MiniPod for Network Automation

Mubeen Abdul

Supervised by Adam Sampson

Final Year Dissertation

BSC Computer Science

Declaration

I, Mubeen Abdul confirm that this work submitted for assessment is my own and is expressed.

in my own words. Any uses made within it of the works of other authors in any form (e.g.,

ideas, equations, figures, text, tables, programs) are properly acknowledged at any point

of their use. A list of the references employed is included.

Signed: Mubeen Abdul

Date: 26/04/2023

Abstract

The aim of this project is to develop a MiniPod, an automation tool that can automate the configuration process for devices in networks. The MiniPod will be able to configure various network parameters, such as setting up interfaces, configuring IP addresses to interfaces, configuring DHCP, OSPF, VLANs, and ACLs, as well as implementing security measures.

In the background section, this project will provide a detailed explanation of how networks work, the different network protocols, and ways to make networks secure. Furthermore, this project will discuss the existing automation technologies on the market, including ACI, NSX, Ansible, and Puppet.

The project requirements will include the types of configurations the MiniPod can automate, along with a couple of use cases. The MiniPod's functionality will be self-evaluated, and a comparison between the time taken for manual configuration and the time taken for the MiniPod to do the same tasks will be examined. The evaluation will be performed using a spine and leaf 2-tier network topology.

Table of Content

1. Introduction

1.1 Brief Introduction

Many businesses, service providers, and datacentres require large networks to function and communicate effectively. Manually setting up these devices for large networks is extremely time-consuming and requires a lot of architectural planning to deploy a secure and efficient network. Configuring, managing, and operating physical and virtual devices inside a network are all tasks that can be automated. This is known as network automation. As repetitive tasks are handled and maintained automatically, and normal network operations and functions are automated, network service availability increases. Automation also reduces human error and lowers businesses' expenses on operating networks.

In addition to these benefits, configuring a network from a specification ensures that the network is set up according to a pre-defined standard. This helps to ensure consistency and reduce the possibility of user misconfigurations. By following a specification, the network can be deployed more quickly and efficiently. Additionally, using a specification allows for easier maintenance and troubleshooting in the future, as the network is built according to a documented standard.

1.2 Motivation

Although network automation seems an easy solution and a no-brainer for every network to have, the biggest factor for it not happening is, there are too many legacy devices being used. Legacy devices are devices that are no longer in production and these devices don’t support automation tools that are currently out for new devices. Although upgrading to new devices that support automation would be ideal, it would be too expensive to change and unfeasible for many to upgrade large scale networks without shutting down their workloads.

My project aims to bridge the gap between legacy devices and automation by building a MiniPod that will automate device setups.

1.3 Aims & Objectives

The aim of this project is to create an automation MiniPod that will automate configurations for devices in a network quickly and efficiently.

The main objectives for this project will be:

- To research the configurations of setting up routers and switches in a network
- To develop an automation tool that will automate setting up legacy devices in a 2-tier Spine and leaf network topology.
- Compare the speed of this tool against manually configuring devices in a network.

2. Background

It is essential to understand how networks work before attempting to create an automation MiniPod. This section will investigate a typical network architecture and explore the different attributes needed to build an efficient and secure network.

2.1 How networks work

In its simplest form, a computer network consists of two or more computers that are linked to each other wirelessly or by a cable and can communicate with each other to share, transmit and exchange data. Networks that cover small geographic locations such as homes and offices are known as local area networks (LAN) and networks that cover large geographical areas are known as wide area networks (WAN). These can contain up to millions of devices across the world.[1]

2.2 Traditional Network Architecture

Traditional networks come in a more rigid form. They come as a hierarchical model. In the 2000s, it was a three-tier model where networks had a core layer, distribution layer and an access layer[6]. In figure 1, it shows a simplified diagram of what a three-tier model would look like.

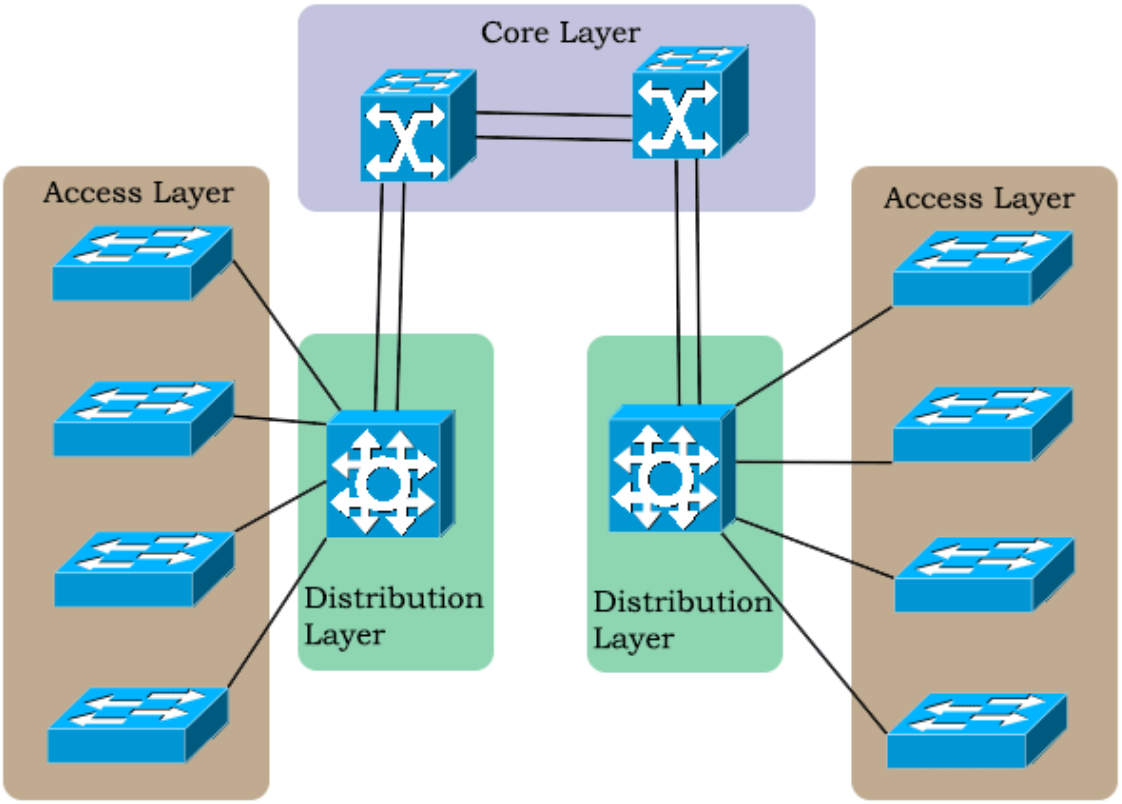

Figure 1. Diagram of an example of a three-tier model without redundancy [8]

Core layer is where the routing devices or layer 3 switches are kept. This layer is like the brain of the network and every routing decision or routing to a different network is done through this layer. The devices at this layer are connected with high-speed connections, such as Gigabit Ethernet link. The core layer devices are also connected to the distribution layer switches. The distribution layer switches are not as fast as core layer switches; however, they are connected to other access layer switches. These switches are not connected to each other and to connect or communicate with other switches or networks, they must go through the core switches which will then forward to other networks and switches. Lastly, we have the access layer and like the name suggests, these switches provide access to devices such as PCs, servers, printers, and other end devices to the network. The purpose of this layer is only to provide access to the end devices. At the access layer, this is where all the different services and security is configured. [5]

Around the 2010-decade, new deployment came around which was the 2-tier or spine and leaf model[6].

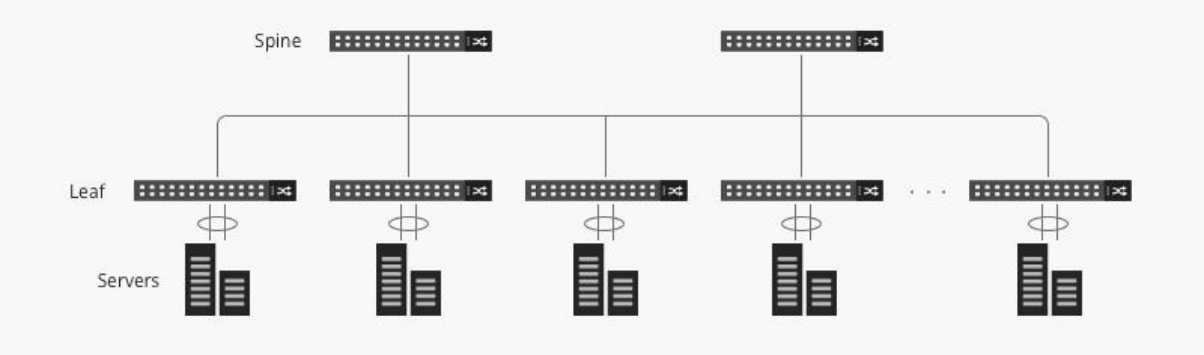

Figure 2 Diagram of a spine-leaf architecture [26]

This was where instead of having three layers, only two layers were used – the access and distribution/core as shown in figure 2. The access layer is named leaf, and the other layers are named spine. With only having two layers, the spine and leaf switches are interconnected with fast links, however, the leaf switches do not connect with each other [9]. This deployment started becoming popular as it aids low latency traffic by always having the same number of hops away from its next destination.[7] Hops in networking refer to the movement of data from one network node to another.

