NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY

By

FOO MUN YAO

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

COMUNNICATION AND NETWORKING

Faculty of Information and Communication Technology

(Kampar Campus)

JANUARY 2020

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: Network Topology Visualizer using server flow affinity

Academic Session: 202001

I FOO MUN YAO

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in

Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

	Verified by,
[illegible]	[illegible]
(Author's signature)	(Supervisor's signature)

Address:

1, Hala Bandar Baru Tambun 10

Bandar Baru Tambun

31400, Ipoh, Perak

Dr Aun YiChiet

Supervisor's name

Date: 22/04/2020

Date: 22/04/2020

NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY

By

FOO MUN YAO

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

COMUNNICATION AND NETWORKING

Faculty of Information and Communication Technology

(Kampar Campus)

JANUARY 2020

DECLARATION OF ORIGINALITY

I declare that this report entitled "**NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature	:	[illegible]
Name	:	FOO MUN YAO
Date	:	22/04/2020

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Dr Aun Yichiet and my moderator Dr Gan Ming Lee, for their guidance, encouragement and critiques of this project. I would also like to thank Dr. Dr Aun Yichiet, for his advice and assistance in keeping my progress on schedule. Many thanks as they are the one who gave me this chance to engage in an AI topology visualizer project. It is my first step at an AI related project.

I would also like to extend my thanks to the laboratory assistants of the FICT department for their help in offering me the resources in this project. Finally, I must say thanks to my parents and my family for their love, support and continuous encouragement throughout the course.

ABSTRACT

Automatic discovery of network topology is very important and has practical significance due to the fact that network is becoming more and more complex. Technology of computer network is changing and developing rapidly, the importance of scientific and effective network management is becoming increasingly significant. This project is a Network Topology Visualizer for academic purpose. The aim of this paper is improve upon methods and techniques to discover physical and logical topology, then ultimately, incorporating AI to visualize the topology. It will provide readers with the methodology, concept and design Network topology discovery. “Divide and conquer” method will be utilised to segment a physical network for the sake of easing the process of data collection. Then, with the help of machine learning technique known as artificial neural network will be used to help identifying the devices in a topology. The end result of the machine learning is that we are able to obtain an acceptable accuracy result of 75%, although, there is still a lot of room for improvement. The latter part of the project has led to the visualization of the topology data gained from the neural network. The visualizer tool in the project is coded 100% in python, where a JSON configuration file is used to generate the topology of the network to a PDF file. Another JSON files is created in order for the user to see in-depth information on the network. After the development of the visualizer, discussion and the results of the visualizer are made to compare with 2 other network topology visualizer researched online.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

LIST OF ABBREVIATIONS

3D	Three-dimensional
Ac	Aggregation and User
Ag-Ac	Aggregation and Access switch
ANN	Artificial neural network
Cs-Ag	Core and Aggregation
CSV	Comma-separated values
FYP	Final Year Project
GUI	Graphical user interface
ICMP	Internet Control Message Protocol
IDE	I*ntegrated development environment*
IDS	Intrusion Detection Systems
IP	Internet Protocol
LAN	Local Area Network
LLDP	*Link Layer Discovery Protocol* (*LLDP*)
MAC	Media Access Control
MIB	Management information base
ML	Machine Learning
OFDP	OpenFlow Discovery Protocol
SDLC	Software Development Life Cycle
SDN	Software Defined Networks
SNAMP	Sensor Network Analysis and Management Platform
SNMP	*Simple Network Management Protocol*
UTAR	Universiti Tunku Abdul Rahman
WAN	Wide Area Network
WIP	Work in progress
WSN	Wireless Sensor Networks
PDF	Portable document format
JSON	JavaScript Object Notation
IBM	International Business Machines
MATLAB	MATrix LABoratory
RFC	Request for Comments
YANG	Yet Another Next Generation

IETF	Internet Engineering Task Force
VLAN	Virtual Local Area Network
VID	Virtual Identification
LMYN	Lets Map Your Network
CSS	Cascading Style Sheets
DSL	Domain Specific Language
XML	Extensible Markup Language
HTML	Hypertext Markup Language
CMBD	Configuration management database
CI	Configuration It**e**ms

CHAPTER 1 – Introduction

1.1 Problem Statement and Motivation

As present computer network is becoming larger and extremely complex, and with technology of computer network is changing and developing rapidly, the importance of scientific and effective network management is becoming increasingly significant. However access to accurate network topology is the premise of network management. It is almost impossible to rely solely on artificial work to draw the topology if we are dealing with these large and complex topology. Therefore, automatic discovery of network topology is very important and has practical significance (Tang, 2016).

In logical topology we are mostly worried about the discovery of network layer devices and pin-pointing of logical relationships between them, such as the connectivity between routers or between router interface and the subnet. On the other side of the coin, in physical topology we are mostly focus about the discovery of data link layer and physical layer devices and the connectivity between them, such as the connectivity between switches, switch and bridge, and host and switch or bridge. As to the discovery of logical topology, we can get better topology by analyzing the routing table, using the Ping tool and etc. Moreover, with topology discovery, questions such as availability, performance and reducing the total cost of ownership can be answered with the IT-monitoring technology. It provides a quantitative starting point that facilitates improving IT performance.

Currently, there are much more advanced tools for logical topology discovery, such as OpenFlow, etc. However, it is not enough for network management to know only the logical topology (Duan, 2015). After all, we have to get the relationship of LAN devices through physical topology. But current methods and techniques of physical topology discovery are not so mature. What’s more there are some defects. For example, when SNMP is used for topology discovery, it will be able to tell whether a switch is a core or leaf switch. But in actual fact, it might not be used as a core or leaf switch in a network topology. Moreover, in SNMP, we can see host visibility but not aware of the traffic flow, which is what we are trying to aim answer in this project.

In conclusion, the main purpose of this research paper is to improve upon methods and techniques to discover physical and logical topology, then ultimately, incorporating AI to visualize the topology.

1.2 Project Scope

The purpose of this project is to develop a Topology Discovery and Topology Visualization method/technique which can reliably figure out the bottleneck and hidden security problems of a network effortlessly and quickly by discovering, monitoring and displaying it. This project is using a method known as flow analysis, where we attempt to discover a network topology just from analysing the traffic flow with tools such as WireShark. The WireShark software will be deployed using a PC running on Windows OS. This PC will then further be used for coding, researching, sniffing, and etc, for the rest of the project.

Next, methods of topology discovery and visualization of this project will be initially be tested in a controlled laboratory environment. Therefore, there will be a number of networking devices which is essential for the simulated topology. The networking devices, includes layer-2 switches, layer-3 switches, routers, hubs, modems, and etc. Some of these said devices could be obtained via UTAR FICT labs, but other non-networking devices would be bought if compulsory.

The technique for flow analysis which we have come up with will be the "Divide and Conquer" method. This method requires us to collect a large amount of dataset with the help of Wireshark at different parts of a 3-tier network. To begin collecting actual traffic, we build a physical 3-tier network with the acquired network devices from UTAR's FICT labs, with some configurations we are able to assign each switch according to its hierarchy, namely Core, Distribution and Access. Each layer has its own specific purpose, therefore what we implement into the access layer won't be implemented on the distribution and so on. Next, we begin collecting packets at specific point of the switch, in order to create a massive dataset of packet information at different locations. Then, with the help of machine learning we can then train the computer to identify the devices that are present in an unknown network the next time a similar set of network flow data is present.

Eventually, the project will build an AI Topology Visualizer application which will analyse the traffic flow data recovered from the flow analysis. The complied flow data will then be fed to the AI program which will be coded with python programming language.

1.3 Project Objectives

The objectives of this project are:

1. To develop a method of Topology Discovery just by using the Network Traffic Flow Data. Instead of creating Protocols like most known topology discovery method, we want to discover a network topology just from Traffic Flow Analysis.

2. To develop an AI Topology Visualizer that can be input with the complied Network Traffic Flow Data. Which will be then able to automatically generate a Network Topology just from the network traffic flow data.

3. Main objective is to present the information of the selected network with the state of nodes and connection between each other with an instinctive form in order to support the whole network.

1.4 Impact, significance and contribution

The deliverables of this project will greatly contribute to the research of method to topology discovery. As the method which we wish to develop is unique to the rest, it will be able to improve the operational and intelligence of the management of network equipment and services. Topology discovery is vital for monitoring network state, to ensure optimum network efficiency, and to identify bottlenecks and weakness to ensure best possible network efficiency. Through my research, there is currently no technique which used machine learning in order to discover and identify network devices in a topology.

Moreover, AI network topology visualization application which is based on this technology will be able to tremendously benefit network manager in realizing and manage the network more conveniently. This is because the Artificial intelligence can automatically determines the network topology without the need of manual input. Finally, the AI visualization technology to the network topology which we have developed can be presented directly to people so that the network management has better ocular and interaction.

1.5 Background information

Computer networking is a group of integrated computers configured to one another. Computer networks or data networks are a series of nodes connected via communication channels. The nodes transmit, receive and exchange data between endpoints. The data transmitted is usually in the form of video or voice. How do you form a network between two computers? Wired Ethernet cables or wireless radio waves connect these computers. This mixture of technologies relies on defined algorithms to transmit data between specified endpoints. The endpoints include tablets, mobile devices, computer, servers and etc.

You're probably thinking how important is networking? This mainstream technology has proven to be an efficient way to enhance proficient, flexible and streamlined communication while maximizing on productivity and resources. Computer networks is essential for users on the network to share the resources and comunicate. Can you imagine how our world is going to be like now without emails, online newspapers, blogs, chat and the other services offered by the internet?

Before going into the rest of the research paper, we much first have a core understanding of what is Network Topology. By definition, it refers to the arrangement of a telecommunication network and how many different nodes in that network are connected to one other and how they communicate. Network topologies can be either physical or logical, where physical topology is the placement of the various devices of a network, while logical topology demonstrates how data flows in and out of a network from one device to the next. (Beal, 2011)

With that said, how can we ensure that these communication networks are able to work at its outmost efficiency? Introducing, Topology discovery, the process of discovering and plotting network devices and links which is imperative for a network's efficiency. With the dawn of mobile computing and virtual infrastructure, networks today often change dynamically, and automatic topology discovery is essential for monitoring network state, to identify bottlenecks and failures, and to ensure optimum network efficiency.

In this project there will be elements of Machine Learning as well, basically, Machine learning is one of the application of artificial intelligence (AI) which enables computer systems the ability to automatically acquire and improve from experience without the need of being directly programmed. Machine learning focuses on the advancement of computer programs so that it has the ability to access data and use it learn for themselves. (Varone, 2018)

1.6 Project Organisation

The project organization is described as follows:

Chapter 2 will introduce what is topology discovery and topology visualization, what is active topology discovery, the different types of active topologies discovery is also discussed in this section. Then, we will discuss on what is passive topology discovery along with some previous work of passive topology discovery.

Chapter 3 will look into the research methodology of this project and introduce the methods and technologies involved for a clearer view of this project

Chapter 4 we will look present the preliminary work done and the discussion of the viability of results obtained in chapter 3

Chapter 5 we will look into the discussion on the result based on the topology visualizer and we will compare its performance with the other visualizers researched.

Chapter 6 will conclude the works and results of the project and we will highlight any future direction could be achieved.

CHAPTER 2- Literature Review

2.1 Introduction to Topology Discovery and Visualization

As of today, networks topology is becoming ever increasingly complex, the job of network admin becomes more and more challenging. The need of having a precise overview on the network and topologies, is imperative to being able to build, control and maintain any network. The problem occurs when the network and topologies can change to address changing conditions in the network state and service demand, this can become a huge challenge. Generally, there are two major techniques used today to discovery network topologies, namely, active and passive discovery, which we will further discuss in this literature review.

2.2 Active Topology discovery

In active network topology discovery, the method usually more or less the use the way based on protocols such as ICMP and SNMP. The active monitoring is utilized in order to obtain the information for constructing network topology, which means these method will actively send probes over a network to discover operating system, ports, and hosts. The advantages of active scanning is mainly due to its ability to constantly contacting all hosts on a network to regulate their current status (Webster & Lippmann, 2006). Unfortunately it also has many disadvantages, these include lengthy scan times for large networks, possible crashing services or hosts due to the increase in overhead, a lack of visibility into subnets that are secured by firewalls that block active scans, frequent tendency to trigger false alarms in intrusion detection systems, and the need to provide credentials on certain network hosts to ensure a complete host analysis. (Akande, et al., 2017)

2.3 Types of Active Topology Discovery

When it comes to active topology discovery, there are a plethora of ways that has been proposed in order to solve such problem, one of the many ways includes as such:

1) **Topology discovery based on Simple Network Management Protocol(SNMP)**

   SNMP is a protocol based on IP protocols, where network devices using SNMP can extract the Management Information Base (MIB) (Pandey, Choi, Won & Won-Ki Hong, 2010) information which supports the network management. Lately, most of devices used as gateway supports the SNMP agent, the network topology information is mostly contained in the MIB, when accessing the topology information in the MIB, we can examine the network topology connection. The method based entirely on standard SNMP, the discovery process of the algorithm is simple; the system and network has small overhead and the algorithm is very efficient.

   But upon closer inspection, we are able to find out the limitation of this method: a)We are unable to discover the network device which does not support SNMP or the device that does not include SNMP agent; b) Large amount of redundant information will be present in the routing table; c) A large routing table often lead to multiple IP address, and the routing table based on a unique IP address will cause a number of issues, which includes multicast routing problem, it is not effective to replicate the network topology connection state. Therefore, this technique is effective for backbone network topology discovery, by replicating the overall sate of the network.

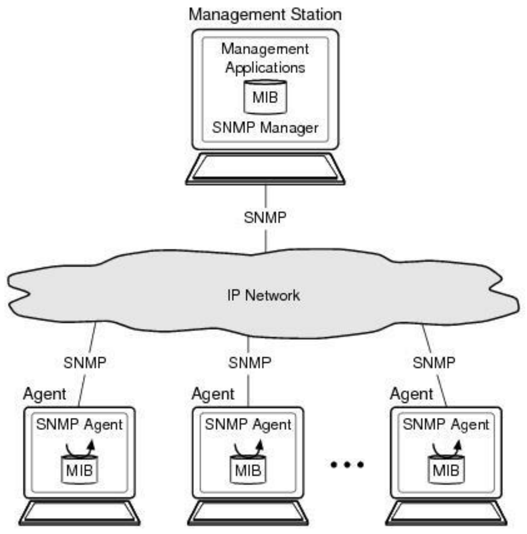


Figure 2.1 Shows MIB accessed with SNMP agent

2) Topology discovery based on OpenFlow Discovery Protocol (OFDP)

In Software-Defined Networking (SDN), to discover the network topology, all current OpenFlow controllers implement the same protocol OFDP (OpenFlow Discovery Protocol). (Azzouni , 2017)

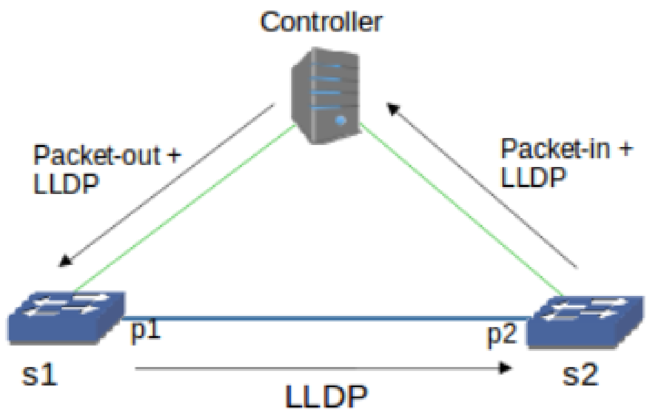


Figure 2.2 Discovering a unidirectional link in OFDP

To discover the mono-directional link from S1 to S2, the controller will be encapsulated in a Link Layer Discovery Protocol (LLDP) packet in a Packet-out message and sends it to S1. The outwards packet-in for S1 will include instruction to send out LLDP packet to S2 via port P1. After P2 received the LLDP packets, S2 will then encapsulate a Packet-in message and tansfers it back to the controller. Controller that receives the LLDP packet and determines that there happen to be a mono-directional link from S1 to S2. Recurring process is done to discover the opposite direction S2 to S1 as well as the other links in the network. Remember that, OFDP packets are sent to the ”normal” multicast MAC (01:23:00:00:00:01) to prevent being affected by 802.1d compliant switches. One of the fundamental advantages of OFDP is that it must offer is an accurate, near real time visibility of the network topology.

But, the fact is OFDP is a non-optimized or buggy topology discovery mechanism can affect routing logic and drastically reduce network performance.(Azzouni , 2017). Limitation of this method includes :a) OFDP is not secure as default OFDP using clear and unauthenticated LLDP packets; b) OFDP is not efficient because the controller sporadically forwards multiple packets to every switch in the network, resulting in decrease in performance.; c) OFDP is not scalable because discovery packets might get dropped or delayed in a multi-controller SDN network.

3) **Topology discovery based on transmission of customized Ethernet frames among installed software agents.**

In this method which is proposed by Nowicki does not require use of administrative access to layer-2 switching devices in order to discover the network topology. Instead it requires the installation of daemons (Master or Slave) with unrestricted access to network interfaces on selected end nodes and injecting layer-2 Ethernet packets which is used to communicate among agents and to detect which packets are passed by switches. It is not necessary to have end nodes connected directly to every switch except switches lowest in hierarchy. The proposed method has three phases. Firstly, the master daemon discovers all slave daemons, then it coordinates interaction among daemons to detect switches to which the daemons are connected followed by detection of the upper level of switches with no directly connected daemons. Two custom MAC addresses are used in addition to the actual MAC addresses of the end points that run the daemons: unicast address 1 (UA1, in this example: 00:00:5E:00:52:02) and unicast address 2 (UA2 , 00:00:5E:00:52:03). The advantages proposed by this method, is that it is capable of discovering hierarchical network topology without installing an agent connected directly to each switch (Unlike SNMP and MIB). Which makes it very accessible and easy to setup, due to the fact that not all devices may be capable of cooperating using simple network management protocol (SNMP) and share their management information base (MIB) structure and contents (Stallings, 1999).

However, this method is unable to achieve what it claims to do without making a number of assumptions: a) assumes that no layer-2 packet filtering is employed (no Media Access Control (MAC) address whitelisting); b) assumed that no multiport repeaters (hubs) are used which should be valid for every modern LAN. This makes the technique less accurate and reliable when put into more practical situations.

4) Topology Visualization based on Simple Network Management Protocol (SNMP)

Xiangyu Li came up with a SNMP-based technique of network topology visualization system, which can assist a network administrator to manage and comprehend the network more conveniently. This network topology visualization technology is divided into topology discovery and visualization. The two steps of the study are as follows, firstly, network topology discovery algorithm based on SNMP protocol is suggested to recognise the network topology discovery automatically. Then, the Transit-Stub model is analysed based on which the network topology is drawn.

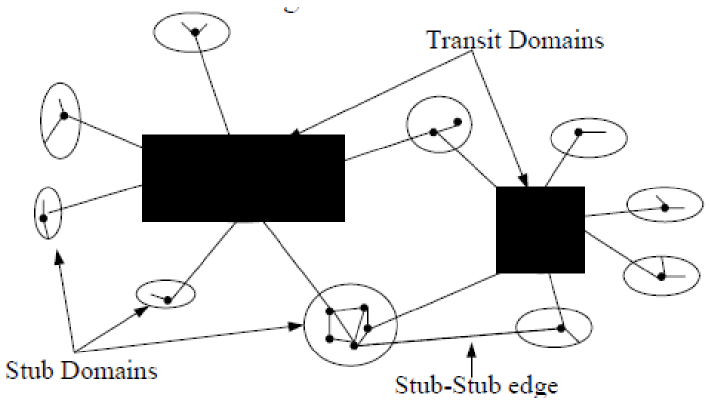


Figure 2.3 Shows a Transit Stud model generated

Flash technology is then used to solve the problem of the Flash ActionScript and backend database interaction and the real-time. Flash technology is known to have huge advantages on graphic processing and portability, moreover its Action Script programming language is very suitable for dynamic graphical development. (Li , 2011)

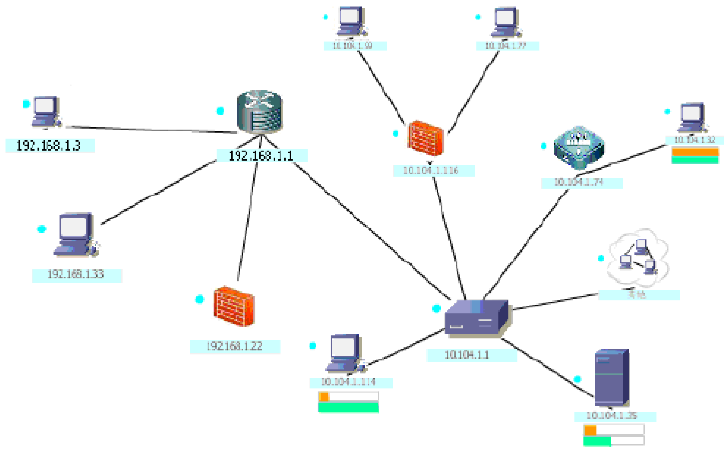


Figure 2.4 is a multi-level network topology generated by Flash

What Xiangyu Li is able to successfully achieve is to present the information of the Targeted network status of nodes and connection with an intuitive form in order to support the whole network. Unfortunately, topology visualization system proposed in this paper take certain advantages when compared with similar systems, there are some areas which needs polishing. These areas includes: a) Improvements on the flexibility of network topology discovery. It need a kind of algorithm which can quickly and accurately detect the network topology of a completely unknown structural network; b) Incorporating better integration of Visualization technology. Web technology development and graphics processing technology development, Adobe AIR technology and the 3D technology is more widely used. If successfully used into network topology visualization design and implementation, Visualization effect will be in a whole new perspective; c) To include more modern network management functions, in order to truly utilize the value of the this technology.

5. **Topology Visualization based on Sensor Network Analysis and Management Platform (SNAMP)**

   An ingenious method was discovered by Yu yang et al. to visualize the topology of Wireless Sensor Networks (WSNs). Due to resource constraining features and the apparent lack of user-network interfaces present in WSN, it suffers from limited visibility into the applications. Therefore, Yu yang et al seek to solve such issue with the self-developed Sensor Network Analysis and Management Platform (SNAMP), a novel multi-sniffer and multi-view visualization platform for WSNs. With the use of SNAMP, data released by each sensor nodes is collected by a multi-sniffer data collection network and forwarded to a multiple view visualization mechanism.

Figure 2.5 GAINS series nodes which is used for emitting data

SNAMP will be able to specify network topology, abnormalities, hardware resource depletion, network performance, and other sensing data in WSNs, which then helps developers by adding application specific visualization functions, which will simplify the research and development of various sensor networks and shorten the time from laboratory to applications.

