K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to the University of Mumbai)

CHAPTER 1
INTRODUCTION

This chapter briefly describes network automation concepts, the problem statement, and enterprises to network automation and various types of automation and objective.

The technology behind automation enables a process or procedure to be carried out with little assistance from people. Automation or automatic control is the use of various control systems for machinery, factory operations, boilers, heat-treating ovens, and other applications with little or no human intervention. [1].

Like this, there are various levels of networking automation, ranging from automating tasks in a single device to automating workflows, such as backing up configurations or configuring a routing protocol across several devices, to higher levels of the hierarchy known as cross-domain automation.

Computer networks have expanded in complexity and dynamism. The difficulty for computer network providers then arises from the availability and dependability of network devices. Network engineers utilize a well-known utility as a secured shell to set up network equipment (SSH). However, the manual configuration takes time because it requires entering passwords, logging in and out, and other repetitive procedures for each device.

An application programming interface (API) for network automation can cut down on the repetition and time required for network maintenance [2]. One of the tasks is to keep an eye on the network for vulnerabilities. The automation network can be used to configure users as well as alter static and dynamic routing. Thus, we may say that network automation manages network devices through the application of programming logic, enabling network managers to configure network devices automatically.

Begin constructing automation at the device level by establishing tasks to automate the essential operations and growing it up from there to the Domain level [3]. Any task that is clearly defined and often repeated can be automated. Device Automation refers to the grouping of these tasks. It has been utilized for fault management and service level monitoring, but as the demands of the business have increased, so have the problems and opportunities.

Up until recently, network engineers had to employ time-consuming techniques and be familiar with proprietary protocols and technology.

Network engineers created Network Automation techniques to automate repetitive daily processes to cut costs and increase efficiency. With the help of significant networking firms, an open-source community was established with the aim of implementing automation applications, primarily using common programming languages like Python and standard interfaces (SSH, REST).

By utilizing industry-standard protocols like OpenFlow, SDN was able to minimize vendor dependence to a certain extent. A pioneer of the software-defined network revolution, OpenFlow is a low-level hardware-based protocol. It allows the network device's control plane to be deconstructed from the data plane.

The control plane is also engaged in the network's data flow. The manager that directs traffic through a device is referred to as the data plane.

SDN makes use of the distinction between the control plane and the data plane, removes the control plane from the device, and assumes control of how the network acts so that the network devices only concentrate on forwarding.

IT administrators may simply manage all network components and provision network services using an SDN application. Some of the benefits of SDN include the ability to automate networks, program traffic, and boost agility. On the other hand, there is regrettably no backward compatibility with legacynetworks or non-SDN networks.

1.1 Challenges in Network Configuration

The main reason for an organization to adopt network automation is to reduce the time that is needed to maintain and deploy changes on the network. Although time is crucial not all organizations choose to move to network automation. Each network nowadays is unique and the same goes for the network devices, this is something that discourages enterprises from moving to automation on their network because it usually involves the upgrade of the current equipment or moving for example on SDN because these unique devices are sometimes difficult if not impossible to be part of an automated network. Another hitch in moving towards network automation is the lack of a vendor-free standardized schema in conjunction with an affordable environment for testing. Except for time-saving networks, automation helps on troubleshooting problems on a network. The network administrators had to manually configure each device through CLI and in the case of a change that had to be done in all the devices, such as an addition of a new VLAN, they had to go to its device and set it up. This was not only time-consuming, but it also maximizes the possibility of an error. Moreover, it is dangerous to apply changes on a network during work hours and there are enterprises that work every day and there is a tight window, usually on holidays, for applying changes. Surveys that have been conducted showed that the most usual reason for downtime on a network is human

errors. The most common human error in networking is the misconfiguration of network devices. It is a common task of a network administrator to apply an updated configuration file to a bunch of network devices.

1.2 Automating Network Operations

Network automation does not apply only to configuring devices, on the contrary, the most significant part of network automation that helps to reduce human errors is that it gives administrators the ability to automate procedures that runs compliance and validation checks against the current configuration or any configuration that is about to be deployed [4].

As a result, this reduces delivery times of the network changes and the risk of an outage or service distribution also minimizes the possibility of human error and ensures the alignment with the network policies [17].

Another procedure that can be automated is troubleshooting. When a problem arises on a network the first step in solving the problem is to collect information. The gathering of data from each device can be laborious and take a lot of time, but this is necessary because typically during this time, the network, or a section of it, is down.

We can automate all the instructions required to collect the information required for troubleshooting and have real-time access to this information by using network automation. We may check them in real time by programmatically retrieving this information. A third part of network automation is known as automated monitoring. It involves checking real-time information and deciding what actions to take if certain parameters, such as MTU, change. Hardware failure-related disruptions can be avoided with the aid of automated monitoring.

1.3 Problem Statement

Due to the COVID-19 pandemic, information technology is at an all-time high. Most of the time, organizations ranging from business to education employ this technology. Computer networks are used in information technology to organize and integrate data. For networked devices, a manageable network configuration will be simpler to maintain and less likely to cause communication issues. In the past, network managers had to manually set up every network device. This procedure is ineffective and time-consuming. Although it is relatively slow, automated network configuration can get around the repetitive process.

1.4 Objectives

- **Discard manual labour:** By automating the network, repeatable, predictable network changes can take the place of manual labour.
- **Secure Remote Connection:** SSH (Secure Shell) is a secure method for remote access as it includes authentication and encryption.
- **Accelerate change:** Execute network changes that were previously infrequently done because they required physical labour, a lot of time, and a lot of resources.
- **Permitted access:** Rather than all users having the credentials only the authentic user can access it.

1.5 Network Automation Drawbacks

Although there are numerous benefits to network automation, as with everything in life, there are also drawbacks. This is also true for businesses.

Network engineers must have a thorough understanding of how network devices operate and behave in order to introduce automation to network devices. Automating a network while still learning about the devices that make up the network may result in serious failures and downtime.

Network automation reduces but does not eliminate human error. Compared to a human committing a configuration error in one device, the impact of an automated error is substantially greater.

For instance, Google Services experienced a significant outage that was both broad in scope and long-lasting (it lasted roughly 4 hours). According to the incident report, the outage was caused by a combination of two typically benign misconfigurations and a specific software issue. First, in the impacted regions, network control plane jobs and the infrastructure that supports them were set up to terminate in the case of a maintenance event. The network control plane's many instances of cluster management software were also designated as being qualified for inclusion in a specific, rather infrequent maintenance event type. Thirdly, a specific fault in the software that scheduled maintenance allowed it to simultaneously reschedule numerous independent software clusters, critically even if those clusters were in various physical locations.

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to the University of Mumbai)

CHAPTER 2

LITERATURE SURVEY

This chapter presents the literature review of the project implemented. It provides excerpts from various papers published on the field.

In **'Performance analysis on network automation interaction with network devices using Python'** [5] the primary goal was to demonstrate how automation is growing in popularity due to its many advantages, particularly given the growing number of network devices. Network automation offers thousands of advantages for businesses, enables the deployment of numerous devices in a matter of minutes, and removes the possibility of human error. The word automatic is defined as behaving or operating in a manner independent of external influence and human control.

In **'Network Automation and Abstraction using Python Programming Methods'** [6] we investigated network programmability based on scripting techniques and traditional programming languages. It has been enhanced and inspired by SDNs. To speed up equipment configuration and make maintenance simpler, this paper presented some novel techniques for automating the configuration of network devices: by identifying and addressing security flaws. These approaches, which enable the unitary control of an expanding number of devices, are what networks will look like in the future. It is possible to automate the configuration and monitoring of any device, irrespective of the vendor, on SDN devices as well as other networking solutions.

In '**As-RaD System as a Design Model of the Network Automation Configuration System Based on the REST API and Django Framework** [7] they put forth a different design for a network automation system. To improve the performance of network automation, the model system was designed using a controller application that utilized the REST API (Representational State Transfer Application Programming Interface) architecture and was created using the Django framework and Python programming language. The As-RaD System design paradigm automates networking operations with a simple GUI and employs a web-based application for maintenance. The Cisco CSR1000V is one of the network devices utilized in this study since it allows REST API connectivity to manage its network configuration and can be installed on the server as well.

In **'Network Automation Methodology for detecting Rogue Switch'** [8] it provided a solution to detect malicious switches on a network using a GNS3 network emulator, Python for automation, Cisco iOS configurations, Wireshark packet analyzer, and Docker. On the other hand, an unmanaged switch operates

as a "plug and play" gadget. It merely enables the devices to communicate with one another and is pre-configured. They lack an IP address, and some even lack a MAC address, making them incredibly challenging to track. The objective is accomplished by regularly employing packet analyzers to filter and analyze network traffic for any broadcast storms or new ARP packets, and by using automated techniques to successfully trace the malicious host linked to the rogue switch. The method for detecting a rogue switch on a wired network had been successfully demonstrated in this article, but there is plenty of room for improvement in the future to turn this solution into a fully functional centralized management tool for efficient network management with advanced fault detection features.

In **'Network Automation and Abstraction using Python'** [9] network programmability is a trend that is based on scripting techniques and traditional programming languages used for managing and monitoring network parts. A network's diverse composition and the total number of devices are both rising. The conventional techniques for configuring network equipment take time, especially when we consider the vendor-specific expertise required. Using open standards like OpenFlow, the Software Defined Networks (SDN) idea seeks to end vendor dependence. The "traditional" non-SDN legacy networks, on the other hand, must stay up and adapt to dynamic network changes. Using methods that are simpler to follow and apply on a broad scale, network automation can reduce operational expenses (OPEX) by improving not just the time required to configure network devices but also the effectiveness of network maintenance. Network programmability makes it possible to configure infrastructure in a reliable and dynamic way by automating deployments, streamlining the network, and minimizing human error. All significant vendors, including Cisco, began promoting the ability of networks to be configured via software. All new automation implementations are built using standardized interfaces like Secure Shell SSH or even RESTful web services, as well as general programming languages like Python and Java.

In **'Network Automation using Ansible for EIGRP Network'** [10] network automation has evolved into a solution that emphasizes efficiency in all areas. Furthermore, communication and computer networks rely on a platform that provides the necessary technological infrastructure for packet transfer through the Internet using routing protocols. The Enhanced Interior Gateway Routing Protocol (EIGRP) is a hybrid routing protocol that combines the properties of both distance-vector and link-state routing methods. The traditional technique to configure EIGRP was inefficient and required repeated processes compared to the network automation concept. Network automation helps to assist network administrators in automating and verifying the EIGRP configuration using scripting. In order to configure EIGRP routing and advanced settings in the GNS3 environment, this study used network automation using Ansible. The EIGRP routing protocol, a default static route, and advanced EIGRP settings were the main topics of this research. Automated scripting to setup IP addresses to the interfaces was also covered. Ansible executed the

programming on Network Automation Docker and sent the router configurations. SSH was used by the network automation docker to connect to other routers.

In order to confirm the accuracy of the EIGRP setups, the running configuration between the conventional technique and automated scripting in the Ansible playbook was compared throughout the testing phase. The results demonstrated that Ansible successfully and error-free distributed the configuration to the routers. Network managers can minimize human error, cut down on time-consuming tasks, and offer device visibility throughout the network environment with the aid of Ansible. EIGRP authentication and hardening method implementation can increase network security levels for future research.

In '**Network Automation: Enhancing Operational Efficiency Across the Network Environment**' [] paper implemented network automation using Ansible to configure BGP routing and advanced configuration in the live network environment. This study is focused on automated scripting to configure IP Addresses to the interfaces, BGP routing protocol, and configuration backup. Ansible ran the scripting on Network Automation Docker and pushed the configurations to the routers. The network automation controller communicated with other routers via SSH. The findings show that Ansible has successfully deployed the configuration to the routers with no errors. Ansible can help network administrators minimize human mistakes, and time-consuming, and enable device visibility across the network environment. This study can help network administrators minimize human mistakes, reduce time-consuming and enable device visibility across the network environment. Implementing different types of authentications and hardening processes can enhance the network security level for future studies.

