University of Mumbai

Computer Network Automation Using Python and GNS3

Submitted at the end of semester VII in partial fulfillment of requirements For the degree of

**Bachelor of Technology**

by

# Dhruti Sangal Roll No: 1913076

# Krishna Kumar Pal Roll No: 1913097

**Vrunda Patel Roll No: 1913101**

Guide

**Mrs. Jyoti Varavadekar**



# Department of Electronics and Telecommunication Engineering

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

**(Autonomous College Affiliated to University of Mumbai)**

## Batch 2019 -2023

**Certificate**

This is to certify that the dissertation report entitled **Computer Network Automation Using Python and GNS3** submitted by Dhruti Sangal, Krishna Kumar Pal and Vrumda Patel at the end of semester VII of LY B. Tech is a bonafide record for partial fulfillment of requirements for the degree of Bachelors in Technology in Electronics and Telecommunication Engineering of University of Mumbai.

Guide Head of the Department

Principal

Date: -5-2023

Place: Mumbai-77

## Certificate of Approval of Examiners

We certify that this dissertation report entitled **Computer Network Automation**

**Using Python and GNS3**is a bonafide record of project work done by Dhruti Sangal, Krishna Kumar Pal and Vrumda Patel during semester VII.

This project work is submitted at the end of semester VII in partial fulfillment of requirements for the degree of Bachelors in Technology in Electronics and Telecommunication Engineering of University of Mumbai.

Internal Examiners

\_ External/Internal Examiners

Date: -5-2023

Place: Mumbai-77

## DECLARATION

We declare that this written report submission represents the work done based on our and / or others’ ideas with adequately cited and referenced the original source. We also declare that we have adhered to all principles of intellectual property, academic honesty and integrity as we have not misinterpreted or fabricated or falsified any idea/data/fact/source/original work/ matter in my submission.

We understand that any violation of the above will be cause for disciplinary action by the college and may evoke the penal action from the sources which have not been properly cited or from whom proper permission is not sought.

|  |  |
| --- | --- |
| **Signature of the Student Roll No.:** 1913076 | **Signature of the Student**  **\_ Roll No.:** 1913097 |
| **Signature of the Student Roll No.:** 1913101 |  |

## Date: -5-2023

**Place: Mumbai-77**

# ABSTRACT

Network programmability is a trend that is based on scripting techniques and traditional programming languages used for managing and monitoring network parts. It has been enhanced and inspired by Software Defined Networks (SDN). In order to speed up equipment configuration and make maintenance simpler, we have presented some novel techniques for automating the configuration of network devices. Identifying and addressing security flaws also strengthens network stability and enhances network security. These approaches, which enable the unitary control of an expanding number of devices, are what networks will look like in the future.

The software's Graphical User Interface (GUI) allows users to conduct both fundamental network automation activities, such as configuring security and configuration settings. The numerous options that the user must connect to and configure network devices using Python and its libraries are demonstrated and discussed in the application's code.

**Keywords:** Network automation, software-defined networks, computer network operations, network management, python scripting.

# Contents

|  |  |  |  |
| --- | --- | --- | --- |
| **Abstract**…………………………………………………………………………………… | | | 5 |
| **List of Figures**……………………………………………………………………………. | | | 7 |
| **Introduction** ……………………………………………………………………………… | | | 8 |
| **1.** | **Network Automation**…………………………………….……………………... | | 10 |
|  | 1.1 | Challenges in network configuration……………………………………... | 10 |
|  | 1.2 | Problem Statement ………………………………………………………... | 10 |
|  | 1.3 | Objectives………………………………………………………………... | 11 |
| **2.** | **Literature Survey**………………………………………………………………. | | 12 |
| **3.** | **Technologies and Techniques**…...…………………………………………… | | 14 |
|  | 3.1 | Program development information…………...…………………………… | 14 |
|  | 3.2 | Python modules and libraries….…………………………………………. | 14 |
|  | 3.3 | VScode …………………………………...……………………………… | 15 |
|  | 3.3 | Virtual Environment…….…….…………………………………………. | 15 |
| **4.** | **Program Implementation** …...………………………………………………… | | 21 |
|  | 4.1 | Objective …………….…………………………………………………... | 21 |
|  | 4.2 | Network Topology………………………………………………………. | 21 |
|  | 4.3 | Application Demo…………………………………………………….… | 22 |
|  | 4.4 | Automation Actions.…………………………………………………….. | 24 |
| **5.** | **Conclusions** …………………………………….……………………………… | | 26 |
| **6.** | **Future Scope**……………….…………………………………………………... | | 26 |
| **Bibliography** ……………………………………………………………………………... | | | 27 |
| **Acknowledgments**………………………………………………………………………. | | | 28 |