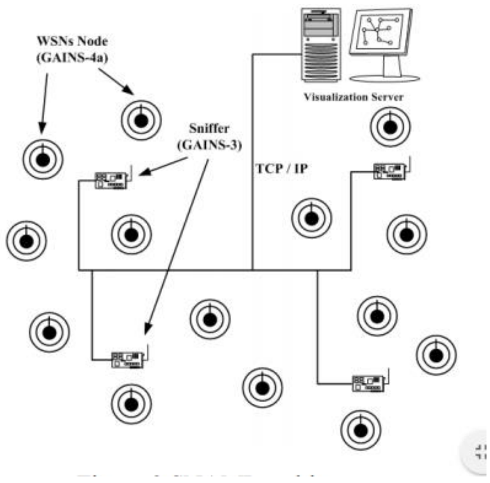


Figure 2.6 SNAMP archtecture

Without a doubt, SNAMP substantially enhances the visibility into WSNs applications. It is capable of avoid overload of the wireless transmission medium with visualization data and does not interfere in network activities. Another strength of SNAMP is beneficial for pre-deployment debugging of WSNs applications and provides a flexible architecture for adding new functions easily. (Yang, 2006)

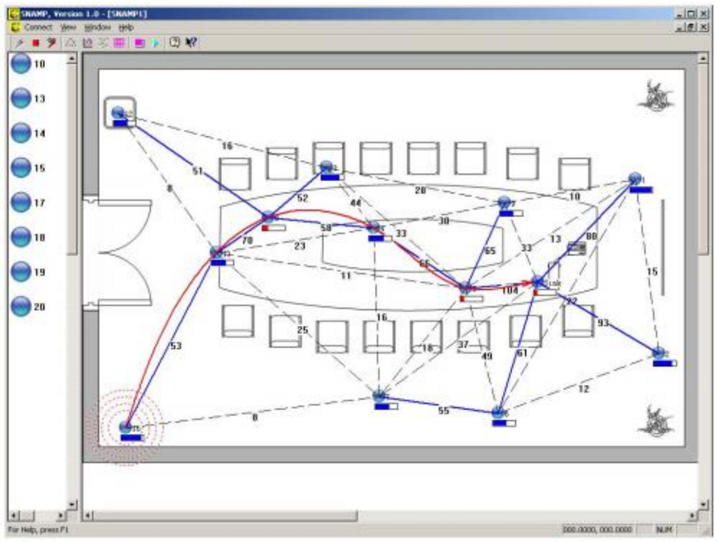


Figure 2.7 Visualization framework and topology view

Of course, SNAMP is a method with flaws as well, a) disadvantage of being passive and lack flexibility in configuration in an event of fault occurrence; b) Not a portable visualization platform for large scale WSNs, which makes it less practical in outdoor situations.

2.4 Passive Topology discovery

Unlike active discovery which we have discussed previously, passive topology discovery involves the observation of network traffic that is already operational on the network without any form of traffic modification or interruption of the network's performance. Comparatively, existing active monitoring techniques such as SNMP, ICMP and Network mapper (Nmap) can cause adverse effects to the infrastructure equipment in a time critical network due to the extra overhead on the monitored network paths. Passive monitoring utilizes tools such as mirror or span ports and network taps to unobtrusively capture network traffic. But of course, it does have its weakness as well, these weakness includes the failure to notice nodes whose traffic bypassess the scanner, the inability to pinpoint the endpoint of broadcast messages, and the incapability to identify the source and destination of messages when the addresses are altered when in transit.(Akande, et al., 2017)

2.5 Types of Passive Topology discovery

When it comes to passive topology discovery, there are limited of ways that has been proposed in order to solve such problem as it is a relatively new field of research, one of those ways includes as such:

1. **Python Passive Network Mapping (P2NMAP)**
   P2NMAP is an open source technique developed by Chet Hosmer to map networks by utilizing the python programming language without the need of sending a packet onto the network. PNAMP works to identifying unusual behaviour in a monitored network. According to Hosmer, information of every node with an IP address on an existing network can be identified depending on how long the network is monitored, the longer the more accurate. More information, such as how long the devices the network is connected is part of its capability as well. More details on a network can be deduced using Hosmer's technique, there details includes where and what devices have communicated, and the time where the communication is established. The python script is 2 basic parts, the main program begins by setting up a network interface in promiscuous mode, and then the script will open up a raw socket which will then listens and read packets. PacketExtractor( ) function is called to decode the packets collected previously. After the decoding is completed it will update a list with packets which meets certain port criteria. (Hosmer and Kessler, 2015)

Based on my research, the advantages of this technique proposed by Chet Hosmer includes, the fact that it has zero overhead or impact when deployed on a network, with long enough time, it has the ability to uncover hosts and services that are typically unknown or are overlooked by active scanning methods. The disadvantages outlined in this project, is that a complete map of the network will take a longer time to compile. Due to its passive nature, this technique has difficulties in identifying details such as specific operating systems, hardware types and vulnerabilities.

CHAPTER 3: Proposed Method/Approach

3.1 Design Specification

After much deliberation on what Software Development Life Cycle (SDLC) we should be using for this project, we decide to settle on, the Kanban Model. The Kanban Model is based on 3 basic principles: 1) **Visualize the workflow**: The visual representation of development process helps determine the current context of your tasks. It's very useful when you work on a big project with multiple task. 2) **Limit the amount of work in progress (WIP)**: This aids by balancing the flow-based method so you don't start and put too much work in one task, you will get a capability to control the available resources and avoid the possible idleness. 3) **Continuous measurement and improvement of the life cycle**: The possibility to make changes during the working process is a distinctive feature of the Agile methodology. Finally, everything is visualised in a Kanban Board based on these 3 principles.

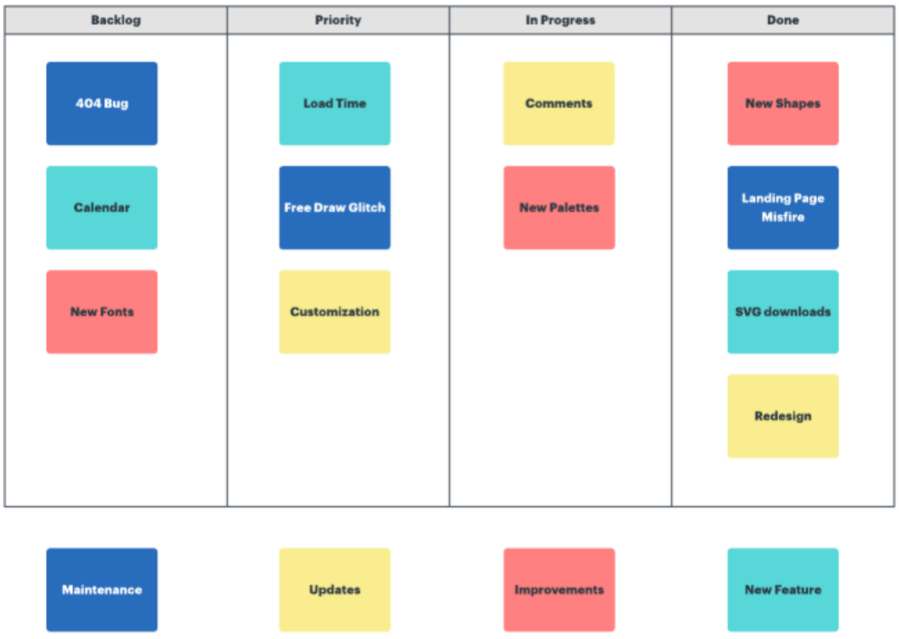


Figure 3.1 Shows example of a Kanban board with Kanban Cards

Basically, a Kanban board consists of clearly labelled columns that denote work yet to be done, work in progress, and finished work. Generally, it is up to you to decide how you want the column to be labelled to fit your progress. Every in-progress columns must have a set limit to the number of items that can be in a column at any given time. Work items will move through the Kanban board's columns as they move through the work process.

3.1.1 Backlog

Naturally, when the project just started, every task that had to be done will be placed in the backlog, but how do we place a task here if we don't have any? Therefore it is our job to identify and research what needs to be done before placing a Kanban card into this column. For example, in this project, developing a method for flow analysis is mandatory, we will then label this task under "Flow Analysis Method". The problem statement of this task is formed in order to find out the challenges of this task. After that, there will be a discussion done with the project supervisor to outline problem statement that I could have failed to identify. Once everything is completed, the first Kanban Card will be generated, and placed on the Kanban board.

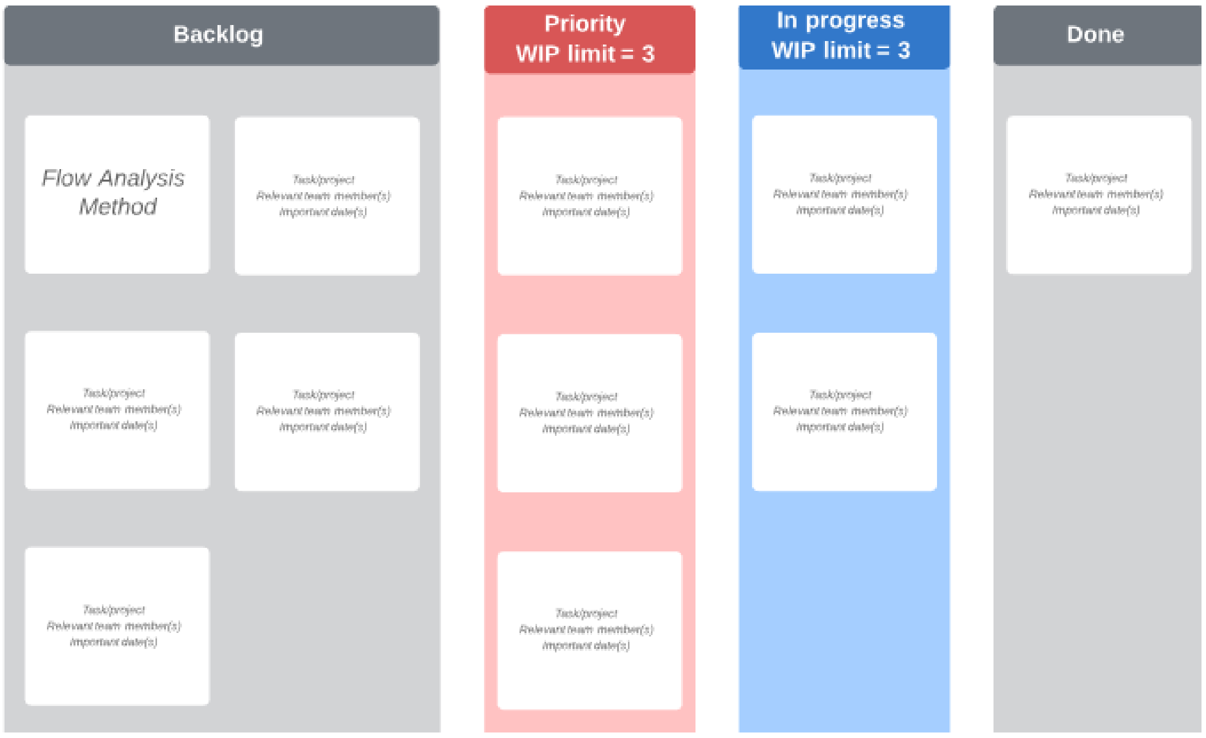


Figure 3.2 shows the Kanban Card being placed on the Backlog Column

3.1.2 Priority

In the second column, the “priority”, is when task from the first section have been analysed and assessed based on the importance of the task. Tasks that is deemed critical to the project, they are moved to the Estimated section. For example, during the project, we discovered a bug or we plan to design a GUI for the application, more task will be added to the “backlog”, but all these task are further evaluated and we determined that “Flow analysis method” is critical for this project. Therefore it is placed in the “priority” column.

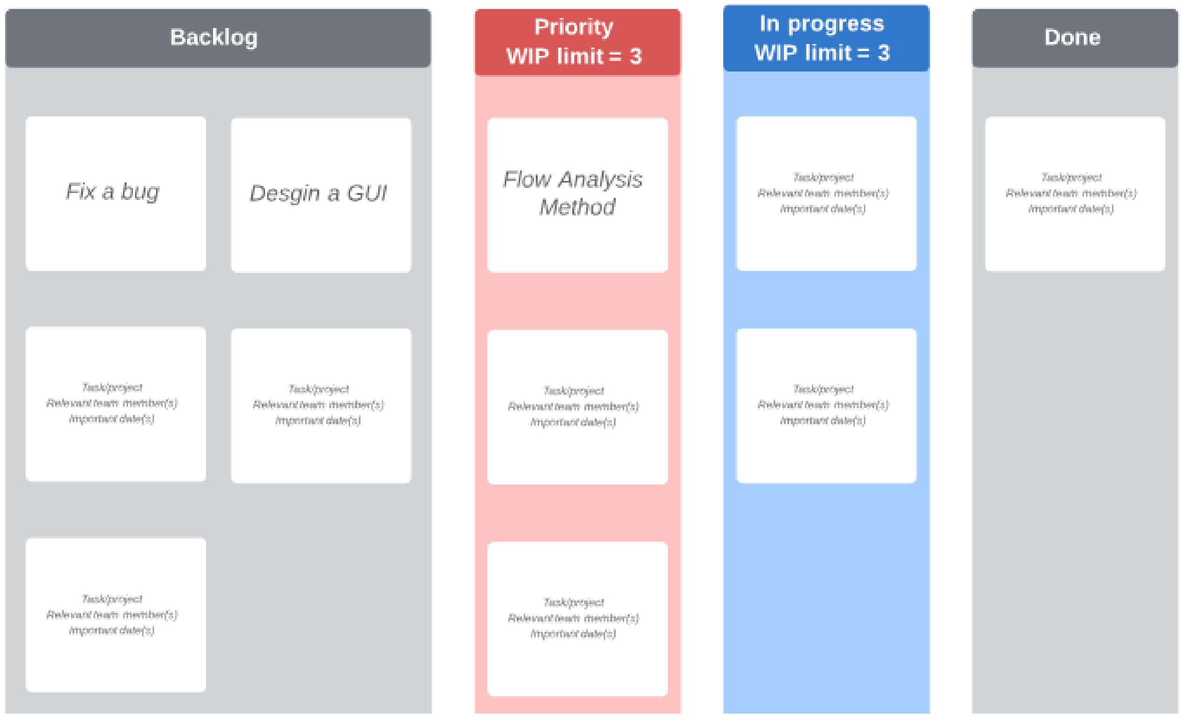


Figure 3.3 shows the Kanban Card being placed on the Priority Column

The limiting WIP principle is used. For every phase which a development process consists of, a limit should be set to the highest number of tasks that can be developed at the same time. Consequently, you are able to define the maximum allowable number of Kanban cards for every area of the Kanban board. By doing so, it give you a possibility to finish the proper task faster and increase the quality of work. Plus, such method will reduce the number of errors, which means that you won’t have to waste your time to fix them.

3.1.3 In progress

When we have decided to take a task which needs to be developed, the task is moved from the “Priority” column to the “In Progress” column where work begins for the task. Part of the “In Progress”, also includes testing, namely alpha and beta testing. Where in alpha testing is sort of a acceptance testing, where it is stress tested in order to identify all possible bugs/errors before the tasks is moved to “Done” column. Alpha testing is mostly conducted internally in a controlled environment, preferably in the lab. On the other hand, beta testing is where it is actually tested by “real users” in a real and more practical environment. Beta testing is also known as a form of external testing, where real feedback is given to the developers

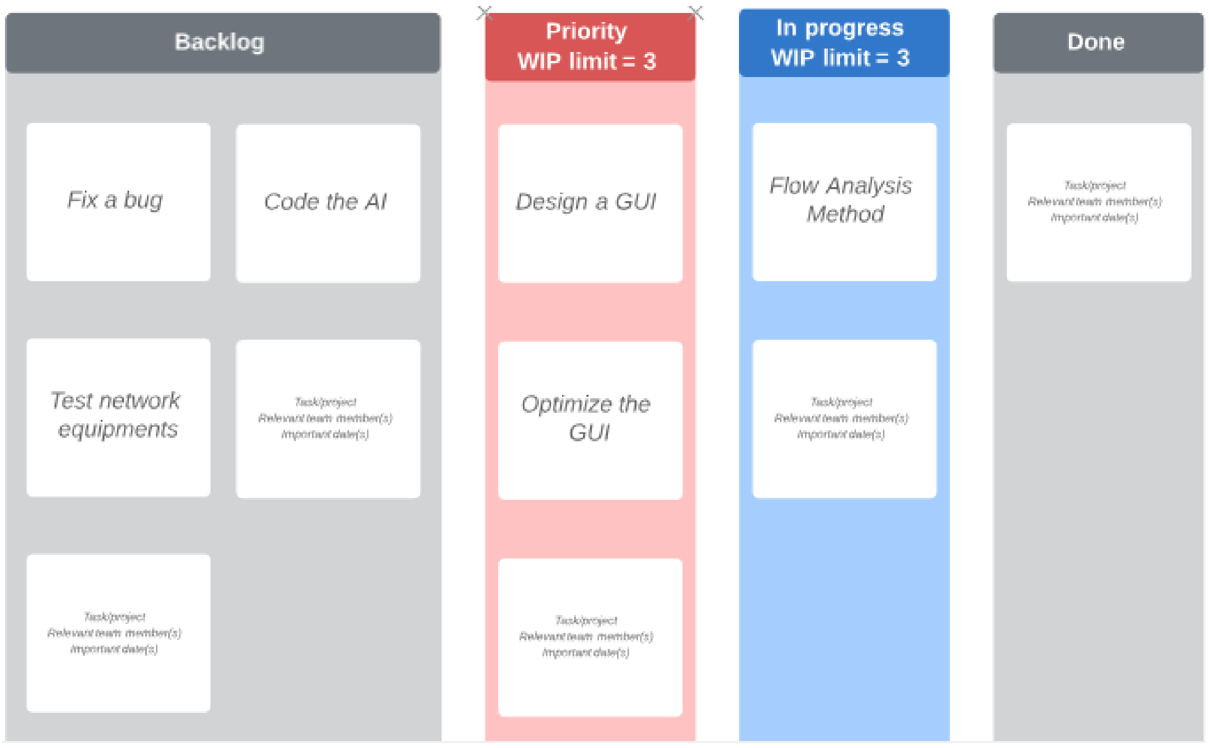


Figure 3.4 shows the Kanban Card being placed on the In Progress Column

3.1.4 Done

Finally, once a task is repetitively tested and ensured completed it will be moved to the final column, the “Done” column. In this section, maintenance will be done is necessary.

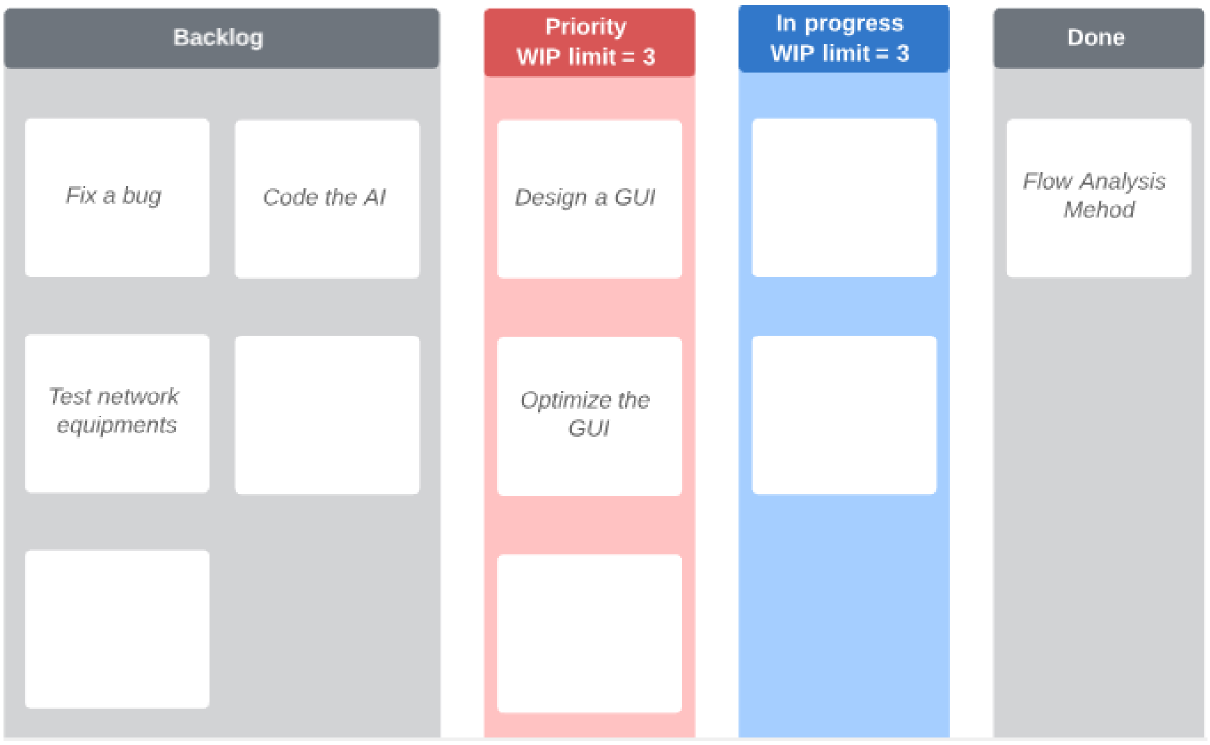


Figure 3.5 shows the Kanban Card being placed on the In Progress Column

3.1.5 Software Involved

These are the software that were used throughout the development of this project:

1. **Wireshark**

   Wireshark is the one of the world's leading and widely-used network open-source protocol analyser. It allows users to observe what is happening a network at a miniscule level. Wireshark is also the current standard across many commercial and non-profit enterprises, educational institutions, and government agencies. It is primarily used for network analysis, troubleshooting, software and communications protocol development. Clearly, Wireshark will be a fundamental tool for us when analysing, capturing and developing the protocols necessary for the topology discovery.

2. **Anaconda**

   Anaconda is a free to use and open-source distribution of the Python languages for scientific computing specifically in data science, large-scale data processing, predictive analytics, and machine learning applications. The main purpose of Anaconda is to streamline package management and distribution. Due to its easy-to-use nature, Launching python IDEs as well as Jupyter notebooks from anaconda navigator is a breeze, and Anaconda makes it very easy to keep these packages up-to-date. In this project anaconda is compulsory to function as a platform for Machine Learning.

3. **SolarWinds WAN Killer**

   WAN Killer Network Traffic Generator is a networking tool which will allow you effortlessly set the IP address and hostname you want to send the random traffic to. It has the features of specify parameters such as port numbers, packet size, and percentage of bandwidth to use. Its main use is for jobs such as load balancing and testing traffic prioritization. But in our case, the random traffic is generated, so that we can use the data collected to machine learn. Our project initially only tested in a controlled environment where the network is built in the lab, therefore artificial packets needs to be generated to simulate a more realistic environment.

3.1.6 Hardware involved

These are the basic hardware that were used throughout the development of this project:

1. **Layer 2 Switch**
   A type of network switch or device that operates on the data link layer and make use of MAC Address to determine the path where the frames are to be forwarded. It utilize hardware based switching methods to link and transfer data in a local area network (LAN).

2. **Layer 3 Switch**
   A type of network switch that merges the functionality of a switch and a router. It works like a switch, which are able to connect devices that are on the same subnet or virtual LAN at fast speeds. Moreover, it has IP routing capabilities built into it to function as a router.

3. **Router**
   A router is a Layer 3 network gateway device, meaning that it connects two or more networks together. Routers works by deciding whether the source and destination address are connected on the same network or whether which data must be transported from one network to another network.

3.2 System Design / Overview

In this section, we will look at details of how the project is designed and the process of collecting data, labelling data and machine learning with ANN.

3.2.1 Network Design

As stated before, the project will be first conducted in a controlled environment, where a physical network will be built in the lab environment, namely UTAR's FYP lab. The purpose of building this simple network is so that outside factor (connection from the internet) will not be a factor that affects the final result of this project. Therefore, we will first begin by visualizing what we are trying to build with a simple network in Cisco packet tracer.