This '**Network Automation using Ansible for Cisco Routers Basic Configuration**' [] paper is illustrating a method of configuring network devices by using automation, reducing time for equipment configuration and easier maintenance. It uses Ansible in the Ubuntu environment as the controller, and Cisco routers as the managed nodes. These methods represent the future of networks, allowing the management of an increased number of devices in a unitary way. They have demonstrated the importance of automation in a network this is not aware the of OpenFlow SDN protocol and have demonstrated that using Ansible, network engineers do not need to configure by themselves each individual device, they just need to create the proper infrastructure and by implementing automation scripting. The network controllability becomes easier and changes can be faster deployed, maybe even automatically, as a response to events that take place in the network. So the legacy network elements become similar to SDNs.

CHAPTER 3

TECHNOLOGIES AND TECHNIQUES

Important libraries that were utilized in the code for the applications will be briefly described in this chapter, along with an explanation of why they were chosen over alternatives.

3.1 Program Development Information

The application's purpose is to demonstrate different ways to connect and configure network devices. The implementation will be done in Python and the code will follow most of the pep 8 instructions. PEP 8 is a style guide for Python code that gives coding conventions. Following a style guide when coding an application improves the readability of code, and makes it consistent and well-maintained.

Some of the recommendations of the PEP 8 style guide that was followed are below:

➢ Indentation 4 spaces per indentation level, continuation lines align wrapped elements

➢ Blank lines two lines for top-level functions single line for methods

➢ Imports each import on a separate line except if it is in the format "from library importsomething". All the imports are at the top of the code.

➢ Snake case naming style refers to the style of variable name writing. Each word starts with a lowercase letter and the space is replaced with an underscore.

Secure methods will be used, for example, the connectivity with the network devices will be accomplished through SSH. By using methods and functions of the imported modules the application's security feature is enhanced. when the user is prompted to input a password to login in the application, the password is masked with asterisks. Multithreading will be used to speed up the execution of the code.

3.2 Python Modules and Libraries

3.2.1 Netmiko

It is an open-source multi-vendor library; it enables the configuration of devices from several vendors using Python. Cisco IOS, Juniper, Arista, HP, and Linux are a few of the platforms that Netmiko supports. It might also work with other vendors including Alcatel, Huawei, and Ubiquity, though there hasn't been much testing done with them. To make SSH connections to network devices less complicated, more adaptable, and user-friendly, Netmiko operates on top of Paramiko. Although Netmiko is simple to use and it only supports a few suppliers' devices. Contrarily, Paramiko can be used to communicate with any SSH-

compatible device. Paramiko is an alternative option for devices that do not support APIs. It is a multi-vendor SSH Python library that simplifies the process of connecting to network devices via SSH. This library adds vendor-specific logic to Paramiko, which is the de-facto SSH library in Python.

For example, to issue a command to a network device and obtain the returned data, you would typically need to:

- Set up an SSH connection.
- Understand the command prompt string for the given device and vendor.
- Issue the command, and then perform the necessary string handling to understand when the device has finished sending its response.
- Handle the various nuances that come with string handling such as paging and terminal widths.

Netmiko abstracts many of these complexities, by providing a Python library with a set of easy-to-use methods.

Below are some of the use cases that Netmiko can be used for:

- **Configuration backups** - Automate the retrieval of the output of the running configuration on a scheduled basis.
- **Security audits** - Run a command to understand if the device is running a vulnerable software version.
- **Automate troubleshooting**- Automate the process of running various commands to troubleshoot an issue.

Features:
- **Structured parsing** - Supports parsing via the TTP, TextFSM and Genie parsing libraries.
- **Multi-vendor** - Supports a large set of multi-vendor devices.
- **Device configuration** - Provides methods for applying configuration from a list of commands or a file of commands.
- **Device config** - Supports various methods for reading configuration from devices.
- **Stability tuning** - Supports various options for ensuring stability for slow devices or network transports

3.2.2 Telnet

Telnet is a network protocol that allows for two-way, collaborative, text-based communication between two computers as well as remote computer access. It uses the TCP/IP networking protocol to create remote sessions in response to user commands.

Telnet's workings

A remote computer, usually a server, can have its command line opened via the client-server protocol called Telnet. This utility allows users to ping a port and determine whether it is open. Telnet emulates a physical terminal connected to a machine by employing what is known as a virtual terminal connection emulator, or an abstract instance of a connection to a computer. For users sending data files, FTP can be utilised in addition to Telnet. Telnet, also allows users to connect remotely to a machine. To access the distant computer, they are required to enter their login and password combination, which permits the execution of command lines as though they were physically present in front of the computer. Users IP addresses will always match the computer they are currently signed into, not the one they used to physically connect, regardless of where they are physically located.

Using Telnet

On a server, Telnet can be used for several tasks, such as file editing, running different programmes, and checking email. Some servers allow remote connections via Telnet so users can access open data to access basic games or check the weather. Many of these features are still functional in older systems that require access to specific data. Telnet allows users to connect to any programme, including web servers and ports, that uses text-based, unencrypted protocols. The telnet connection will ping the port to determine whether it is open or not when users open a command prompt on the remote machine, type the word telnet, and the name or IP address of the remote machine. A blank screen indicates that a port is open, whereas an error message indicating that a port is connecting indicates that a port is closed.

Security

Telnet is an insecure, unencrypted protocol. Anyone who keeps an eye on a user's connection can read their plaintext username, password, and other private information entered during a Telnet session. With this knowledge, access to the user's device is possible.

Protocols associated to SSH

Some contemporary systems only permit command-line connections via a virtual private network (VPN) or using Secure Shell (SSH), an encrypted tool akin to Telnet. Many professional organisations demand the

use of SSH, PuTTY, or other options instead of Telnet due to security concerns. The main reason SSH is preferred over other alternatives is that it encrypts all data traveling across the communication connection.

Benefits of Telnet

- It gives someone remote access to their computer system.
- Telnet gives the user greater access with fewer data transmission issues.
- Telnet is very time-efficient.
- With the use of telnet, two systems with different operating systems can be connected.

Negative aspects of Telnet

It is somewhat intricate, making it challenging for beginners to understand.

The fact that the data is transferred here in plain text makes it less secure.

Because the remote and local devices are not properly interconnected, some capabilities are disabled.

The three following modes are used by most telnet implementations

1. Default option: This option is utilized if no other modes are selected. In this mode, the client does the echoing. In this mode, when a character is entered, the client displays it on the screen but does not send it until the entire line has been entered.

2. Character Mode: In this mode, each character entered by the user is transmitted to the server by the client. Characters are typically echoed back by the server in this type of mode and displayed on the client's screen.

3. Line Mode: Client-side line editing, such as character erasure and echoing, is available. The entire line will be sent from the client to the server.

3.2.3 IP Address

The addresses that the user enters the program will be checked to ensure that they are legitimate IP addresses using this module library. The power of Python is demonstrated by modules like these. Using this module, it just takes three to four lines to determine whether an IP address is genuine or not. Ip address provides the capabilities to create, manipulate and operate on IPv4 and IPv6 addresses and networks. The functions and classes in this module make it straightforward to handle various tasks related to IP addresses, including checking whether two hosts are on the same subnet, iterating over all hosts in a particular subnet, checking whether a string represents a valid IP address or network definition.

3.2.4 Tkinter

This is the module library that will be used to create the application's GUI. Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tkinter GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps:

- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the widgets to the GUI application.
- Enter the main event loop to act against each event triggered by the user.

Here are some common use cases for Tkinter

- **Creating windows and dialog boxes**: Tkinter can be used to create windows and dialog boxes that allow users to interact with your program. These can be used to display information, gather input, or present options to the user.
- **Building a GUI for a desktop application**: Tkinter can be used to create the interface for a desktop application, including buttons, menus, and other interactive elements.
- **Adding a GUI to a command-line program**: Tkinter can be used to add a GUI to a command-line program, making it easier for users to interact with the program and input arguments.
- **Creating custom widgets**: Tkinter includes a variety of built-in widgets, such as buttons, labels, and text boxes, but it also allows you to create your own custom widgets.
- **Prototyping a GUI**: Tkinter can be used to quickly prototype a GUI, allowing us to test and iterate on different design ideas before committing to a final implementation.

In summary, Tkinter is a useful tool for creating a wide variety of GUIs, including windows, dialog boxes, and custom widgets. It is particularly well-suited for building desktop applications and adding a GUI to command-line programs.

What are Widgets?

Widgets in Tkinter are the elements of a GUI application that provides various controls (such as Labels, Buttons, Combo Boxes, Checkboxes, menu bars, Radio Buttons, and many more) to users to interact with the application. These are also referred to as the fundamental structure of the Tkinter program.

3.2.5   SQLITE

It is a C library that provides a lightweight disk-based database that does not require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage.

Python SQLite3 module is used to integrate the SQLite database with Python. It is a standardized Python DBI API 2.0 and provides a straightforward and simple-to-use interface for interacting with SQLite databases.

This API opens a connection to the SQLite database file. You can use ": memory:" to open a database connection to a database that resides in RAM instead of on disk. If the database is opened successfully, it returns a connection object. When a database is accessed by multiple connections, and one of the processes modifies the database, the SQLite database is locked until that transaction is committed. The timeout parameter specifies how long the connection should wait for the lock to go away until raising an exception. The default for the timeout parameter is 5.0 seconds. If the given database name does not exist then this call will create the database. You can specify the filename with the required path as well if you want to create a database anywhere else except in the current directory.

3.2.6   Threading

A thread is a unique execution flow. This implies that there will be two simultaneous events in your program. However, most Python 3 implementations only give the impression that the various threads are running concurrently. It can be tempting to imagine threading as having two (or more) distinct processors running your program, each of which is carrying out a separate task concurrently. The threads will only be active one at a time, even if they are running on different CPUs.

A non-standard Python implementation, writing some of your code in another language, or using multiprocessing, which has some additional overhead, are required to get many tasks operating at once. Threading might not speed up all tasks due to how the C Python implementation of Python functions. Threading is an excellent option for tasks that spend a significant amount of time waiting for outside events. It is possible that problems that heavily rely on the CPU and spend little time waiting for outside events are not at all faster. This holds true for Python code that executes on the default C Python implementation.

3.3 Various Routing Configurations

Routing is one of the most essential procedures in data communication. It ensures that data travels from one network to another with optimal speed and minimal delay, and that its integrity is maintained in the process.

Broadly, routing is performed in two different ways:

- Dynamic routing continuously updates its routing table with paths and their cost/metric, while making optimal routing decisions based on changing network operating environments.

- Static routing performs routing decisions with preconfigured routes in the routing table, which can be changed manually only by administrators. Static routes are normally implemented in those situations where the choices in route selection are limited, or there is only a single default route available. Also, static routing can be used if you have only a few devices for route configuration and there is no need for a route change in the future.

3.3.1 Static Routing

The routing technique which needs a manual configuration is static routing. Most network administrators rely on observing static routing. The usage of this routing will be high in the places that follow constant parameters in-network and environment. The static routes are fixed & do not change if the network occurs or is reconfigured. It is used on a router to maximize the routing efficiency & to provide backup if other information fails to be exchanged.

It exploits the paths between the two ways, and they cannot automatically be updated. Thus, we must manually reconfigure static routes when the network changes. It uses low bandwidth. It can be used in those areas where the network traffic is predictable & designed. It cannot be used in the vast and continuously changing network because it can't react to the network change.