2.3 Internet Protocols

For computers to communicate with each other in a network, there are network protocols that they follow. These protocols operate at different layers of the OSI model. The most common protocol used in networks at the network layer is called IP (Internet Protocol). Each device in a network is assigned a unique internet protocol address (IP). Depending on the network, this address can either be an IPv4 address or an IPv6 address. IPv4 address is a 32-bit binary address and is the most common internet protocol to be used in all networks. Although IPv4 is the most used internet protocol, it has a limited number of unique addresses, with only 4.3 billion available. Due to the rapid growth of the internet, all possible IPv4 addresses were exhausted in 2019[3]. In response to this, IPv6 was introduced as a new internet protocol with a much larger address space of up to 340

trillion trillion trillion unique addresses. Currently most commercial and home networks still haven't made the transition to fully support and utilise using IPv6 and still rely upon IPv4 addresses to communicate[2].

2.4 Discovery Protocols

Link Layer Discovery Protocol (LLDP) is a vendor-neutral networking protocol that operates at the data link layer and facilitates the discovery of information about neighbouring devices. LLDP packets are sent every 30 seconds by default and include device type, port ID, and system name. LLDP is supported by a wide range of networking vendors and can be used on most networking devices.

Given that Cisco is the most widely used networking equipment vendor[34], this project will be examining Cisco-specific technologies, including the Cisco Discovery Protocol (CDP). As I will be using Cisco devices in my network topology, CDP is particularly relevant to this project. CDP operates at the data link layer and is designed specifically to provide neighbouring information for Cisco devices. CDP packets contain more detailed information compared to LLDP, including device type, model number, IP address, and the interfaces that connect them. CDP packets are sent out every 60 seconds by default. [32]

2.5 DHCP

In networks, the IP address can be manually configured on each device or using Dynamic Host Configuration Protocol (DHCP), it can be allocated an address automatically. Usually for smaller networks, DHCP is setup on a router and for larger networks there is almost always a dedicated DHCP server. DHCP eliminates errors caused by typing mistakes or the assignment of the same address to two end devices when IP addresses are manually configured by a user. When a newly connected PC or end device is connected to a network topology that has a DHCP server, it must go through four steps to obtain an IP address: Discover, Offer, Request, and Acknowledge. (DORA). In order to locate the DHCP server, the PC will first transmit a Discover message. This communication will be received by the DHCP server, which will then provide the PC with an IP address to use. Finally,

the DHCP server will recognise the PC's approval by sending a message confirming that it accepts the provided IP address. [4]

2.6 Switches

A network switch allows different devices to communicate with each other within the same network. Switches usually have many ports and are essential for large networks that contain many end devices. There are 2 types of switches, a layer 2 switch (data link layer) forwards data by using its media access control (MAC) address table to find the device it needs to send to. A MAC address, assigned by manufacturers, is a unique hardware address located at the data link layer that identifies network interface controllers (NICs) and is globally unique[35]. When a computer tries to communicate with another device in the same network for the first time, the switch will look at the destination MAC address of the frame. Since the switch does not recognise this MAC address, the computer will send out an Address Resolution Protocol (ARP) request. The switch will broadcast to all the devices in the network and the device that is destined to receive this information will reply to this request with its MAC address. Through this process, the switch will learn about this devices MAC address and add this to its MAC address table.[27]

Layer 3 switches can do all the things layer 2 switches do but also contain an IP routing table and can have routing functionalities such as static routing and dynamic routing [28]. Routing will be explained later.

2.7 Routers

For a computer to send and receive information to a different device that is not within its network, it will need to be connected to a router. Routers allow communications to be established and data to be transmitted between different networks. Routers will analyse the data that is incoming by looking at which network address it is destined for and forward this data using its routing table into that network. [11]

2.8 Routing Protocols

When there are multiple routers connected to each other or for routers to send data to different networks, different routing protocols can be used to find the optimal path for data to reach its destination. The most basic approach to setting up routing is static routing. For static routing the routing entries must be manually configured by a network administrator, and this involves writing out all the neighbouring networks connected to the router. Static routing is the easiest to implement, however, if any changes were to happen in the network, like a new device being added, all the devices would have to be reconfigured to take this change into account and this can be a very time-consuming process, especially for large networks.[12]

For large networks, dynamic routing protocols are used. Some of the routing protocols used are OSPF and EIGRP.

Open Shortest Path First (OSPF) is a linked state routing protocol which computes the best path in a network by calculating the shortest path from the source network to the destination network. It does this by finding out all the neighbouring routers by sending hello packets out each OSPF enabled interfaces. Once a neighbouring router acknowledges the hello and sends one back, OSPF will store that neighbouring routers information in a table which is shared and then each router will run a Dijkstra's algorithm[36] to identify the shortest path between routers in the same network topology and give it a cost value. OSPF will store the best routes in a routing table [10]. OSPF will send a hello packet every 10 seconds between routers, therefore, if any changes were to occur in the network it would update the routing table quickly[13]. One of the main advantages of using OSPF is that it's not a Cisco proprietary tool, allowing it to be used for any device.

Enhanced Interior Gateway Routing Protocol (EIGRP) is a distance-vector routing protocol. Like OSPF, EIGRP will send hello packets to learn the information about neighbouring routers and save that information in the topology table. When identifying neighbouring routers, it will learn about the reported distance between the two routers and adds this to the routing table. Using the reported

distance values, EIGRP will examine all the accessible paths to the destination and use the best path calculated[14]. For EIGRP, there are other factors that will affect in determining the best path. These factors are the bandwidth, reliability in the connection, the delay, and the load on the link [10].

One of the disadvantages of using EIGRP is this is a proprietary protocol designed by Cisco, meaning it doesn't support devices from other vendors[10]. This results in OSPF being preferred over EIGRP for routing in networks.

Border Gateway Protocol (BGP) is another routing protocol which is mainly used to connect networks to the internet. It is an exterior gateway protocol which directs packets between groups of networks that have a unified routing policy (autonomous system). BGP is usually configured on edge routers which are routers that are located at the network boundary and allows the internal network to connect to external networks [18]. Although BGP can be implemented on LANs, it is not desirable due to its convergence time, if a router shuts down in a network, the time it takes for BGP to restore its routing will take 3 minutes as opposed to EIGRP and OSPF having very low convergence times. [17]

2.9 VLAN

Virtual LAN is a network that is created virtually inside a network. VLANs are configured on the ports of switches and only allows data to be sent between the same VLANs[15]. Creating VLANs for large networks are very useful, as it keeps information between each network separate. For example, in a university you would not want communication from a computer science department to go to all the university departments as it would create unnecessary load on the bandwidth. To combat this issue, you would create separate VLAN for the computer science department and other departments.

One way to configure a router to route traffic between multiple VLANs is to use a "router on a stick". Router on a stick is a networking configuration that allows a router to route traffic for multiple VLANs using a single physical interface. This configuration is often used in networks that have a layer 2 switch that lacks the ability to route VLAN packets. Instead of relying on the switch to route traffic

between VLANs, a router is used to perform this function. Each VLAN is given its own IP subnet in a router-on-a-stick configuration, and VLAN tagging is used to identify traffic from each VLAN. Each VLAN's communication is recognised by a distinct VLAN tag, and the switch sends all VLAN traffic to the router on a single interface. When the router receives the VLAN-tagged traffic, it examines the VLAN tag to determine the appropriate destination for the traffic. The router then routes the traffic to the correct VLAN on the network. [33]

2.10 Security Methods

Network security is essential to protect businesses or commercial networks from experiencing attacks or downtime. A method of preventing attacks from happening is by filtering the incoming traffic that is received in networks using access control lists (ACL). ACL is stateless, this means it will just look at individual packets. ACL can be configured on routers and switches, with ACL you can configure exactly what traffic can pass through by defining different source IP addresses that can be blocked when packets pass through, this is known standard ACL. This can be taken a step further and include filtering destination address of packets and include port numbers such as port 22 for SSH or TCP/UDP packets, this is known as extended ACL. [22]

Outside ACL's, there are dedicated devices for filtering traffic. Most common is traditional firewall devices and newer versions called New Gen firewalls. Most firewall devices support stateful filtering, this is the opposite of stateless where it doesn't need a permit statement for every packet, as it recognises replies from already established connections. With traditional firewalls, these are like ACL's where it controls the flow of traffic by looking at source, destination IP address and port numbers, however, this isn't enough to protect against cyber threats. Hence, why new gen firewalls are also used. New gens firewalls have all the capabilities of traditional firewalls and have added capabilities such as built-in intrusion detection system, packet inspection capabilities, web filtering and many more.[23]

The only firewalls I will be using is access lists as that is directly implemented on routers and can be automated, whereas dedicated devices do not come under my MiniPod.

2.11 Remotely Accessing Devices

Telnet, one of the oldest network protocols still in use, allows network engineers to log into devices in a network from another device, enabling remote access and configuration. This can be very convenient for network engineers for making changes on WANs that cover multiple countries as they can remotely login from one country to a device in another country and make changes. Typically, when telnetting a router, users will open the command line and remotely access a device by logging with a username and password and then have full access. However, Telnet is not a secure protocol as it is unencrypted leaving it vulnerable to information leak during communication.[19]

Secure Shell (SSH) is another network communication protocol which serves the same purpose as telnet but encrypts all the data that is passed between the devices when it is remotely accessed. SSH not only provides encryption, but also better authentication than Telnet. With SSH, users typically use a cryptographic key for authentication, providing an additional layer of security over Telnet's password-based authentication [20].