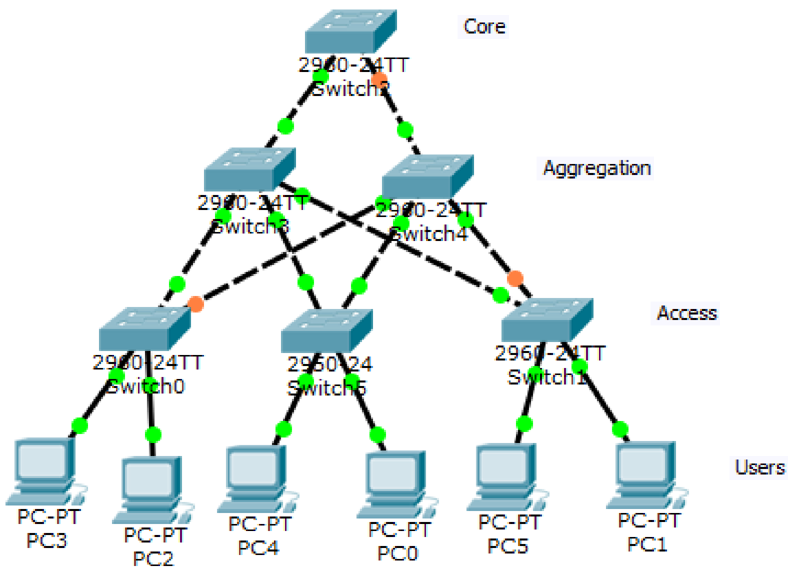


Figure 3.6 Shows the initial network design

As shown in figure 3.6, this is known as a logical Three-layered Hierarchical Model, one of the most common design for any large network. The important concept of this design is to group devices into three tiers. Devices that belongs in the same tier have similar role. Once the role of all devices is defined, we can begin to define the interconnections between devices. This design outlines three layers, namely, Core layer, Aggregation layer and Access layer. Each layer serves has its own specific purpose, so whatever we apply into the access layer won't be implemented on the aggregation and core.

Due to the restrain of budget, we could not acquire enough switches to build a full Three-layered Hierarchical Model, so we opt to scale down the network design by limiting only one switch per layer. For example, one switch for access switch, one for aggregation and finally one switch for core layer. But we are still capable of obtaining enough user devices at UTAR's FYP lab, in this project there will be 3 users, 2 PC desktops and 1 Laptop. The end product of the scaled-down network design should be something like shown in Figure 3.7

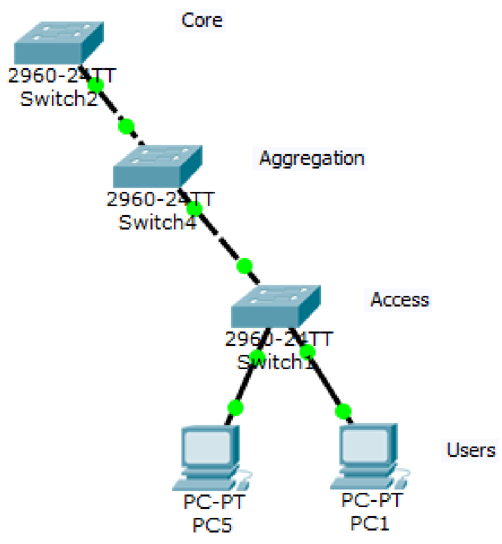


Figure 3.7 Shows the final network design

Once the final designed is determined, we build the logical design into actual physical design with the switches borrowed from UTAR's FYP lab. The switches borrowed are 2 Cisco 2960 Series switch and 1 Huawei Quidway S5700 series switch. The implementation of the physical network is shown in Figure 3.8

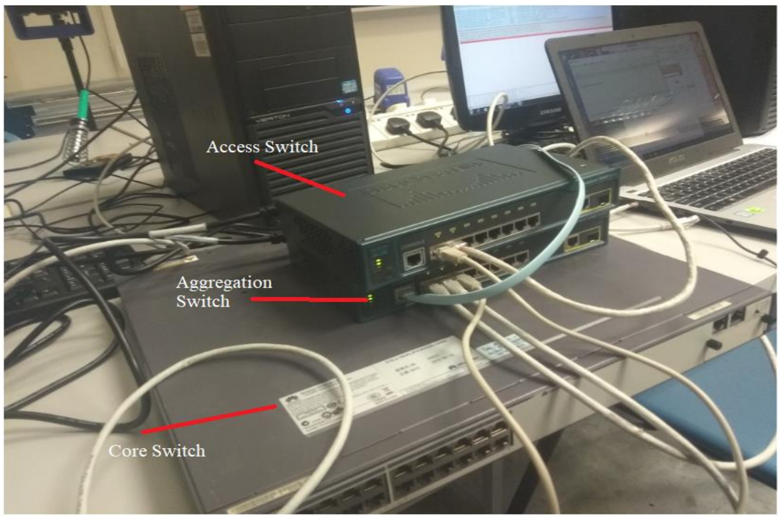


Figure 3.8 shows the physical network design

3.2.2 Collection of network traffic

Once our physical network is completed and ready to go, we have to first figure out what technique we will use to collect the network. After much deliberation we will be using a technique known as the principle of divide and conquer. Why divide and conquer? In a network, the packets flows can be overwhelming for a person to study and analyse. Therefore, with divide and conquer, we can split a large network into smaller sub-network. Which we will then attempt to collect traffic flow from the smaller sub sections.

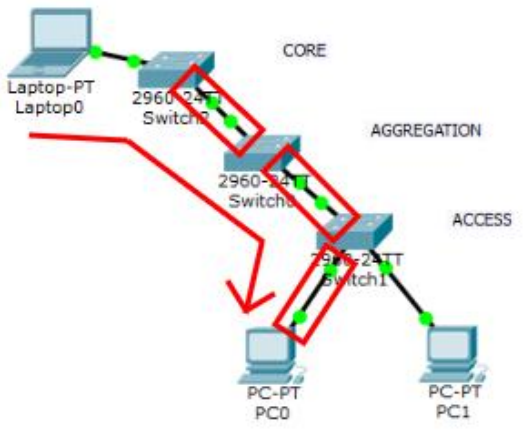
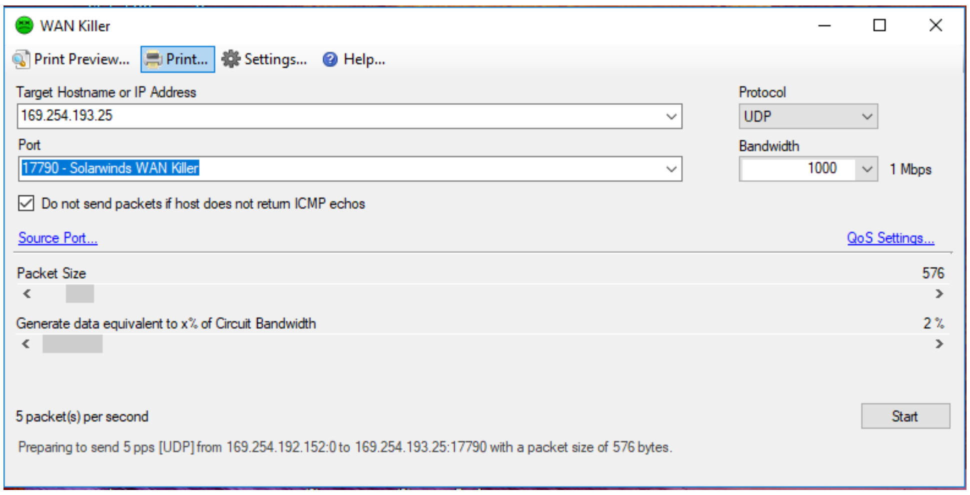


Figure 3.9 shows the “Divide and conquer”

We divide and conquer by pumping in randomly generated traffic with the help of WAN Killer software which is installed in my laptop. My laptop will then start inputting random traffic, starting from the core switch, the packets will then flow to the users, since I have previously set the software to target the user.

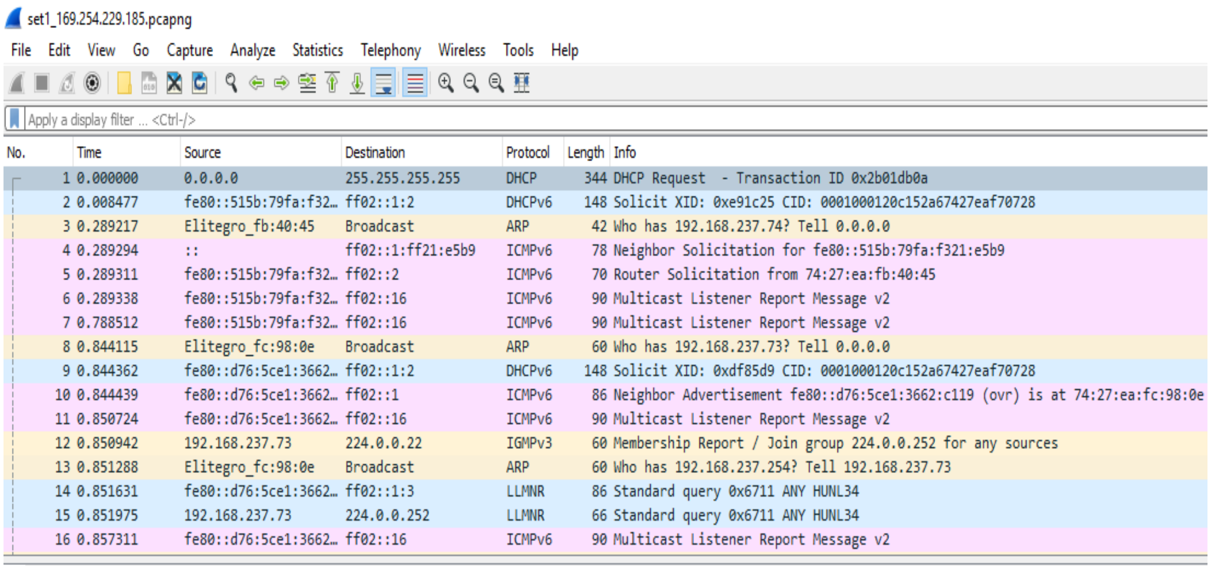


Target Hostname or IP Address: 169.254.193.25
Protocol: UDP
Port: 17790 - Solarwinds WAN Killer
Bandwidth: 1000 1 Mbps
☑ Do not send packets if host does not return ICMP echos
Packet Size: 576
Generate data equivalent to x% of Circuit Bandwidth: 2 %
5 packet(s) per second
Preparing to send 5 pps [UDP] from 169.254.192.152:0 to 169.254.193.25:17790 with a packet size of 576 bytes.

Figure 3.10 shows the interface of “WanKiller”

Figure 3.10 Shows an example of WAN Killer interface, this tools is used to generate a set of random traffic to be analysed

We can now begin collecting packets at different point of the network as highlighted at Figure 3.9. Collecting packets at these different points allows us to identify devices in a network based on the traffic or applications of network packets or flows later in the project.


set1_169.254.229.185.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	0.0.0.0	255.255.255.255	DHCP	344	DHCP Request - Transaction ID 0x2b01db0a
2	0.008477	fe80::515b:79fa:f32…	ff02::1:2	DHCPv6	148	Solicit XID: 0xe91c25 CID: 0001000120c152a67427eaf70728
3	0.289217	Elitegro_fb:40:45	Broadcast	ARP	42	Who has 192.168.237.74? Tell 0.0.0.0
4	0.289294	::	ff02::1:ff21:e5b9	ICMPv6	78	Neighbor Solicitation for fe80::515b:79fa:f321:e5b9
5	0.289311	fe80::515b:79fa:f32…	ff02::2	ICMPv6	70	Router Solicitation from 74:27:ea:fb:40:45
6	0.289338	fe80::515b:79fa:f32…	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
7	0.788512	fe80::515b:79fa:f32…	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
8	0.844115	Elitegro_fc:98:0e	Broadcast	ARP	60	Who has 192.168.237.73? Tell 0.0.0.0
9	0.844362	fe80::d76:5ce1:3662…	ff02::1:2	DHCPv6	148	Solicit XID: 0xdf85d9 CID: 0001000120c152a67427eaf70728
10	0.844439	fe80::d76:5ce1:3662…	ff02::1	ICMPv6	86	Neighbor Advertisement fe80::d76:5ce1:3662:c119 (ovr) is at 74:27:ea:fc:98:0e
11	0.850724	fe80::d76:5ce1:3662…	ff02::16	ICMPv6	90	Multicast Listener Report Message v2
12	0.850942	192.168.237.73	224.0.0.22	IGMPv3	60	Membership Report / Join group 224.0.0.252 for any sources
13	0.851288	Elitegro_fc:98:0e	Broadcast	ARP	60	Who has 192.168.237.254? Tell 192.168.237.73
14	0.851631	fe80::d76:5ce1:3662…	ff02::1:3	LLMNR	86	Standard query 0x6711 ANY HUNL34
15	0.851975	192.168.237.73	224.0.0.252	LLMNR	66	Standard query 0x6711 ANY HUNL34
16	0.857311	fe80::d76:5ce1:3662…	ff02::16	ICMPv6	90	Multicast Listener Report Message v2

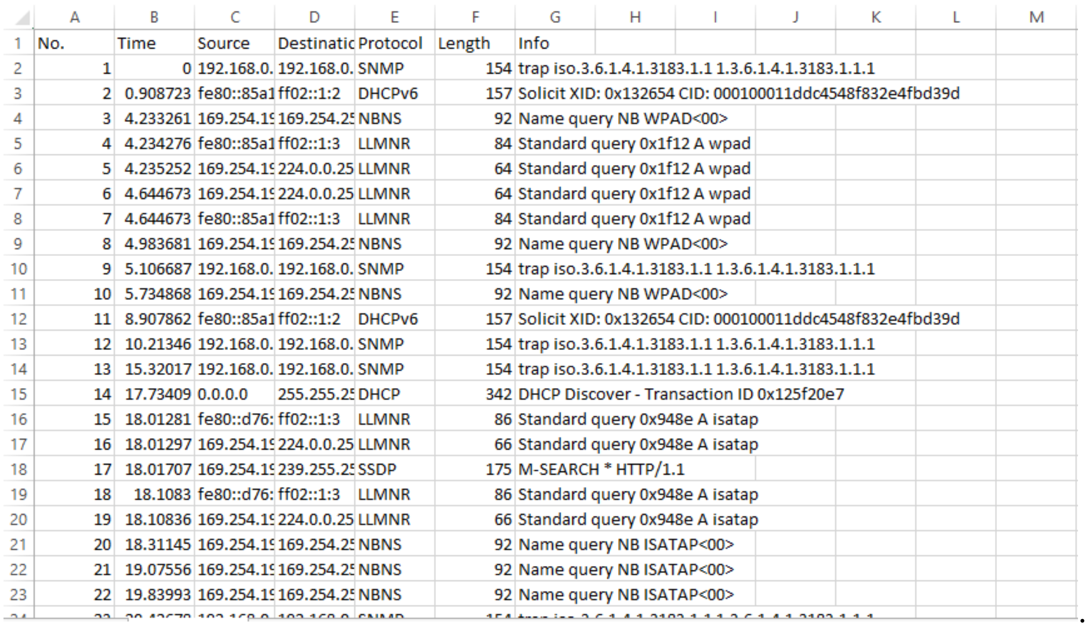
Figure 3.11 Example of Wireshark captures

Figure 3.11 shows an example of wireshark data collected at the network. In the case of network data, as shown in the Wireshark screenshot above, packets are described with the following features, the time frame of arrival, frame length, the IP address of the source, IP address of the destination, network protocol and info about the frame.

3.2.3 Labelling and filtering of data

Raw data collected is useless since there exist some data that cannot be used for machine learning. For example, in our case we are in the position of a passive observer, in order to build our classifier, we will only focus on protocols and size-related metrics and not on features based on content. In this section we will show how we label and filter necessary data for Machine Learning.

The first step will be converting the data into a more useful format, in our case a CSV file, the data can be then used to analysed by Jupyter

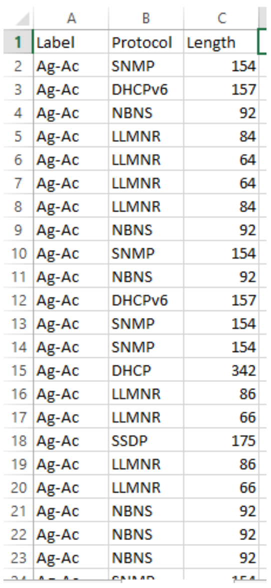


No.	Time	Source	Destinatic	Protocol	Length	Info
1	0	192.168.0.	192.168.0.	SNMP	154	trap iso.3.6.1.4.1.3183.1.1 1.3.6.1.4.1.3183.1.1.1
2	0.908723	fe80::85a1	ff02::1:2	DHCPv6	157	Solicit XID: 0x132654 CID: 000100011ddc4548f832e4fbd39d
3	4.233261	169.254.1	169.254.2	NBNS	92	Name query NB WPAD<00>
4	4.234276	fe80::85a1	ff02::1:3	LLMNR	84	Standard query 0x1f12 A wpad
5	4.235252	169.254.1	224.0.0.25	LLMNR	64	Standard query 0x1f12 A wpad
6	4.644673	169.254.1	224.0.0.25	LLMNR	64	Standard query 0x1f12 A wpad
7	4.644673	fe80::85a1	ff02::1:3	LLMNR	84	Standard query 0x1f12 A wpad
8	4.983681	169.254.1	169.254.2	NBNS	92	Name query NB WPAD<00>
9	5.106687	192.168.0.	192.168.0.	SNMP	154	trap iso.3.6.1.4.1.3183.1.1 1.3.6.1.4.1.3183.1.1.1
10	5.734868	169.254.1	169.254.2	NBNS	92	Name query NB WPAD<00>
11	8.907862	fe80::85a1	ff02::1:2	DHCPv6	157	Solicit XID: 0x132654 CID: 000100011ddc4548f832e4fbd39d
12	10.21346	192.168.0.	192.168.0.	SNMP	154	trap iso.3.6.1.4.1.3183.1.1 1.3.6.1.4.1.3183.1.1.1
13	15.32017	192.168.0.	192.168.0.	SNMP	154	trap iso.3.6.1.4.1.3183.1.1 1.3.6.1.4.1.3183.1.1.1
14	17.73409	0.0.0.0	255.255.2	DHCP	342	DHCP Discover - Transaction ID 0x125f20e7
15	18.01281	fe80::d76:	ff02::1:3	LLMNR	86	Standard query 0x948e A isatap
16	18.01297	169.254.1	224.0.0.25	LLMNR	66	Standard query 0x948e A isatap
17	18.01707	169.254.1	239.255.2	SSDP	175	M-SEARCH * HTTP/1.1
18	18.1083	fe80::d76:	ff02::1:3	LLMNR	86	Standard query 0x948e A isatap
19	18.10836	169.254.1	224.0.0.25	LLMNR	66	Standard query 0x948e A isatap
20	18.31145	169.254.1	169.254.2	NBNS	92	Name query NB ISATAP<00>
21	19.07556	169.254.1	169.254.2	NBNS	92	Name query NB ISATAP<00>
22	19.83993	169.254.1	169.254.2	NBNS	92	Name query NB ISATAP<00>

Figure 3.12 Show the Initial CSV reading

As you can see at figure 3.12, the list of items in the dataset is an ethernet frame, it is now parallels to one line in the CSV file. Each IP packets has its own line of information. Next, we have to start filter out information which we cannot use for machine learning. Firstly, since IP address can be dynamic for different network, it is unreliable to use both source and destination address will not be used to build the classifier. Info property is too complicated to be used into machine learning algorithm, therefore it will be omitted. Time is an unreliable data, since we will later combine several dataset together to create a big data model, it will be slightly inconsistent. Finally, we are left with protocols and data length will be used.

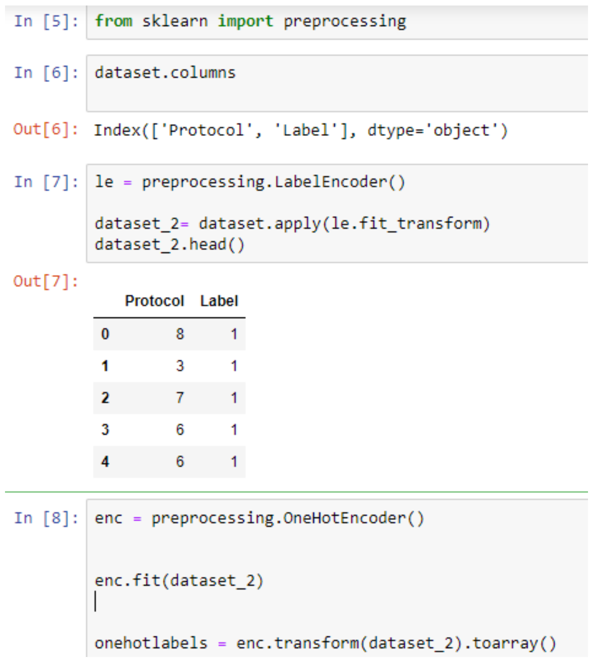
Once that is said and done, we can begin labelling out data, in supervised learning, it is imperative to label the data so that we can efficiently train the neural network to learn the correlation between labels and data.



	A	B	C
1	Label	Protocol	Length
2	Ag-Ac	SNMP	154
3	Ag-Ac	DHCPv6	157
4	Ag-Ac	NBNS	92
5	Ag-Ac	LLMNR	84
6	Ag-Ac	LLMNR	64
7	Ag-Ac	LLMNR	64
8	Ag-Ac	LLMNR	84
9	Ag-Ac	NBNS	92
10	Ag-Ac	SNMP	154
11	Ag-Ac	NBNS	92
12	Ag-Ac	DHCPv6	157
13	Ag-Ac	SNMP	154
14	Ag-Ac	SNMP	154
15	Ag-Ac	DHCP	342
16	Ag-Ac	LLMNR	86
17	Ag-Ac	LLMNR	66
18	Ag-Ac	SSDP	175
19	Ag-Ac	LLMNR	86
20	Ag-Ac	LLMNR	66
21	Ag-Ac	NBNS	92
22	Ag-Ac	NBNS	92
23	Ag-Ac	NBNS	92

Figure 3.13 Filtered and labelled CSV reading

As shown in Figure 3.13, the label of our devices will the different sections where we collected out traffic. The labels are labelled as, Ag-Ac (Aggregation and Access switch), Ac-U (Aggregation and User) and finally Cs-Ag (Core and Aggregation) Labelling this way is significant for us to identify the topology of the network. Now comes another issue, we have faced, the problem with our dataset is the fact that, both "Label" and "Protocols" belong to the string data type. In order to successfully utilize our dataset for machine learning we have to convert these values to integers. This is where "One Hot Encoding" comes in, one hot encoding is a process where categorical variables are changed into a form of "zeroes and ones" so that ML algorithms are able to do a better job in prediction.