It is applicable for small networks and easy to configure. The configuration of the system depends on the size of the net. The small networks are accessible to, but as the web grows, applying changes to all the routes can be difficult. There are various incidents where a static route is the most logical & efficient method for the path. It is the opposite of dynamic routing. Dynamic routing is a system in which routers will automatically adjust to the changes in network traffic. It is considered the purest form of routing & requires extensive manual processes.

Even in typical traffic conditions or during a failure in the network environments, they will not get rerouted because they cannot change or tune by themselves even though we have certain choices to save whenever there are issues. The work of the router is to forward packets from the source device to the destination

device. In between there may be several routers. The router uses a database known as the routing table to forward these packets.
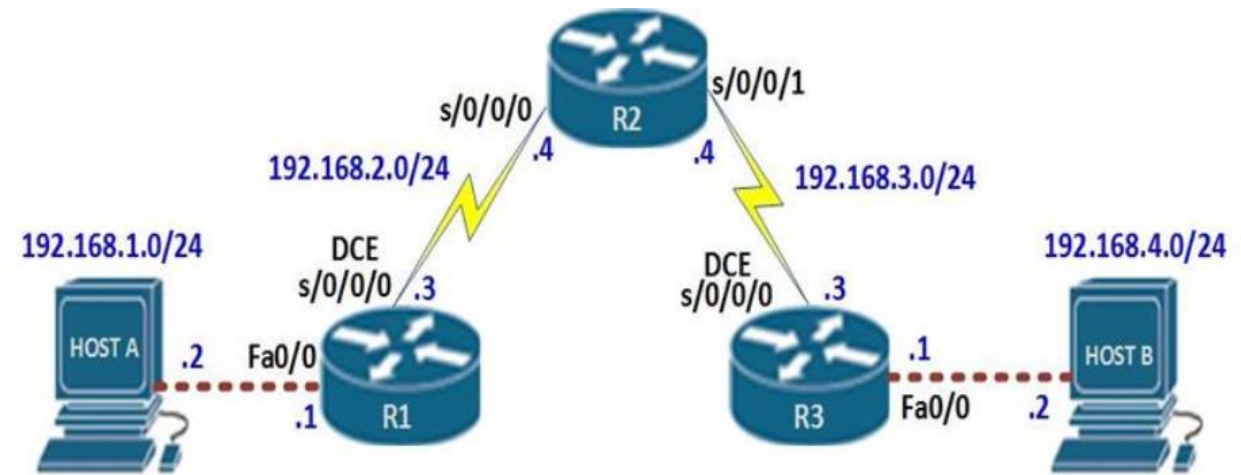
Figure 3.1: Static Routing

In our scenario, this means that;

Host A can ping R1

R1 can ping R2's s0/0/0 interface but not interface s0/0/1

R2 can ping R1's s0/0/0 interface but not interface fa0/0 or HOST A

R2 can ping R3's s0/0/0 interface but not interface fa0/0 or HOST B

R3 can ping R2's s0/0/1 interface but not interface s0/0/0

HOST B can ping R3.

Neither host can ping the other

R1 and R3 cannot ping each other.

- Static routes are one way we can communicate to remote networks. In production networks, static routes are mainly configured when routing from a particular network to a stub network.
- Stub networks are networks that can only be accessed through one point or one interface.
- The 192.168.1.0/24 and 192.168.4.0/24 networks are stub networks. This means that for hosts in these network segments only have one way to communicate with other hosts, which is R1 and R3 for the 192.168.1.0/24 and 192.168.4.0/24 networks respectively.

3.3.2   RIP Routing

Routing Information Protocol (RIP) is a dynamic routing protocol that uses hop count as a routing metric to find the best path between the source and the destination network. It is a distance-vector routing protocol and works on the Network layer of the OSI model.

Hop Count

Hop count is the number of routers occurring in between the source and destination network. The path with the lowest hop count is considered as the best route to reach a network and therefore placed in the routing table. RIP prevents routing loops by limiting the number of hops allowed in a path from source and destination.

Features of RIP

- Updates of the network are exchanged periodically.
- Updates (routing information) are always broadcast.
- Full routing tables are sent in updates.
- Routers always trust routing information received from neighbour routers.
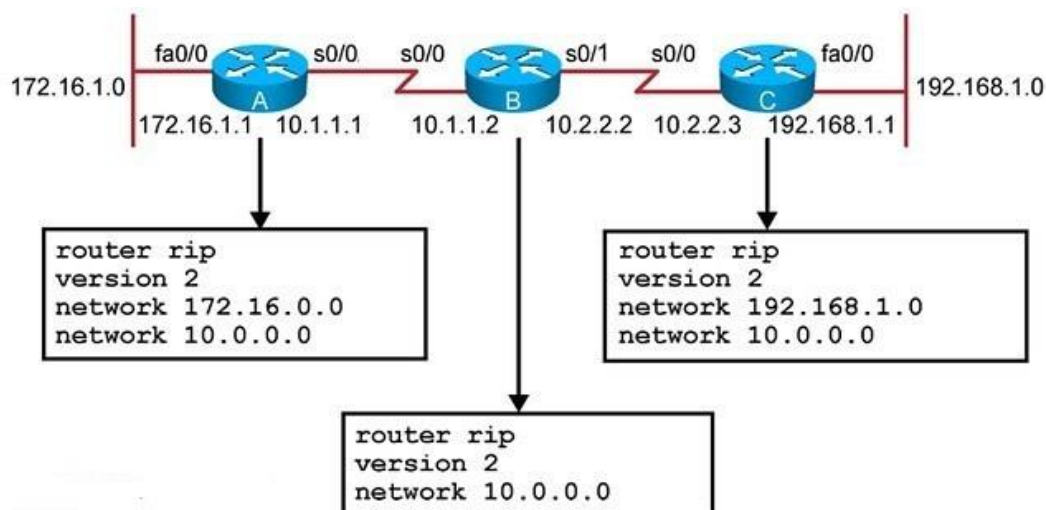
Figure 3.2: RIP Routing

Normal utilization of RIP:

- **Small to medium-sized networks**: RIP is normally utilized in little to medium-sized networks that have moderately basic directing prerequisites. It is not difficult to design and requires little support, which goes with it a famous decision for little organizations.

- **Legacy organizations:** RIP is as yet utilized in some heritage networks that were set up before further developed steering conventions were created. These organizations may not merit the expense and exertion of overhauling, so they keep on involving RIP as their directing convention.

- **Lab conditions:** RIP is much of the time utilized in lab conditions for testing and learning purposes. A basic convention is not difficult to set up, which pursues it a decent decision for instructive purposes.

- **Backup or repetitive steering:** In certain organizations, RIP might be utilized as a reinforcement or excess directing convention, on the off chance that the essential steering convention falls flat or encounters issues. RIP isn't generally so productive as other directing conventions; however, it may very well be helpful as a reinforcement if there should be an occurrence of crisis.

3.3.3 OSPF Routing

Open Shortest Path First (OSPF) is a link-state routing protocol that was developed for IP networks and is based on the Shortest Path First (SPF) algorithm. OSPF is an Interior Gateway Protocol (IGP).

In an OSPF network, routers or systems within the same area maintain an identical link-state database that describes the topology of the area. Each router or system in the area generates its link-state database from the link-state advertisements (LSAs) that it receives from all the other routers or systems in the same area and the LSAs that it generates.

An LSA is a packet that contains information about neighbours and path costs. Based on the link-state database, each router or system calculates a shortest-path spanning tree, with itself as the root, using the SPF algorithm.
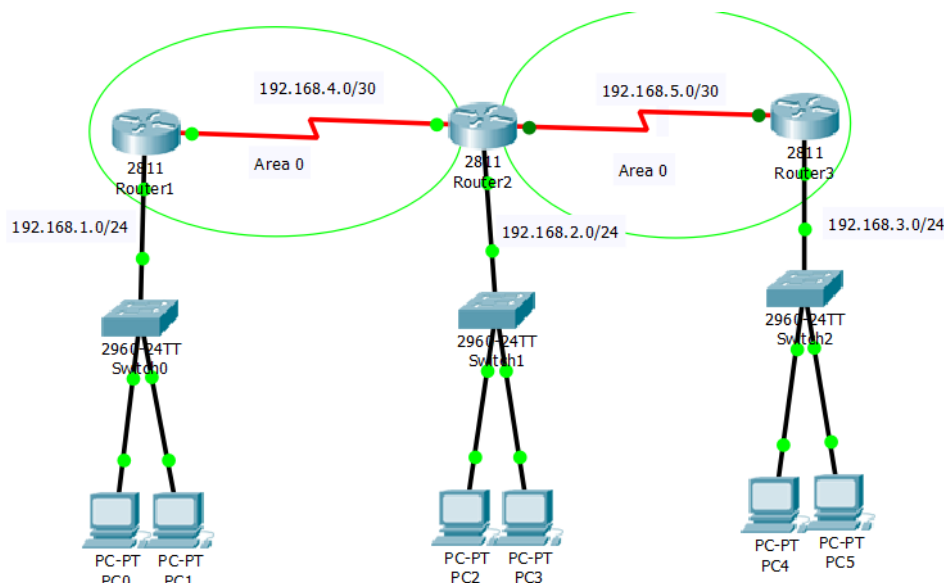
Figure 3.1: OSPF Routing

OSPF has the following key advantages:

- Compared with distance-vector routing protocols such as the Routing Information Protocol (RIP), OSPF is more suitable for serving large, heterogeneous internet works. OSPF can recalculate the routes in a short amount of time when the network topology changes.

- With OSPF, you can divide an Autonomous System (AS) into areas and keep area topologies separate to decrease the OSPF routing traffic and the size of the link-state database of each area.

- OSPF provides equal-cost multipath routing. You can add duplicate routes to the TCP stack using different next hops.

It is a widely used and supported routing protocol. It is an intradomain protocol, which means that it is used within an area or a network. It is an interior gateway protocol that has been designed within a single autonomous system. It is based on a link-state routing algorithm in which each router contains the information of every domain, and based on this information, it determines the shortest path. The OSPF achieves this by learning about every router and subnet within the entire network. Every router contains the same information about the network. The way router learns this information by sending LSA (Link State Advertisements). These LSAs contain information about every router, subnet, and other networking information. Once the LSAs have been flooded, the OSPF stores the information in a link-state database known as LSDB. The main goal is to have the same information about every router in an LSDB.

3.3.4 EIGRP Routing

Enhanced Interior Gateway Routing Protocol (EIGRP) is a dynamic routing network-layer Protocol that works on protocol number 88. EIGRP supports classless routing, VLSM, route summarization, load balancing, and many other useful features. It is a Cisco proprietary protocol, so all routers in a network that is running EIGRP must be Cisco routers but now EIGRP is moving towards becoming an open standard protocol.