2.12 Current Network automation technology

2.12.1 New Automation

Currently, there is network automation available for companies to use, however, the main problem that arises, is that for companies to switch to automate their networks, they would need to switch out their networking device. I.e., routers and switches.

Current automation offered by companies is through software defined networking. This is as the name suggests, networking but using scripts and software. ACI is a form of that. Application Centric Infrastructure (ACI) is one of the most well-known automation tools, due to it being made by Cisco[16]. This technology offers a way to centrally manage your network devices. You get the ability

to log into a Graphical User Interface (GUI) on the cloud and either write a script or manually click on a router and change the configuration, this is also known as the management plane. Before you make your changes, you can simulate what effect the changes will make to your network and after you approve, you can push the changes through. All this is completed on a controller. The controller can be a physical device or on the cloud as mentioned above. With the ACI deployment, the control plane and data plane are separated. Control plane is the brains of your network devices. It decides where to route it and does the computations necessary to route your packet. The data plane is the "dumb" part of your routing device. It gets a message and just sends it through. There are no computations completed. By separating the control and data plane, the devices can be controlled by their control plane which is in the controller. Businesses will save a lot of money as configuration time goes down by a significant amount since, in the GUI, you select the routers you wish to change or make a configuration on and run the script. This will then make the changes for all the routers.

Although this seems like a must have for every business to improve their network management and savings, the architecture for this automation comes at a costly price. For example, a starter kit equipment for ACI which includes layer 3 switches, a controller, and highspeed cables, it can cost up to $350,000![21] As well as the cost of purchasing the devices, you would need to switch over your whole network to these devices. The time and cost to do this as well as having downtime on your network while the switch is happening, does not appeal to businesses and hence they are refraining from switching over.

VMWare NSX is another network automation tool that is a competitor for Cisco's ACI technology. NSX is very similar to ACI, where it has a management plane for the user to make any changes for the network through its GUI, and it also has the control and data plane. The main difference with NSX, is their technology focuses on virtualised networks, meaning their software management tools can connect devices and computers over the internet [25]. However, ACI's tools already does this while being able to manage physical networks.

Another automation tool that is currently in the market is SolarWinds Network Manager [26]. Again, this is very similar to Cisco's ACI technology with having a management plane, control plane and data plane but has some very minor differences such as connectivity and supported application, which can include devices that are not compatible for cisco ACI technology.

2.12.2 Similar Network Automation Tools

Ansible is an open-source IT automation tool used for configuring, deploying, and controlling servers, network devices, and software applications automatically. Ansible uses playbooks, in which it defines automation tasks using the declarative language YAML. With its declarative approach you can define a playbook that specifies the desired state of the network device. This playbook can include tasks such as configuring interfaces, setting up VLANs, and configuring routing protocols. Ansible will then execute the necessary commands on the network device to bring it to the desired state. Additionally, it does not require any agents to be installed on target machines. This means that no software is required to be installed on the nodes that are being managed. Instead, it interacts with them and completes tasks using the SSH protocol. [30]

Ansible has the ability to configure tasks across numerous servers simultaneously and automate complicated multi-tier applications. Additionally, Ansible offers a variety of modules and plugins that allow users to manage a range of different systems and services, including cloud infrastructure, network hardware, and databases. Ansible places a high priority on security and includes functions such as role-based access control, SSH key administration, and encryption of sensitive data.

While Ansible is a powerful tool there is a few disadvantages. Ansible has a steep learning curve with users of this tool having to be familiar with writing in YAML and being able to define the tasks for the tool to execute. The execution speed for running tasks can be slow, especially when dealing with a large number of target machines. Additionally, troubleshooting and debugging Ansible playbooks can be challenging for novice users because the error messages are not always obvious in diagnosing problems.

Puppet is another popular automation tool for networking. It follows a client-server architecture in which a centralized server, known as the Puppet Master, communicates with all nodes that it manages. However, unlike Ansible, Puppet requires agents to be installed on all nodes being managed for it to communicate changes.

Puppet uses its own declarative language called Puppet DSL, which allows for defining configurations, services, packages, and other elements. With Puppet, you can define a manifest that describes the desired state of the network device. This manifest can include resources such as users, groups, and services, as well as networking resources such as interfaces, bridges, and VLANs. Puppet will then apply the changes necessary to bring the device to the desired state. One advantage of Puppet is its reporting mechanism, which tracks changes made to systems for compliance auditing and troubleshooting. [31]

However, one potential disadvantage of Puppet is that its configuration language also has a steeper learning curve than other automation tools. Additionally, some users may prefer Ansible's agentless architecture for its simplicity and ease of deployment.

While both Puppet and Ansible offer free versions of their tools, large-scale networks may require the use of commercial versions, which can be costly. For instance, Ansible's commercial version, called Ansible Tower, can cost up to $17,500 per year, which includes support for managing up to 100 nodes and 24/7 support. Similarly, Puppet's commercial version is also highly priced, with a cost of $199 per node, per year, including support.[29]

2.13 Conclusion

In conclusion, my project aims to bypass the big hurdle of cost for companies by creating a MiniPod that will automate network engineer tasks. While there are advanced tools available for these tasks, the MiniPod is designed to be a cost-effective solution for companies that can't afford to replace all

their networking devices. It's important to note that the MiniPod is not intended to replace these advanced tools or provide features like network change simulation. However, it will automate repetitive tasks and help network engineers work more efficiently. Overall, the MiniPod will be a valuable tool for companies looking to streamline their network engineering processes without breaking the bank.

3. Requirements Analysis

This section outlines the requirements that is required of the MiniPod. The requirements have been listed below along with their priority using the MoSCoW method. The requirements have been divided into functional and non-functional requirements.

3.1 Functional Requirements

ID	Description	Priority
FR 1	MiniPod must automate OSPF routing protocol configuration on routers	Must
FR 2	MiniPod could automate EIGRP routing protocol configuration on routers	Could
FR 3	MiniPod must give IP addresses to interfaces(ports) on a router	Must
FR 4	MiniPod must setup DHCP on a router	Must
FR 5	MiniPod could setup Telnet	Could
FR 6	MiniPod must setup SSH	Must
FR 7	MiniPod could setup Access List for allowing and denying IP addresses	Could
FR 8	MiniPod must setup Access List for allowing and denying port numbers	Must
FR 9	MiniPod could set up VLANs on network device	Could

3.2 Non-Functional Requirements

ID	Description	Priority
NFR 1	Time for automation must be faster than manual configuration	Must
NFR 2	The MiniPod should be easy to use	Should
NFR 3	The MiniPod could have an interface	Could
NFR 4	The MiniPod should have a user profile i.e. login privileges for admins only	Should

3.3 Use Case

Use Case	Configure Router
ID	1
Description	User runs the MiniPod on a router and router has the configurations setup
Primary Actor	User, MiniPod, Router
Secondary Actor	
Main Flow	1. User defines router configuration on MiniPod 2. User loads MiniPod onto device 3. MiniPod is run on the router
Post Condition	1. Router successfully configured according to user's criteria

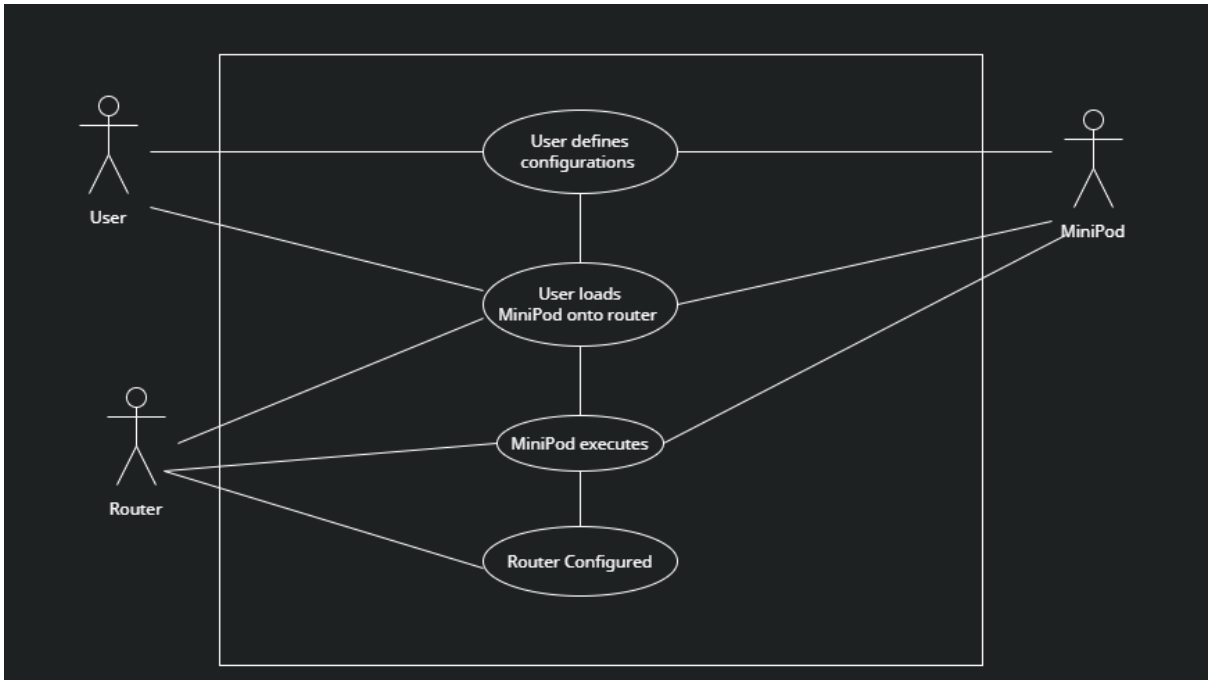

Figure 3 Use case diagram for ID1

4. Evaluation Strategy

4.1 Evaluation

This project will be self-evaluated. Due to the skill required to configure routers and switches in networks, it is not possible for me to carry out a user evaluation to compare the time it takes for my automation MiniPod and compare it to a manual configuration as it would require a skilled network engineer to carry this out.