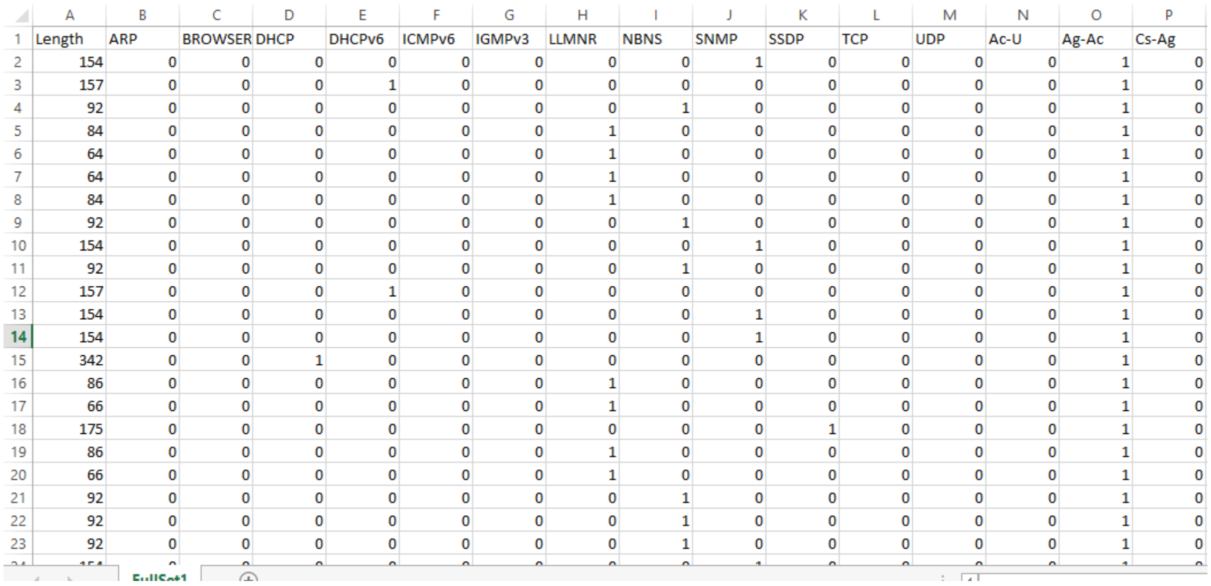
```
In [5]: from sklearn import preprocessing

In [6]: dataset.columns

Out[6]: Index(['Protocol', 'Label'], dtype='object')

In [7]: le = preprocessing.LabelEncoder()

        dataset_2= dataset.apply(le.fit_transform)
        dataset_2.head()

Out[7]:
```

	Protocol	Label
0	8	1
1	3	1
2	7	1
3	6	1
4	6	1

```
In [8]: enc = preprocessing.OneHotEncoder()


        enc.fit(dataset_2)


        onehotlabels = enc.transform(dataset_2).toarray()
```

Figure 3.14 Snippet of data processing code

Figure 3.14 Shows a snippet of python code used to one hot encode the dataset. The end result will be the image shown below. Now the CSV file is ready to be put into a ML algorithm to be trained.



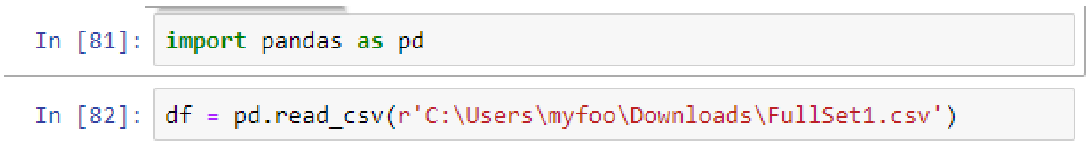
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Length	ARP	BROWSER	DHCP	DHCPv6	ICMPv6	IGMPv3	LLMNR	NBNS	SNMP	SSDP	TCP	UDP	Ac-U	Ag-Ac	Cs-Ag
2	154	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
3	157	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0
4	92	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
5	84	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
6	64	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
7	64	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
8	84	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
9	92	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
10	154	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
11	92	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
12	157	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0
13	154	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
14	154	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
15	342	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
16	86	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
17	66	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
18	175	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0
19	86	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
20	66	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
21	92	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
22	92	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
23	92	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0

Figure 3.15 Final CSV reading

3.2.4 Training the data

Now that the data is ready to be trained, we have to first determine what kind of machine learning which we want to go for. We first have consider what we are trying to achieve with the machine learning, we want our machine to be able to automatically classify and predict the type of devices in a network from the patterns of data (Network Traffic), therefore, what better choice we have other than an Artificial neural network? Moreover, previously we have already prepared a set of data that are labelled and ready to be trained, hence, when it comes to the types of learning it will of course be supervised learning for the ANN. Our neural network environment will be set up in a Jupyter Notebook with keras, tensorflow, pandas, scikit-learn packages installed. The dataset that I will be using is what we complied at the previous section.

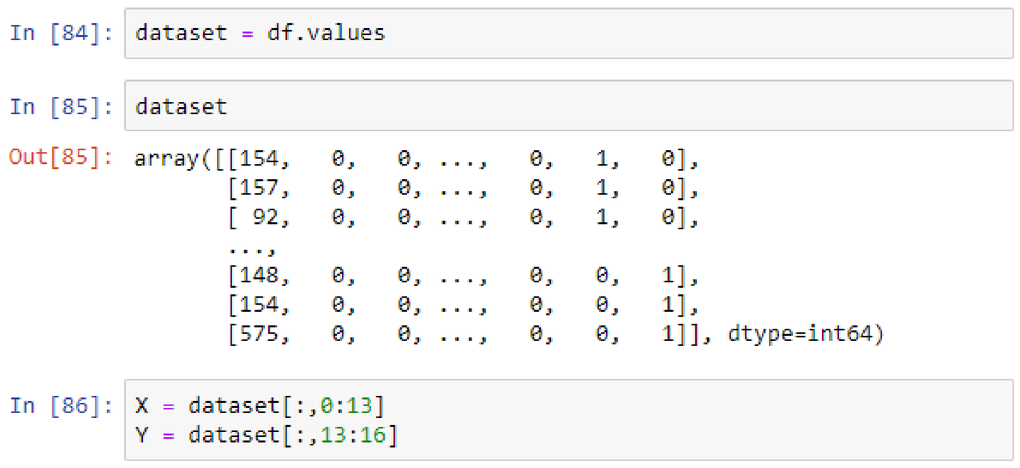
We begin telling our notebook that we will using pandas package by importing it, then, we will read the CSV file 'FullSet1.csv" and store it in the variable 'df'. The code should shown in Figure 3.16



```
In [81]: import pandas as pd

In [82]: df = pd.read_csv(r'C:\Users\myfoo\Downloads\FullSet1.csv')
```

Figure 3.16 Shows the importing of package and CSV file

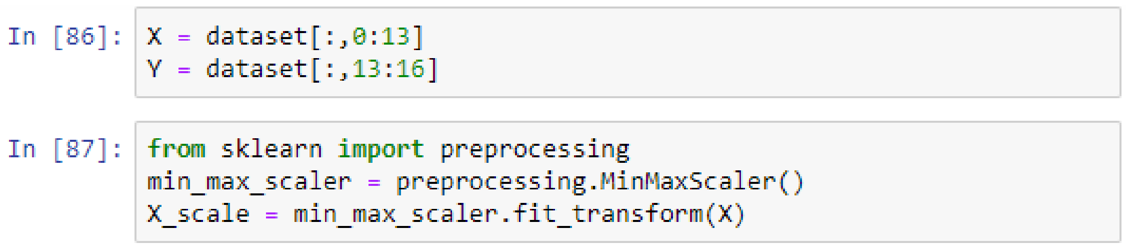
Next, we proceed by converting our dataframe into an array, we just store the values of df into the variable 'dataset'. Now, we split the dataset into X (input features) and Y (features that we want to predict). For that to happen, we just have to assign the first 13 columns of our array to a variable called X and the 3 columns of our array to a variable called Y.



```
In [84]: dataset = df.values

In [85]: dataset

Out[85]: array([[154,   0,   0, ...,   0,   1,   0],
                [157,   0,   0, ...,   0,   1,   0],
                [ 92,   0,   0, ...,   0,   1,   0],
                ...,
                [148,   0,   0, ...,   0,   0,   1],
                [154,   0,   0, ...,   0,   0,   1],
                [575,   0,   0, ...,   0,   0,   1]], dtype=int64)

In [86]: X = dataset[:,0:13]
         Y = dataset[:,13:16]
```

Figure 3.17 Shows the conversion of data to array

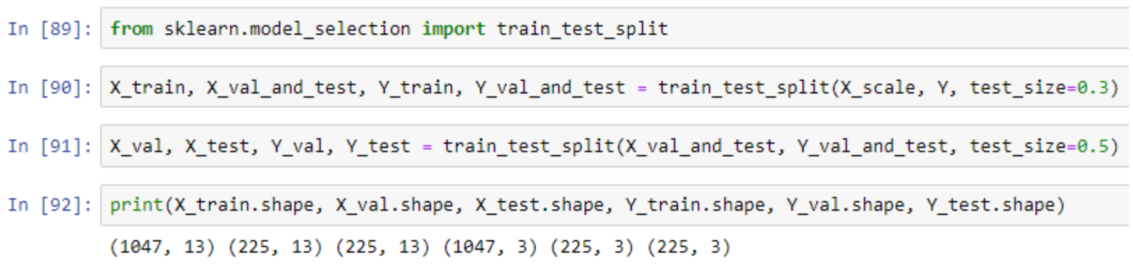
For the sake of reducing errors between expected values and predictions, the scaling of inputs and outputs are important factor. Hence, data scaling is imperative during the pre-processing step when dealing with neural network. We can easily solve that with a function called the min-max scaler, that can help us scale the dataset so that all the input features will be either “Ones or zeroes” inclusive.



```
In [86]: X = dataset[:,0:13]
         Y = dataset[:,13:16]

In [87]: from sklearn import preprocessing
         min_max_scaler = preprocessing.MinMaxScaler()
         X_scale = min_max_scaler.fit_transform(X)
```

Figure 3.18 Shows the scaling of array data

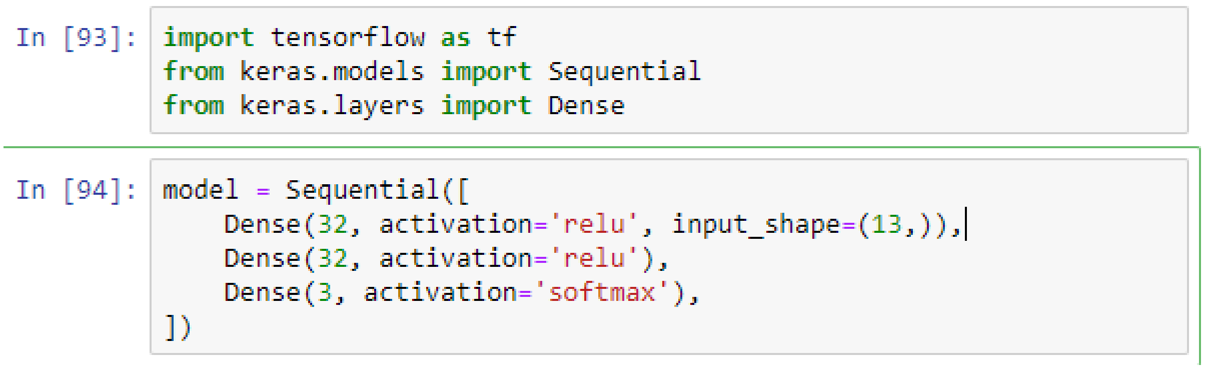
The final step in processing data, is the splitting of the dataset into training set, a validation set and a test set. As observed on figure 3.19, the training set ends up with 1047 data points while the validation and test set has 225 data points each. 13 inputs for the X variables, meanwhile the Y variables has 3 features to predict.



```
In [89]: from sklearn.model_selection import train_test_split

In [90]: X_train, X_val_and_test, Y_train, Y_val_and_test = train_test_split(X_scale, Y, test_size=0.3)

In [91]: X_val, X_test, Y_val, Y_test = train_test_split(X_val_and_test, Y_val_and_test, test_size=0.5)

In [92]: print(X_train.shape, X_val.shape, X_test.shape, Y_train.shape, Y_val.shape, Y_test.shape)

         (1047, 13) (225, 13) (225, 13) (1047, 3) (225, 3) (225, 3)
```

Figure 3.19 Shows the splitting of datasets

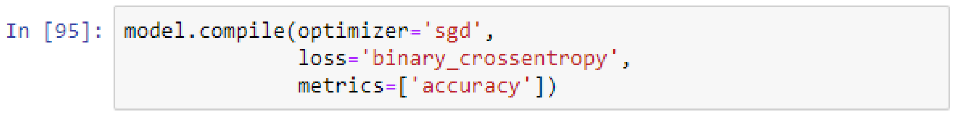
Since the model we will be using for the neural network will be a sequential model, we will have to build the code this way, as shown in figure 3.20



```
In [93]: import tensorflow as tf
         from keras.models import Sequential
         from keras.layers import Dense

In [94]: model = Sequential([
             Dense(32, activation='relu', input_shape=(13,)),
             Dense(32, activation='relu'),
             Dense(3, activation='softmax'),
         ])
```

Figure 3.20 Shows the Code of the neural network model

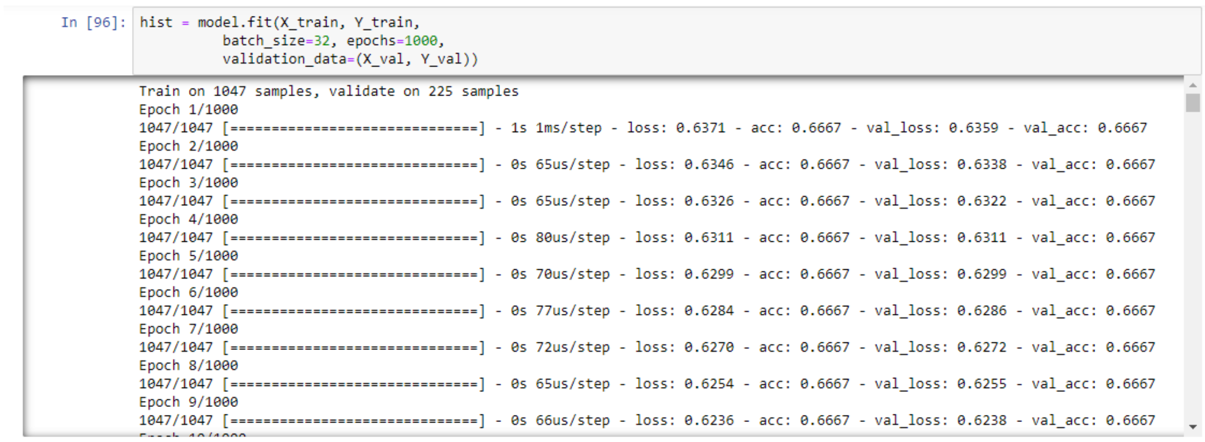
The model consist of 2 dense hidden layer with 32 neuron and based on ReLU activation, 1 dense output layer with 3 neuron and based on Softmax activation. With this our model architecture is complete, we can now move on to configuring the model



```
In [95]: model.compile(optimizer='sgd',
                       loss='binary_crossentropy',
                       metrics=['accuracy'])
```

Figure 3.21 Show the model configuration code

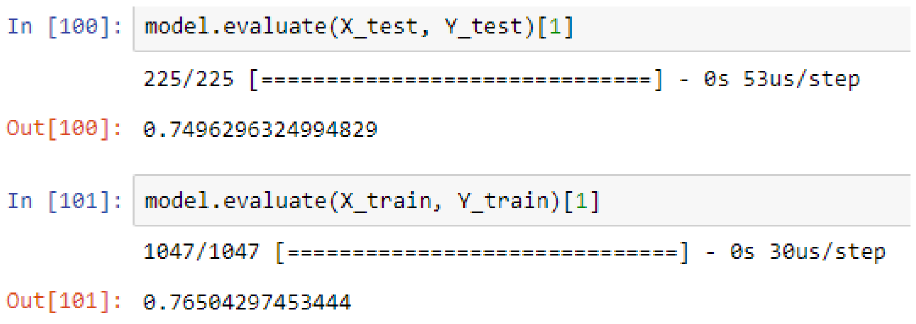
The settings for our model configuration will be as such, “stochastic gradient descent” as the algorithm to use for optimization, binary cross entropy which is the loss function for outputs that take values of “ones” and “zeroes”, and finally, the accuracy metrics that we want to track apart from the loss function.



```
In [96]: hist = model.fit(X_train, Y_train,
                    batch_size=32, epochs=1000,
                    validation_data=(X_val, Y_val))

Train on 1047 samples, validate on 225 samples
Epoch 1/1000
1047/1047 [==============================] - 1s 1ms/step - loss: 0.6371 - acc: 0.6667 - val_loss: 0.6359 - val_acc: 0.6667
Epoch 2/1000
1047/1047 [==============================] - 0s 65us/step - loss: 0.6346 - acc: 0.6667 - val_loss: 0.6338 - val_acc: 0.6667
Epoch 3/1000
1047/1047 [==============================] - 0s 65us/step - loss: 0.6326 - acc: 0.6667 - val_loss: 0.6322 - val_acc: 0.6667
Epoch 4/1000
1047/1047 [==============================] - 0s 80us/step - loss: 0.6311 - acc: 0.6667 - val_loss: 0.6311 - val_acc: 0.6667
Epoch 5/1000
1047/1047 [==============================] - 0s 70us/step - loss: 0.6299 - acc: 0.6667 - val_loss: 0.6299 - val_acc: 0.6667
Epoch 6/1000
1047/1047 [==============================] - 0s 77us/step - loss: 0.6284 - acc: 0.6667 - val_loss: 0.6286 - val_acc: 0.6667
Epoch 7/1000
1047/1047 [==============================] - 0s 72us/step - loss: 0.6270 - acc: 0.6667 - val_loss: 0.6272 - val_acc: 0.6667
Epoch 8/1000
1047/1047 [==============================] - 0s 65us/step - loss: 0.6254 - acc: 0.6667 - val_loss: 0.6255 - val_acc: 0.6667
Epoch 9/1000
1047/1047 [==============================] - 0s 66us/step - loss: 0.6236 - acc: 0.6667 - val_loss: 0.6238 - val_acc: 0.6667
```

Figure 3.22 Shows the process of training the neural network

As shown in Figure 3.22, function ‘fit’ are called as we are fitting the parameters to the data. X_train and Y_train is the data specified to train the neural network. Afterwards, we specify the size and how long we want to train for with batch_size and epochs function, in our case it will be 32 and 1000. Lastly, we are able specify our validation data so that the model will show us how we are performing on the validation data at each point.



```
In [100]: model.evaluate(X_test, Y_test)[1]

          225/225 [==============================] - 0s 53us/step

Out[100]: 0.7496296324994829

In [101]: model.evaluate(X_train, Y_train)[1]

          1047/1047 [==============================] - 0s 30us/step

Out[101]: 0.76504297453444
```

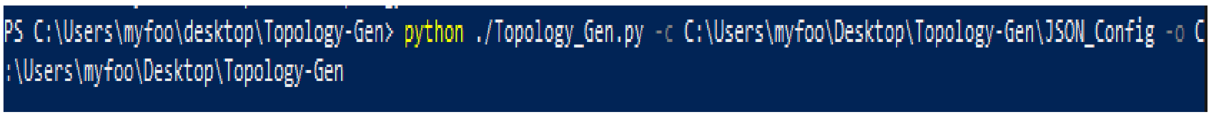
Figure 3.23 Shows the final accuracy of the test and train model

Finally, we are able to obtain the test and train accuracy around 75% as shown in figure above

3.2.5 Generation of network topology

As part of the requirement for the FYP the generation/visualization of the topology from the result of the AI model is crucial in order to achieve our main objective. Therefore, in order to fulfil such requirement, a topology visualizer is developed using Python and JSON programming language. This python tool is used to generate a number random network topologies which include, servers, clients, switches and routers. The user can key in parameters specified in a JSON configuration file. Finally, the generated topologies are written into JSON output files and plotted in PDF format for the user to see.

In order to run the topology visualizer, the following command have to be type in "Window Powershell", *"$ ./Topology_Gen.py -c <configFile> -o <outputFile>"* where <*configFile*> is the configuration file path and <*outputFile*> is the output file name without the extension. The topology generator will create two output files: <*outputFile*>.json and <*outputFile*>.pdf, if requested. As sample is shown below.



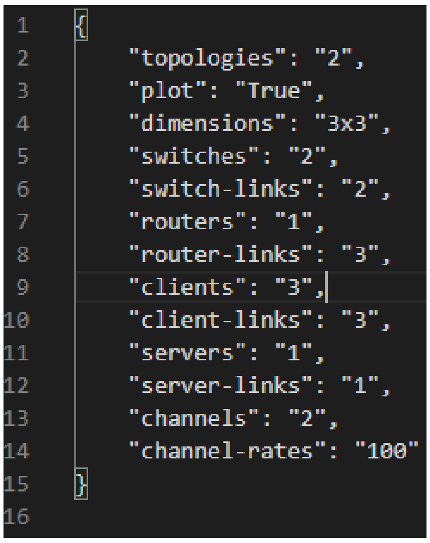
```
PS C:\Users\myfoo\desktop\Topology-Gen> python ./Topology_Gen.py -c C:\Users\myfoo\Desktop\Topology-Gen\JSON_Config -o C
:\Users\myfoo\Desktop\Topology-Gen
```

Figure 3.24 Shows powershell to input the commands

As stated previously, user can manually key in the parameters using a JSON configuration file. All parameters are required to be keyed in and explained as follows:

- **Topologies**: This parameter is how many topologies is generated. For example, if 3 is typed in, 3 topologies will be generated.
- **Plot**: Whether or not to plot the topology in PDF, True or False can be set
- **Dimensions** : Area of dimension where the topology is generated
- **Switches**: The number of switches to be generated
- **Switch-links**: States the number of links each switch can be connected to the adjacent routers. This parameter can be specified with an exact number of link or a range of the format of minimum to maximum. For example, "3-7" links
- **Routers**: The number of routers to be generated
- **Router-links**: States the number of links each router can be connected to the adjacent routers. This parameter can be specified with an exact number of link or a range of the format of minimum to maximum. For example, "3-7" links
- **Clients**: The number of clients in the topology.

- **Client-links**: States the number of links each client can be connected to the adjacent routers. This parameter can specified with an exact number of link or a range of the format of minimum to maximum. For example, “3-7” links
- **Servers**: The number of servers in the topology.
- **Server-links**: States the number of links each server can be connected to the adjacent routers. This parameter can be specified with an exact number of link or a range of the format of minimum to maximum, For example, 3-5
- **Channels**: Number of channels can be used to connect the topology nodes. Data rate is different for each channel
- **Channel-rates**: List out the channel rates, the number in this list should be identical to the value of the channel parameter.



```
{
    "topologies": "2",
    "plot": "True",
    "dimensions": "3x3",
    "switches": "2",
    "switch-links": "2",
    "routers": "1",
    "router-links": "3",
    "clients": "3",
    "client-links": "3",
    "servers": "1",
    "server-links": "1",
    "channels": "2",
    "channel-rates": "100"
}
```

Figure 3.25 Shows input JSON configuration file

An example of the topologies generated is shown below, with yellow representing the switches, red representing the routers, blue representing the client and green representing the server. The output is according to the data keyed-in the JSON file.