Compared to prior network protocols, EIGRP enables routers to exchange data more quickly. As Cisco planned to link tens of thousands of teleworker home offices to its network, a difficulty comparable to those encountered by other large, spread organizations, the need for a scalable routing protocol got increasingly critical. In order to handle changing requirements without having to modify the wide area network routing design, Cisco set out to create a straightforward protocol, which resulted in the development of EIGRP.
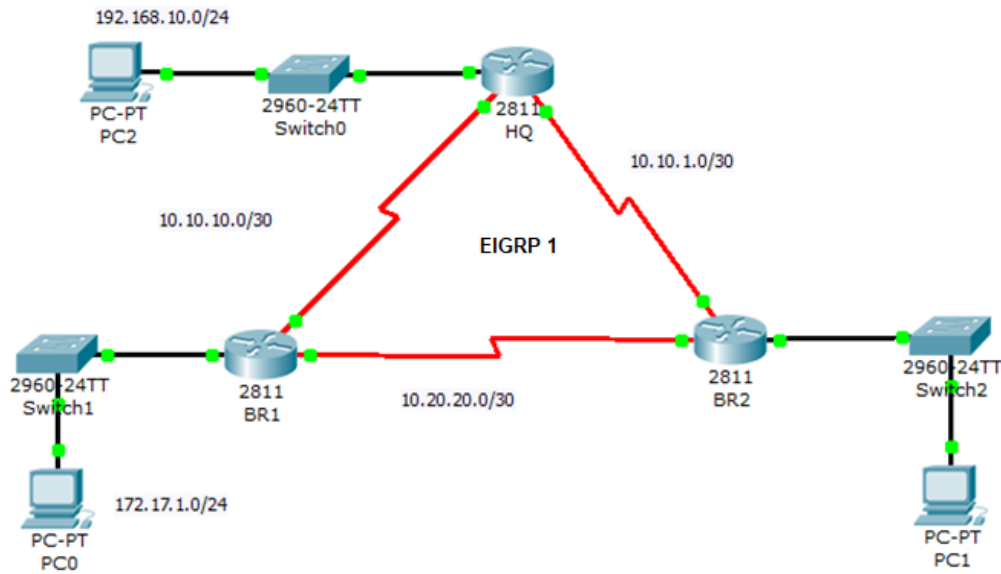
Figure 3.2: EIGRP Routing

What are the fundamental components of EIGRP?

In a network of routers and links, the diffusing update algorithm (DUAL), which is at the heart of EIGRP, selects the most efficient and least expensive routes to reachable destinations. The decision is made based on the distance and whether the final path has loops. The procedure keeps track of the routes communicated to its neighbors and retains the information required to use the DUAL finite state machine to determine the least expensive route.

EIGRP further uses a reliable transport method to guarantee that all EIGRP packets are sent to neighbors in the correct sequence. The transport enables the simultaneous transmission of packets via multicast and unicast.

Features EIGRP offers

- Automatic redistribution of routes between IGRP and EIGRP.
- Ability to turn off and on EIGRP and IGRP on individual interfaces of the router.
- Incremental Updates that save network bandwidth and speed convergence.
- Reduced router CPU load, as compared to IGRP.
- EIGRP prevents routing loops on the network
- Supports variable length subnet masks (VLSM)
- Automatic Route Summarization

3.3.5 Comparison Between Routing

RIP Routing	OSPF Routing	EIGRP Routing
RIP stands for Routing Information Protocol	OSPF stands for Open Shortest Path First	EIGRP stands for Enhanced Interior Gateway Routing Protocol
RIP is a distance vector protocol	OSPF is a **link-state** protocol	EIGRP Is derived from Integrated Gateway Routing Protocol
The metric used Is **hop**.	The metrics used are bandwidth and delay.	The metrics used are bandwidth, delay, load, and reliability
RIP uses Distance vector algorithm to calculate the best path	OSPF uses the SPF algorithm to calculate the best path.	EIGRP uses Diffusing update algorithm to calculate the best path
In RIP, networks are not divided into areas or tables.	Routing with OSPF is done in Autonomous System, Areas, Stub Areas and Backbone areas.	Routing with EIGRP is done in Neighbour Tables, Topology tables, and Routing tables.
The max**imum** hop count is 15.	No hop counts.	The **maximum** hop count is 255.

3.3.6 VLAN

Virtual local area networks have become crucial for organizations with complex networking systems. Organizations require solutions that allow them to scale their networks, segment them to increase security measures, and decrease network latency. While LAN is used to connect a group of devices such as computers and printers to a server via cables, VLANs allow multiple LANs and associated devices to communicate via wireless internet.

By using the virtual LAN (VLAN) idea, we can logically segment the devices on layer 2 (data link layer). The broadcast domain is often divided by layer 3 devices; however, switches can also divide the broadcast domain using the VLAN concept.
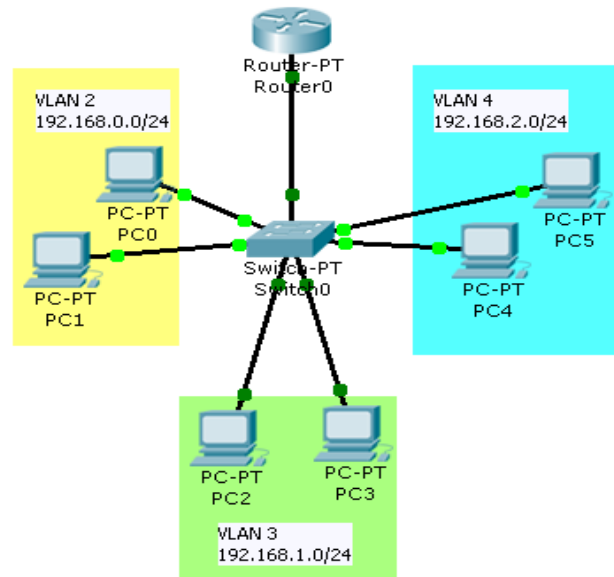
Figure 3.3: VLAN

A broadcast domain is a section of a network where every device within the same broadcast domain will receive a packet that is broadcasted there. All broadcast packets will be received by the devices in the same broadcast domain, but only switches can receive them because routers don't forward broadcast packets. Inter VLAN routing is required to forward packets to different VLANs (from one VLAN to another) or broadcast domains. Different small-size sub-networks are created through VLAN, which are relatively simple to manage.

Below are the five different types of virtual networks:

- **Management virtual local area network**: A smaller network designed to help manage traffic coming from devices, application logging, and monitoring. The major advantage of using this network is improved network security.

- **Voice virtual local area network**: It can be used to manage traffic generated from VoIP equipment or devices such as IP phones, VoIP endpoints, and voice systems.

- **Native virtual local area network**: This network serves as a common identifier on opposing ends of the trunk links to carry untagged traffic generated by a system or devices configured with the native VLAN via a switch port.

- **Default virtual local area network**: A default virtual network with all the access ports until they're assigned to other networks such as voice or management virtual networks.

- **Data virtual local area network**: Divides the entire network into two major groups known as users and devices. This network cannot carry management or voice traffic.

3.3.7 VPN

A virtual private network gives an encrypted and secure path called a tunnel from the source end to the destination end through the internet. The ease of using the internet has increased the use of data transfer through the networks and this leads to the increase in VPN construction by lots of companies. The term VPN refers to a logical connection between two devices i.e., sender and receiver to transfer data. Confidentiality and data integrity is maintained in VPN so that when anyone tries to get a sense of shared data gets nothing in return because the original data is encrypted and they do not have the key to decrypt the data to know what the actual data is. VPN has the implementation of a secure network using switching or routing capabilities. It also uses integrity checking to make sure that VPN is monitoring correctly the packets so that they are not modified maliciously along its route. VPN is basically the combination of tunneling and encryption. VPN is also used for remote access, file sharing, and conferencing.

Virtual private network = Tunnelling + Encryption

Tunneling is the technique of securing the movement of data from one to another network. It is also known as port forwarding and transfers the data through a secure connection. Data is monitored as it travels through the tunnel so that there is not any malicious modification in data through its route. Data is divided into packets and are encrypted as they go through the tunnel again decrypted as they reach the destination.
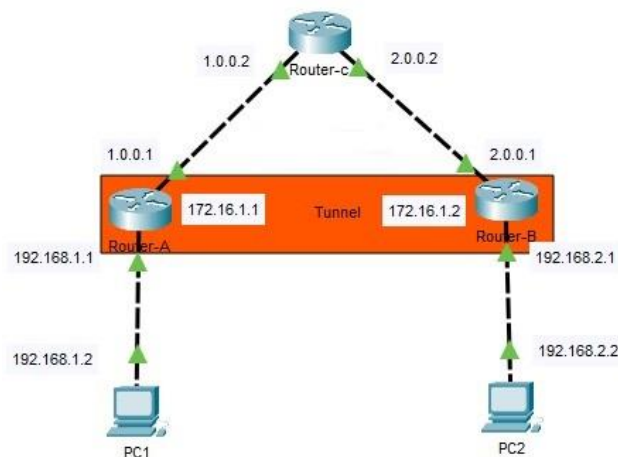
Figure 3.4: VPN Configuration

There are various protocols that provide tunneling some of them are explained below.

- **Point-to-Point Tunneling Protocol (PPTP):** It was developed by Microsoft and is released with Windows 95. It encrypts the data and divides it into packets and then sends it through the tunnel.
- **Layer Two Tunneling Protocol (L2TP):** It is used with IPSec (Internet Protocol Security) to create a more secure tunnel than PPTP. It creates two layers of encryption by wrapping the encrypted data again.
- **Secure Socket Tunneling Protocol (SSTP):** It is available only for Windows operating systems, it transfers the data through a secure sockets layer which is also known as SSL. It is also used with Transport Layer Security (TLS)

3.4 VScode

Visual Studio Code is a free open-source text editor by Microsoft. VS Code is available for Windows, Linux, and macOS. Although the editor is relatively lightweight, it includes some powerful features that have made VS Code one of the most popular development environment tools in recent times.

Visual Studio Code is a source code editor that can be used with a variety of programming languages. A notable feature is an ability to create extensions that add support for new languages, themes, debuggers, time travel debuggers, perform static code analysis, and add code lines using the Language Server Protocol. Visual Studio Code includes multiple extensions for FTP, allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software. Visual Studio Code allows users to set the code page in which the active document is saved, the newline character, and the programming language of the active document. This allows it to be used on any platform, in any locale, and for any given programming language [14].
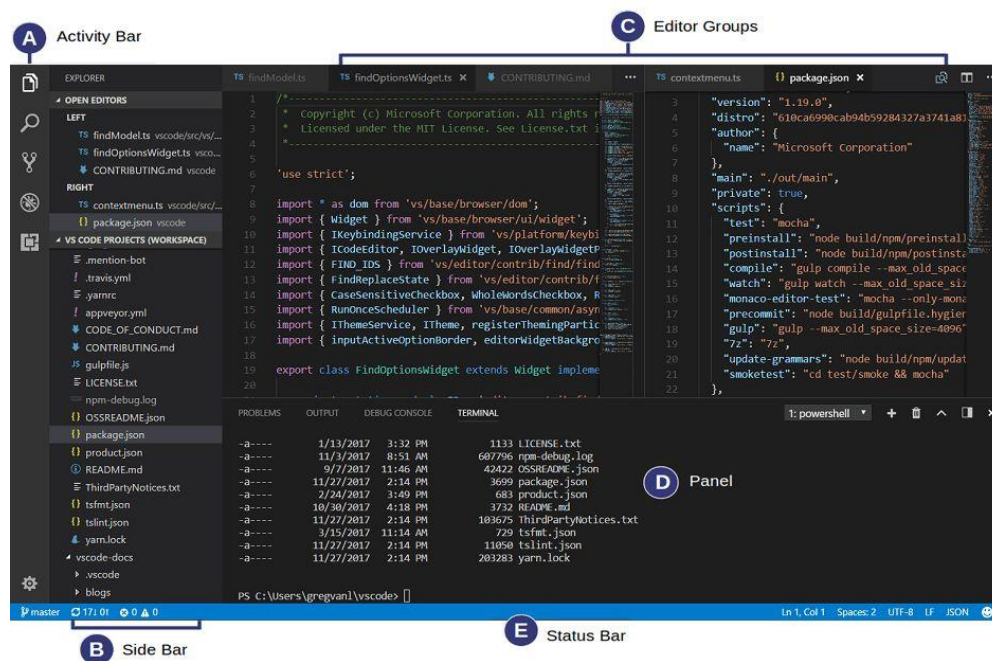
Figure 3.5: VScode

The VS Code user interface allows for a lot of interaction compared to other text editors.