For the evaluation, I will be using the network simulation software GNS3. I will have a small network topology that will consist of devices that can be configured by the MiniPod. This network topology will be a basic version of the 2-tier spine and leaf topology that was discussed in section 2.2. I will time myself in configuring these devices and check that the network is functioning then repeat the process using the MiniPod. To make sure there is no bias, I'll have instructions for configurations written down and repeat the manual configuration multiple times and take an average of the time it took to complete.

4.2 Testing

The MiniPod will undergo testing throughout development to ensure that it is functional, and the requirements of the project are being met. Regular testing can also discover bugs and errors during the process. As more and more functionality are implemented, these functions will need to be tested right after and before implementing new and extra features to ensure the most important requirements of the project is met.

5. Project Management

5.1 Methodology

An Incremental methodology approach will be used for this project. This process splits the development into 4 categories. First, is the requirements phase, where all the system requirements are identified. Secondly, is the design and development phase, and the system functionality is implemented during this phase. Next, the System is tested and ensures the functionality is performing as expected. Finally, the implementation phase is where the current system, if successful after the testing phase, will be implemented as the final product. [24]

This methodology approach is very convenient and suitable for this project as it is easier to test and debug and allow errors to be identified in the early stages of development.

5.2 Gantt Chart

A Gantt chart was created to visualises a timetable that has been set for semester 2, shown in figure 3, noting all the tasks needed to be carry out with appropriate time scales throughout the semester and ensuring there is enough time to implement all functionality.

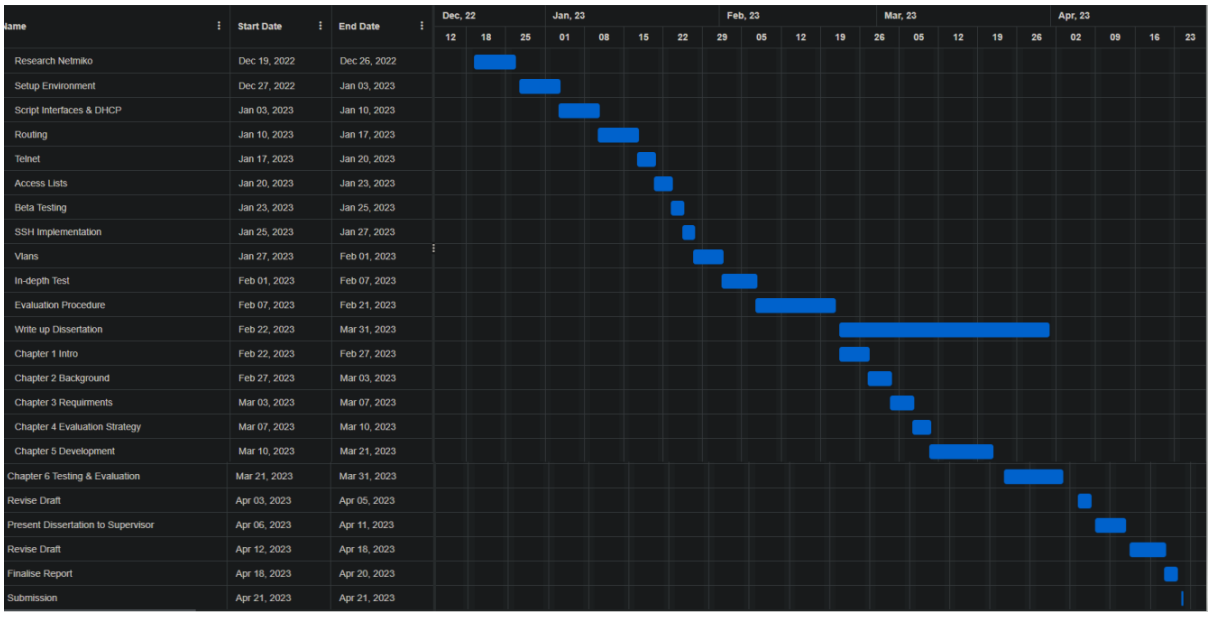

Figure 4 Gantt Chart

5.3 Risk Management

In this section a list of risks has been identified that may occur during the development process. Recognising these risks early and planning around these risks are important to having a successful project where all the requirements can be met. Each risk will have a corresponding impact and probability, highlighted by a colour to indicate the risk severity, using the risk table guide.

5.3.1 Risk Table Guide

		Impact				
Probability		Trivial	Minor	Moderate	Major	Severe
	Almost Certain	Low Risk	High	High	Extreme	Extreme
	Likely	Low Risk	Moderate	High	High	Extreme
	Possible	Low Risk	Moderate	Moderate	High	Extreme
	Unlikely	Low Risk	Moderate	Moderate	High	High
	Rare	Low Risk	Low Risk	Moderate	Moderate	High

5.3.2 Rate of Probability Guide

Probability	**Rate of Probability**	**Description**
Rare	<10%	Not expected to occur
Unlikely	10-25%	More likely not to occur than occur
Possible	25-50%	May or may not occur
Likely	50-75%	More likely to occur than not
Almost certain	>75%	Expected to occur

5.3.2 Impact Description Guide

Impact	**Description**
Trivial	Very little impact on project
Tolerable	Manageable impact on project
Moderate	Average impact on project
Major	Significant impact on project
Severe	Chance of project failure

5.3.3 Risk Management Table

Risk	Mitigation Plan
Loss of Data Type: Technology Probability: Rare Impact: Severe	Work will regularly be backed up and stored on Heriot watt OneDrive. Git hub will also be used for version control and work will regularly be committed to the workspace to minimize loss of data.
PC failure Type: Technology Probability: Rare Impact: Major	Ensure work is saved and in the case of a PC failure then the work can easily be continued on a different PC or use any on campus systems to continue work.
Power outage (Energy crisis) Type: Technology Probability: Possible Impact: Major	Due to the energy crisis and possible blackouts happening in the winter period, I must ensure that work is backed up regularly and be aware of any planned power outages that get announced by the UK government as to plan and work around the downtime.
Project Delay Type: Estimation Probability: Possible Impact: Major	In the event of a delay due to unpredictable factors or wrong time estimation for tasks, the timetable must be readjusted, and the supervisor must be notified.
Bugs Type: Technology Probability: Almost Certain Impact: Minor - Severe	Bugs are very common to appear in the development phase, therefore testing will be carried out throughout development to identify bugs as early as possible and avoid delays.
Incompatible Framework (Python, Net Miko) Type: Technology Probability: Unlikely Impact: Moderate	Research will be done initially to identify the most suitable frameworks to use for development, if a framework is found to be incompatible then an alternative will be found and used instead.
Incompatibility with GNS3 Type: Technology Probability: Unlikely Impact: Major	If there are any issues when using GNS3 then another network simulation software will be used, Cisco packet tracer can be used instead which is also a network simulation tool. In the extreme case, physical devices can be used to test my MiniPod.
Health Issue/Illness Type: People Probability: Possible Impact: Major	In the event of any health issue, this will result in work being stopped for a brief period and the timeline would need to be readjusted to ensure all tasks are completed on time. Supervisor will need to be notified and if needed, a mitigation form should be filled out for any extensions required.

5.4 PLES

5.4.1 Professional

As this is an honours project being carried out at Heriot Watt University, I must conduct myself professionally throughout the dissertation and ensure that I follow the conduct rules provided by the University. I also must follow the standards and code of conduct set by the British Computer Society and work is done ethically and responsibly.

5.4.2 Legal

As my project is a self-evaluation project where I test my MiniPod, I won't need to worry about GDPR laws as I won't need to gather any confidential information and carry out any investigation with users. The software I will be using to simulate networks (GNS3) is open source. However, to use Cisco devices in GNS3, a license is required to download and run the devices. Fortunately, I already have devices installed, which were provided by the university, so I do not need to worry about obtaining a license. Therefore, for this project I will not have any legal issues.

5.4.3 Ethical

The ethics form for the project was approved by Heriot Watt. I stated in the ethics form that I will be self-evaluating my project, by comparing the time it takes to manually configure a network that contains devices, against the time it takes for my MiniPod to do the same test. If during the project, the evaluation plan changes, I will ensure to update the ethics and notify the ethics board to get reapproved.

5.4.4 Social Issues

This MiniPod will aim to benefit network engineers in setting up networks. The MiniPod will automate the process of setting up devices in a network and will save a lot of time and avoid any human generated errors. This can also cut costs for companies as less time will be sent in setting up

networks and companies will not need to buy newer expensive devices that have built in automation capabilities and purchase automation software.

6. Implementation

This section will go over the technology used to implement the MiniPod and discuss the virtual network software that was used to test on. Below in figure 6 is an overview showing the workflow of the MiniPod and all the configurations that it will carry out.