**List of Figures**

|  |  |
| --- | --- |
| 1. GNS3 Setup Window…………………………………………………………………. | 16 |
| 2. GNS3 GUI…………………………………………………………………………….. | 17 |
| 3. GNS3 Doctor …………………………………………………………………………. | 18 |
| 4. VMware Workstation 16 player ………………………………………………............ | 18 |
| 5. GNS3 VM Preferences……………………………………………………………….. | 19 |
| 6. GNS3 Servers Summary……………………………………………………………… | 20 |
| 7. Network Topology……………………………………………………………………. | 21 |
| 8. Application Interface…………………………………………………………………. | 22 |
| 9. Add Device Functionality……………………………………………………………. | 22 |
| 10. Add Device Flow Chart………………………………………………………………. | 23 |
| 11. Automation Actions Available………………………………………………………... | 24 |

# INTRODUCTION

Automation is the technology that allows a process or procedure to be carried out with little help from humans. The use of various control systems for equipment such as machinery, factory processes, boilers, heat-treating ovens, and other applications with little or no human intervention is known as automation or automatic control [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 legacy networks or non-SDN networks.

## NETWORK AUTOMATION

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

### 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 on 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 mis-configuration 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. **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.

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

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.

### Problem Statement

Information technology is very high because of the COVID-19 pandemic. Organizations from business through education tend to use this technology most of the time. Information technology uses computer networks for the integration and management of data. A manageable network configuration for networked devices will be easier to maintain and reduce communication problems. Traditionally, network administrators must configure each network

device manually. This process takes time and is inefficient. Automated network configuration can overcome the repetitive process, but it is relatively slow.

### Objectives

* **Eliminate manual tasks:**

Automating the network replaces manual tasks with predictable, repeatable network changes.

### Accelerate service:

A critical benefit: provide key data services faster, optimize network performance, and speed the rollout of new services and applications.

### Make changes faster:

More frequently execute network changes that were previously rarely performed because they were manual, time-consuming, and resource intensive.

### Authorized access:

Rely on network analytics for insight into performance, utilization, security, and resource allocation that help resolve issues far faster than via manual techniques.

**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 completely 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 de-schedule numerous independent software clusters, critically even if those clusters were in various physical locations.

## LITERATURE SURVEY

### In ‘Performance analysis on network automation interaction with network devices using python’

[5] the main aim was that how automation is becoming a trend these days due to its tremendous benefits, especially with the increasing numbers of network devices. The word automatic is being defined as acting or operating in a manner independent of external influence and human control, network automation offers thousands of benefits for companies, it allows the configuration of many devices within minutes, and eliminates the chances of human error.

In **‘Network automation and abstraction using Python Programming Methods’** [6] we looked into 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 the 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 Frame-work**’ [7] they proposed an alternative model of a network automation system. The model system was implemented with a controller application that used REST API (Representational State Transfer Application Programming Interface) architecture and built by the Django framework with Python programming language to increase the performance of network automation. The design model, called the As-RaD System, uses a web-based application for maintenance and automates networking tasks with easy GUI. The network devices used in this research include the Cisco CSR1000V because it supports REST API communication to manage its network configuration and could be placed on the server either.

In **‘Network Automation Methodology for detecting Rogue Switch’** [8] it provided a solution to detect malicious switches on a network using GNS3 network emulator, Python for automation, Cisco ios configurations, WireShark packet analyzer and Docker. An unmanaged switch on the other hand behaves like a “plug and play” device. It comes pre-configured and simply allows the devices to communicate with one another. They do not have an IP address and some also have no MAC address

which makes them very difficult to trace. Goal is achieved by continuously filtering and analyzing network traffic for any broadcast storms or new ARP packets using Packet Analyzers and then effectively tracing the malicious host connected to the rogue switch by deploying automation techniques. This article successfully demonstrates how to detect a rogue switch on the wired network; however, there is a lot of future scope for enhancing this solution into a fully functional centralized management tool for effective network management and with advanced fault detection features.