To simplify the user experience, VS Code is divided into five main regions:

- The activity bars
- The sidebar
- Editor groups
- The panel
- The status bars

3.5 GNS3:

GNS3 is software that is used to emulate, configure, and test a network environment. It is open-source free software and can be downloaded from the official website. GNS3 consists of two components. The all-in-one software (GUI) is a graphical user interface and the Virtual Machine (VM) is a server that runs in a virtual environment and provides better topology size and device support. The installation is straightforward, and the default options should be used [15].

Figure 3.6: GNS3 Setup

After the installation and booting of the GNS3 GUI on the Servers Summary window, the PC's name that the GNS3 is installed must be shown and it should also have a green light on the left.
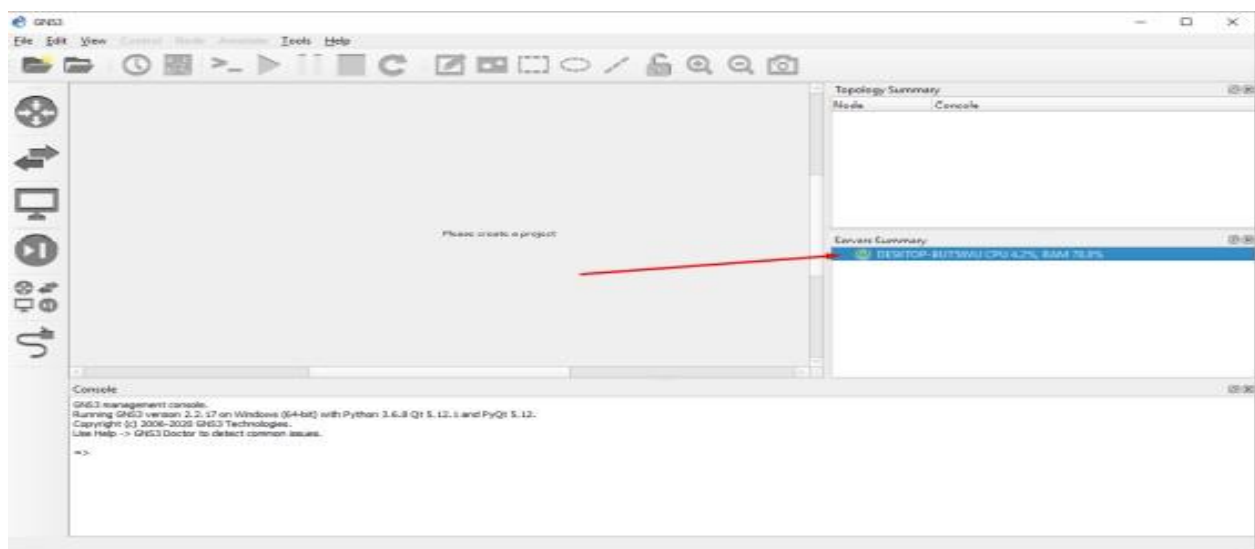
Figure 3.7: GNS3 UI

If there is nothing on that window or the light is red try restarting the GNS3, restarting the PC, or check if the firewall or antivirus stopped the GNS3 service from running. If it is not running, make sure if necessary that permission was given to GNS3 through firewall and antivirus. GNS3 Doctor (Help -> GNS3 Doctor) is a helpful module to check if there is something that blocks GNS3 from working correctly.
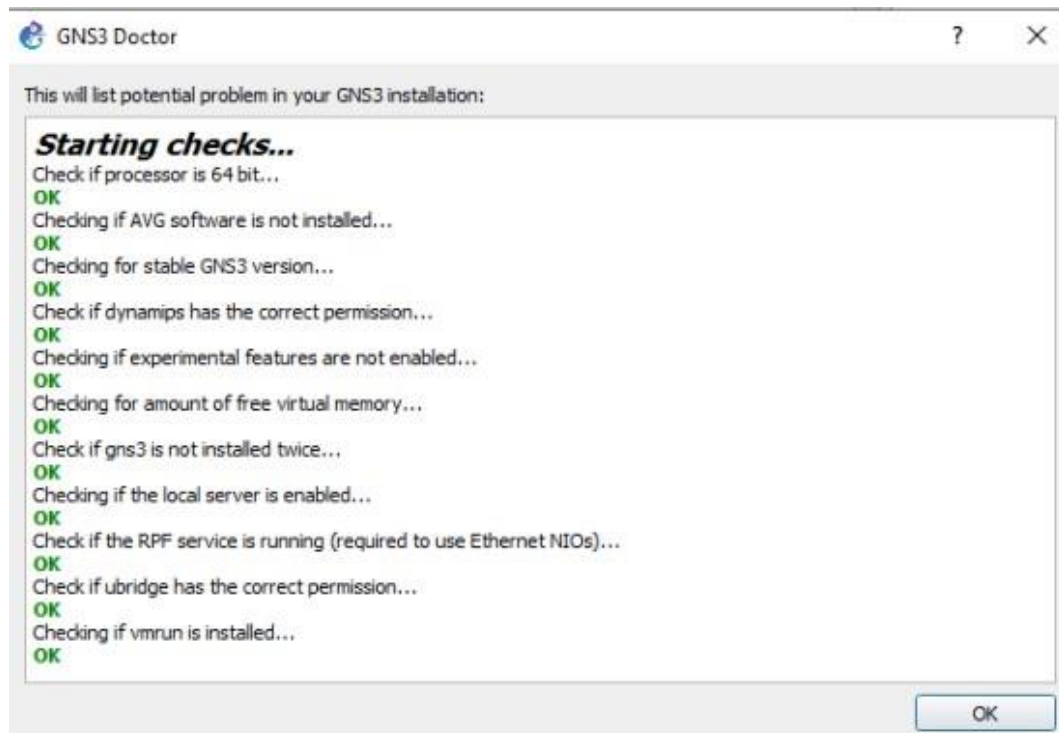
Figure 3.8: GNS3 Doctor

CHAPTER 4

PROGRAM DESIGN

This chapter gives us a brief introduction about the objective of our desired problem statement, the GNS3 working network topology, and finally the various hardware setup & the UI design development.

4.1 Objective

The user will be able to add a device's information (IP, enable password, type, hostname, device username, device password, and enable secret), which will be stored in the SQLite database, through the application's graphical interface. When the credentials are used to open this application, a list of all the saved devices will be accessible to configure. The user will then need to choose which protocols the user wants to configure, and they can also choose which type of connection they want to form either Telnet or SSH. Users can add or remove configurations and see the previous activity done on that device using this application

4.2 Network Topology

We have performed all the above-mentioned protocols using Cisco routers and switches and then verify it on GNS3 Software.
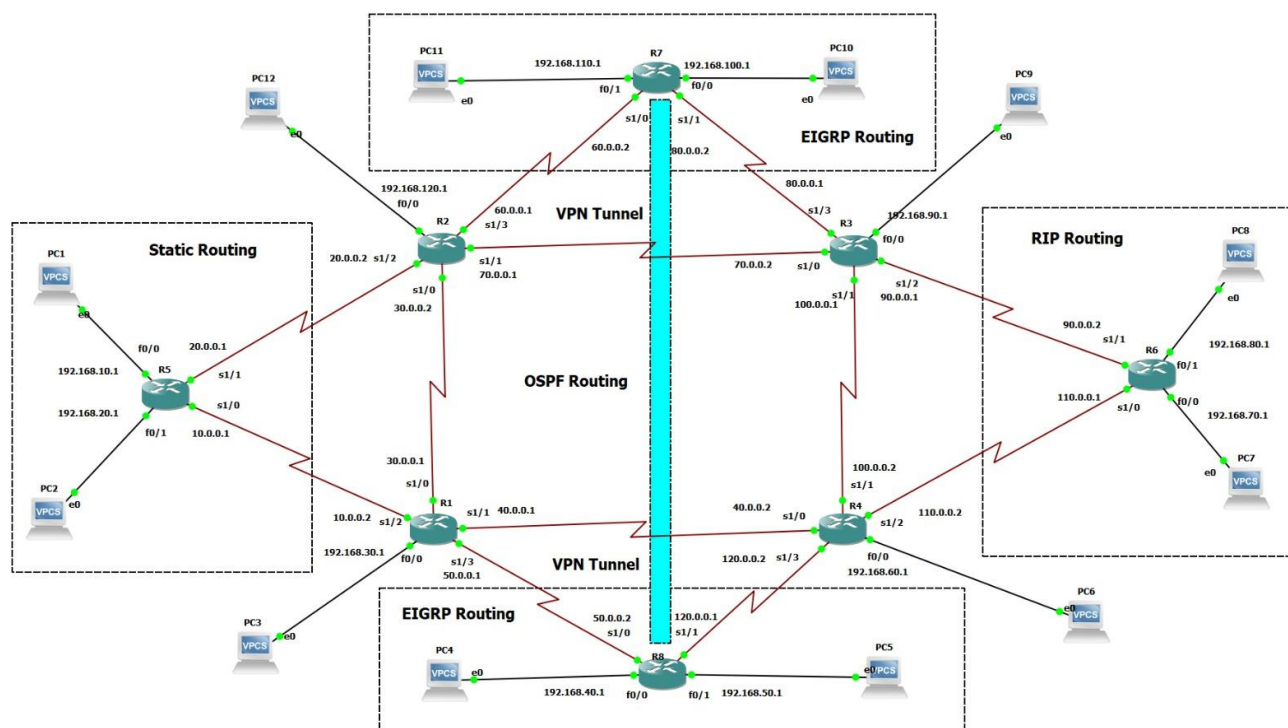
Figure 4.1: GNS3 Network Topology

4.3 Project Flow Chart

Below is the flowchart explaining the workflow of our application (Cybernation).
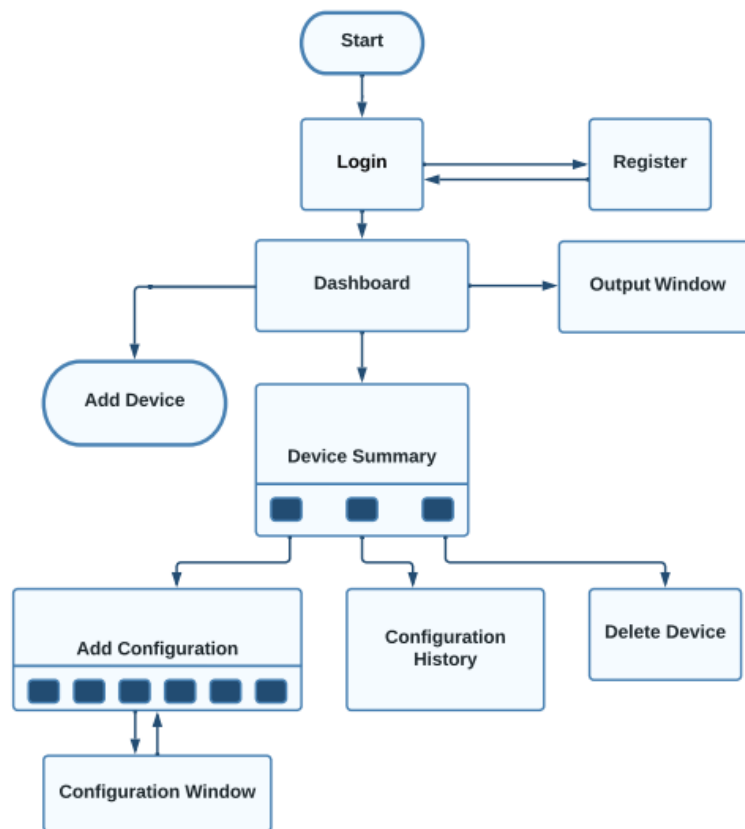
Figure 4.2: Project Flow Chart

