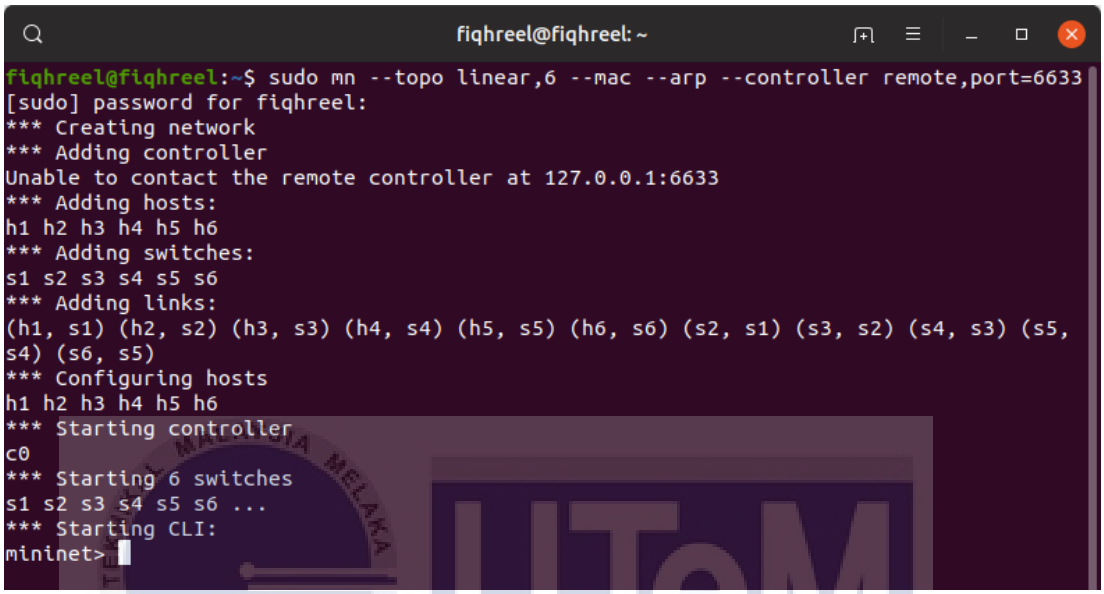
Figure 5.1: Topology A implementation

5.2.1.2 Topology B Environment Setup

Refer to figure 5.2, the topology b uses linear topology with six hosts connected to the six switches. This topology has Openflow controller connected and using port

6633. Form the figure 5.2 also can see every one host connected to one switch. Host one and host two will be configure as a HTTP server.

Figure 5.2: Topology B Implementation



5.2.1.3 Topology C Environment Setup

Refer to figure 5.3, for the topology c, it uses a customs topology. For the custom topology, it need to create using a python coding. after create a python coding it need to be open to Mininet using a single command. It shows in figure 5.3. this topology using a three connected switches plus a host and three servers for each switch.