## TECHNOLOGIES AND TECHNIQUES

### Program development information

The application’s purpose is to demonstrate different ways to connect and configure network devices thus commented code will be present in the code to exhibit an alternative solution. 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, 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 import something”. 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 instead of Telnet and if there will be a need for data serialization JSON will be used instead of Pickle. By using methods and functions of the imported modules the application’s security feature is enhanced. For instance, the password that the application needs to connect to the network devices is encoded and encrypted, and when the user is prompted to input a password, the password is masked with asterisks. Multithreading will be used to speed up the execution of the code.

### Python modules and libraries

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 options for devices that do not support APIs.

[**Netmiko**](https://github.com/ktbyers/netmiko) 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 for you, 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

1. **Ip address**: The addresses that the user enters into 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 as opposed to tenths of lines. [Ip address](https://docs.python.org/3/library/ipaddress.html#module-ipaddress) 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 or not two hosts are on the same subnet, iterating over all hosts in a particular subnet, checking whether or not a string represents a valid IP address or network definition, and so on.
2. **Tkinter**: This is the module library that will be used to create the application's graphical user interface. 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 Tk GUI toolkit.

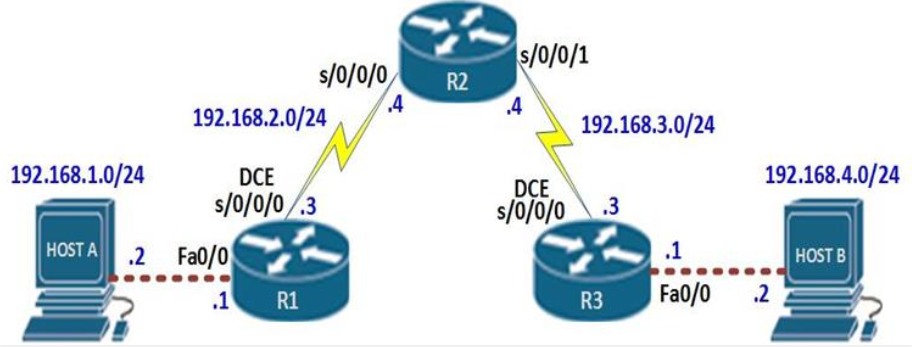
Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps:

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

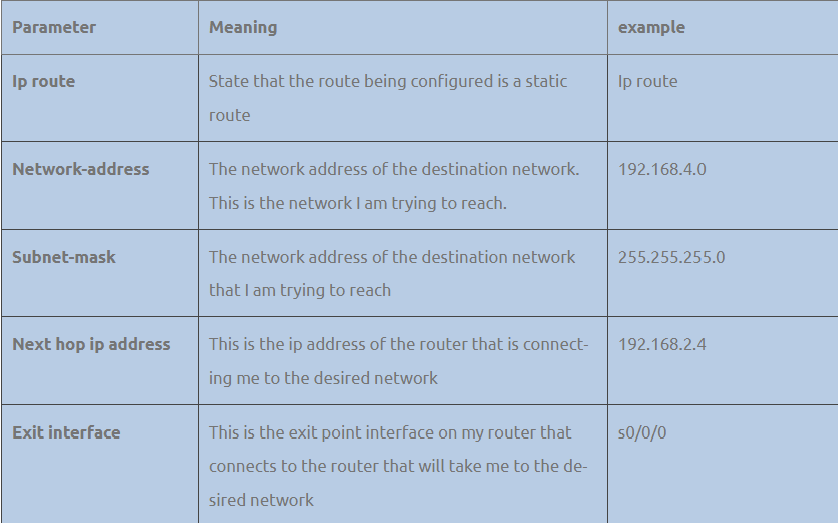
**3.3 Various Routing Configurations**

**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 & does not change if the network occurs or are 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 can’t automatically be updated. Thus you 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 can’t be used in the vast and continuously changing network because they 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 won’t get rerouted because they can’t change or tune by themselves even though we have certain choices to save whenever there are issues.
* A BFD (Bidirectional forwarding detection) can be configured along with the static method, so whenever there is a failure in BFD, it switches to an alternative route.
* 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.