4.4 Hardware Setup

We have configured four routers and switches in the below hardware setup using our application i.e., Cybernation.
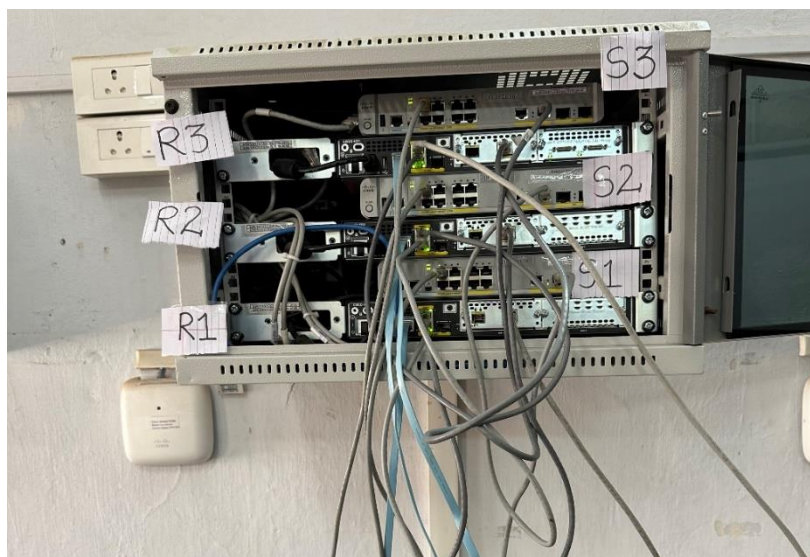
Figure 4.3: Hardware Router Setup
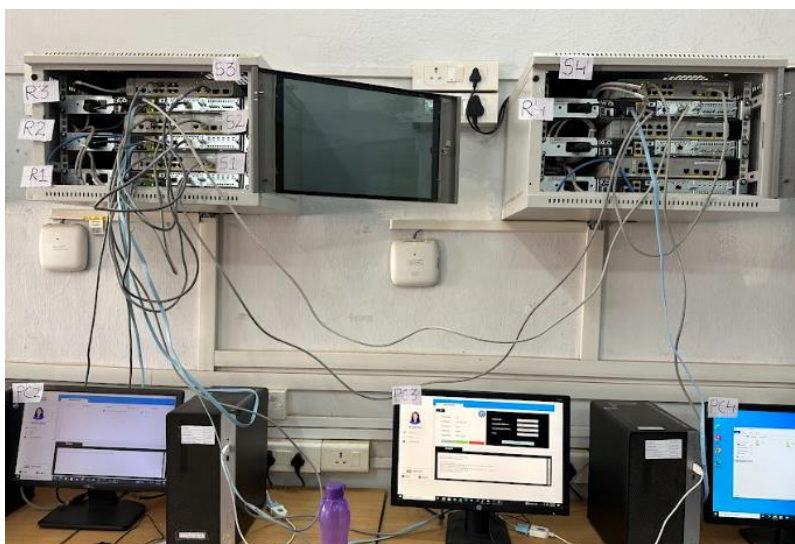
Figure 4.4: Hardware Setup

Network IP Addresses	
Network 1	
Router 1	**IP Address**
g 0/0/0	192.168.10.1
g 0/0/1	10.0.0.1
g 0/1/0	40.0.0.2
Switch 1	192.168.10.2
PC 1	192.168.10.5
Network 2	
Router 2	**IP Address**
g 0/0/0	192.168.20.1
g 0/0/1	20.0.0.1
g 0/1/0	10.0.0.2
Switch 2	192.168.20.2
PC 2	192.168.20.5
Network 3	
Router 3	**IP Address**
g 0/0/0	192.168.30.1
g 0/0/1	30.0.0.1
g 0/1/0	20.0.0.2
Switch 3	192.168.30.2
PC 3	192.168.30.5
Network 4	
Router 4	**IP Address**
g 0/0/0	192.168.40.1
g 0/0/1	40.0.0.1
g 0/1/0	30.0.0.2
Switch 4	192.168.40.2
PC 4	192.168.40.5

Result of verifying after the configuration
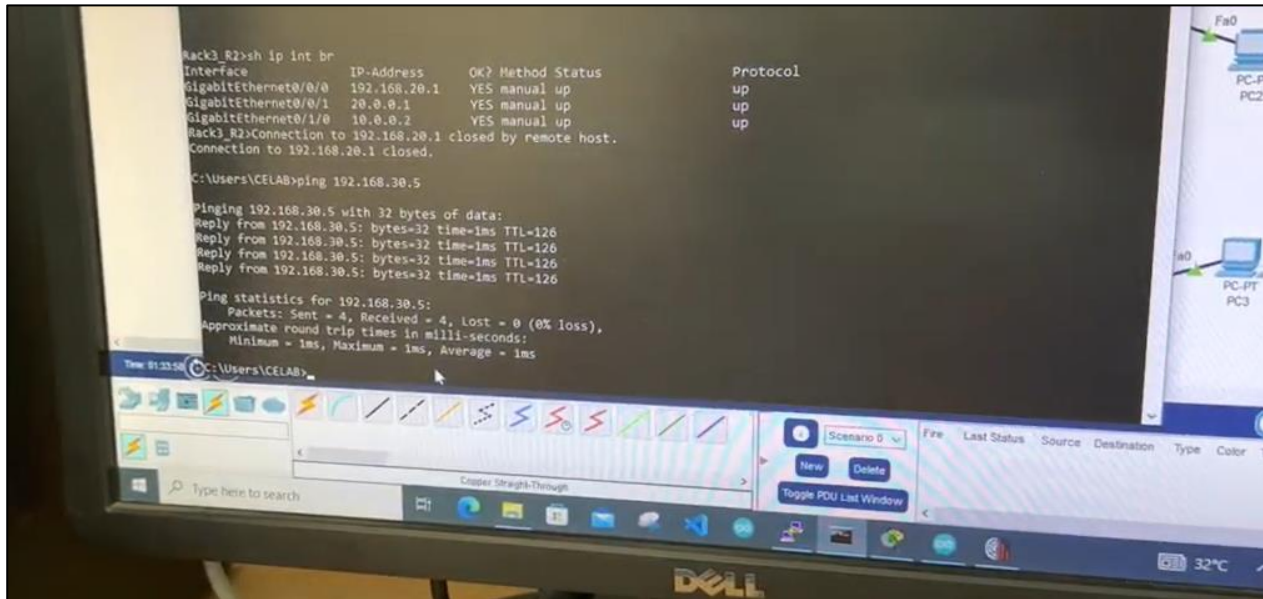
Figure 4.5: Hardware OSPF Result
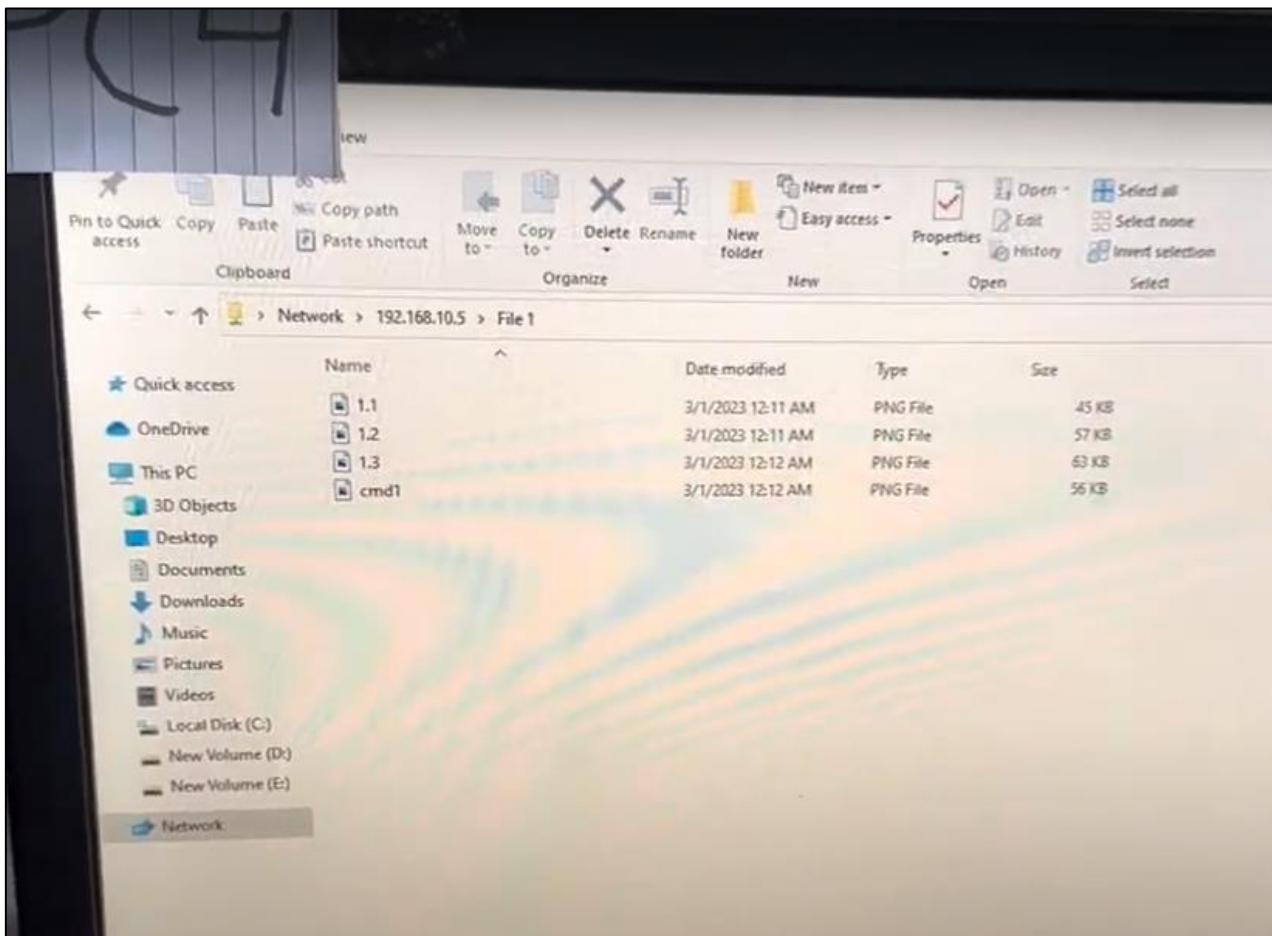
Figure 4.6:File Sharing Result
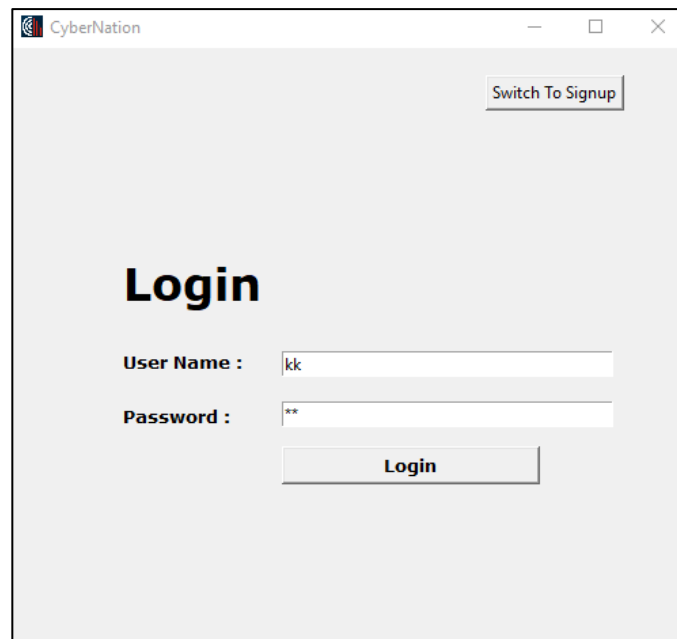
4.5   Application Demo
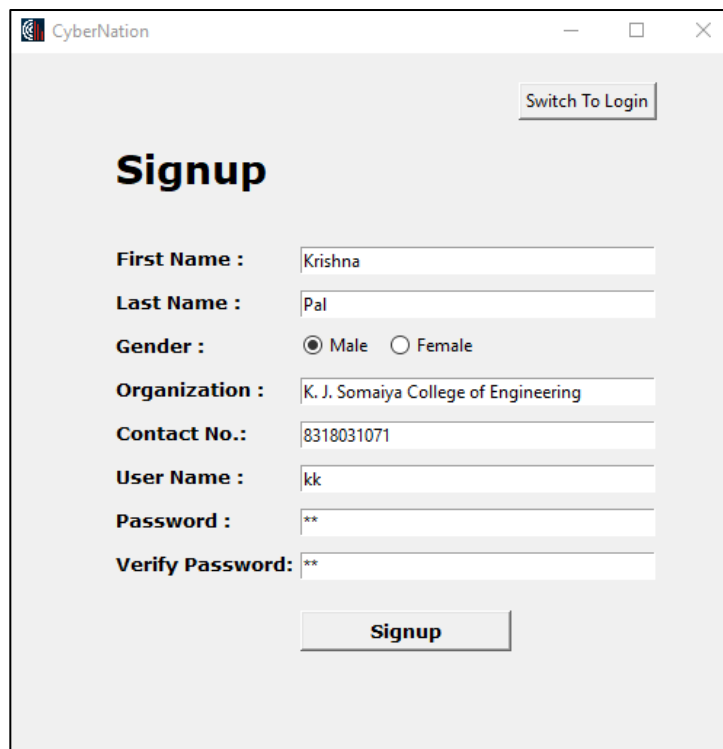
4.5.1   Login and Signup

