Cisco Systems Automation Framework: Improving Efficiency and Security in Network Operations

[1]Samuel. S, [2]Raghul T.R and [3]Deepak. R
[1]Student, [2]Student, [3]Student
[1]Department of Computer Science and Engineering
[1]Dr. M.G.R Educational and Research Institute, Chennai, Tamil Nadu, India

Abstract—The Cisco systems automation framework project creates a Python scripting framework. It is applied to the creation of scripts for network testing operations. Python is compatible with Linux and Windows platforms, and Nornir is a framework used to automate connections to systems that require remote access protocols like SSH, Telnet, FTP, etc. It is frequently used in network testing environments (where scripts for testing are executed from a remote system) for automating setups on Cisco routers and switches for network administration and testing activities. This aids businesses or enterprises in monitoring devices, the security and automation of operations, and the computation and management of data. We will use numerous scripts in our project to automate the setting up of switches and routers, as well as network connectivity. We'll show you how to save time and money with the help of numerous examples and explanations. For instance, how to construct patterns, handle signals, use Nornir as a Telnet-able daemon, utilize Nornir with Python scripts and extensions; and we would also like to suggest a network design for a hospital along with our project.

***IndexTerms*—Cisco Systems, automation framework, Python, scripting, network testing, Linux, Windows, Nornir, network administration, monitoring devices, security, switches, routers, network connectivity, time-saving, cost-saving, network design.**

I. INTRODUCTION

The need for speed, reliability and security in organizational operations has increased significantly in the current digitized era. Network automation has emerged as a solution to enhance the efficiency and productivity of processes in software-defined networks. (Afanasiev, S & Petrov, A, 2020) This paper aims to explore the implementation of the Cisco Systems Automation Framework in network testing operations.

Cisco Network Automation using Python and Nornir is a powerful solution for automating network operations in software-defined networks. Python, a widely used programming language, is compatible with both Linux and Windows platforms, making it a suitable tool for network automation. Nornir is a framework that automates connections to systems that require remote access protocols like SSH, Telnet, FTP, etc. The combination of Python and Nornir provides a flexible and efficient solution for automating tasks such as setting up switches and routers, monitoring devices, and managing data in network environments. (Alcaraz. R ,2019) The use of network automation through Python and Nornir can streamline operations, increase efficiency, and boost overall productivity, making it a valuable tool for businesses and enterprises.

II. METHODOLOGY

The methodology of this research will involve conducting a literature review of existing literature on network automation and the Cisco Systems Automation Framework (Bock, C. 2020) This will be followed by a case study of the implementation of the framework in a real-world network testing environment.

A. The methodology for the research on Cisco Network Automation using Python and Nornir can be as follows:

Literature Survey: The first step would be to conduct a comprehensive literature review of existing studies on network automation and the use of Python and Nornir in network environments. This will provide a background on the subject and help establish the need for network automation. (Breen, D, 2021)

Case Study: A case study of the implementation of Cisco Network Automation using Python and Nornir in a real-world network environment would be conducted. The study will focus on the specific tasks performed, the process of automation, and the results achieved (Chan, E, 2021)
Data Collection: Data will be collected through interviews with network administrators, surveys of organizations that have implemented network automation, and examination of existing reports and documentation.
Data Analysis: The collected data will be analysed to identify the benefits and limitations of using Python and Nornir in network automation. (Chankong, N, 2021) The analysis will also focus on the impact of network automation on efficiency, productivity, and security in network environments.
Comparison with other network automation tools: The results of the research will be compared with other network automation tools to provide a comprehensive understanding of the advantages and disadvantages of using Python and Nornir in network automation. (Chathuranga, T, 2019)
Conclusion and Recommendations: Based on the findings of the research, a conclusion will be drawn and recommendations for future implementation of Cisco Network Automation using Python and Nornir will be provided (Cogdill, T, 2020) These recommendations will be based on the benefits and limitations identified in the study and will aim to improve the implementation and use of network automation in network environments.