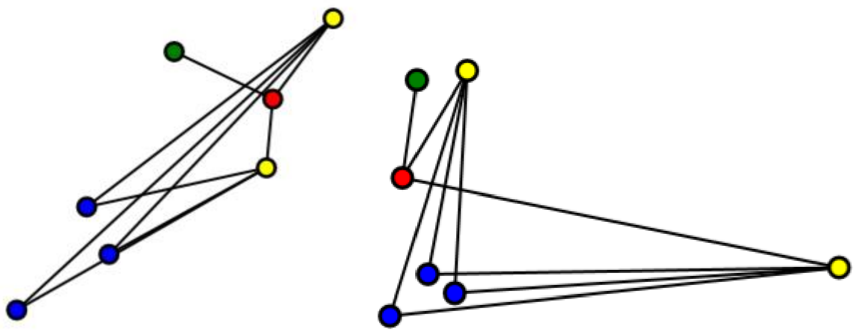


Figure 3.26 Shows example of topology generated

Without a doubt, we cannot consider this as a topology vizualization without the topology visualized in a structure where it resembles what human can understand. Therefore, the program is further enhanced to resemble hierarchical topology.

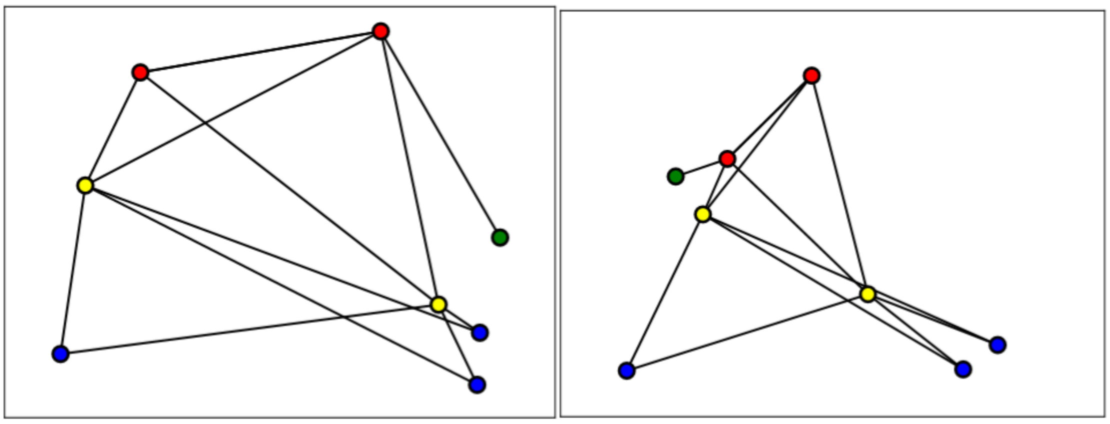


Figure 3.27 Shows example of hierarchical topology generated

As you can see above, a hierarchical topology can be observed with the routers (RED) at the very top, connecting down to the switches (YELLOW) with the server (GREEN) branching off from one of the router. Finally the clients (BLUE) is connected at the very bottom.

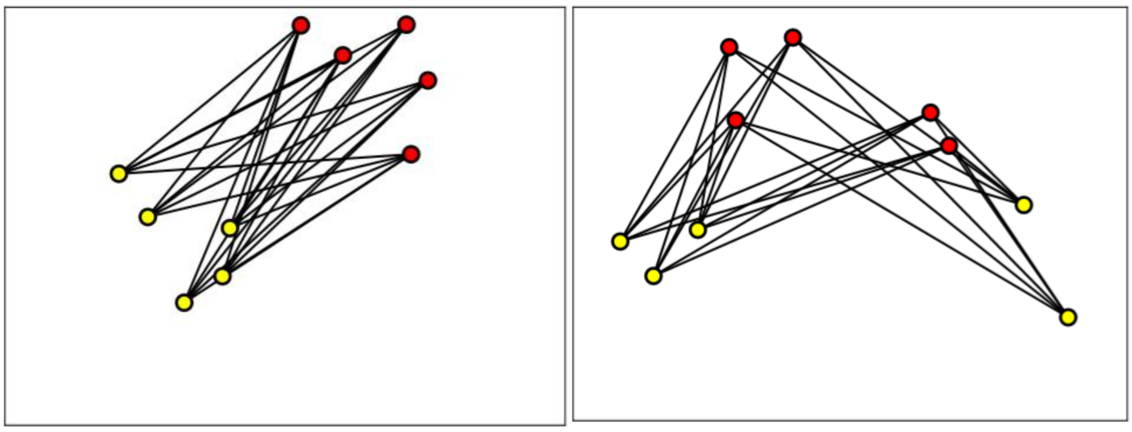
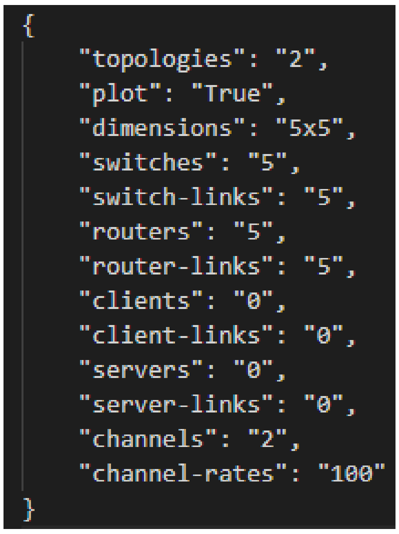


Figure 3.28 Shows example of leaf-spine topology generated

Moreover, with this tool we are also able to generate a leaf-spine topology where each "leaf" (YELLOW) switches are able to connect with the "spine" (RED) router. But the limitation with that, we have to manually remove the connection between router – router.



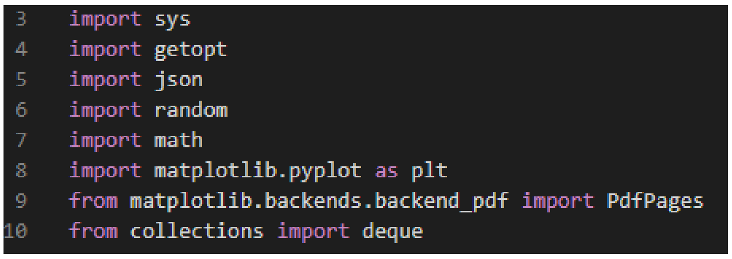
```
{
    "topologies": "2",
    "plot": "True",
    "dimensions": "5x5",
    "switches": "5",
    "switch-links": "5",
    "routers": "5",
    "router-links": "5",
    "clients": "0",
    "client-links": "0",
    "servers": "0",
    "server-links": "0",
    "channels": "2",
    "channel-rates": "100"
}
```

Figure 3.29 Shows example of leaf-spine JSON configuration file

The Leaf - Spine topology can be achieved with such input of the JSON file, by only allowing router and switches to exist in the topology, we are able to achieve topology design that has only two layers, the Leaf layer and Spine layer.

3.3 Code explanation of topology visualizer

In this section, we will begin to dive deeper into the coding aspect of the topology visualizer. First and foremost, as previously stated, the visualizer will be coded in python. Why python? Due to its extensive support libraries, more specifically, we will be using *matplotlib.pyplot* as a way for us to plot point to represent nodes in a network topology. Additionally, we will be using JSON file as the input of the parameter of the topology visualizer, thanks to the simplicity and it is easier to work with python.

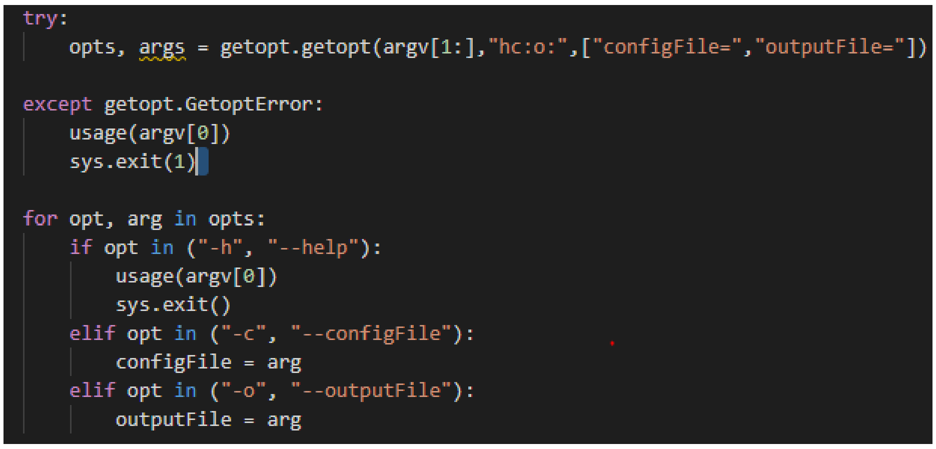


```python
import sys
import getopt
import json
import random
import math
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
from collections import deque
```

Figure 3.30 Shows the imported libraries

Firstly we have to import libraries that are necessary for the program to work, I will be highlighting some of the libraries that are important.

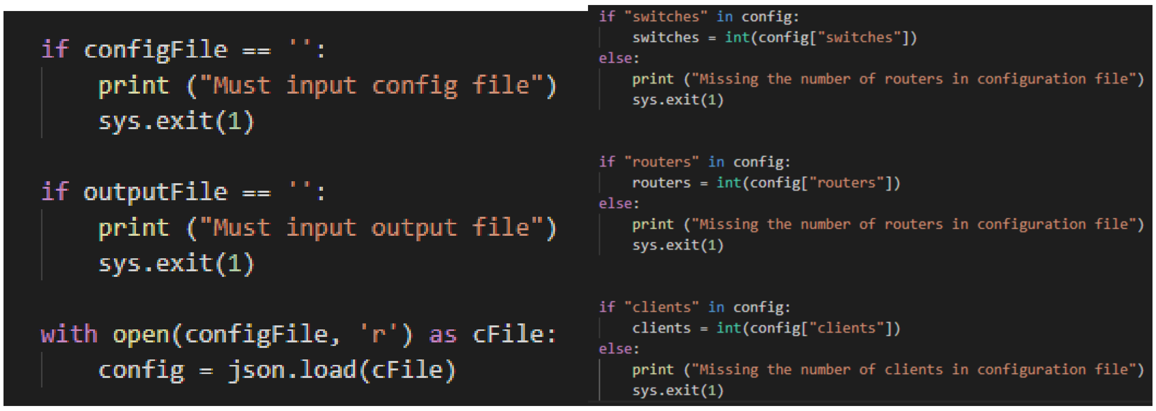
- “import getopt” is used for parsing the command line arguments in sys.argv, it is important for us in order to get JSON configuration files using PowerShell.
- “import json” is necessary for us to read and write JSON files in this tool
- “import matplotlib.pyplot” is used for plotting point which we will use to represent nodes in a network topology
- “import PdfPages” is for creating a pdf file after the plotting the points



```python
try:
    opts, args = getopt.getopt(argv[1:],"hc:o:",["configFile=","outputFile="])

except getopt.GetoptError:
    usage(argv[0])
    sys.exit(1)

for opt, arg in opts:
    if opt in ("-h", "--help"):
        usage(argv[0])
        sys.exit()
    elif opt in ("-c", "--configFile"):
        configFile = arg
    elif opt in ("-o", "--outputFile"):
        outputFile = arg
```

Figure 3.31 Shows parsing code section

“getopt.getopt” will be used to parse command line options and the parameter list. “args” is the argument list to be parsed. The options that are available, include help, the config file and the required output file

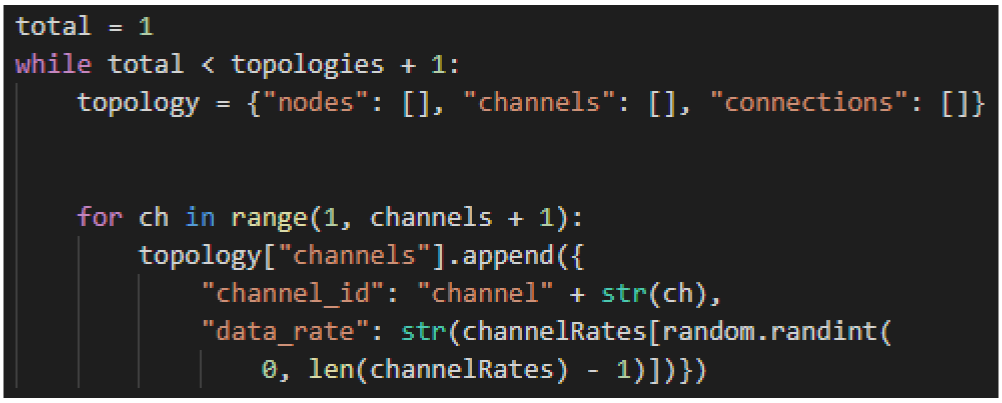


```
if configFile == '':
    print ("Must input config file")
    sys.exit(1)

if outputFile == '':
    print ("Must input output file")
    sys.exit(1)

with open(configFile, 'r') as cFile:
    config = json.load(cFile)
```

```
if "switches" in config:
    switches = int(config["switches"])
else:
    print ("Missing the number of routers in configuration file")
    sys.exit(1)

if "routers" in config:
    routers = int(config["routers"])
else:
    print ("Missing the number of routers in configuration file")
    sys.exit(1)

if "clients" in config:
    clients = int(config["clients"])
else:
    print ("Missing the number of clients in configuration file")
    sys.exit(1)
```

Figure 3.32 Shows the condition checking code

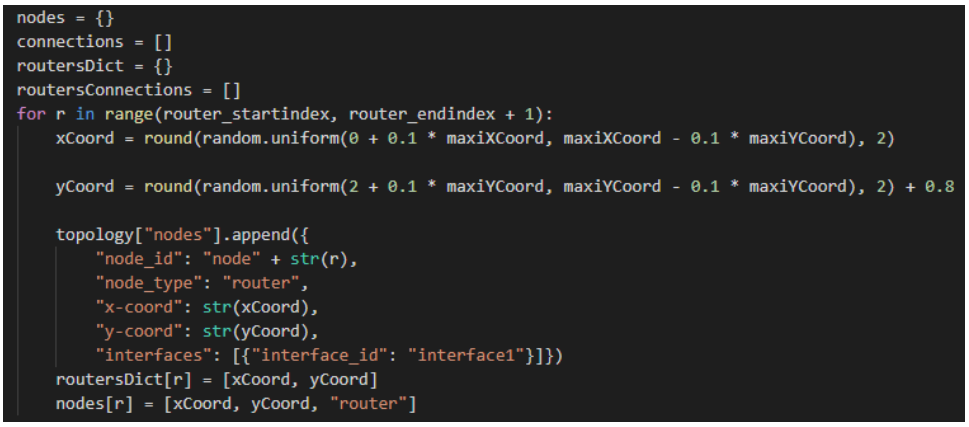
This part of the code mainly just deals with the condition checking and parameter checking, this is to ensure the user have successfully input the configuration files and specified the output files. After the configuration files are available and opened, parameter checking is done to ensure the values in the JSON configuration files in keyed in correctly. All parameter in the JSON config file is checked, if no errors were found, the values will be assigned into the variables.



```
total = 1
while total < topologies + 1:
    topology = {"nodes": [], "channels": [], "connections": []}


    for ch in range(1, channels + 1):
        topology["channels"].append({
            "channel_id": "channel" + str(ch),
            "data_rate": str(channelRates[random.randint(
                0, len(channelRates) - 1)])})
```

Figure 3.33 Shows the creation of dictionaries and data entry

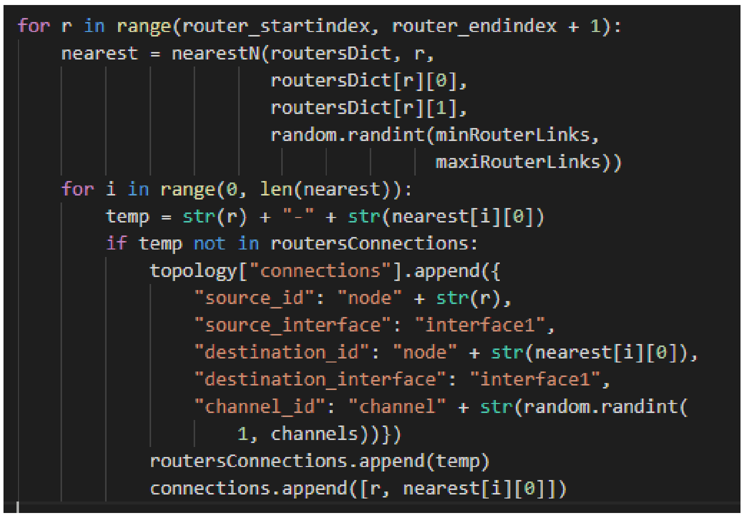
We create a dictionary of list, which we will be using to store more detail information about the network topology. The lists includes nodes, channels and connections. The purpose of the for loop is to key-in channel information in JSON format into the list so that it can be output later.



```
nodes = {}
connections = []
routersDict = {}
routersConnections = []
for r in range(router_startindex, router_endindex + 1):
    xCoord = round(random.uniform(0 + 0.1 * maxiXCoord, maxiXCoord - 0.1 * maxiYCoord), 2)

    yCoord = round(random.uniform(2 + 0.1 * maxiYCoord, maxiYCoord - 0.1 * maxiYCoord), 2) + 0.8

    topology["nodes"].append({
        "node_id": "node" + str(r),
        "node_type": "router",
        "x-coord": str(xCoord),
        "y-coord": str(yCoord),
        "interfaces": [{"interface_id": "interface1"}]})
    routersDict[r] = [xCoord, yCoord]
    nodes[r] = [xCoord, yCoord, "router"]
```

Figure 3.34 Shows the generation of X and Y coordinates

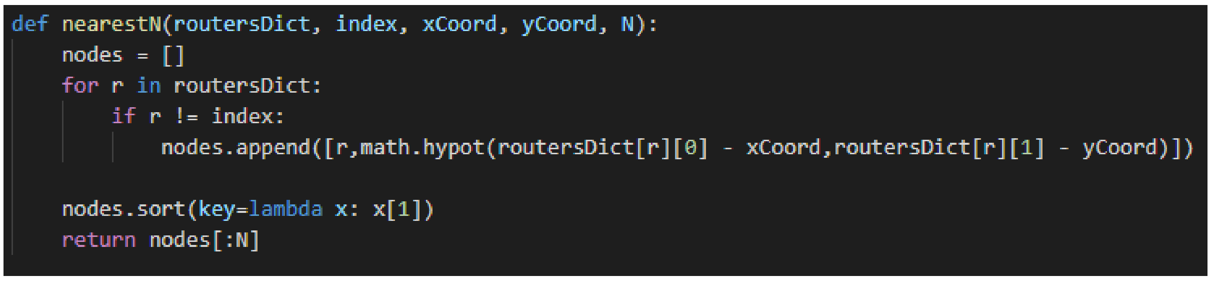
This code basically, add nodes into a topology by randomly assigning X and Y coordinate value. Some restrictions can be observed at the Y coordinate's random.uniform, this is done to allow better positioning of the nodes on a PDF. The next portion is similar to earlier, where for loop is to key-in nodes information in JSON format into the list so that it can be output later. Lastly it also stores X coordinate, Y coordinate and device type into a nodes array for node plotting later.



```
for r in range(router_startindex, router_endindex + 1):
    nearest = nearestN(routersDict, r,
                       routersDict[r][0],
                       routersDict[r][1],
                       random.randint(minRouterLinks,
                                      maxiRouterLinks))
    for i in range(0, len(nearest)):
        temp = str(r) + "-" + str(nearest[i][0])
        if temp not in routersConnections:
            topology["connections"].append({
                "source_id": "node" + str(r),
                "source_interface": "interface1",
                "destination_id": "node" + str(nearest[i][0]),
                "destination_interface": "interface1",
                "channel_id": "channel" + str(random.randint(
                    1, channels))})
            routersConnections.append(temp)
            connections.append([r, nearest[i][0]])
```

Figure 3.35 Shows the generation of X and Y coordinates

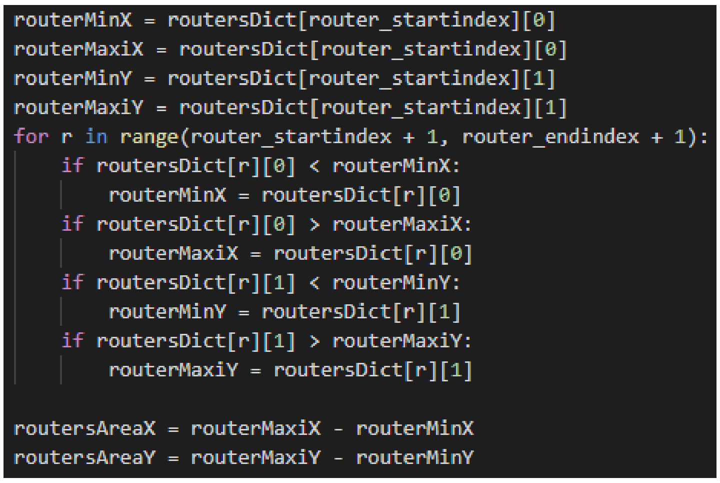
Purpose of this nested for loop is to mainly call the nearestN function (Nearest Node), what it does is basically to connect each routers to nearest neighbouring routers. The next for loop is the same as previously discussed, to key-in connections information in JSON format into the list so that it can be output later. This is what we will be using to our advantage so that we are able to generaye visually identifiable network topology structure.



```
def nearestN(routersDict, index, xCoord, yCoord, N):
    nodes = []
    for r in routersDict:
        if r != index:
            nodes.append([r,math.hypot(routersDict[r][0] - xCoord,routersDict[r][1] - yCoord)])

    nodes.sort(key=lambda x: x[1])
    return nodes[:N]
```

Figure 3.36 Shows the nearestN function

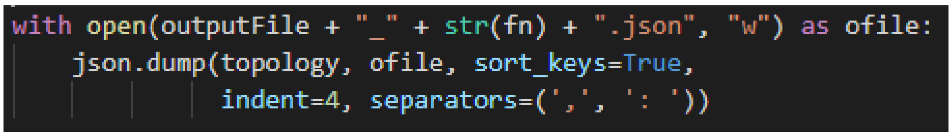
The purpose of the nearestN (Nearest Node) function is to returns a list of the N nearest node indexes along with their distances. lambda x:x[1] is an anonymous function with a single argument, x is a list, the lambda will then returns the first element of the list. This is then used as the key for the sort.



```
routerMinX = routersDict[router_startindex][0]
routerMaxiX = routersDict[router_startindex][0]
routerMinY = routersDict[router_startindex][1]
routerMaxiY = routersDict[router_startindex][1]
for r in range(router_startindex + 1, router_endindex + 1):
    if routersDict[r][0] < routerMinX:
        routerMinX = routersDict[r][0]
    if routersDict[r][0] > routerMaxiX:
        routerMaxiX = routersDict[r][0]
    if routersDict[r][1] < routerMinY:
        routerMinY = routersDict[r][1]
    if routersDict[r][1] > routerMaxiY:
        routerMaxiY = routersDict[r][1]

routersAreaX = routerMaxiX - routerMinX
routersAreaY = routerMaxiY - routerMinY
```

Figure 3.37 Shows the calculation of routers area

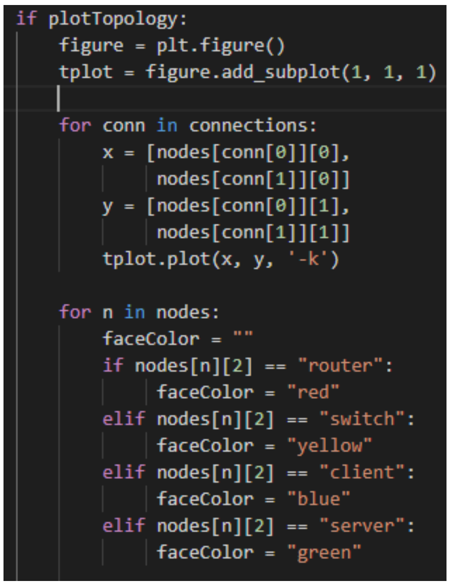
This section of the code is to calculate the area where routers are located by determining the minimum and maximum of X coordinate and the minimum and maximum of Y coordinate. This is vital for other type of nodes (switch,server, clients) so that it does not clump into the area of the routers.



```
with open(outputFile + "_" + str(fn) + ".json", "w") as ofile:
    json.dump(topology, ofile, sort_keys=True,
              indent=4, separators=(',', ': '))
```

Figure 3.38 Shows the writing of JSON output file

Used for writing out the JSON file form the topology dictionary data we have collected form earlier



```
if plotTopology:
    figure = plt.figure()
    tplot = figure.add_subplot(1, 1, 1)

    for conn in connections:
        x = [nodes[conn[0]][0],
             nodes[conn[1]][0]]
        y = [nodes[conn[0]][1],
             nodes[conn[1]][1]]
        tplot.plot(x, y, '-k')

    for n in nodes:
        faceColor = ""
        if nodes[n][2] == "router":
            faceColor = "red"
        elif nodes[n][2] == "switch":
            faceColor = "yellow"
        elif nodes[n][2] == "client":
            faceColor = "blue"
        elif nodes[n][2] == "server":
            faceColor = "green"
```

Figure 3.39 Shows the plotting of nodes and its connections

This section of the code is to mostly, plot all connections into a PDF in a solid black line style. Next we have to colour code the nodes depending on the type of devices it have to represent.