Here Users can log in if they had already created their account or they can switch to the signup window to create one and then login into the application

Figure 4.7: Login Window

Figure 4.8: Sign up Window

4.5.2 Dashboard

In Dashboard, there are three sections sidebar, device list, and output. In the sidebar, users can see their profile such as name, organization name, contact number, and gender, and at the bottom, there are three buttons – add device, close, and logout. In the device list user will see all the saved devices that they had added and in the output section they will see all activity that they had on in that session

Figure 4.9: Dashboard

4.5.3 Add and Delete Device

In adding the device user needs to enter device information such as Hostname, Ip address, Subnet Mask, Description, Device type, Username, and Password, and Enable Secret
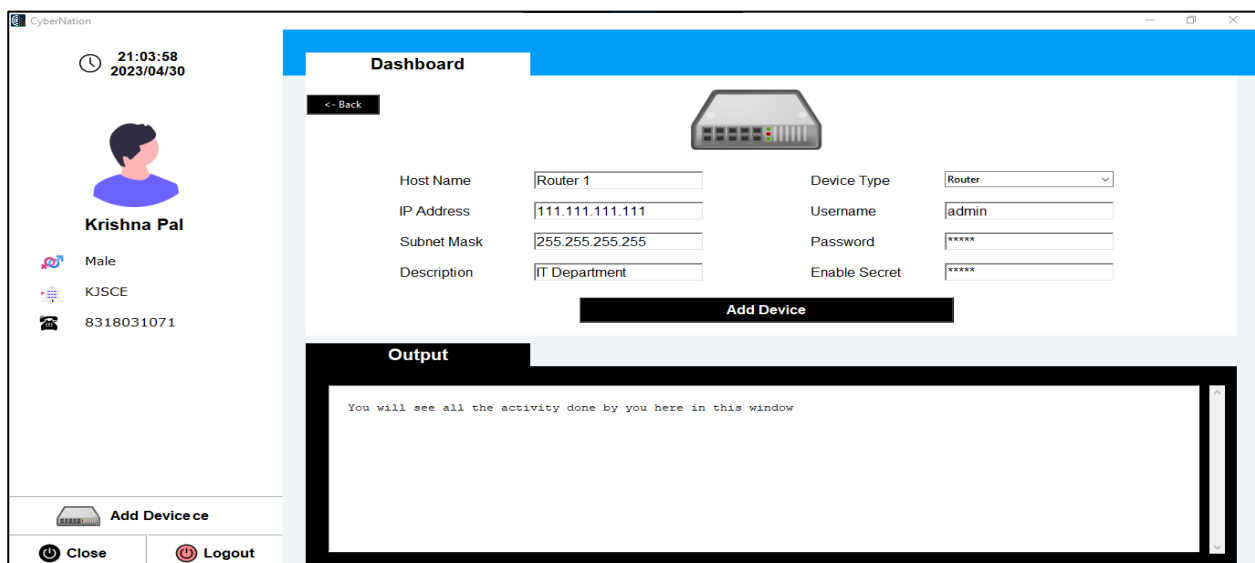

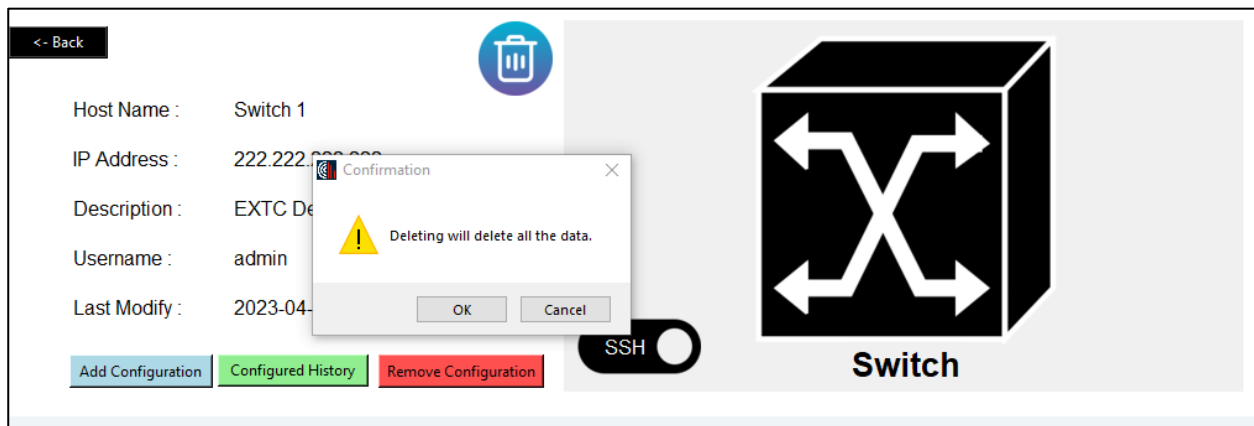*Figure 4.10: Add device*

Users can also delete the device

Figure 4.11: Delete the device

4.5.4 Add and Remove Configuration

On clicking on add configuration user can select which protocols he/she wants to config and enter the required data to config and the output will reflect in the output section.
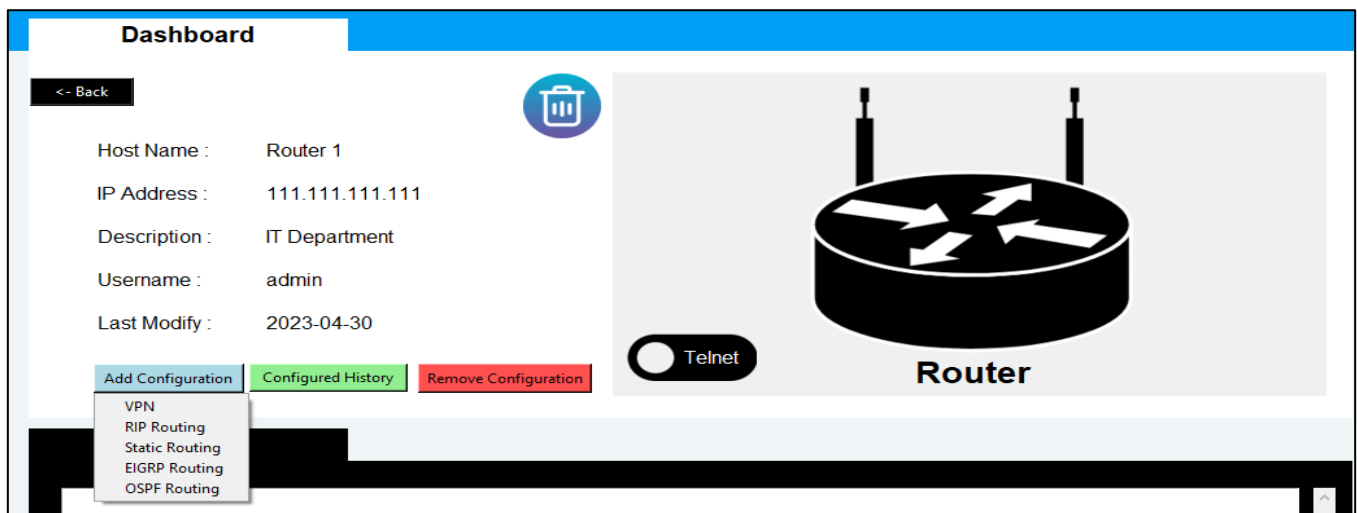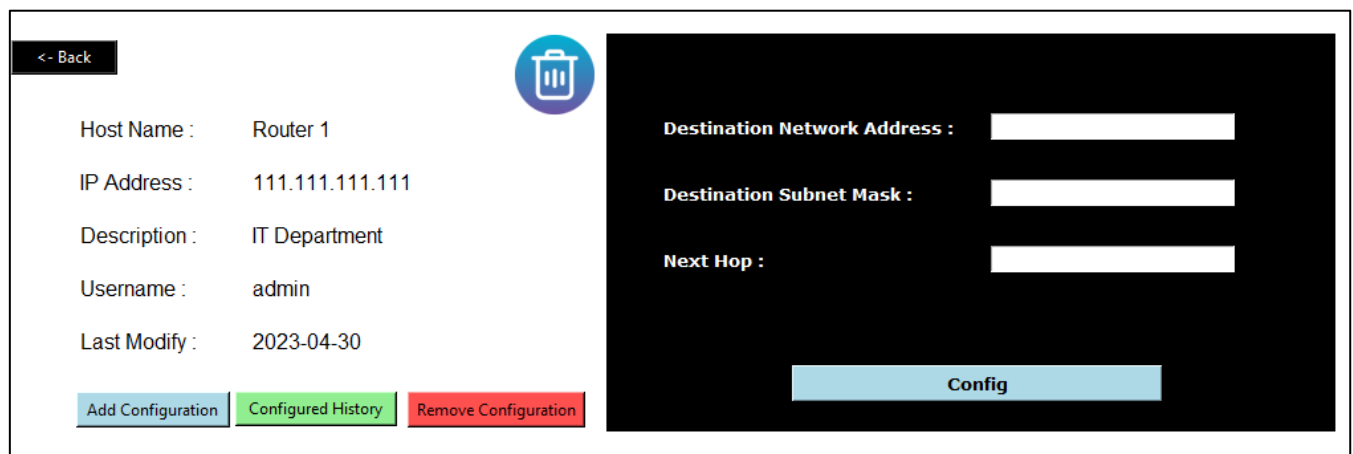