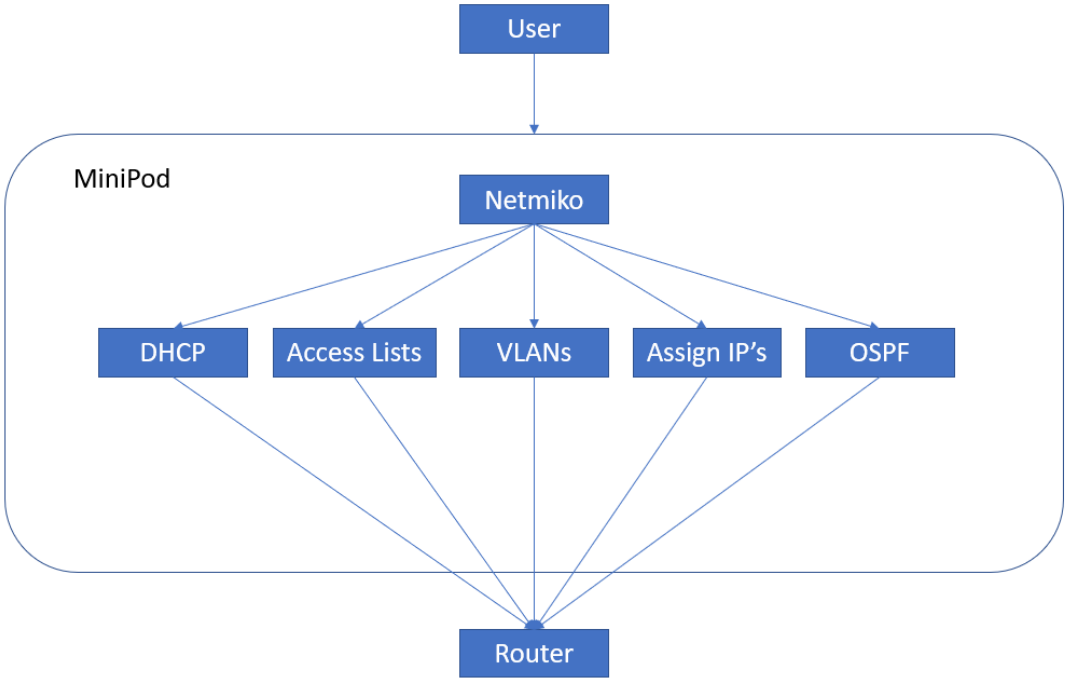

Figure 5 MiniPod system overview

The MiniPod will carry out 5 configurations: DHCP, Access Lists, VLANs, Assigning IP's and OSPF routing. It's important to note that Netmiko is used to send these configurations to the device that is being configured and the configurations will need to be suitably configured within the Python script and this will be explained throughout this section.

6.1 Netmiko

The MiniPod was developed in Python using a library known as Netmiko[38]. Netmiko is a library that is based on Paramiko[37], which is a Python implementation of the SSH protocol for secure remote access to network devices. Netmiko simplifies the process of network automation by providing an easy-to-use interface for configuring and managing network devices such as routers,

switches, firewalls, and load balancers using Python scripts. With Netmiko, network engineers can automate the configuration of these devices and improve network efficiency and reliability.

Like Ansible, it uses the Secure Shell (SSH) protocol which allows it to connect securely to a network device. Once an SSH connection is established, configuration commands can be sent to devices and executed, with the results presented to users. As Netmiko uses the same set of standardized APIs across different networking devices, this results in the same syntax and commands being used across many different networking devices and from different vendors, making it simpler for users to configure and administer multiple devices at the same time. Some vendors that Netmiko can be run on include Cisco, Arista, and Juniper.[43]

There are a few reasons why it's better to use Netmiko over current technologies that were mentioned. For example, Netmiko is purpose-built for network automation, whereas Ansible and Puppet are generic automation tools that can also be used for network automation. This means that Netmiko offers features and capabilities that are specifically tailored to the needs of network automation, making it more efficient and easier to use.

Since Netmiko is a python library, it can be easily integrated into existing Python scripts and workflows, whereas Ansible and Puppet are complex systems, and these require significant configuration and setup. This results in Netmiko being more flexible and easier to use for network engineers who are already familiar with Python.

Furthermore, Netmiko provides a simple and consistent API for interacting with network devices, making it easier to learn and use than Ansible and Puppet. Also, the Netmiko API is well-documented and has a large community of users and contributors, making it easier to find help and support when needed.

One important factor is Netmiko provides support for a wide range of network devices and vendors[43], including legacy devices and devices that do not support modern automation protocols.

Whereas Ansible and Puppet may not support some older devices or may require custom modules to be developed.

In summary, Netmiko is a powerful and efficient library for network automation that offers features and capabilities specifically designed for network engineers. Its simplicity, flexibility, and wide range of device support make it a valuable tool for network automation tasks.

With Netmiko it will only send and receive output of commands you write, and I had to extend its functionality by compiling and configuring various network settings, such as DHCP, OSPF, ACL, VLAN, and assigning Ips. And ensuring with python scripts that this was configured successfully.

Other options I considered included PExpect and Paramiko. PExpect[39] is a Python module that provides a simple way to automate interactions with applications that require text input and output, such as CLI interfaces. However, I found that Netmiko provided a more robust solution as it was specifically designed for network device automation and had more comprehensive documentation. On the other hand, Paramiko[37] is a Python library for secure shell (SSH) connections, and while it is a powerful library, it required more low-level configuration and customization than Netmiko, making Netmiko a better fit for this project.

6.2 Network Virtualisation Tool - GNS3

To test the automation tool that was created, a virtual network environment software was used known as GNS3 (Graphical network Simulator 3)[44]. GNS3 is an open-source network virtualisation tool that allows users to build network topologies and test network configurations, simulate network traffic, and troubleshoot issues before deploying them in a production environment. GNS3 allows users to test a variety of devices from different vendors and allows scripts to be run on these devices. By using a virtual environment, it saves me from spending hundreds on hardware and avoid the time-consuming process of physically setting these devices up. Additionally, there was the flexibility of being able to test my tool anytime, anywhere using GNS3. An alternative that was considered is VIRL[41], a Cisco-specific tool, but it requires a license to use and has limited support

for serial interfaces. Another option was EVE-NG[42], while this is a clientless tool, its lack of online support made me less comfortable using it. Therefore, I preferred using GNS3 for the project.

As Cisco is the world's largest provider for networking devices[40], it made sense to use these vendors devices when trying to run my tool. GNS3 already has layer 2 switches pre-installed, however routers are required to be imported. For this I used the Cisco 2691 router appliance which was supplied to me by the university and installed this onto my GNS3.

A GNS3 virtual machine was also required, and this was installed with VMware's Workstation Pro[45]. This allowed me to to run a ubuntu docker where I was able to run python scripts within GNS3.

To run Python scripts on the Docker, I needed to connect it to the internet and install the necessary packages. To do this, I used the cloud node within GNS3 to bridge to my physical network. However, I spent a lot of time trying to figure out why it wasn't working on my laptop, which uses Wi-Fi. Eventually, I realized that bridging only works for Ethernet connections. To resolve this, I had to use my home computer instead where an ethernet connection was used. After internet access was setup, I was able to install python and Netmiko on the docker by using the pip command.

6.3 Connecting to a Device

When implementing the code for the tool, I first had to make a connection to the device I was configuring. Then for each of the different configurations I was trying to achieve, these were split up into different python methods.

To make a connection to a device using Netmiko, the device type, IP address, username and password had to be defined. The IP, username and password were obtained by the user, and by default, the device was specified to be Cisco. If the device was from a different vendor, this would need to be changed within the code. However, it is pre-defined as Cisco since Cisco is more widely used than other vendors[40]. Then using this information, the script would attempt to make the SSH

connection to the desired device. If any of this information is not correct, then on the command line it would show up with an error stating that the connection was failed. After a successful connection, the tool is now entered into the configuration mode on the device, and it would then go on to do some configurations by using the send command method.

6.4 SSH

One problem that occurred right away was attempting to automate the process of SSH being set up on a device. Automating the process of setting up SSH is not possible using Netmiko because SSH requires initial manual configuration to be done on the device to enable the SSH service and generate cryptographic keys. This manual configuration process cannot be automated through Netmiko as it involves a level of interaction that cannot be done through automated scripts. However, once SSH is enabled and the necessary keys are generated, Netmiko can be used to automate the subsequent SSH login and configuration processes.

One possible implementation of automating SSH is by using a physical wire to connect to a devices port which would give it out-of-band access, which means the device you are making configurations from does not need to be connected to the same network. Through this method I would be able to make up an attempt of making an SSH automation configurations. However, GNS3 does not allow this as the only dedicated connection I can get is by opening the router terminal, I do not need a wire for this as its already a part of my pc, and within the router terminal on GNS3 I can’t load and run python scripts. I tried researching into different methods and on how I would be able to do this implementation on GNS3 and trying to do these implementation but it was not successful due to time constraints and to ensure that other configurations were implemented, I was not able to achieve SSH automation.

6.4 Assigning IP Addresses

To automate the process of assigning IP addresses to a router's interface, I needed to determine the IP address of the neighbouring routers and assign an appropriate IP address within the same

network. This is where I chose the approach of using a discovery protocol. I first implemented this using LLDP but upon running the script for this I realised that this was not supported on the Cisco 2691 router I was using, and this would be the case for legacy devices as well.

```
# Run the "show cdp neighbors" command
output = net_connect.send_command('show cdp neighbors detail')

# Extract the IP addresses and interfaces from the output
neighbours = re.findall(r"IP address: (\d+\.\d+\.\d+\.\d+).*?Interface: (\S+)", output, re.DOTALL)
```

Figure 6 Extracting CDP Information

Therefore, I had to make use of Cisco's discovery protocol CDP. With CDP I ran a command (show CDP neighbours' detail) on the router where it would output the neighbouring devices information, and this would contain the device name and its IP address for the corresponding port number it was connected to.