3.4 Implementation Issues and Challenges

The main issues faced while implementing this project is the fact that technique of traffic flow analysis which we choose to implement, the "Divide and Conquer", requires a huge amount of dataset for a wide variety devices. For example, in this project by only identifying different types of switches (Core,Aggregation,Access) would require a relatively large dataset. Therefore as the project progress further, we will have to start adding routers, servers, hubs and many other type network devices, the collecting of this large dataset will become very challenging. This is also true when the dataset is placed into ML to be process, the bigger the data the more time and resources is necessary to process them. In this project we are unable to obtain an excellent test and train accuracy, this could be the cause of the lack of data that is fed into the neural network.

Another challenge faced while doing this project, is the lack of research at this field, more specifically, in passive network topology discovery. Most of the industry standards today relies on active discovery, which indirectly cause a severe lack of funding on this research field. Companies like IBM, Cisco, WhatsUp Gold, and many more, which developed their own network discovery technique, is usually unknown to public as it is a company's secret. This adds more to the challenge of this project. Finally, when it comes to the machine learning aspect, a lot of studies is done to improve the Intrusion Detection Systems (IDS), studies done for machine learning to identify network topologies is quite lacking.

Moreover the challenges faced when developing the topology visualizer the learning curve when learning the python language and learning the Matplotlib. In part, it's due to the fact that I have no experience in python language and the usage of MATLAB prior to the start of the project. The learning curve can be daunting task initially but eventually I was able to overcome it. Next, during the development phase, it was tough to decided how complex I want the tool to be, highly complex might lead to incomplete visualizer. Therefore, I've decided to go with a simpler topology visualizer with being able to achieve the initial objectives set in this project.

3.5 Timeline

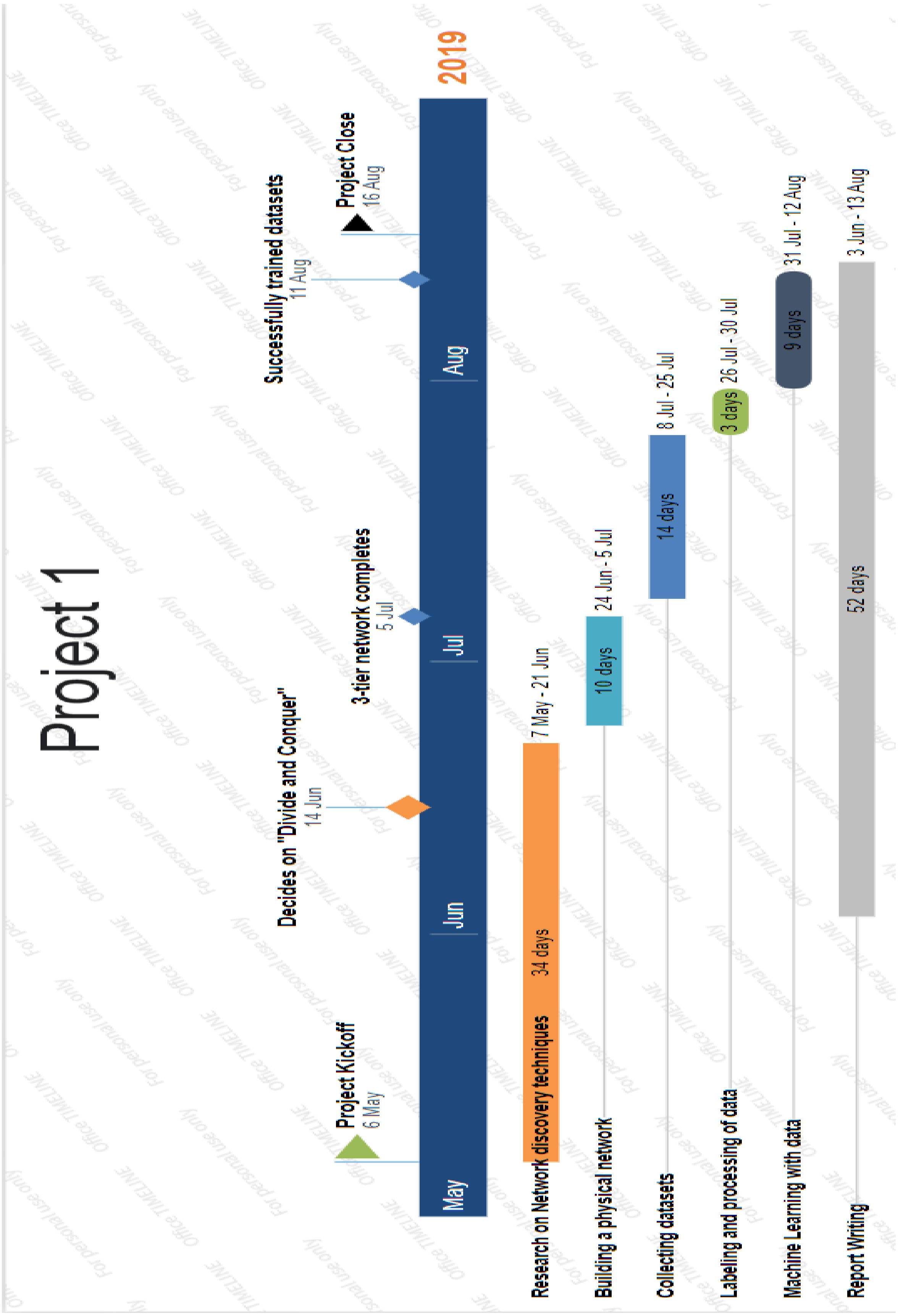


Figure 3.23 Shows the Gantt Chart for project 1

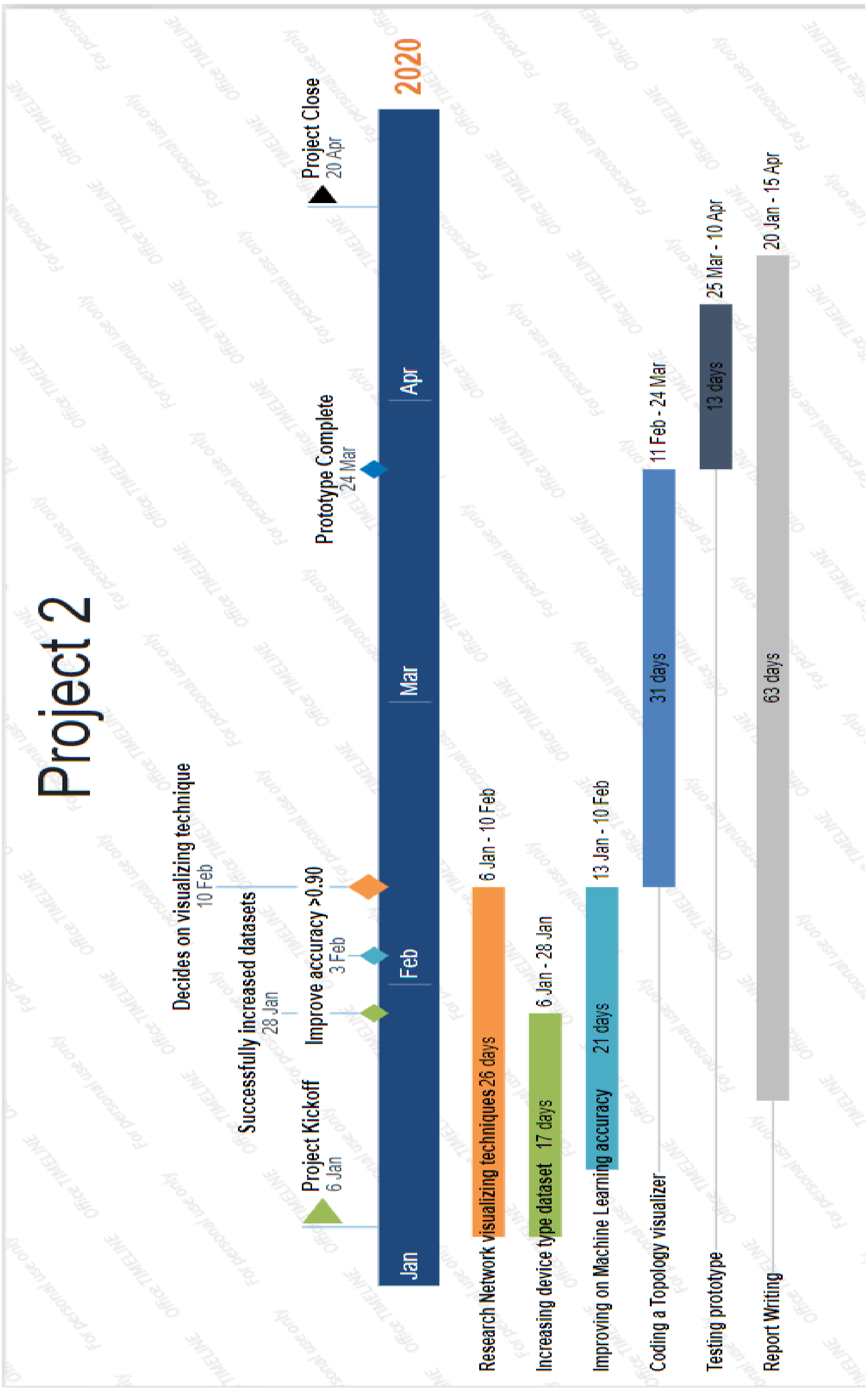


Figure 3.24 Shows the Gantt Chart for project 2

CHAPTER 4: Preliminary Work

At the beginning of the Project 1, the amount of time required to research essential techniques of passive network discovery was severely underestimated. Due to the lack of useful documentation, research papers and journals, most of my time is wasted on blind research. If I was able to successfully research a feasible technique earlier, this project would have went much smoother. Nevertheless, based on the end result of this project, I was still able to output a neural network machine learning accuracy of 75%, it does have room for improvement. Based on a research done by Kuldeep singh et al, they were able to output a results show an impressive classification accuracy of 91.875 % with the help of Bayes Net classifier. Moreover, a program completed of Chet Hosmer clearly shown that although passive network discovery does have its weakness, it is still feasible and will continue to evolve.

Although there exist some papers that mentions the limitations of this methods, like a research done by Ayodeji Akande et al on the "Limitations of Passively Mapping Logical Network Topologies". The research also stated that it is possible to acheive logical topology discovery from passively captured network traffic, the data collected can be used for mapping out legacy network. It also can be used to identify network misconfiguration, or possible intrusions.

Moreover, the method of "divide and conquer" proposed in this project, is sufficiently documented at other fields. But one research in particular, "An Efficient Divide and Conquer Approach for Big Data Analytics in Machine to Machine Communication " by Mukesh Nair et al, highlighted that it is possible to use this technique to be used for acquisition, aggregation and analyzing the data depending on context

During Project 2, research was done more toward the development of topology visualizer. The problem statement of most topology visualizer today seem to be scalability of the visualizer and user intuitiveness. For example, a research done by Yuki Kumata et al on the "Mesh Net Viewer: A Visualization System for Wireless Mesh Networks" who stated that as the number of node increase in a network topology, the network structure becomes complex with every increasing node. Therefore not a lot of visualizers are able to display superior scalability and still be able visualize the network topology intuitively.

CHAPTER 5: Results and Discussions

In this chapter, comparisons will be made between the results of the Topology Visualizer developed and the Topology Visualizer that were researched throughout this project. The Topology Visualizer which we will further make comparisons is *RFC8345-based Network Topology Visualizer by Hagiwara and LetsMapYourNetwork by Pramod Rana.* In terms of what to compare, we will be mainly focusing on 2 aspects **user intuitiveness** and **flexibility**

5.1 User Intuitiveness

When developing a topology visualizer, one of its biggest objective is to allow visualization of the network architecture that provides a complete understanding of the entire network. For a network engineer troubleshooting the network, it is imperative to have a thorough picture of all the working systems that are connected to your network. Therefore, having a visualizer with high user intuition (User-friendly) and easy to use is significant for the success of this project.

Beginning with my own Topology Visualizer, mainly the program is coded using the python, therefore, as a tool for visualization of the network topology, "*matplotlib.pyplot*" is generally used to interactively plot points and simple cases of programmatic plot generation.

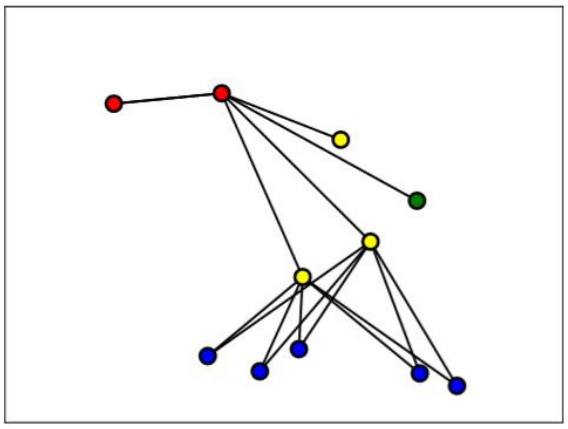
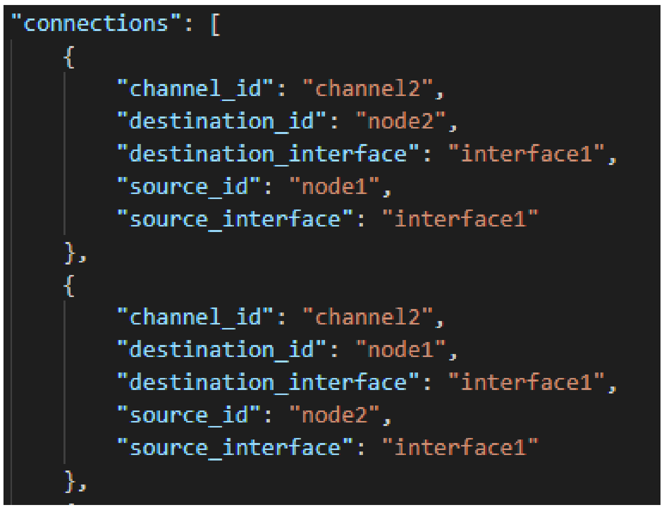


Figure 5.1 Shows the example of topology generated

As you can see, due to the limitation of *matplotlib.pyplot* we can only generally plot the nodes in colours to represent different types of devices. In our case, RED for routers, YELLOW for switches, GREEN for servers and finally BLUE for users/clients. This might be less user-friendly as users have to know beforehand the colour representation of each devices. Device connection wise, it's rather intuitive, users can easily observe the connections between the nodes. Which nodes are connected to which, how many connections are there, and the number of connection at each node, all of which is obviously shown.

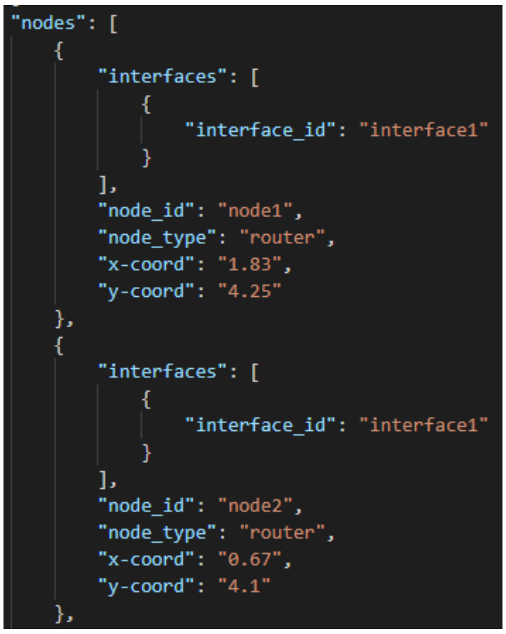


```
"connections": [
    {
        "channel_id": "channel2",
        "destination_id": "node2",
        "destination_interface": "interface1",
        "source_id": "node1",
        "source_interface": "interface1"
    },
    {
        "channel_id": "channel2",
        "destination_id": "node1",
        "destination_interface": "interface1",
        "source_id": "node2",
        "source_interface": "interface1"
    },
```

Figure 5.2 Shows outputted JSON file showing the connections

As part of the topology visualizer, along with a topology generated, a separate JSON file will be outputted to show in detail the information between the connections.

- **channel_id**: This will show which particular channel this connection is from
- **destination_id**: This will show which destination node the connection is connected
- **destination_interface**: Which destination interface is the connection connected
- **source_id**: The source/node ID of the connection begins
- **source_interface**: Which source interface is the connection connected



```
"nodes": [
    {
        "interfaces": [
            {
                "interface_id": "interface1"
            }
        ],
        "node_id": "node1",
        "node_type": "router",
        "x-coord": "1.83",
        "y-coord": "4.25"
    },
    {
        "interfaces": [
            {
                "interface_id": "interface1"
            }
        ],
        "node_id": "node2",
        "node_type": "router",
        "x-coord": "0.67",
        "y-coord": "4.1"
    },
```

Figure 5.3 Shows outputted JSON file showing the nodes and interfaces

Moreover, more information regarding the nodes is also generated as part of the JSON output file. Each node will have its own unique information presented to the user. The information is shown as below:

- **node_id**: This will show the node ID of the node
- **node_type**: Node type will detail the type of device the node belongs, Eg: Switch, Router, Server, Client
- **x-coord**: The X coordinate of the node
- **y-coord**: The Y coordinate of the node

Afterwards we can move on to *RFC8345-based Network Topology Visualizer (NetoViz) by Hagiwara*, Netoviz is a mainly javascript coded project by Hagiwara who is a network engineer. It is a tool which is used to visualize network topology data that based on RFC8345. Before we move forward, to summarize what is RFC8345, it is an abstract YANG data model that are used to represent topologies and network. The purpose of the data model is to work as a base model for topology and inventory data models. It is developed as a product by the Internet Engineering Task Force (IETF).

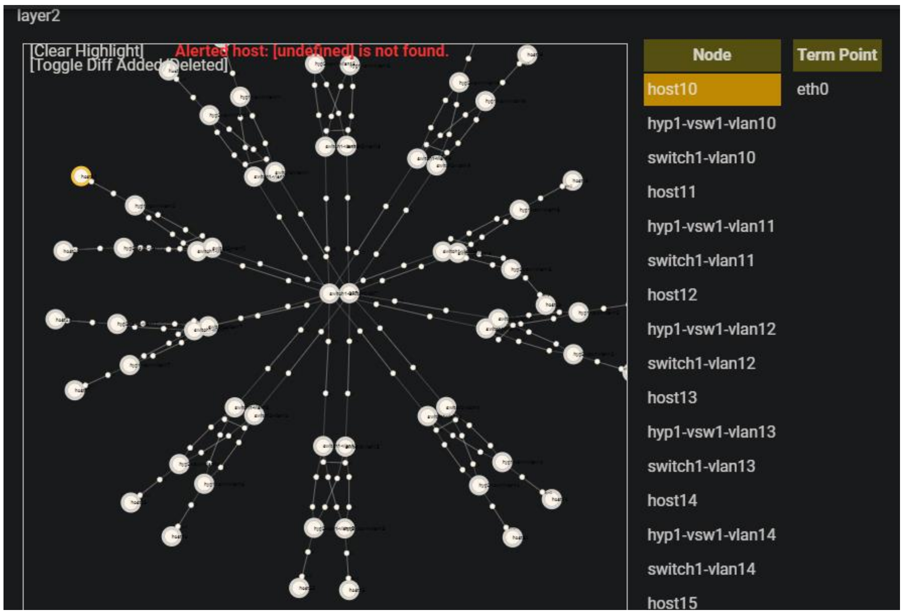


Figure 5.4 Shows an example of NetoViz generated topology

As you can see, the impressive topology generated by this tool is shown above. An example of a layer 2 topology is generated with each of the nodes highlighted at the right side. Instead of colour coding each type of node, upon closer inspection, the nodes are overlays with text indicating what node it belongs. For example in this case below, it is a VLAN 16 of physical switch 1.

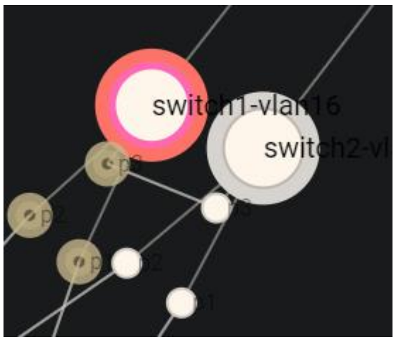
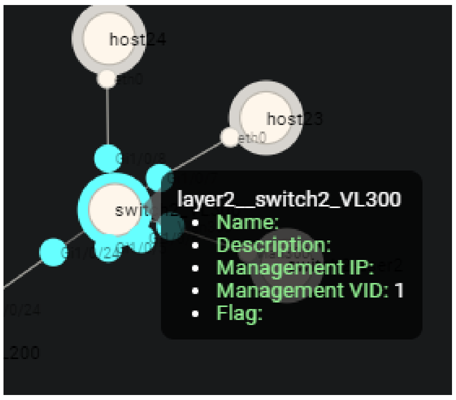


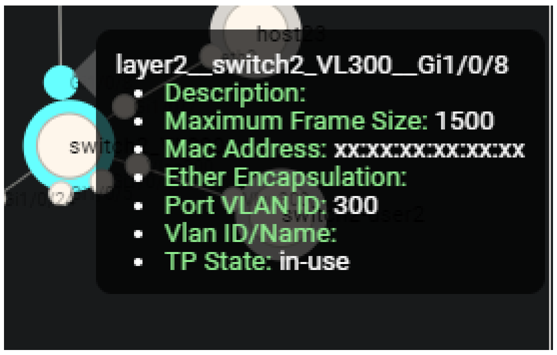
Figure 5.5 Shows a closer look at the nodes of NetoViz


layer2__switch2_VL300

- Name:
- Description:
- Management IP:
- Management VID: 1
- Flag:

Figure 5.6 Shows a closer look at more information of NetoViz

Instead of separately generating another JSON file for viewing in depth information of the topology, simply by hovering over the nodes more information such as Name, Description, Management IP, Management VID and Flag can be observed


layer2__switch2_VL300__Gi1/0/8

- Description:
- Maximum Frame Size: 1500
- Mac Address: xx:xx:xx:xx:xx:xx
- Ether Encapsulation:
- Port VLAN ID: 300
- Vlan ID/Name:
- TP State: in-use

Figure 5.7 Shows a closer look at more information of NetoViz

Apart from that, when hovering over the individual port of the nodes, even more detail information will pop out. Each detailing the network information of the port, such as Maximum Frame Size, the Mac address, encapsulation, port VLAN ID, VLAN name and the TP state. Also, the nodes generated is a not static, huge flexibility is allowed as the nodes can be click and dragged around the screen if it get too congested to be observed properly.