****

* 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 hosts can ping each 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 that has an AD value of 120 and works on the Network layer of the OSI model. RIP uses port number 520.

#### **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. The maximum hop count allowed for RIP is 15 and a hop count of 16 is considered as network unreachable.

### ****Features of RIP****

1. Updates of the network are exchanged periodically.

2. Updates (routing information) are always broadcast.

3. Full routing tables are sent in updates.

4. Routers always trust routing information received from neighbor routers. This is also known as *Routing on*rumors.

### ****RIP versions:****

There are three versions of routing information protocol – **RIP Version1**, **RIP Version2**, and **RIPng**.

### ****Normal utilization of RIP:****

1. **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.
2. **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.
3. **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.
4. **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 very well may 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 itself generates.

An LSA is a packet that contains information about neighbors 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.

**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 internetworks. 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.

The OSPF stands for **Open Shortest Path First**. 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 goal of routing is to learn routes. The OSPF achieves by learning about every router and subnet within the entire network. Every router contains the same information about the network. The way the 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 LSDBs.

**3.3.4 EIGRP Protocol**

### VScode

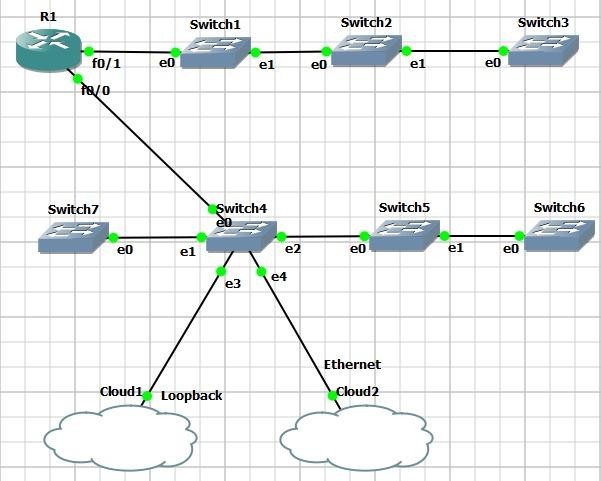
Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS, and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages and runtimes.

## PROGRAM DESIGN

### Objective

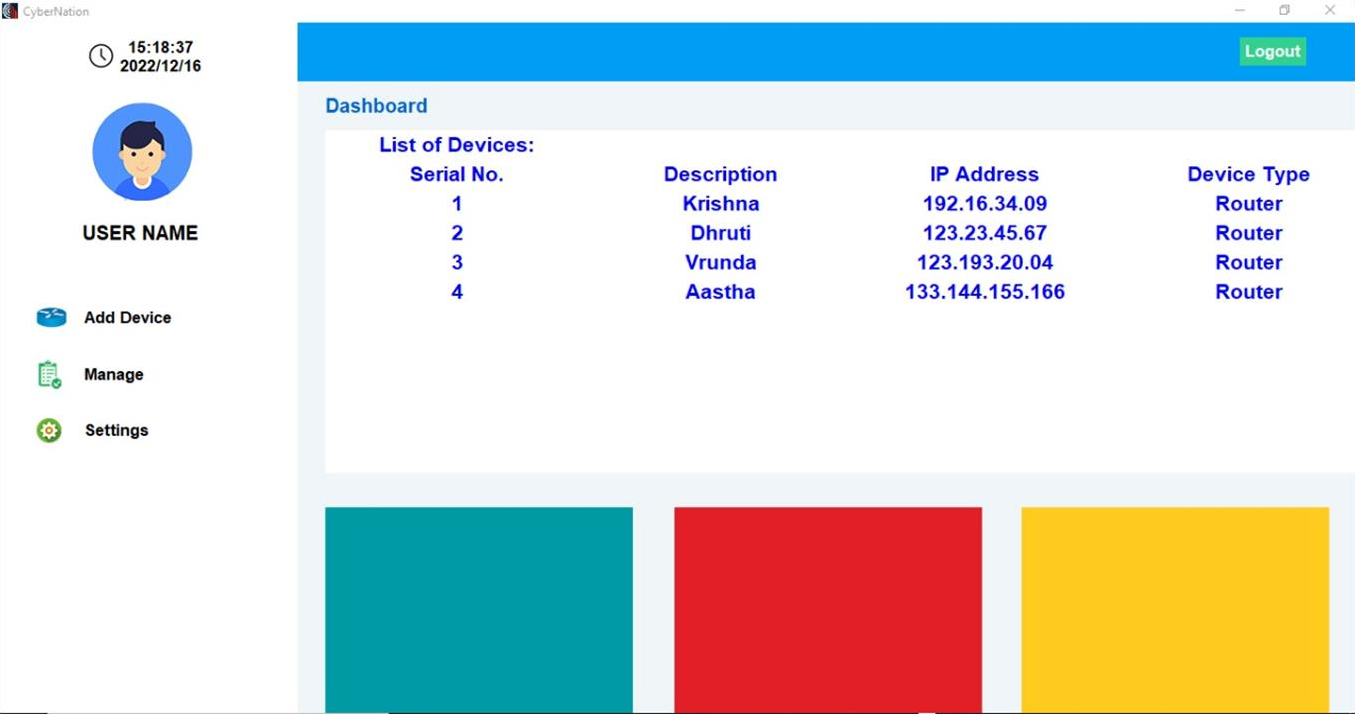
The user will be able to add a device's information (IP, enable password, type, hostname, SSH username, SSH password), which will be recorded in a CSV type file, through the application's graphical interface. When the credentials are used to open an SSH connection with the device, they are decrypted after being stored in an encrypted manner. A list of all the saved devices will be accessible for selection. It is possible to choose one or more devices. The user will then need to pick which automated method will be used after selecting the devices, and depending on the operation, additional parameters may need to be configured.