Figure 5.3: Topology C Implementation

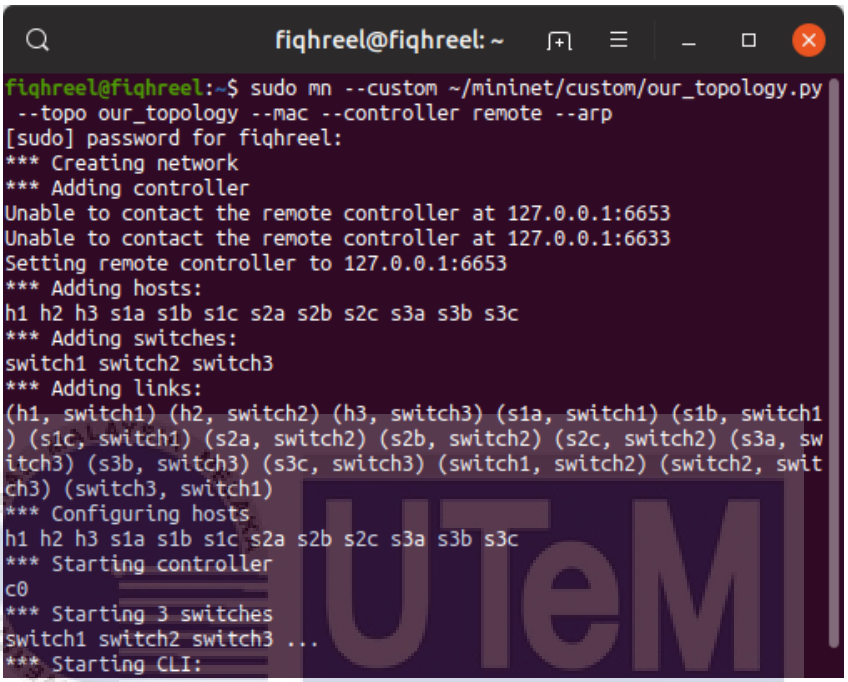


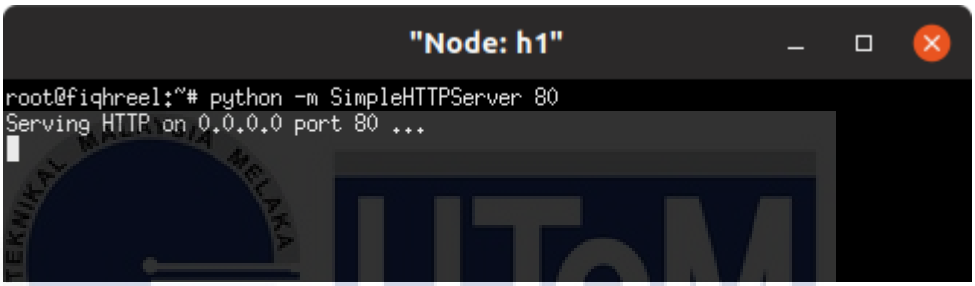
Figure 5.4 : Custom topology coding

```
from mininet.topo import Topo

class MyTopo( Topo ):

    def __init__( self ):

        # Initializing topology
        Topo.__init__( self )

        # Adding hosts, servers and switches
        host1 = self.addHost( 'h1' )
        host2 = self.addHost( 's1a' )
        host3 = self.addHost( 's1b' )
        host4 = self.addHost( 's1c' )
        host5 = self.addHost( 'h2' )
        host6 = self.addHost( 's2a' )
        host7 = self.addHost( 's2b' )
        host8 = self.addHost( 's2c' )
        host9 = self.addHost( 'h3' )
        host10 = self.addHost( 's3a' )
        host11 = self.addHost( 's3b' )
        host12 = self.addHost( 's3c' )
        switch1 = self.addSwitch( 'switch1' )
        switch2 = self.addSwitch( 'switch2' )
        switch3 = self.addSwitch( 'switch3' )
        linkopts = dict(bw=1000, delay='5ms',loss=50)
        linkopts2 = dict(bw=100, delay='5ms', loss=50)
```

5.2.1.4 Server Pool Setup

To configure HTTP server, it shows on figure 5.5 below.