To extract the necessary IP address information and the interface number from the output, I used the 're' Python module to search for a match containing all the IP addresses and interface number. The "re.findall()" method is used with a regular expression pattern as its first argument. The regular expression pattern has two capture groups: (\d+\.\d+\.\d+\.\d+) and (\S+).

The first capture group \d+\.\d+\.\d+\.\d+ matches IP addresses in the format of four decimal numbers separated by periods (e.g., 192.168.1.1). The second capture group .*?Interface: (\S+): This matches any characters (including newlines) up to the first occurrence of the word "Interface" followed by a non-whitespace character.

The "re.DOTALL" flag is passed as the third argument to the "re.findall()" method, which allows the dot . character in the regular expression pattern to match newline characters as well.

Therefore the "re.findall()" method returns a list of tuples, where each tuple contains two items: the IP address as a string and the name of the interface as a string.

Once the information was parsed in Python, I calculated a suitable IP address for that interface by incrementing the network part of the IP address by one.

One limitation of assigning the IP address in this manor is there is no way for me to obtain the subnet mask from CDP packets and automatically determine it. Hence, by default I have left the subnet mask to be 255.255.255.0 where this will allow for 254 unique IP addresses (class C address).

One disadvantage of using CDP to determine the IP addresses is CDP packets are not secure. CDP packets are not encrypted and since they contain information about the device names and IP addresses, this can easily be intercepted and read by an attacker on the network. This can lead to security risks such as CDP packet spoofing, eavesdropping, information disclosure and even Denial of Service attacks.

In hindsight, although CDP was not the best approach in assigning IP's, I used this approach as I was able to get neighbouring routers information, such as routers IP addresses and port numbers. I realise now that this could be implemented a different way where I get the neighbouring router information from the user and assign IP's interfaces this way. This is where the user will define the IP addresses for the routers, and the script will use this information and assign IP's.

Additionally, this approach only works for IPv4 address and not IPv6. I have chosen to ignore assigning IPv6 addresses for now since most networks are still using IPv4[46] and have not fully migrated to IPv6. This approach could be extended further later to consider assigning IPv6 addresses to interfaces.

6.5 DHCP

The user will be given the option to set up a DHCP server on the desired device at the beginning of running the MiniPod, as not all devices are suitable for this purpose. To achieve DHCP automation I made the same assumption that the network address is still a Class C address. By using the same IP address that was used to connect to the device, the DHCP network address was calculated by

stripping the last octet of the address and this was passed into the configure DHCP method along with the device IP.

```
def configure_dhcp(net_connect, DHCP_ip, deviceIP):
    dhcp_config_commands = [
        'ip dhcp pool LAN',
        f'network {DHCP_ip} 255.255.255.0',
        f'default-router {deviceIP}',  # Default gateway
        'exit',
        f'ip dhcp excluded-address {deviceIP}',  # Exclude the router IP from the DHCP pool
    ]
    output = net_connect.send_config_set(dhcp_config_commands)
    print(output)
```

Figure 7 DHCP automation code

Now within this method it contained the configuration commands for setting up DCHP on a router. First the DHCP pool is created, and this is given a default name “LAN”. The DHCP IP variable is used to specify the network address of the DHCP pool, along with the subnet mask, which will determine the number of unique IP addresses the DHCP server can lease out. To prevent the same IP from being assigned to multiple end users in the network, I excluded the device's IP address from the DHCP pool using the device IP variable. A default gateway address is configured as this is required for DHCP. The default gateway address, used for connecting the local network to external network, needs to be setup for the DHCP with the address being the same address of the router, therefore the device IP variable was reused to achieve this.

6.6 OSPF

```
def configure_ospf(net_connect):
    ospf_config_commands = [
        'router ospf 1',
        'network 0.0.0.0 255.255.255.255 area 0',  # Advertise all connected networks to area 0
        'exit',
    ]
    output = net_connect.send_config_set(ospf_config_commands)
    print(output)
```

Figure 8 OSPF automation code

In figure 9, this code sets up Open Shortest Path First (OSPF) routing protocol on a router. It creates a list of OSPF configuration commands which includes:

“router ospf 1” - This command enables OSPF process on the router and sets the process ID to 1.

“network 0.0.0.0 255.255.255.255 area 0” - This command advertises all the connected networks to area 0, which means that OSPF will include all the router interfaces in its routing table. The 0.0.0.0 address allows the router to include all connected networks in the OSPF routing process.

The approach of using the 0.0.0.0 address has a limitation, as it will advertise all connected networks to a single area. This will include loopback interfaces and interfaces to other routers, and this can result in unnecessary traffic on the network. If I had more time, I would have improved this configuration by specifying specific network statements for each network that is connected.

6.7 Access Lists

As described in 2.9, extended ACLs would be most suitable to implement here. I have decided to block all incoming traffic except ICMP traffic, used for pings, and allow SSH (TCP port 22) traffic, as these are the traffic I know will be in use. Below in figure 10.

```
def configure_access_lists(net_connect):
    acl_config_commands = [
        "access-list 101 permit icmp any any",  # allow ICMP
        "access-list 101 permit tcp any any eq 22", # allow SSH (port 22)
        "access-list 101 deny ip any any",  # block everything else
    ]
    output = net_connect.send_config_set(acl_config_commands)
    print(output)
```

Figure 9 Extended ACL Code

One mistake that I made when implementing extended ACLs was that I did not initially realize that the statements are evaluated in a first come first serve basis (FCFS). In my case, I had placed the statement to deny all traffic at the start, which caused issues when trying to use ICMP and SSH. It

was only then that I realized the importance of considering the order of statements in an ACL, and made the necessary adjustments to ensure proper traffic flow.

6.8 VLANs

The VLAN automation implementation was set up by achieving the router on a stick configuration. I first experimented with trying to receive an input from the user by asking how many VLANs they would like to have setup on the router and from there I would have used that to configure the desired amount of VLANs. However, as I'm using Netmiko which uses an SSH connection to connect to the device when trying to set up the VLANs the SSH connection would close, and I would receive an error message in the terminal indicating that the connection was closed by the remote device. Initially, I had the user input configured to be requested after the SSH connection was established. However, when I tried to change this to prompt for input before the connection was established, I still received the same error, and the connection was closed.

```
def configure_vlan(net_connect):
    VLAN_config_commands = [
        'interface FastEthernet0/1.10',
        'encapsulation dot1Q 10',
        'description admin VLAN',
        'ip address 192.168.10.1 255.255.255.0',
        'exit',
        'interface FastEthernet0/1.20',
        'description CS VLAN',
        'encapsulation dot1Q 20',
        'ip address 192.168.20.1 255.255.255.0',
        'exit',
    ]
    output = net_connect.send_config_set(VLAN_config_commands)
    print(output)
```

Figure 10 VLAN automation code

Therefore, the VLAN implementation shown in figure 11 will setup 2 VLANs using the commands. The configuration commands shown specify the physical interface the VLAN is being configured on where the ".10" will create a sub interface on the physical interface FastEthernet0/1. The second

command 'encapsulation dot1Q 10' sets the VLAN ID of the sub-interface to 10 using the 802.1Q VLAN encapsulation protocol. The third command 'description admin VLAN' is an optional command that adds a description to the sub-interface to help identify it. This is useful for trying to name different VLANs for different networks. The fourth command assigns an IP address to the sub-interface with a subnet mask. This IP address can be used for routing and management purposes on the VLAN. Given more time, I would have expanded the code and would have changed it to where it would take in a text file that was written by the user where it contained the desired VLAN configurations that they would have liked to set up.

Whenever a different configuration is run on the device, the Command Line Interface will display an output that highlights all the configurations being carried out and shows if the configurations were successful.

7. Evaluation & Testing

This section describes the self-evaluation that was carried out on the MiniPod with all the configurations that were implemented in the previous section being evaluated. Testing will be shown as well using the same topology that is used for the evaluation. An evaluation of the requirements will be shown as well as explaining all the requirements that were achieved and some not achieved.

7.1 Self-Evaluation

As mentioned in the evaluation strategy, due to the knowledge required to carry out these network configurations it was not possible to carry out this evaluation with users and it would need to be carried out by myself. Therefore, using the GNS3 software I will be creating a simple spine and leaf topology where a new spine is added to the network and this spine device would need to be configured.