### Network topology

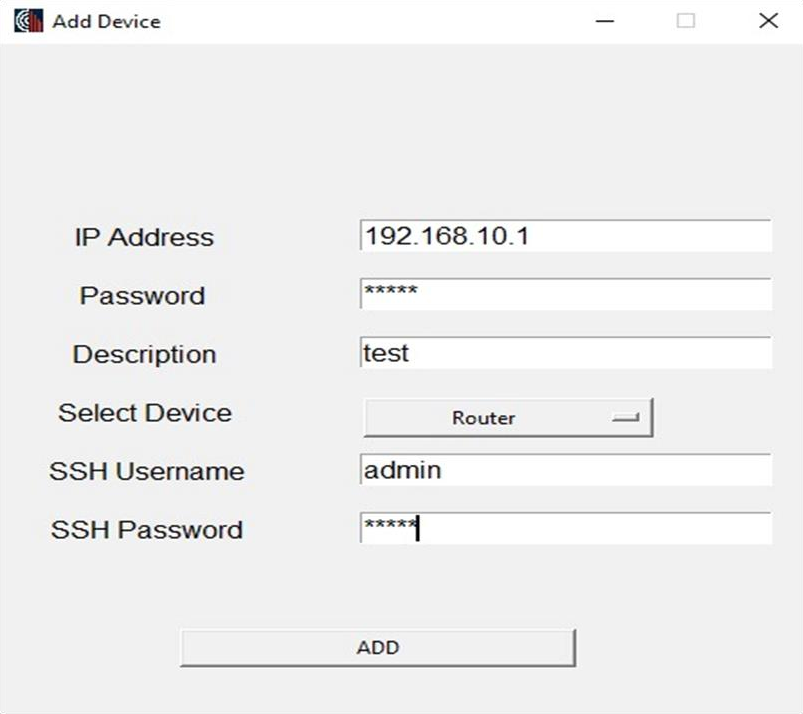


**Figure 7: Network topology**

### Application demo



**Figure 8: Application Interface**



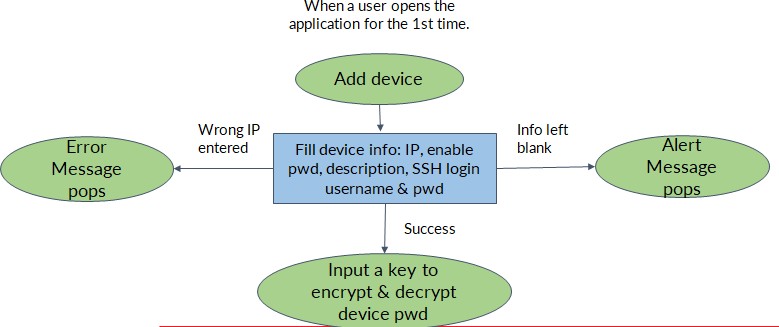
**Figure 9: Add device functionality**

### Add devices Section:

The first thing that the user has to do when the application is executed for the first time is to add network devices. To add a network device all information have to be filled (IP address, Enable password, Description, SSH login username and SSH login password), if not a pop-up message appears informing the user.