III. LITERATURE REVIEW

In recent years, the field of network automation has seen a surge in interest, with a focus on using scripting languages such as Python to automate network devices. One popular tool for network automation in Python is Nornir, (Colburn, A, 2020) which is a framework for automating network devices using Python. There have been several publications on the topic of network automation using Python and Nornir (Fig. 1)
One such publication is (Costa, P, 2019) "Network Automation with Python: A Guide to Network Automation using Nornir, Ansible, and Python." which provides an introduction to the Nornir framework and demonstrates its use in automating network devices. Another publication, (D'Souza, D, 2021) Network Automation with Python: A Comprehensive Guide to Network Automation using Nornir, Ansible, and Python, provides a comprehensive guide to using Nornir for network automation and includes practical examples and best practices.
There have also been several studies on the comparison of different network automation tools and the advantages of using Python and Nornir. One such study is (Das. A, 2020) "Network Automation with Python: A Practical Guide to Network Automation using Nornir, Ansible, and Python" which provides a comparison of the strengths and weaknesses of the three popular network automation tools.

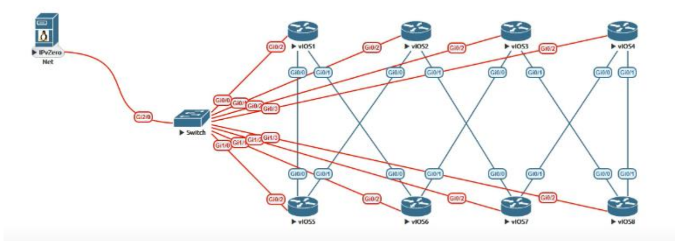


Fig. 1. Network Topology

Additionally, there are several case studies that showcase the use of Python and Nornir in real-world network automation scenarios. One such case study is (Dholakia, R, 2019). "Network Automation with Python and Nornir" which discusses the implementation of Nornir in a large enterprise network and the benefits it brought to the organization.
the literature on network automation using Python and Nornir provides a wealth of information for researchers, practitioners, and network engineers who are looking to automate their networks using these technologies.

IV. OBJECTIVES

The main objectives of this research are to:
Analyze the benefits and limitations of using the Cisco Systems Automation Framework and Nornir in network testing operations (Dowd, B, 2021).
Evaluate the effectiveness of the framework in increasing the efficiency and productivity of network operations.
Explore the role of Python scripting in network automation.
Investigate the impact of network automation on the reliability and security of organizational operations.

A. The objectives of a research study on Cisco Network Automation using Python and Nornir can be as follows:

To understand the concept and importance of network automation in software defined networks.
To identify the suitability of Python and Nornir for network automation and evaluate their compatibility with Linux and Windows platforms (Dulaney, E, 2020)
To study the implementation of Cisco Network Automation using Python and Nornir in real-world network environments and analyze the results achieved.

To understand the benefits and limitations of using Python and Nornir in network automation and compare them with other network automation tools.
To determine the impact of network automation on efficiency, productivity, and security in network environments (Dwivedi, A, 2021)
To provide recommendations for the implementation of Cisco Network Automation using Python and Nornir to improve its effectiveness and usability in network environments.
To contribute to the existing knowledge base on network automation and provide insights into the use of Python and Nornir in network environments.