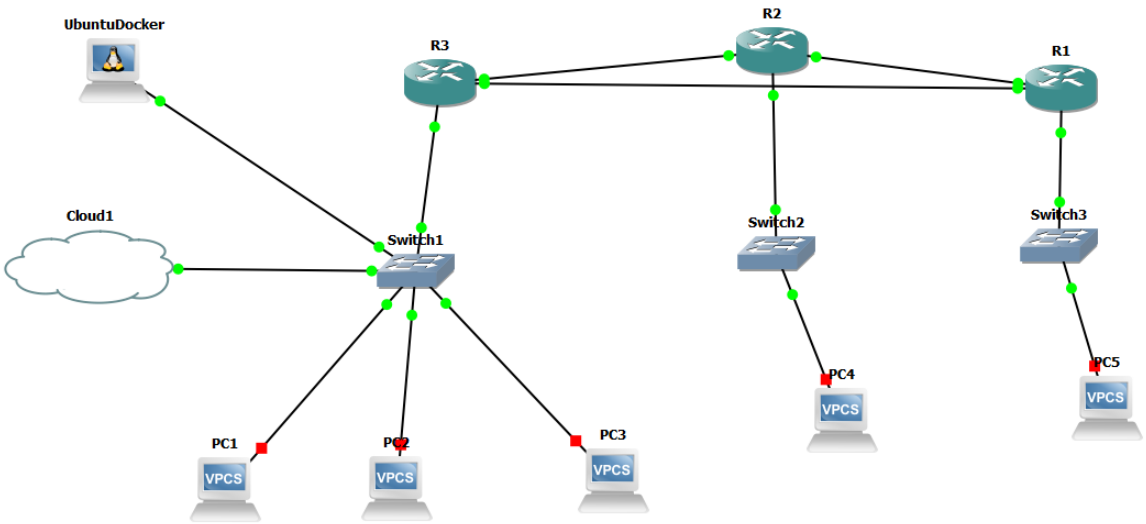

Figure 11 Spine & Leaf Topology in GNS3

In figure 12 the spines of this topology consist of routers that are named R1, R2 and R3. R3 is the newly added device, and this device will have the configurations setup by the MiniPod. This will be achieved by the ubuntu docker that is connected to the layer 2 switch switch 1. The cloud node is in

the topology as this was used to install the necessary packages to allow the ubuntu docker to run the tool. The switches are the leaf nodes in this topology and the PCs represent the end devices in this topology.

The goal for this evaluation is to show the MiniPod is faster at configuring all the configurations for the device R3 than it is to manually configure them. The tasks that need to be carried out to successfully configure this are listed below:

1. SSH into R3
2. Give the 2 interfaces on R3 IP addresses that are suitable to be within the same network of R1 and R2
3. Set up DHCP server for R3 for end nodes PC1,2,3 and exclude the R3 address from the DHCP pool
4. Set up OSPF routing protocol on R3 to allow routing between the spine router
5. Configure 2 VLANs using the router-on-stick configuration.
6. Configure extended ACLs which only allows ICMP and SSH traffic

These tasks cover all aspects of what has been implemented and will showcase the time savings of using the MiniPod compared to manual configuration.

Before carrying out this evaluation one of the one of the router's interfaces has already been assigned an IP address and SSH to allow for connections from the Ubuntu Docker and allow the user to manually SSH into R3. This will then allow them to carry out the configuration from here onwards.

To minimize bias as much as possible, the process will be manually repeated three times. To accurately reflect manual configurations, command shortcuts that are typically used in the CLI will be employed. For instance, instead of typing out the full command "configure terminal", "conf t" will be used, or the TAB key will be used to allow the CLI to complete the command when partially typed.

While seemingly minor, these manual practices are common and can save time when configuring manually.

After repeating the manual configuration 3 times and running the tool 3 times here is the results presented in the table below.

	Run 1	Run 2	Run 3	Average
Manual	4 min 54 s	6 min 2 s	5 min 7 s	5 min 21s
MiniPod	22s	22s	22s	22s

As shown in the table, the tool is consistent across repeated runs and significantly faster than manual configurations. As seen in the manual configurations, the times varied with each run, with some taking much longer than others, thus inflating the average. This was due to typographical errors I made while trying to enter the commands as quickly as possible. Such errors can be common in the real world, and this highlights the advantages of using an automation tool, which reduces the likelihood of errors.

7.2 Testing

As mentioned in the implementation, GNS3 was used to run the MiniPod and as described in the project management section I followed an incremental methodology where a configuration was implemented and tested right away to ensure it was working and moved onto the next feature. To test the functionality and ensure the configurations were being successfully setup, I had print statements showing the output of the configuration commands being ran by the MiniPod on the command line interface of the ubuntu docker and if any errors were run into this would be highlighted.

```
root@UbuntuDocker:~# python3 final.py
Enter device IP address: 192.168.1.130
Enter username: mubeen
Enter password: password
Do you want to configure DHCP? (y/n): y
[('192.168.50.2', 'FastEthernet0/1,'), ('192.168.25.2', 'FastEthernet1/0
,')]
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface FastEthernet0/1
R3(config-if)#ip address 192.168.50.3 255.255.255.0
R3(config-if)#no shutdown
R3(config-if)#end
R3#
interface FastEthernet0/1 ip address 192.168.50.3 255.255.255.0
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface FastEthernet1/0
R3(config-if)#ip address 192.168.25.3 255.255.255.0
R3(config-if)#no shutdown
R3(config-if)#end
R3#
interface FastEthernet1/0 ip address 192.168.25.3 255.255.255.0
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#ip dhcp pool LAN
R3(dhcp-config)#network 192.168.1.0 255.255.255.0
R3(dhcp-config)#default-router 192.168.1.130
R3(dhcp-config)#exit
R3(config)#ip dhcp excluded-address 192.168.1.130
R3(config)#end
R3#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#router ospf 1
R3(config-router)#network 0.0.0.0 255.255.255.255 area 0
R3(config-router)#exit
R3(config)#end
R3#
```

Figure 12 CLI showing the output of commands being configured

```
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#interface FastEthernet0/1.10
R3(config-subif)#encapsulation dot1Q 10
R3(config-subif)#description admin VLAN
R3(config-subif)#ip address 192.168.10.1 255.255.255.0
R3(config-subif)#exit
R3(config)#interface FastEthernet0/1.20
R3(config-subif)#description CS VLAN
R3(config-subif)#encapsulation dot1Q 20
R3(config-subif)#ip address 192.168.20.1 255.255.255.0
R3(config-subif)#exit
R3(config)#end
R3#
configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#access-list 101 permit icmp any any
R3(config)#access-list 101 permit tcp any any eq 22
R3(config)#access-list 101 deny ip any any
R3(config)#end
R3#
```

Figure 13CLI showing the output of commands being configured continued

In figures 13 and 14 it is shown the configurations being set up by the tool. First it will ask the user to input the the device IP along with the username and password to establish an SSH connection. Next it will ask if DHCP should be configured and after this all the configurations will be carried out.

7.3 Requirements Updated

After completing the tool, the requirements can now be updated showing which have been completed and incomplete giving an explanation.

ID	Description	Priority	Complete?
FR 1	MiniPod must automate OSPF routing protocol configuration on routers	Must	Yes

FR 2	MiniPod could automate EIGRP routing protocol configuration on routers	Could	No
FR 3	MiniPod must give IP addresses to interfaces(ports) on a router	Must	Yes
FR 4	MiniPod must setup DHCP on a router	Must	Yes
FR 5	MiniPod could setup Telnet	Could	No
FR 6	MiniPod must setup SSH	Must	No
FR 7	MiniPod could setup Access List for allowing and denying IP addresses	Could	No
FR 8	MiniPod must setup Access List for allowing and denying port numbers	Must	Yes
FR 9	MiniPod could set up VLANs on network device	Could	Yes

Overall, the major requirements were implemented, with the most important configurations that you would expect to be carried out on a networking device being completed. The requirements that had the highest priority were completed, with the exception of automating SSH. This was explained as not possible in the implementation. Additionally, Telnet was decided to be left out as this protocol is not secure, and it would be pointless to set it up when there would already be an established and secure SSH connection. Although there wasn't enough time to implement EIGRP routing, the OSPF routing protocol has been automated and this should be sufficient. Filtering IP addresses using access lists was not implemented due to time constraints.

Conclusion

In conclusion, the goals that were set out at the beginning of the report were achieved successfully. Common network configurations were researched, along with existing network automation technologies. With the use of Python and the Netmiko library, the MiniPod was created, and the implementation was successful, as most of the important requirements were completed, such as setting up interfaces with IP addresses, OSPF, DHCP, access lists, and VLANs. Some configurations were not implemented, but these were low priority ones. The tool was heavily tested, following the methodology that was described, with testing occurring as soon as something new was implemented. The tool was tested on a network simulation software called GNS3, and the self-evaluation shows the tool outperforming manual configuration in setting up a device in a 2-tier spine and leaf topology. The project is a success due to it achieving its main aim which was to create an automation tool to configure routers. This has been completed and the results are as expected which is the script is faster than manual configurations. At the beginning of the project, a significant amount of time was spent trying to fully set up GNS3 and troubleshoot errors in the environment. This resulted in delays and impacted the Gantt chart. When working on each requirement, a lot of research was conducted to ensure that each aspect was properly configured. As a result, some lower-priority configurations were not implemented. Overall, it can be considered the project was a success as all the goals set out were completed and a successful tool was created that is considerably faster than manually configuring the same tasks.

8.1 Limitations

While the MiniPod automation tool was successful in achieving most of the set goals, there were some limitations that were encountered during the implementation. Some limitations of the automation MiniPod tool are listed below.

When the tool is configuring the IP for interfaces, it will not assign an IP to an interface if that interface is not connected to any other device. Due to the nature of how this process was coded it

requires a device to be connected to an interface as it uses a discovery protocol to calculate a suitable IP address for it to give.

As explained previously, the VLAN implementation had problems with trying to be configured by getting a user input as the SSH connection would close every time it was tested. Therefore, if any changes to the number of VLAN's were to be changed including specifying which interface the VLAN would be setup on then it would require the commands being changed within the code.

8.2 Future Work

The MiniPod automation tool is capable of automating many of the common configurations that a network engineer would typically perform. While the tool is functional, there are some areas for improvement that could enhance its capabilities, as outlined below.