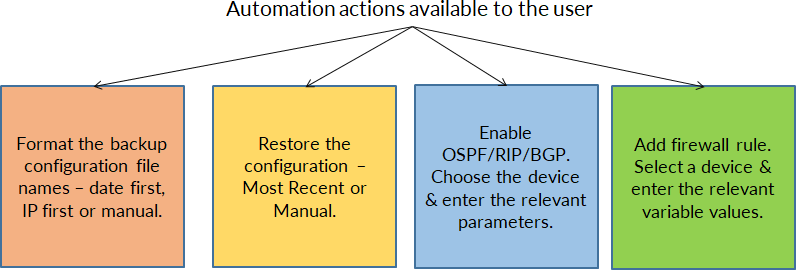
Another check that takes place is if the IP that the user has entered is a valid IP address if not again a pop-up informs the user that the IP is not valid.

When all information’s has been entered correctly a window opens asking the user to input a password. This password will be used to encrypt and decrypt the devices password to and from a configuration file. When the user enters the password, a message appears to inform that the device was added successfully.



**Figure 10: Add device flow chart**

### Automation actions



**Figure 11: Automation Actions Available**

**The backup configuration** option stores the configuration of the devices that the user chooses at the backup\_files directory which is in the projects folder. The user has 3 options about the format of the files name and should select one of them.

The options are:

* **Date first:** The files name starts with the current date with format dd\_mm\_yyy\_hh\_mm\_ss following by the string backup\_of\_ and then the IP of the Device
* **IP first:** It is the reverse naming format of the Date First option, for example 192.168.10.10\_backup\_of\_10\_11\_2020\_10\_15\_30
* **Manual:** An entry box opens, and the user can manually enter the second half of the files name. The first half is the device’s IP.

To **restore the configuration file** in a device the user has to select one of the two options available. The options are:

* **Restore most recent backup:** The application search in each device’s that has been selected backup folder finds the most recent one and restores it to the device.
* **Manual restore:** This option is not suitable for a large number of devices because a window will open for each device for the user to choose the file that he wants to restore to the device.

To **enable OSPF/RIP/BGP** the user should first choose the devices for which he wants to enable routing algorithms.

**Add firewall rule:** To enter an access list in a network device the user has to select the devices that it wants to enable the access list and then complete all the variables needed. When the application has completed the access list creation the user can check the output in the bottom of the application window.

## CONCLUSIONS

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. Additionally, some of the code needs to be updated because of the majority of the application's demonstration purposes.

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 shown that network engineers may use Python to automate the configuration of devices rather than having to manually configure each one. They only need to set up the necessary infrastructure and automation scripts. As a result of network events, network controllability improves and changes can be implemented more quickly. So, with SDNs, the components of older networks are becoming comparable.

## 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 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.

# BIBLIOGRAPHY

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

*MACRo 2015* 2.1 (2017): 95-103.

1. Choi, Brendan. "GNS3 Basics." *Introduction to Python Network Automation*. Apress, Berkeley, CA, 2021. 415-489.
2. 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.
3. Choi, Brendan. "Introduction to Python Network Automation." *Introduction to Python Network Automation*. Apress, Berkeley, CA, 2021. 1-21.
4. Milios, George. "Network Automation using Python." (2021).
5. 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."
6. Alberto Simon Fernandez. “Automating the configuration of networking devices with Python.” [https://upcommons.upc.edu](https://upcommons.upc.edu/)
7. James, Vineet, "Network Automation Methodology for Detecting Rogue Switch" (2019).

*Technical Library*. 337. <https://scholarworks.gvsu.edu/cistechlib/337>

# ACKNOWLEDGEMENTS

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 with this opportunity.

We would like to thank Prof. Jyoti Varavdekar, 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.