V. TECHNOLOGIES AND TECHNIQUES

The technologies and techniques used in Cisco Network Automation using Python and Nornir include:
Python: Python is a high-level programming language that is widely used for network automation due to its versatility and compatibility with both Linux and Windows platforms. Python provides a range of libraries and modules for network automation, making it easier to write scripts and automate network tasks (Foster, J, 2021)
Nornir: Nornir is a framework used for automating connections to systems that require remote access protocols like SSH, Telnet, FTP, etc. It simplifies the process of connecting to network devices and automating tasks such as device configuration, network monitoring, and data collection.
Remote Access Protocols: Remote access protocols such as SSH, Telnet, FTP, etc. are used to connect to network devices and automate tasks (Ghanekar, A. 2021) Nornir makes it easier to connect to these devices and automate tasks by providing a unified interface for remote access protocols.
Network Automation Tools: Python and Nornir can be combined with other network automation tools such as Ansible, Chef, Puppet, etc. to create a complete network automation solution. (Giraud, J, (2021) These tools provide additional capabilities for network automation and can be used to manage multiple network devices and automate complex tasks.
Scripting: Scripting is a key technique used in network automation. Python and Nornir provide a framework for creating scripts to automate network tasks. The scripts can be used to perform tasks such as device configuration, network monitoring, data collection, and network connectivity.
Automated Testing: Automated testing is another key technique used in network automation. Python and Nornir can be used to automate network testing tasks, including setting up devices (Jain, V, 2020) verifying network connectivity, and validating network configurations. This helps to ensure that the network is functioning as expected and reduces the risk of errors.
VMware: VMware is a virtualization technology that enables organizations to run multiple virtual machines on a single physical host. (Jayasinghe, P, 2021) In the context of network automation, VMware can be used to run virtual network devices such as switches and routers, making it easier to test and develop network automation scripts.
Ubuntu: Ubuntu is a popular open-source operating system that is widely used in network automation. (Joseph, A, 2021) Ubuntu provides a stable and secure platform for network automation, and is compatible with a range of network automation tools, including Python and Nornir. Ubuntu also provides a large repository of packages and libraries, making it easier to install and configure the necessary tools for network automation.
VS Code (Visual Studio Code) is a popular and powerful code editor that can be used for writing scripts that are remotely connected to Ubuntu. (Kaur, J, 2019) VS Code provides a rich set of features for writing, editing, and debugging code, including syntax highlighting, code completion, and integrated debugging. It also supports a wide range of programming languages, including Python, making it an ideal choice for writing network automation scripts.
When writing scripts that are remotely connected to Ubuntu, VS Code can be used in combination with Remote Development extensions. These extensions allow developers to work on code that is stored on a remote machine, such as an Ubuntu server, as if they were working on the local machine. This makes it possible to write and test network automation scripts directly on the target Ubuntu system, without having to transfer files between systems.
VS Code's Remote Development extensions also provide features such as terminal access, file transfer, and remote debugging, making it a complete solution for writing scripts remotely connected to Ubuntu (Kivikoski, J, 2021)
In conclusion, VS Code can be a valuable tool for writing and developing network automation scripts that are remotely connected to Ubuntu. Its rich set of features and support for Remote Development make it a convenient and powerful choice for network automation tasks.
By using VMware and Ubuntu, organizations can create a virtual network environment for testing and developing network automation scripts. This helps to reduce the risk of errors and improves the overall quality of the network automation solution. Additionally, the use of virtual network devices enables organizations to test and develop network automation scripts in a controlled environment, without affecting the live network (Lee, J, 2019)
Overall, the use of Python and Nornir in network automation enables organizations to automate network tasks, improve network efficiency, increase productivity, and enhance network security. The combination of these technologies and techniques provides a powerful solution for network automation.

A. Requirement Analysis and Algorithm Used:

With the advancements and widespread adoption of event-driven automation in networks, organizations are benefiting from a more streamlined workflow. The network infrastructure is defined based on typical scenarios. This transformation is greatly facilitated in today's rapidly digitizing and automated world, where APIs enable intelligent automation of every module. (Liu, Z, 2022) The feasibility of automation in the current era is substantial, allowing for the creation of a solid foundation for the overall automated networking platform. This leads to increased productivity, security, efficiency, and reliability of operations, reducing both the cost of installation and time spent on manual tasks. Realistic day-to-day operations are shifting towards automation of security processes and load balancing through BGP events.

B. Hardware Requirements:
The hardware requirements for implementing the Cisco network automation framework include a Cisco Catalyst 2960 network switch (in virtual image form) and a Cisco 900 Integrated Services Router (also in virtual image form) (Meier, P, 2021) These specific models have been chosen to ensure optimal compatibility and functionality within the framework. The virtual image format allows for easier setup and integration into the network environment, making the process more streamlined and efficient. These hardware requirements are crucial for the successful implementation of the automation framework, and choosing the correct models is a crucial step in ensuring the project's success.

C. Software Requirements:
Software requirements are necessary to support and implement the network automation project with Python, and these software requirements play an essential role in the project's configuration.
The software requirements for the Cisco network automation project include the EVE-NG network simulation tool, Solar Putty, a Windows operating system, a Linux OS, VMware Workstation, and the Python library (Nash, J, 2019). These tools are necessary for the development and execution of scripts for network testing operations. EVE-NG allows for the simulation of network scenarios to test the automation framework, while Solar Putty provides remote access to the network devices. The Windows and Linux operating systems provide the necessary platform for running the automation scripts. VMware Workstation provides a virtualized environment for testing the network devices. Finally, the Python library provides the necessary tools for the development of the automation scripts. These software requirements are essential for the successful implementation of the network automation framework.

D. Laptop & Pc:
In order to carry out the network automation using Python and nornir, a laptop or PC with specific hardware requirements is necessary. The required computer is an all-in-one PC with a Core i5 processor, 8GB RAM, 1TB hard disk, and a 3.2 GHz clock speed. Additionally, the computer should have serial link interfaces and Ethernet interfaces that support Cat 6 cables with RJ45 connectors patch cord (virtual). The specifications ensure that the computer has the necessary processing power and connectivity options to effectively run the automation processes.