Figure 4.12: Add Configuration

Figure 4.13: Add config window

On clicking on remove configuration user can select which protocols he/she wants to remove and enter the required data to remove config and the output will reflect in the output section.
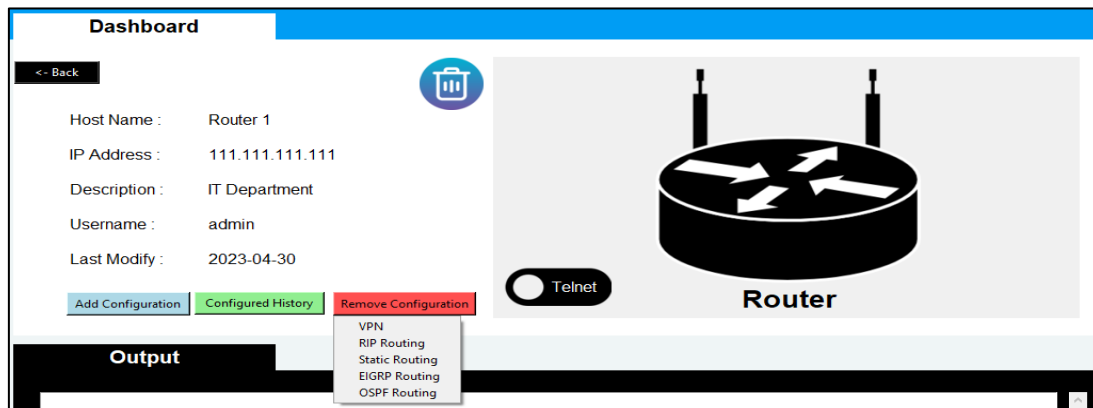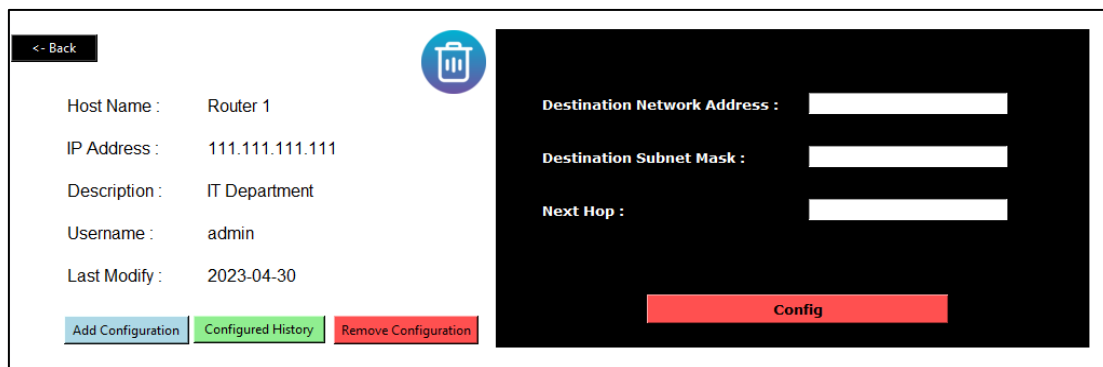

Figure 4.14: Remove Configuration

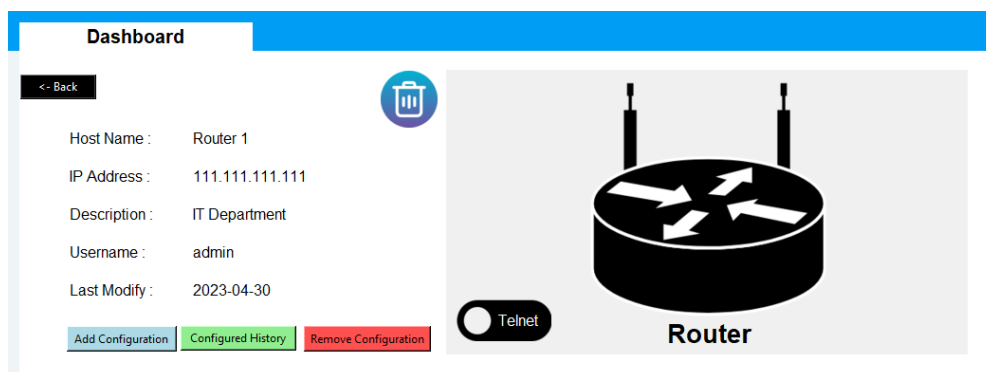*Figure 4.15:Remove config window*

4.5.5 Device Info and Config History
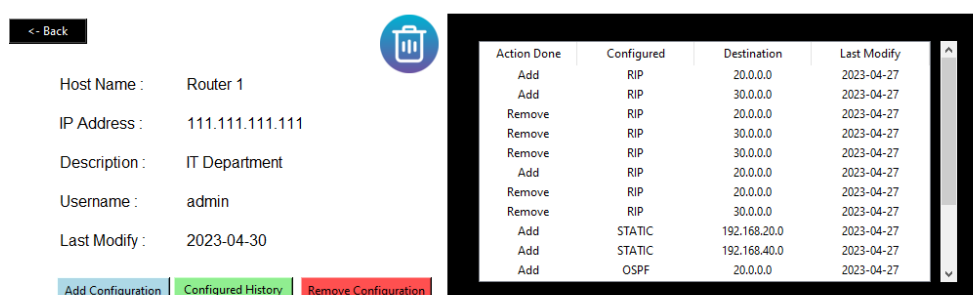

Figure 4.16: Device Info

Figure 4.17: Config History

CHAPTER 5

CONCLUSION

This application's purpose is to demonstrate the basic idea around network automation with python and as a starting point for an application that can be used in real environment. It needs improvements on the user inputs check and it needs to be customized to fill to the needs of the network environment that it will run.

The goal of this application is to serve as both a starting point for an application that can be applied in a real setting and a demonstration of the fundamental concept surrounding network automation with Python. It needs to be adapted to meet the requirements of the network environment in which it will operate and the user inputs check needs to be improved. Users need to have some background knowledge of networking then only users can use this application without any flaw.

An automation plan will help organizations with change control, architecture, security, and operational management. When automated systems continuously scan the network, troubleshooting may be done fast and easily. The idea of software controllability is growing throughout the networking industry thanks to the creative applications of SDN. It is possible to automate the configuration and monitoring of any device, irrespective of the vendor, on SDN devices as well as other networking solutions.

We have demonstrated that by using Python, network engineers do not need to configure by themselves each individual device, they just need to create the proper infrastructure and by implementing automation scripting. The network controllability becomes easier and changes can be faster deployed, maybe even automatically, as a response to events that take place in the network. So, the legacy network elements become like SDs. We have offered examples of two automation libraries based on Python and Secure Shell connections.

We have added two different types of connectivity i.e., Telnet and SSH. So, if network engineers have any issues with ssh or ssh does not support the device they can use a telnet connection to configure that device. Also, the data stored in the database is only available on the device on which engineers have created their account, they need to add or create their account again to use our application on another device.

Organizations will benefit from an automation strategy with benefits on change control, architecture, security, and operational managmemt. Troubleshooting can be made easily and quickly when automated systems examine the network continuously.

CHAPTER 6

FUTURE SCOPE

This application's purpose is to demonstrate the basic idea around network automation with Python and as a starting point for an application that can be used in a real environment. It needs improvements on the user inputs check and it needs to be customized to fill to the needs of the network environment that it will run. Also, because most of the demonstration purpose of the application there are parts of the code that needs to be improved.

The future of network automation looks promising, with the continued growth of new technologies and the increasing demand for faster and more efficient network operations.

Here are some key trends and developments that are shaping the future of network automation:

Artificial Intelligence and Machine Learning: Advances in AI and machine learning are enabling network automation tools to become more intelligent and adaptive, allowing them to respond to network issues in real time.

Intent-Based Networking: Intent-based networking (IBN) is a new approach to network automation that uses machine learning and natural language processing to understand the intent behind network operations. IBN aims to simplify network operations and make it easier for network administrators to manage complex networks.

Cloud-Based Network Automation: Cloud-based network automation tools are becoming more popular, offering greater scalability, flexibility, and cost savings. Cloud-based tools also enable network administrators to automate tasks across multiple network environments.

Software-Defined Networking (SDN): SDN is a network architecture that separates the network control plane from the data plane, allowing for greater programmability and automation. SDN is becoming more widespread, enabling network administrators to manage networks more efficiently and automate network tasks.

DevOps Integration: DevOps practices are being integrated into network automation, allowing for greater collaboration between development and operations teams. This integration is helping organizations to streamline network operations and improve overall efficiency. The future of network automation is bright, with new technologies and developments enabling network administrators to automate more tasks and operate networks more efficiently. As network automation becomes more widespread, it will become increasingly important for network administrators to develop the necessary skills and expertise to manage and maintain automated networks.

BIBLIOGRAPHY

[1] Mihăilă, Paul, et al. "Network Automation and Abstraction using Python Programming Methods."

[2] Choi, Brendan. "GNS3 Basics." Introduction to Python Network Automation. Apress, Berkeley, CA, 2021. 415-489.

[3] Santyadiputra, G. S., I. M. E. Listartha, and G. A. J. Saskara. "The effectiveness of Automatic Network Administration (ANA) in network automation simulation at Universitas Pendidikan Ganesha." Journal of Physics: Conference Series. Vol. 1810. No. 1. IOP Publishing, 2021.

[4] Choi, Brendan. "Introduction to Python Network Automation." Introduction to Python Network Automation. Apress, Berkeley, CA, 2021. 1-21.

[5] Milios, George. "Network Automation using Python." (2021).

[6] Abdelhak, B. O. U. M. E. Z. R. A. G. "A Proposed of Novel Network Management Platform for Network Automation and Programmability with Implementation on GNS3."

[7] Alberto Simon Fernandez. "Automating the configuration of networking devices with Python." https://upcommons.upc.edu

[8] James, Vineet, "Network Automation Methodology for Detecting Rogue Switch" (2019).

[9] Choi, Brendan. "Building a Python Automation Lab Environment." Introduction to Python Network Automation: The First Journey. Berkeley, CA: Apress, 2021. 515-548.

[10] Abdelhak, B. O. U. M. E. Z. R. A. G. "A Proposed of Novel Network Management Platform for Network Automation and Programmability with Implementation on GNS3."

[11] R. Emiliano and M. Antunes, "Automatic network configuration in a virtualized environment using GNS3," 2015 10th International Conference on Computer Science & Education (ICCSE), Cambridge, UK, 2015, pp. 25-30, doi: 10.1109/ICCSE.2015.7250212.

[12] Mazin, Aladhami Mahmood, Ruhani Ab Rahman, and Murizah Kassim. "Performance analysis on network automation interaction with network devices using Python." 2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE). IEEE, 2021.

[13] Zaidi, Syed Muhammad Asad, et al. "SyntheticNET: A 3GPP compliant simulator for AI enabled 5G and beyond." IEEE Access 8 (2020): 82938-82950.

[14] Zaidi, Syed Muhammad Asad, et al. "SyntheticNET: A 3GPP compliant simulator for AI enabled 5G and beyond." IEEE Access 8 (2020): 82938-82950.

[15] Al-Mekhlal, Monerah, et al. "Network Automation Python-based Application: The performance of a Multi-Layer Cloud Based Solution." 2022 IEEE International Conference on Omni-layer Intelligent Systems (COINS). IEEE, 2022.

K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to the University of Mumbai)

ACKNOWLEDGEMENT

We appreciate the opportunity to work on this topic for our major project at K. J. Somaiya College of Engineering. We are grateful to Dr. Shubha Pandit, our principal, and Dr. Bharati Singh, our HOD, for giving us this opportunity.

We would like to thank Prof. Jyoti Varavadekar, our project guide, for allowing us to complete this project under her expert guidance. We are incredibly grateful for her time and care. Throughout the course of this short project, she taught us extremely valuable skills. Her helpful comments and frequent support helped us to focus and shape our learning throughout the process. Each of us researched the subject using various sources of information and did our best to make this major project a success