Finally, we are going to compare with *LetsMapYourNetwork(LMYN) by Pramod Rana*. LMYN is a tool that aims to deliver an easy to use interface for network engineer to map out their network in graphical form, just like the NetoViz the tools is mainly coded in javascript. The use of Neo4j bloom is what helped the plotting of nodes in this topology visualizer.

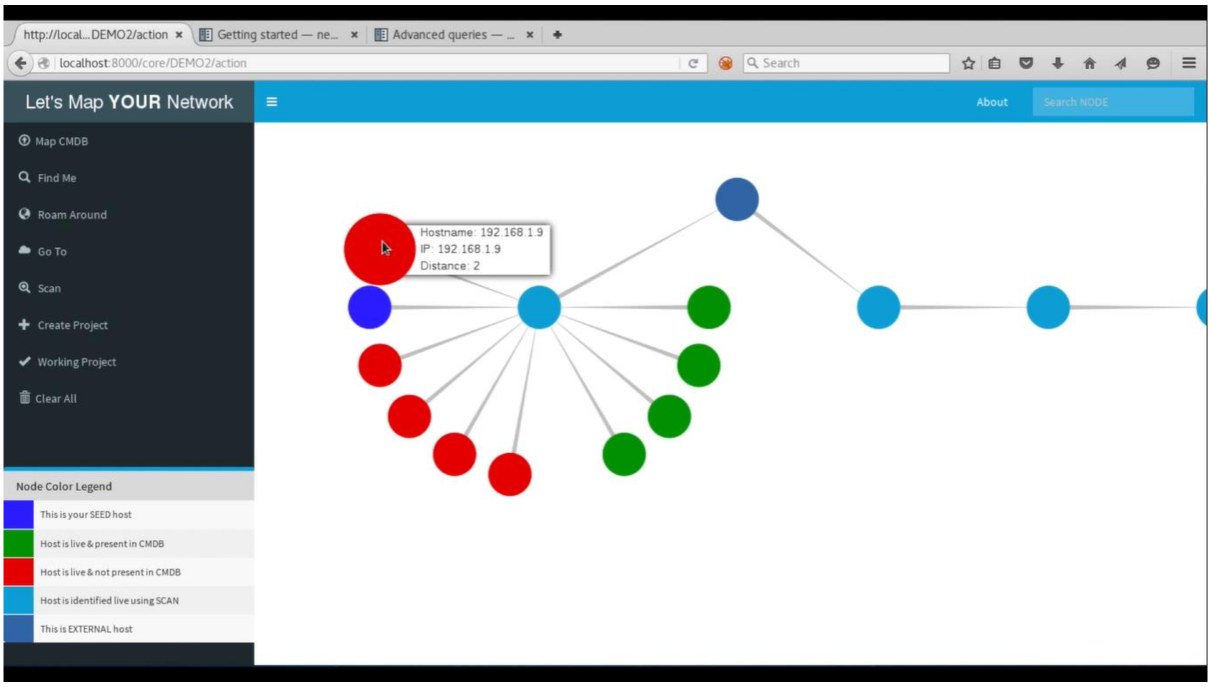
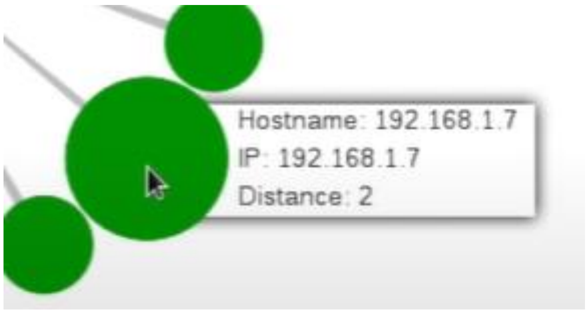


Figure 5.8 Shows an example topology generated by LMYN

At first glance, just like most Topology Visualizer, it will create nodes for user to observe with different colours representing different states of the nodes. According to the official documentation, the colours are represented at such:

- 🟦 This is SEED node
- 🟦 Node has external (public) IP
- 🟦 Node has internal (private) IP
- 🟩 Node present in CMDB file and is live
- 🟥 Node present in CMDB file but not live
- ▨ Node is down at this moment
- ⬜ Node is either Inline Router or VPC Peer

A colour node legend can also be observed on the bottom left corner for the user to reference in case it is forgotten.



Hostname: 192.168.1.7
IP: 192.168.1.7
Distance: 2

Figure 5.9 Shows a closer look at more information by LMYN

Similarly, upon hovering over the nodes, more details will be shown to the user, but rather than really complex information, simple information such as Hostname, IP address and the Hop distance of the nodes. Moreover, also similar to NetoViz the nodes and be dragged and readjusted to different locations on the screen to allow better visibility.

In conclusion, there are various similarities and differences when comparing my own Topology Visualizer with the other two Visualizer. To begin with the node generation/plotting, when it comes to the generation of the nodes, all 3 of the visualizer opt of a simple circle shape as the representation of a node rather than placing an image of the devices. For example, cisco packet tracer which shows a detailed image of the device. But when to comes to design, more obvious differences can be observed. For example in my own visualizer, I've used colour coding to separate the different type of devices that are present in the topology, meanwhile in LMYN colour coding I used for states of the devices rather than the types of devices. In NetoViz case, no colour is chosen to represent anything, it opted for text overlays to convey information to the user. Next, when it comes to the display of more detailed information, my visualizer mainly focuses toward the generation of a JSON file, where most of the network information resides. While the other 2 visualizers, have a hover feature that allows for better user intuition. But for the information displayed, I will point out that LMYN is rather lacking as to compare with my visualizer and NetoViz. NetoViz and my visualizer put much more emphasis on the network information that need to be displayed to users that are potentially network engineers or administrators. They require more detailed information such as the destination and source of the connection, which interface is the node connected, and the type of device the nodes belongs. The click and drag feature is also something that is rather unique as to compare with my visualizer, due to the nature of which my tool is created, the topologies can only be generated on PDF, therefore such features is impossible for my current state. But definitely needs to be considered due to its convenience to allow better visibility. Finally, the one the major similarity of all 3 visualizers is the ability to create different visually identifiable structure eg: hierarchical, leaf-spine and etc.

	My Topology Visualizer	NetoViz by Hagiwara	LetsMapYourNetwork by Pramod Rana
Node shape	• Circle in shape	• Circle in shape	• Circle in shape
Colour coding	• Yes	• No	• Yes
Colour Meaning	• Different colour for different devices	• No	• Different colour for different device states
Hover feature	• No	• Yes, with most information shown	• Yes, with more information shown
Display of more info	• Generate a separate JSON file • More detailed	• Hover over node to see most info • Most detailed	• Hover over node to see some info • Least detailed
Click and Drag node	• No, static node at a PDF file	• Yes	• Yes
Create Topological Structure	• Yes	• Yes	• Yes but limited

Table 5.1 Comparison of user intuitiveness

5.2 Scalability

When literature review was done for this project, what we have discovered is that, one of the major problems faced by most researcher and developers when creating a topology visualizer is the scalability of their visualizers. With network topology today consisting of more than hundreds of devices and nodes, the ability for the visualizer to seamlessly expand in scale, reconfiguring the network topology on demand, adding new flows or virtual networks without the need to upgrade the device the visualizer is running on is vital.

Beginning with my visualizer, one of the core advantages when it comes to plotting the nodes using matplot.pyplot is that node generation will be incredibly light weight when it comes to computational load. So when it comes to scalability of adding more devices into a topology, it will not be a big problem

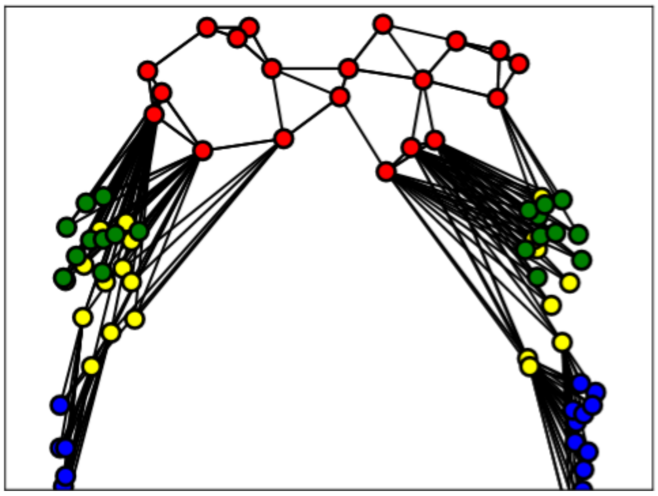


Figure 5.10 Shows a 80 nodes topology example

For example, in this case, we added 20 routers, 20 switches, 20 servers and 20 client nodes, a grand total of 80 nodes concurrently in one topology. With a medium spec laptop I used to run the topology visualizer tool, I was able to generate it within 2 seconds. Moreover the process of adding new nodes and connection, is as simple as changing the numbers in the configuration JSON files.

Next we will be discussing the scalability aspect of NetoViz, one of the main reasons this visualizer was able to create such as amazing user interface that are not only interactive but also user friendly is due to its usage of Vue, CSS and Ruby. But this also brings a huge disadvantage when it comes to the scalability when expanding the topology.

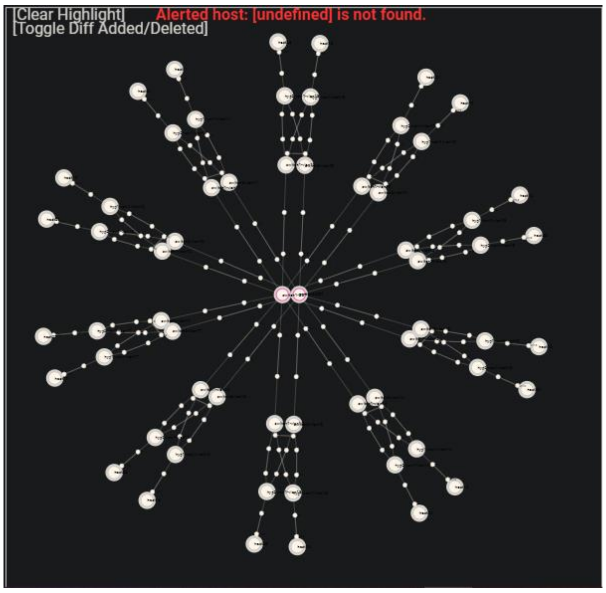
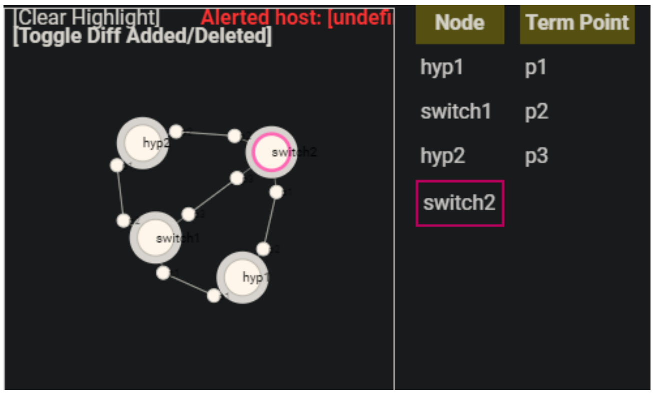

[Clear Highlight]
[Toggle Diff Added/Deleted]
Alerted host: [undefined] is not found.

Figure 5.11 Shows a 62 nodes topology example by NetoViz

For example, when I was running this topology on my google chrome browser, lagging issues begin to occur. The topology only have a total of 62 nodes, and it is enough to induce lag to my medium spec laptop, this show that the visualizer is highly dependent of the hardware specification of the user. The problem gets worse, when I attempt to drag the nodes around, it will occasionally freeze before moving again. With that said, it is still remarkable that it is able to perform well in smaller topologies. Like the one shown below



Node	Term Point
hyp1	p1
switch1	p2
hyp2	p3
switch2	

Figure 5.12 Shows a smaller topology example by NetoViz

One of the limitation stated in the official documentation of NetoViz, is that this visualizer only work with RFC 8345 YANG data models. Consequently, this means that when it comes to expanding the topology, it requires a huge amount of coding based on YANG language. It is not as simple as changing some figure in a JSON file, its requires the user to download YANG models repository on github, then construct the topology data using DSL (Domain Specific Language), converting JSON to XML, and finally the XML data will be used to visualize the topology

Finally, discussing about the scalability of Let's Map Your Network (LMYN), with reasons similar to NetoViz, this visualizer was able to create a user friendly UI that is interactive, is due to its usage of CSS and HTML. But what really set this software apart from the rest is the usage of Neo4j Bloom. So, what make Neo4j Bloom so special? It is a graph database management system that can visualize data which they claimed to have unlimited scalability. But to be fair, there are still lagging issues as well.

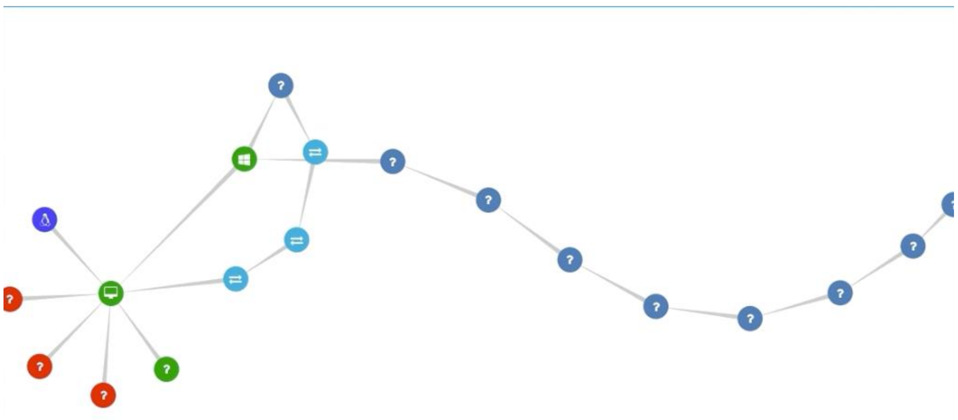


Figure 5.13 Shows a 19 nodes topology example by LMYN

For example, when running this topology, without doing much, the software is already slightly lagging, this can be partially caused by the number of prerequisite software that are required to run before the visualizer can work. The list of software that are required to run are as such, Neo4j server, RabbitMQ server, Oracle JDK 8, Microsoft Visual Studio C++ and Celery.

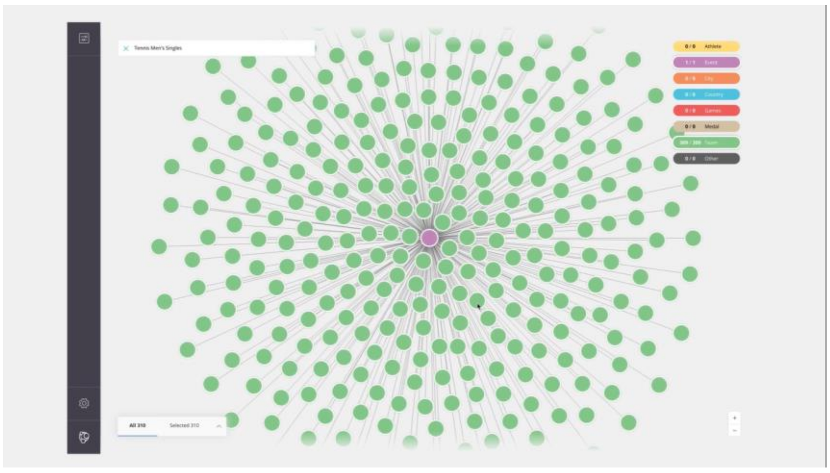


Figure 5.14 Shows a 309 nodes topology example of Neo4j Bloom

This is an example taken from Neo4j Bloom website, we can clearly see the capabilities of its "infinite" scalability when it comes to visualization. The total of 309 nodes are easily generated without much effort needed, as most of its processing is done at the Neo4j Server.

The topology data that is required by LMYN to generate it topology is a CMBD (configuration management database) file, this can be a main disadvantage if this visualizer in term of scalability. This is because, when we talk about CMBD, one of its biggest challenges face is maintenance. A fully populated CMDB requires all sorts of complex relationships and dependencies within, it a nightmare to alter any CI due to their associated dependencies and relationships. In order for a visualizer to be easily scalable, it must be able to constantly receive changes and new nodes.

To summarize, there are a lot of lesson can be learnt in terms of scalability from the other topology visualizers. Firstly, when developing a visualizer, there should be a balance between the simplicity and complexity of the tool, what I mean by that is, when developing my topology visualizer in order to achieve high scalability, I've made my visualizer tools as simple as possible in order to accommodate more nodes. But, this resulted in lack of certain features that allow for better user intuitiveness. While on the other side of the coin, NetoViz was developed as a very complex topology visualizer, but in-turn sacrificed the scalability of the visualizer. This was greatly achieved by LMYN by using a third party software in order to deal with the node visualization (Neo4j Bloom). By using such method, the visualizer is able to achieve great scalability without the need to rely too much on the hardware of the system running the visualizer.

Additionally, when discussing the type of configuration files that are required to visualizer a topology, there are lessons that we could learn from the other examples. When designing my topology visualizer, I've allowed flexibility and scalability by allowing JSON as the configuration input file. The reasoning behind this is because JSON is minimal, readable format for structuring data. Comparatively, when dealing with network data model, NetoViz opt for a more robust yet complicated standard (RFC 8345). But to be fair, this allows a more mature visualization of the network topology and its data. LMYN took a much different approach by using CMBD to build the network topology. This is partially due to the usage of the third-party software, Neo4j Bloom. Which according to its official documentation, required CMDB for the visualization to work.

	My Topology Visualizer	NetoViz by Hagiwara	LetsMapYourNetwork by Pramod Rana
Node scalability	• Yes, very scalable due to lightweight coding	• Yes, but limited to hardware spec of user	• Yes, very scalable due to 3rd party software (Neo4j Bloom)
Configuration files	• JSON file	• YANG files defined in RFC8345	• CMDB file
How the node were created	• Using matplot.pyplot	• Using of Vue, CSS and Ruby	• Using Neo4j Bloom

Table 5.2 Comparison of scalability

CHAPTER 6: Conclusion

The ability to reliably diagnose the bottleneck and hidden security problems of a network effortlessly and quickly is becoming increasingly significant in time-critical network. By solving this issue, a network engineer can efficiently fix a problem that is occurring in an organization, such solutions will not only minimize cost but also maximize profit. Common approach are today are usually more or less the based on active probing that utilizes protocols such as ICMP and SNMP. The methods will introduce a list of disadvantage, these include lengthy scan times for large networks, possible crashing services or hosts due to the increase in overhead, a lack of visibility into subnets that are secured by firewalls that block active scans, and many more. Hence it is important to further explore the possibilities of passive topology discovery, which seeks to solve these weakness presented by active probing.

First and foremost, early on to the project, a controlled network environment is successfully set up in a lab and packets are able to be generated for data collection. Next, are able to come up with a viable topology discovery technique to aid us in labelling traffic flow, which will enable us to discover possible devices in a network. Then, we are able to train our machine with an acceptable result to identify the type of connections between devices with the help of neural network.

Later into the project, based on the several problem statements of other researches we are able to recognise the challenges face by most network topology visualizer today which is scalability and user intuitiveness. With that said, we set to conquer these challenges by developing our own minimalistic network topology visualizer using python programming language and JSON. After that, we set out to make comparison with other 2 network topology visualizer that are researched online. I would say after the comparison made there are a number of things we have done right and some lessons which we can learn by taking them as an example to further improve upon

6.1 Future Direction.

Although, this report were not able to meet some of the expectation in the AI data discovery aspect, but it should give a better idea behind the effort and the potential of passive network topology discovery. In the future, hopefully, we will be able to improve on some of the aspects when dealing with this type project. Those improvements include, increasing the amount of dataset for different type of network devices, the improvement on the accuracy of the machine learning result and successfully classify them. On the network topology side of things, we look forward to improving the user intuitiveness by adding features such as, able to hover over nodes to show network information, click and drag feature to allow better visibility, and using a third party software in order to deal with the node visualization (Neo4j Bloom).

REFERENCE

Azzouni, Abdelhadi & Boutaba, Raouf & Thi Mai Trang, Nguyen & Pujolle, Guy. (2017). *sOFTDP: Secure and Efficient OpenFlow Topology Discovery Protocol.*

Azzouni, Abdelhadi & Boutaba, Raouf & Thi Mai Trang, Nguyen & Pujolle, Guy. (2017). *Limitations of OpenFlow Topology Discovery Protocol.*

Beal, V., 2011. *What Are Network Topologies?.* [Online] Available at: https://www.webopedia.com/author/Vangie-Beal [Accessed 07 04 2019].

Hosmer, C. and Kessler, G. (2015). *Passive Python network mapping*. 1st ed. Waltham: Syngress, pp.99-103.

J. Akande, A., Fidge, C. and Foo, E. (2017). Limitations of Passively Mapping Logical Network Topologies. *International Journal of Computer Network and Information Security*, [online] 9(2), pp.1-3. Available at: https://www.researchgate.net/publication/313393221_Limitations_of_Passively_Mapping_Logical_Network_Topologies [Accessed 4 Jun. 2019].

Nowicki, K., Malinowski, A., (2015) *Topology Discovery of Hierarchical Ethernet LANs without SNMP support* .

Pandey, S., Choi, M., Won, Y., & Won-Ki Hong, J. (2010). SNMP-based enterprise IP network topology discovery. *International Journal Of Network Management*, *21*(3), 169-184. doi: 10.1002/nem.756

Singh, K., Agrawal, S. & Sohi, B., 2013. *A Near Real-time IP Traffic Classification Using,* Punjab: MECS.

Varone, M., 2018. *What is Machine Learning? A definition.* [Online] Available at: https://www.expertsystem.com/machine-learning-definition/ [Accessed 12 07 2019].

W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2,* 3rd ed.(1999) Reading, MA: Addison-Wesley Longman, Inc.,

X. Li, "*A Method of Network Topology Visualization Based on SNMP*," (2011) First International Conference on Instrumentation, Measurement, Computer, Communication and Control, Beijing, 2011, pp. 245-248. doi: 10.1109/IMCCC.2011.70

Yu Yang, Peng Xia1,, Liang Huang, Quan Zhou, Yongjun Xu, Xiaowei Li SNAMP: A Multi-sniffer and Multi-view Visualization Platform for Wireless Sensor Networks. (2006). 2006 1ST IEEE Conference on Industrial Electronics and Applications, Industrial Electronics and Applications, 2006 1ST IEEE Conference On, 1. https://doi-org.libezp2.utar.edu.my/10.1109/ICIEA.2006.257222

Kumata, Y. and Koyama, A., 2013. Mesh Net Viewer: A Visualization System for Wireless Mesh Networks. *2013 Eighth International Conference on Broadband and Wireless Computing, Communication and Applications*,.