- Building a user interface would be very beneficial as it would streamline the process of a user being able to set different configurations on an interface instead of having to take input from the command line interface.
- Currently the tool will only support Cisco devices, but this can be expanded so it supports networking devices from different vendors such as Juniper, Huawei and others to improve its versatility.

References

[1] www.ibm.com. (n.d.). networking-a-complete-guide. [online] Available at: https://www.ibm.com/uk-en/cloud/learn/networking-a-complete-guide.

[2] Ali, A.N.A., 2012. Comparison study between IPV4 & IPV6. International Journal of Computer Science Issues (IJCSI), 9(3), p.314.

[3] RIPE Network Coordination Centre. (2019). The RIPE NCC has run out of IPv4 Addresses. [online] Available at: https://www.ripe.net/publications/news/about-ripe-ncc-and-ripe/the-ripe-ncc-has-run-out-of-ipv4-addresses.

[4] Droms, R., 1999. Automated configuration of TCP/IP with DHCP. IEEE Internet Computing, 3(4), pp.45-53.

[5] Kindervag, J., 2010. Build security into your network's dna: The zero trust network architecture. Forrester Research Inc, 27.

[6] Pluribus Networks. (2019). Traditional Network Infrastructure Model: Issues Associated with It. [online] Available at: https://pluribusnetworks.com/blog/traditional-network-infrastructure-model-and-problems-associated-with-it/

[7] www.networkcomputing.com. (n.d.). Why Networks Are Evolving Toward Leaf-Spine Architectures | Network Computing. [online] Available at: https://www.networkcomputing.com/networking/why-networks-are-evolving-toward-leaf-spine-architectures.

[8] ComputerNetworkingNotes. (n.d.). Access, Distribution, and Core Layers Explained. [online] Available at: https://www.computernetworkingnotes.com/ccna-study-guide/access-distribution-and-core-layers-explained.html.

[9] Huawei Enterprise Support Community. (n.d.). Spine-Leaf vs Three-Tier Network Architecture. [online] Available at: https://forum.huawei.com/enterprise/en/spine-leaf-vs-three-tier-network-architecture/thread/782313-100723 [Accessed 24 Nov. 2022].

[10] Wijaya, C., 2011, December. Performance analysis of dynamic routing protocol EIGRP and OSPF in IPv4 and IPv6 network. In 2011 First International Conference on Informatics and Computational Intelligence (pp. 355-360). IEEE.

[11] Cisco. (n.d.). How Does a Router Work? [online] Available at: https://www.cisco.com/c/en/us/solutions/small-business/resource-center/networking/how-does-a-router-work.html#~different-types-of-routers.

[12] SearchNetworking. (n.d.). Static Vs. Dynamic Routing: What is the Difference? [online] Available at: https://www.techtarget.com/searchnetworking/answer/Static-and-dynamic-routing#:~:text=What%20is%20static%20routing%3F [Accessed 24 Nov. 2022].

[13] OSPF Support for Fast Hello Packets. (n.d.). [online] Available at: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_ospf/configuration/xe-16/iro-xe-16-book/iro-fast-hello.pdf [Accessed 24 Nov. 2022].

[14] Albrightson, R., Garcia-Luna-Aceves, J.J. and Boyle, J., 1994. EIGRP--A fast routing protocol based on distance vectors.

[15] Harmoush, E. (2016). Virtual Local Area Networks (VLANs). [online] Practical Networking .net. Available at: https://www.practicalnetworking.net/stand-alone/vlans/.

[16] Cisco. (n.d.). Cisco Application Centric Infrastructure - Cisco Application Centric Infrastructure Solution Overview. [online] Available at: https://www.cisco.com/c/en_uk/solutions/collateral/data-center-virtualization/application-centric-infrastructure/solution-overview-c22-741487.html [Accessed 24 Nov. 2022].

[17] Manzoor, A., Hussain, M. and Mehrban, S., 2020. Performance analysis and route optimization: redistribution between EIGRP, OSPF & BGP routing protocols. Computer Standards & Interfaces, 68, p.103391.

[18] What Is BGP? | BGP Routing Explained | Cloudflare UK. (n.d.). Cloudflare. [online] Available at: https://www.cloudflare.com/en-gb/learning/security/glossary/what-is-bgp/.

[19] Fitzgibbons, L. (2021). What is Telnet? Definition from SearchNetworking. SearchNetworking. [online] Sep. Available at: https://www.techtarget.com/searchnetworking/definition/Telnet.

[20] Kovačević, A. (2021). Telnet vs. SSH: How Is SSH Different From Telnet? [online] Knowledge Base by phoenixNAP. Available at: https://phoenixnap.com/kb/telnet-vs-ssh#:~:text=SSH%20serves%20the%20same%20primary [Accessed 25 Nov. 2022].

[21] Duffy, J. (2014). A breakdown of Cisco ACI pricing. [online] Network World. Available at: https://www.networkworld.com/article/2459441/a-breakdown-of-cisco-aci-pricing.html [Accessed 28 Nov. 2022].

[22] Sedayao, J., 2001. Cisco IOS Access Lists. " O'Reilly Media, Inc.".

[23] Neupane, K., Haddad, R. and Chen, L., 2018, April. Next generation firewall for network security: A survey. In SoutheastCon 2018 (pp. 1-6). IEEE.

[24] Hibbs, C., Jewett, S., and Sullivan, M. (2009). The art of lean software development: a practical and incremental approach. ” O’Reilly Media, Inc.”

[25] VMware. (2019). Network Virtualization and Security Platform – NSX | VMware. [online] Available at: https://www.vmware.com/uk/products/nsx.html.

[26] Blog. (2021). What Is Leaf-Spine Architecture and How to Design It. [online] Available at: https://community.fs.com/blog/leaf-spine-with-fs-com-switches.html.

[27] ComputerNetworkingNotes. (n.d.). How Switch learns the MAC addresses Explained. [online] Available at: https://www.computernetworkingnotes.com/ccna-study-guide/how-switch-learns-the-mac-addresses-explained.html.

[28] Study CCNA. (2021). What Is Layer 3 Switch and How it Works in Our Network? [online] Available at: https://study-ccna.com/layer-3-switch/.

[29] Simplilearn.com. (2018). Ansible vs. Puppet: The Key Differences to Know. [online] Available at: https://www.simplilearn.com/ansible-vs-puppet-the-key-differences-to-know-article#:~:text=In%20Puppet%2C%20the%20client%20pulls [Accessed 11 Apr. 2023].

[30] Ansible, Red Hat (2019). How Ansible Works | Ansible.com. [online] Ansible.com. Available at: https://www.ansible.com/overview/how-ansible-works.

[31] clairecadman (n.d.). Introduction to Puppet. [online] puppet.com. Available at: https://www.puppet.com/docs/puppet/6/puppet_overview.html.

[32] Anon, (2023). CDP vs LLDP- Which Protocol is Better? | Best Explained 2023. [online] Available at: https://www.nwkings.com/cdp-vs-lldp [Accessed 11 Apr. 2023].

[33] NetworkAcademy.io. (n.d.). Router on a stick (ROAS). [online] Available at: https://www.networkacademy.io/ccna/ethernet/router-on-a-stick.

[34] IT Services | CR-T - Utah. (2020). 10 of the Most Powerful Enterprise Networking Companies in 2020. [online] Available at: https://www.cr-t.com/blog/10-of-the-most-powerful-enterprise-networking-companies-in-2020/.

[35] Simplilearn.com. (2022). What is a MAC Address, and How to Find It? | Simplilearn. [online] Available at: https://www.simplilearn.com/what-is-mac-address-how-to-find-it-article#:~:text=The%20MAC%20is%20globally%20unique [Accessed 24 Apr. 2023].

[36] Programiz (2019). Dijkstra's Algorithm. [online] Programiz.com. Available at: https://www.programiz.com/dsa/dijkstra-algorithm.

[37] docs.paramiko.org. (n.d.). Welcome to Paramiko's documentation! — Paramiko documentation. [online] Available at: https://docs.paramiko.org/en/stable/.

[38] PyPI. (2019). netmiko. [online] Available at: https://pypi.org/project/netmiko/.

[39] pexpect.readthedocs.io. (n.d.). Pexpect version 4.8 — Pexpect 4.8 documentation. [online] Available at: https://pexpect.readthedocs.io/en/stable/.

[40] IT Services | CR-T - Utah. (2020). 10 of the Most Powerful Enterprise Networking Companies in 2020. [online] Available at: https://www.cr-t.com/blog/10-of-the-most-powerful-enterprise-networking-companies-in-2020/.

[41] learningnetwork.cisco.com. (n.d.). Cisco Learning Network. [online] Available at: https://learningnetwork.cisco.com/s/virl.

[42] www.eve-ng.net. (n.d.). Home -. [online] Available at: https://www.eve-ng.net/.

[43] netmiko. (n.d.). Supported Platforms. [online] Available at: http://ktbyers.github.io/netmiko/PLATFORMS.html [Accessed 25 Apr. 2023].

[44] mother.github.io. (n.d.). Getting Started with GNS3 | GNS3 Documentation. [online] Available at: https://docs.gns3.com/docs/.

[45] https://www.vmware.com/uk/products/workstation-pro.html

[46] er.educause.edu. (n.d.). The Hidden Value of IPv4 Addresses. [online] Available at: https://er.educause.edu/articles/sponsored/2022/8/the-hidden-value-of-ipv4-addresses#:~:text=IPv4%20is%20still%20the%20dominant.