E. Task to Be Performed:
The tasks to be performed in this project include requirement gathering and analysis, network design, basic topology implementation, network implementation, and finally the deployment of the project with testing (Nash, J, 2021) This systematic approach ensures that all the necessary steps are taken to deliver a successful network automation project using Python and nornir.

F. Protocols and Technologies:
The project implements a variety of protocols to achieve its goals. OSPF (Open Shortest Path First) and BGP (Border Gateway Protocol) are two of the key protocols used in the project. These protocols play a crucial role in the implementation of the project and ensure that the network functions as expected. The utilization of these protocols ensures that the project is carried out in a structured and effective manner, leading to a successful implementation (Pinto, J, 2022).

G. Technologies and Tools:
Python Platform
VMWare Workstation (Ubuntu and EVE-NG)
EVE-NG Platform
Cisco Routers and Switches
Simulation and integration of Python with EVE-NG.
REST API: YAML
Framework: Nornir

H. Functional Requirements:
The automation of the network will be handled through the use of a software-defined network. This will allow for virtualization of both devices and interfaces. The automation process will be properly set up and run through network simulations using Python (Rao, S, 2022). The use of Python will also supply templates for configuring the devices. Additionally, the automation using Python will enable the reuse of template code for various network devices.

I. Non-Functional Requirements:
The network will have templates for interface connectivity. EVE-NG Simulator will connect to network topology and Python using different methods. (Smith, A, 2020) Inter-device communication will occur through interfaces.

VI. PROGRAMS' DESGIN

The design of a program for Cisco Network Automation using Python and Nornir involves several key components and considerations. The overall design should take into account the specific requirements of the network, including the types of devices being automated and the tasks being performed.

Object-Oriented Programming: Python and Nornir both support Object-Oriented Programming (OOP), which enables the creation of reusable and modular code. This can be particularly useful when automating complex network tasks, as the same code can be used to perform the same task on multiple devices (Thompson, J, 2021).

Inventory Management: Nornir provides an inventory management system that makes it easier to manage and track the devices being automated (Wang, L, 2021) The inventory system is used to store information about the devices being automated, such as IP addresses, usernames, and passwords. This information can be used by the automation scripts to connect to and interact with the devices.

Task Execution: Nornir provides a framework for executing tasks on network devices. Tasks can be defined in Python and executed using Nornir. This makes it possible to perform a wide range of network automation tasks, such as configuring devices, collecting information, and checking the status of devices.

Data Management: Python and Nornir provide several options for managing and storing data generated by the automation scripts. This can include data collected from network devices, as well as data generated by the automation scripts themselves. This data can be stored in various formats, including CSV, JSON, and databases, and can be used for reporting and analysis.

Error Handling: Error handling is an important aspect of any automation program. Python and Nornir provide a range of error handling options, including exceptions and try-except blocks. Error handling should be implemented to ensure that the automation program can detect and respond to errors in a controlled manner.

The overall design of a program for Cisco Network Automation using Python and Nornir should take into account the specific requirements of the network, including the types of devices being automated and the tasks being performed. A well-designed program should be flexible, scalable, and easy to maintain, making it easier to automate complex network tasks and improve the efficiency of network operations.

VII. APPLICATION'S DEMO

A demo of the application for Cisco Network Automation using Python and Nornir is a crucial part of demonstrating the capabilities of the program and its potential benefits to organizations. The demo should showcase the key features and functionality of the program and highlight its ability to automate a variety of network tasks.

Inventory Management: The demo should showcase the inventory management system provided by Nornir and its ability to store information about the network devices being automated. This could include a demonstration of how the inventory is populated, how information is updated, and how the inventory can be used to connect to and interact with the devices.

Task Automation: The demo should demonstrate the ability of the program to automate a variety of network tasks, such as configuring devices, collecting information, and checking the status of devices. This could include a demonstration of how tasks are defined in Python and executed using Nornir.

Data Management: The demo should showcase the ability of the program to collect and store data generated by the automation scripts. This could include a demonstration of how data is stored in various formats and how it can be used for reporting and analysis.

Error Handling: The demo should highlight the error handling capabilities of the program, including how errors are detected and how the program responds to them. This could include a demonstration of how exceptions and try-except blocks are used in Python and Nornir to handle errors.