Figure 5.5: HTTP Server Configuration

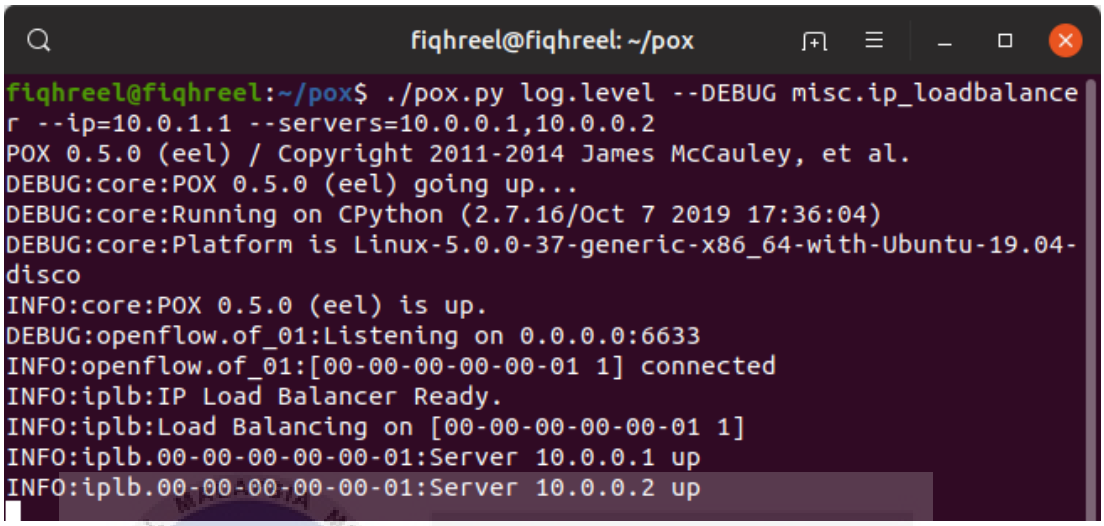


5.2.2 Controller Environment Setup

5.2.2.1 Load Balancing Controller Environment Setup

In this project, it uses Openflow controller connected to the switch. It has several type of controller can be used with Mininet software. For this project it uses pox controller. In figure 5.5 show that pox controller using a Load Balancer protocol. For the network it uses Ip address 10.0.1.1 and server use Ip address 10.0.0.1 for server one and 10.0.0.2 for server two.

Figure 5.6: Openflow Controller Configuration



5.2.3 Network Monitoring Environment Setup

To testing the network, this project will use Wireshark software to enable the simulation captured the parameter needed. The parameter will be results and based on throughput, response time and transaction rate.

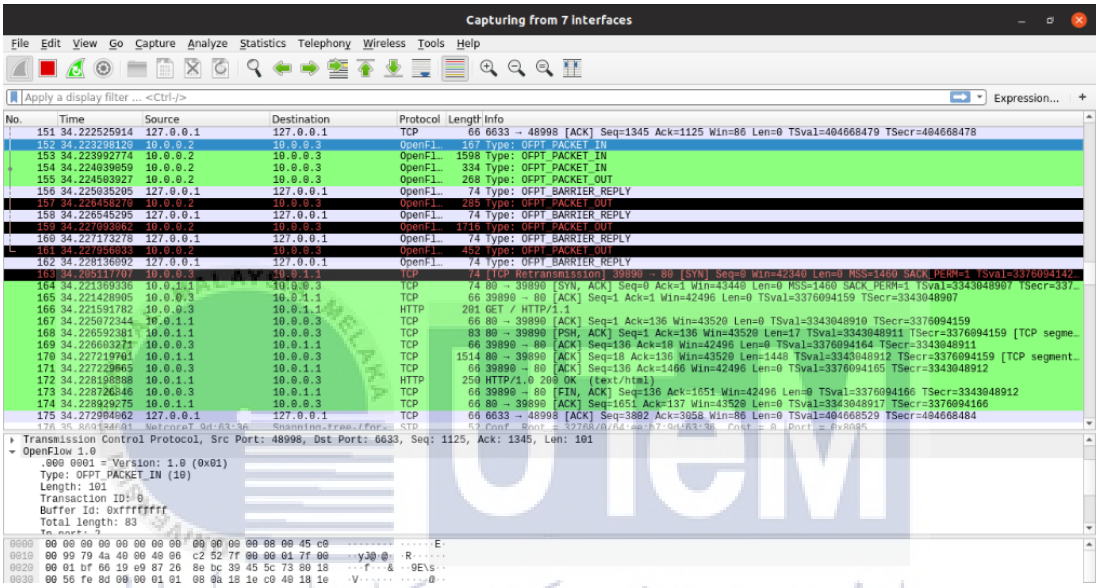
5.2.3.1 Wireshark Environment Setup

To captured packet transfer it need Wireshark to run in the background. The command use to run the Wireshark in the background is:

"$ sudo wireshark &"

To capture Openflow packet, "of" need to be fill in the filter box.

Figure 5.7: Wireshark packet transfer



5.3 Conclusion

As a conclusion, this chapter is discussed about how this project completely implement. It overall focussed on how the network and parameter that has been use, how the method to implement and further to be continued is Testing Phase. Detail explanation on how the data captured using Wireshark to be testing will be discussed on the next chapter.