Rana, P., 2018. *Varchashva/Letsmapyournetwork*. [online] GitHub. Available at: <https://github.com/varchashva/LetsMapYourNetwork> [Accessed 21 April 2020].

Hagiwara, M., 2019. Corestate55/Netoviz. [online] GitHub. Available at: <https://github.com/corestate55/netoviz> [Accessed 21 April 2020].

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S1	**Study week no.: 2**
Student Name & ID: Foo Mun Yao 16ACB05444	
Supervisor: Dr. Aun Yichiet	
Project Title: NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

i. Updated FYP report structure, revised content for Chapter 1 & 2.
ii. Researching on method for data collection

2. WORK TO BE DONE

i. Determining the type of AI for the visualizer
ii. Build a physical network for testing data collection

3. PROBLEMS ENCOUNTERED

i. Hard to decide on what type of AI model to build
ii. Acquire resources for network topology building

4. SELF EVALUATION OF THE PROGRESS

Self-assigned tasks are completed within expected timeframe.

[illegible]
Supervisor's signature

[illegible]
Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S1	**Study week no.: 4**
Student Name & ID: Foo Mun Yao 16ACB05444	
Supervisor: Dr. Aun Yichiet	
Project Title: NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

iii. After extensive research, determine a method of data collection, Divide and Conquer

2. WORK TO BE DONE

iii. Begin collecting dataset for the neural network
iv. Build physical network topology
v. Determine the type of AI

3. PROBLEMS ENCOUNTERED

iii. Dataset is more difficult to collect that expected

4. SELF EVALUATION OF THE PROGRESS

Self-assigned tasks are completed within expected timeframe.

[illegible] [illegible]

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S1	**Study week no.: 6**
Student Name & ID: Foo Mun Yao 16ACB05444	
Supervisor: Dr. Aun Yichiet	
Project Title: NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY	

1. WORK DONE
[Please write the details of the work done in the last fortnight.]

iv. Decide on the AI model to work on, neural network
v. Built simple a Three-layered Hierarchical Model

2. WORK TO BE DONE

vi. Begin collecting dataset for the neural network
vii. Begin working on the AI model

3. PROBLEMS ENCOUNTERED

iv. Deciding on the type of data to be generated in the network topology

4. SELF EVALUATION OF THE PROGRESS

Self-assigned tasks are completed within expected timeframe.

[illegible] [illegible]

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S1	**Study week no.: 8**
Student Name & ID: Foo Mun Yao 16ACB05444	
Supervisor: Dr. Aun Yichiet	
Project Title: NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY	

1. WORK DONE
[Please write the details of the work done in the last fortnight.]

viii. Developed an AI model, ready for training phase
ix. Dataset Collected and preprocessed data from selected dataset.
x. Decided on the packet generator, WANKILLER

2. WORK TO BE DONE

xi. Determine training hyperparameters and other configurations
xii. Begin training phase of AI model

3. PROBLEMS ENCOUNTERED

v. Learning curve of AI without prior knowledge
vi. Generating the right packet based the network topology

4. SELF EVALUATION OF THE PROGRESS

Self-assigned tasks are completed within expected timeframe.

[illegible] [illegible]

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S1	**Study week no.: 10**
Student Name & ID: Foo Mun Yao 16ACB05444	
Supervisor: Dr. Aun Yichiet	
Project Title: NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY	

1. WORK DONE
[Please write the details of the work done in the last fortnight.]

xiii. Able to train the model with acceptable accuracy

2. WORK TO BE DONE

xiv. Further improve on the accuracy of the model
xv. Splitting the dataset into different sets

3. PROBLEMS ENCOUNTERED

vii. During the preprocessing of data, might have led to less accurate model

4. SELF EVALUATION OF THE PROGRESS

Self-assigned tasks are completed within expected timeframe.

[illegible]
Supervisor's signature

[illegible]
Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S1	**Study week no.: 12**
Student Name & ID: Foo Mun Yao 16ACB05444	
Supervisor: Dr. Aun Yichiet	
Project Title: NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY	

1. WORK DONE
[Please write the details of the work done in the last fortnight.]

xvi. Improvements were made on the AI model

2. WORK TO BE DONE

xvii. Begin writing the report

3. PROBLEMS ENCOUNTERED

viii. Insufficient in the variety of dataset might led to the accuracy of the result

4. SELF EVALUATION OF THE PROGRESS

Self-assigned tasks are completed within expected timeframe.

[illegible]
Supervisor's signature

[illegible]
Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S3	**Study week no.: 2**
Student Name & ID: Foo Mun Yao 16ACB05444	
Supervisor: Dr. Aun Yichiet	
Project Title: NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

xviii. Updated FYP report structure, revised content for Chapter 1 & 2

2. WORK TO BE DONE

i. Research Network visualizing techniques
ii. Increase the device type datasets
iii. Improve machine learning accuracy

3. PROBLEMS ENCOUNTERED

ix. Collecting real life dataset might be a big challenge considering the security issues

4. SELF EVALUATION OF THE PROGRESS

Self-assigned tasks are completed within expected timeframe.

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S3	**Study week no.: 4**
Student Name & ID: Foo Mun Yao 16ACB05444	
Supervisor: Dr. Aun Yichiet	
Project Title: NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

iv. Research done for Network visualizing techniques

2. WORK TO BE DONE

i. Deciding on the language for the visualizer
ii. Research on the problem statement of most visualizers today
iii. Increasing the devices dataset
iv. Use GNS3 to visualizer a core network

3. PROBLEMS ENCOUNTERED

x. Create a simulation for network topology

4. SELF EVALUATION OF THE PROGRESS

Self-assigned tasks are completed within expected timeframe.

[illegible]

Supervisor's signature

[illegible]

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S3	**Study week no.: 6**
Student Name & ID: Foo Mun Yao 16ACB05444	
Supervisor: Dr. Aun Yichiet	
Project Title: NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY	

1. WORK DONE
[Please write the details of the work done in the last fortnight.]

v. Decide to develop topology visualizer with python and JSON for lightweight design
vi. Researched on the problem statement of most visualizers today
vii.

2. WORK TO BE DONE

v. Begin working the visualizer
vi. Increasing the devices dataset

3. PROBLEMS ENCOUNTERED

xi. Create a simulation for network topology using GNS3 to visualizer a core network as it is too resource heavy
xii.

4. SELF EVALUATION OF THE PROGRESS

Self-assigned tasks are completed within expected timeframe.

[illegible]
Supervisor's signature

[illegible]
Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S3	**Study week no.: 8**
Student Name & ID: Foo Mun Yao 16ACB05444	
Supervisor: Dr. Aun Yichiet	
Project Title: NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY	

1. WORK DONE
[Please write the details of the work done in the last fortnight.]

viii. Improvement on topology visualizer

2. WORK TO BE DONE

vii. More refining on the topology visualizer
viii. Comparision between my visualizer with other visualizer

3. PROBLEMS ENCOUNTERED

xiii. Learning curve of developing the visualizer due to the fact that I've never learn python prior to this project

4. SELF EVALUATION OF THE PROGRESS

Self-assigned tasks are completed within expected timeframe.

[illegible]
Supervisor's signature

[illegible]
Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S3	**Study week no.: 10**
Student Name & ID: Foo Mun Yao 16ACB05444	
Supervisor: Dr. Aun Yichiet	
Project Title: NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY	

1. WORK DONE
[Please write the details of the work done in the last fortnight.]

ix. In-depth comparision between my visualizer with other visualizer

2. WORK TO BE DONE

x. Begin writing report

3. PROBLEMS ENCOUNTERED

xiv. MCO have restricted some of the resources needed for the project

4. SELF EVALUATION OF THE PROGRESS

Self-assigned tasks are completed within expected timeframe.

[illegible] [illegible]

Supervisor's signature Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project I / Project II)

Trimester, Year: Y3S3	**Study week no.: 12**
Student Name & ID: Foo Mun Yao 16ACB05444	
Supervisor: Dr. Aun Yichiet	
Project Title: NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY	

1. WORK DONE
[Please write the details of the work done in the last fortnight.]

xi. Finalizing the topology visualizer

2. WORK TO BE DONE

xii. Complete the report

3. PROBLEMS ENCOUNTERED

xv. MCO have restricted some of the resources needed for the project

4. SELF EVALUATION OF THE PROGRESS

Self-assigned tasks are completed within expected timeframe.

[illegible] [illegible]

Supervisor's signature Student's signature

UNIVERSITI TUNKU ABDUL RAHMAN

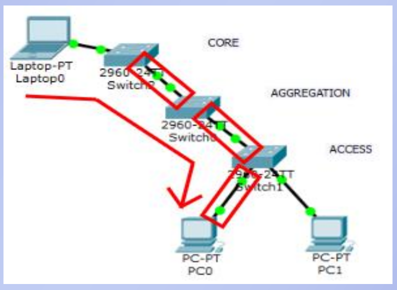
Faculty of Information Technology (Hons) Communication and Networking

NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY

Foo Mun Yao and Dr Aun Yen Chiet

ABSTRACT

Automatic discovery of network topology is very important and has practical significance due to the fact that network is becoming more and more complex. Technology of computer network is changing and developing rapidly, the importance of scientific and effective network management is becoming increasingly significant. This project is a Network Topology Visualizer for academic purpose. The aim of this paper is improve upon methods and techniques to discover physical and logical topology, then ultimately, incorporating AI to visualize the topology. It will provide readers with the methodology, concept and design Network topology discovery. "Divide and conquer" method will be utilised to segment a physical network for the sake of easing the process of data collection. Then, with the help of machine learning technique known as artificial neural network will be used to help identifying the devices in a topology. Afterward, a visualizer tool is created in python, where a JSON configuration file is used to generate the topology of the network to a PDF file. Discussion and the results of the visualizer are made to compare with 2 other network topology visualizer researched online.

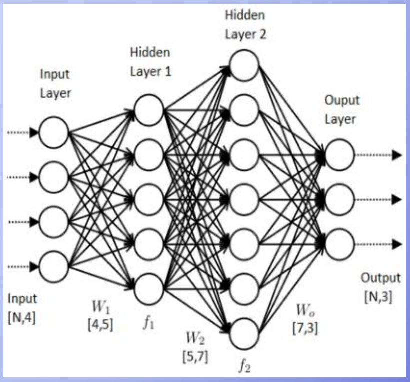


OBJECTIVES

- To develop a method of Topology Discovery just by using the Network Traffic Flow Data. Instead of creating Protocols like most known topology discovery method, we want to discover a network topology just from Traffic Flow Analysis.

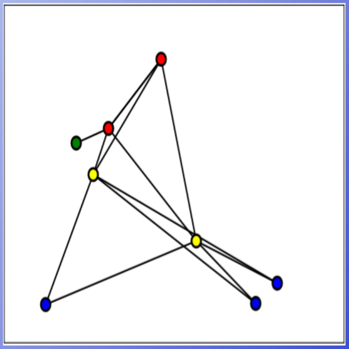
METHODOLOGY

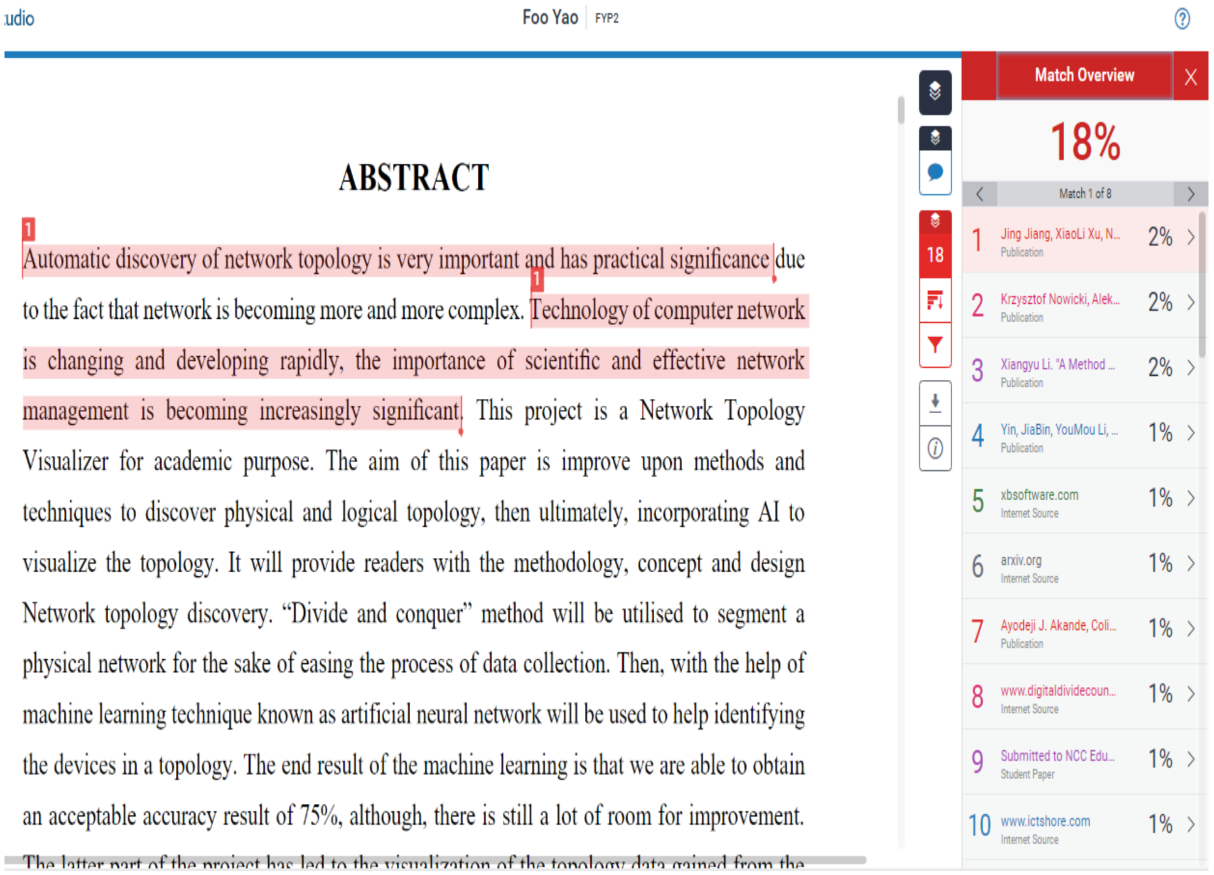
"Divide and conquer" method will be utilised to segment a physical network for the sake of easing the process of data collection. Then, with the help of machine learning technique known as artificial neural network will be used to help identifying the devices in a topology. As for the artificial neural network, we will utilize "supervised learning" in order to train our model. This will aid us in getting the best possible result. After that, a topology visualizer is developed using Python and JSON programming language. This python tool is used to visualize a number random network topologies which include, servers, clients, switches and routers.



RESULTS & DISCUSSION

In summary, the first half of this project has successfully accomplished the objective that is set in Project 1. First and foremost, early on to the project, a controlled network environment is successfully set up in a lab and packets are able to be generated for data collection. Next, are able to come up with a viable topology discovery technique to aid us in labelling traffic flow, which will enable us to discover possible devices in a network. Then, we are able to train our machine with an acceptable result of 75% to identify the type of connections between devices with the help of neural network. Later into the project, based on the several problem statements of other researches we are able to recognise the challenges face by most network topology visualizer today which is scalability and user intuitiveness. Comparisons between our topology visualizer and others were made based on these aspects.

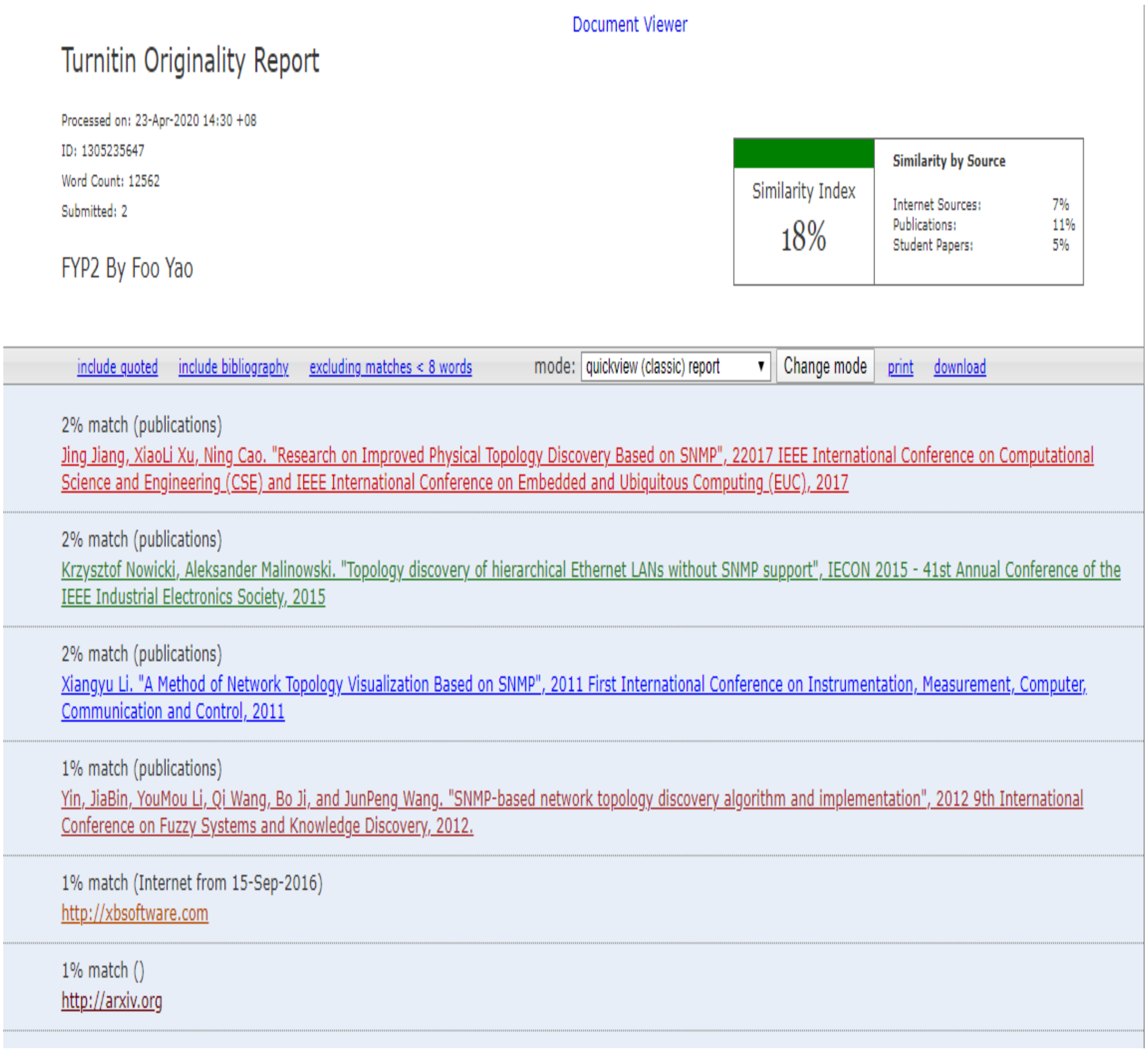




ABSTRACT

Automatic discovery of network topology is very important and has practical significance due to the fact that network is becoming more and more complex. Technology of computer network is changing and developing rapidly, the importance of scientific and effective network management is becoming increasingly significant. This project is a Network Topology Visualizer for academic purpose. The aim of this paper is improve upon methods and techniques to discover physical and logical topology, then ultimately, incorporating AI to visualize the topology. It will provide readers with the methodology, concept and design Network topology discovery. "Divide and conquer" method will be utilised to segment a physical network for the sake of easing the process of data collection. Then, with the help of machine learning technique known as artificial neural network will be used to help identifying the devices in a topology. The end result of the machine learning is that we are able to obtain an acceptable accuracy result of 75%, although, there is still a lot of room for improvement. The latter part of the project has led to the visualization of the topology data gained from the

1	Jing Jiang, XiaoLi Xu, N... Publication	2%
2	Krzysztof Nowicki, Alek... Publication	2%
3	Xiangyu Li. "A Method ... Publication	2%
4	Yin, JiaBin, YouMou Li, ... Publication	1%
5	xbsoftware.com Internet Source	1%
6	arxiv.org Internet Source	1%
7	Ayodeji J. Akande, Coli... Publication	1%
8	www.digitaldividecoun... Internet Source	1%
9	Submitted to NCC Edu... Student Paper	1%
10	www.ictshore.com Internet Source	1%


Document Viewer

Turnitin Originality Report

Processed on: 23-Apr-2020 14:30 +08

ID: 1305235647

Word Count: 12562

Submitted: 2

FYP2 By Foo Yao

Similarity Index	Similarity by Source	
18%	Internet Sources:	7%
	Publications:	11%
	Student Papers:	5%

include quoted include bibliography excluding matches < 8 words mode: quickview (classic) report Change mode print download

2% match (publications)
Jing Jiang, XiaoLi Xu, Ning Cao. "Research on Improved Physical Topology Discovery Based on SNMP", 22017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), 2017

2% match (publications)
Krzysztof Nowicki, Aleksander Malinowski. "Topology discovery of hierarchical Ethernet LANs without SNMP support", IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society, 2015

2% match (publications)
Xiangyu Li. "A Method of Network Topology Visualization Based on SNMP", 2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control, 2011

1% match (publications)
Yin, JiaBin, YouMou Li, Qi Wang, Bo Ji, and JunPeng Wang. "SNMP-based network topology discovery algorithm and implementation", 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery, 2012.

1% match (Internet from 15-Sep-2016)
http://xbsoftware.com

1% match ()
http://arxiv.org

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of	Foo Mun Yao
ID Number(s)	1605444
Programme / Course	Communication and Networking
Title of Final Year Project	NETWORK TOPOLOGY VISUALIZOR USING SERVER FLOW AFFINITY

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 18 **%** **Similarity by source** Internet Sources: 7 % Publications: 11 % Student Papers: 5 %	
Number of individual sources listed of more than 3% similarity: 0	

Parameters of originality required and limits approved by UTAR are as Follows:

(i) Overall similarity index is 20% and below, and
(ii) Matching of individual sources listed must be less than 3% each, and
(iii) Matching texts in continuous block must not exceed 8 words

Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

[illegible]

Signature of Supervisor	Signature of Co-Supervisor
Name: Dr Aun Yichiet	Name:
Date: 22/04/2020	Date:

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	1605444
Student Name	FOO MUN YAO
Supervisor Name	DR AUN YICHIET

TICK (√)	**DOCUMENT ITEMS** Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
✓	Front Cover
✓	Signed Report Status Declaration Form
✓	Title Page
✓	Signed form of the Declaration of Originality
✓	Acknowledgement
✓	Abstract
✓	Table of Contents
✓	List of Figures (if applicable)
✓	List of Tables (if applicable)
	List of Symbols (if applicable)
✓	List of Abbreviations (if applicable)
✓	Chapters / Content
✓	Bibliography (or References)
✓	All references in bibliography are cited in the thesis, especially in the chapter of literature review
	Appendices (if applicable)
✓	Poster
✓	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report. [illegible] (Signature of Student) Date: 22/04/2020	Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction. [illegible] (Signature of Supervisor) Date: Date: 22/04/2020