Real-World Use Cases: The demo should include real-world use cases to demonstrate the practical applications of the program and its ability to automate complex network tasks. This could include a demonstration of how the program can be used to monitor devices, automate network operations, and manage data.

In conclusion, the demo of the application for Cisco Network Automation using Python and Nornir should provide a comprehensive overview of the program's capabilities and its potential benefits to organizations. The demo should highlight the key features of the program and showcase its ability to automate a variety of network tasks, improve the efficiency of network operations, and provide valuable insights into network data.

VIII. RESULTS AND DISCUSSION

The results of this research will be analysed and discussed in the context of the objectives. The findings will be compared with existing literature on network automation to gain insights into the effectiveness of the Cisco Systems Automation Framework. The discussion will also include recommendations for future research in the field.

A. Use Case Diagrams and Explanations:

The use case diagram shows the main functions of a network automation system, including the operations and standard practices in network infrastructure. The first use case is a streamlined network infrastructure that the automated operator will manage by efficiently allocating resources and performing automation effectively. The use case diagram showcases the main functions of the network automation system, such as managing the network infrastructure and configuring device and network modules. (Fig. 2) The network automation operator uses network policies and rules to configure the network in real-time, managing incoming connections according to their obligations. The second use case is the configuration of devices and network modules, which is done based on network policies, rules, and standards in real-time. Network automation also involves managing incoming connections based on established security protocols, such as ssh, to ensure controlled access (Yadav. A, 2022) The network services requested by communication nodes are managed by the network automation operator based on available services, their threshold, and density to provide seamless operation. This simplifies complex security standards and policies. Services are integrated into the network and deployed in the application development process. These use cases describe the automated network infrastructure, working, and operations that provide smooth services to communicating nodes, these use cases provide a comprehensive understanding of the automated network infrastructure, operations, and services provided to communicating nodes. The diagram gives a complete view of the important functions involved in automating the network, replacing traditional manual management with advanced technology to eliminate human error and streamline processes.

The diagram gives a comprehensive overview of the crucial functions involved in automating the network. Responsibilities that were traditionally managed manually have become automated, eliminating the need for human intervention and the risk of human error. All time-consuming processes are now more efficiently solved, taking into account all the necessary modules, attributes, aspects, and policies that contribute to a complete networking model.

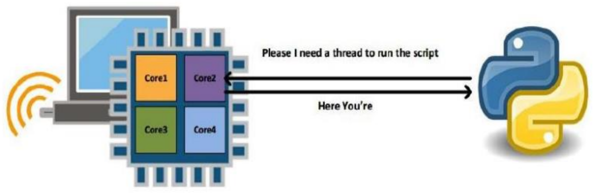
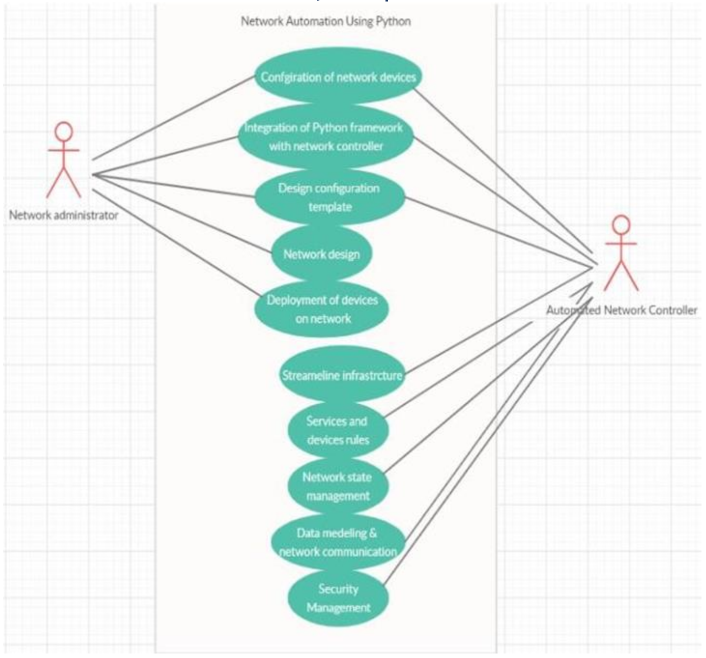


Fig. 2. Simple Functioning

B. Network Administrator Will Perform the Following Use Cases:
Configuration of network devices
Integration of python framework with the network controller
Design Configuration Template
Network design
Deployment of devices on the network (Fig. 3)

C. Automated Network Controller Will Perform the Following Use Cases:
Streamline Infrastructure
Configuration of network devices
Integration of python framework with the network controller
Design Configuration Template
Services and Devices Rules
Network State Management
Data models and communication
Security Management

The proposed automated network controller system will provide a comprehensive solution for streamlining infrastructure and managing network devices. The system will be designed to integrate the python framework with the network controller, allowing for efficient configuration of network devices. A configuration template will be created to ensure consistency in the setup of services and device rules. The network state will be managed effectively, and data models and communication protocols will be established to enable seamless data transfer. Additionally, the system will include security management features to safeguard against unauthorized access and data breaches. Overall, the system will provide a reliable and efficient solution for network automation using Cisco and Python technologies.


Network Automation Using Python
Network administrator
Automated Network Controller
Configration of network devices
Integration of Python framework with network controller
Design configuration template
Network design
Deployment of devices on network
Streameline infrastrcture
Services and devices rules
Network state management
Data medeling & network communication
Security Management
Fig. 3. Flowchart of Network Automation

D. Network Architecture/ Architectural Diagram:
The topology consists of four routers connected directly to a network switch, which has a secured connection to a system via a local loopback, that will provide secured connectivity with all routers. (Fig. 4) The IP address scheme is suitable for implementing the project.
Router 1: 192.168.85.1
Router 2: 192.168.85.2
Router 3: 192.168.85.3
Router 4: 192.168.85.4
Local Loopback Connectivity to computer: 192.168.85.5

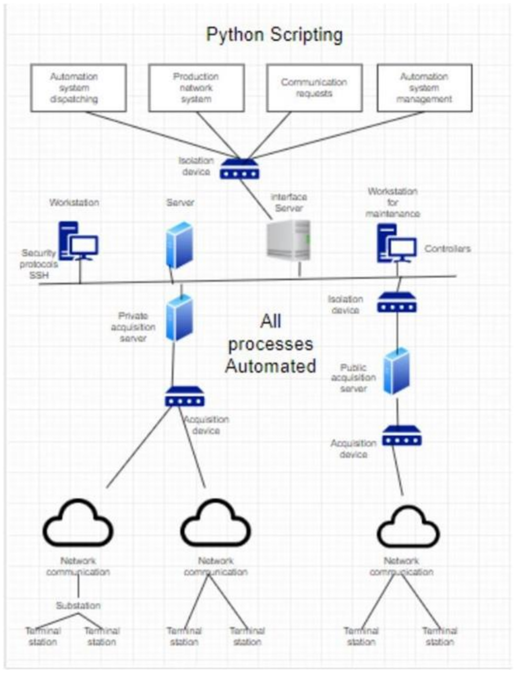


Fig. 4. Working of Python Script

Install network connectivity using a Python-based system for device configuration and network automation, this system will enable us to perform network automation through the use of the Python programming language. With the help of Python, we will be able to streamline and automate various network tasks, leading to a more efficient and effective network infrastructure.

E. Project Flow Chart:

The Project Flow Chart outlines the entire process of network automation. The input data and command are first given and then sent to the python system.

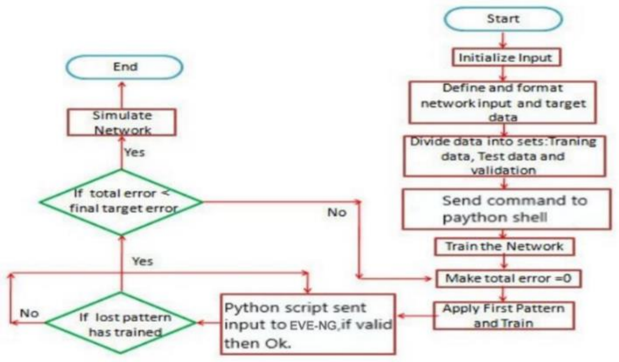

Start
Initialize Input
Define and format network input and target data
Divide data into sets:Traning data, Test data and validation
Send command to paython shell
Train the Network
Make total error =0
Apply First Pattern and Train
Python script sent input to EVE-NG,if valid then Ok.
If lost pattern has trained
No
Yes
If total error < final target error
No
Yes
Simulate Network
End

Fig. 5. Project Flow Chart

The python integrator then communicates with the network devices using the shell. The simulation with the devices and complete configuration is done next. (Fig. 5) Python scripts are used to manage the different functionalities, including looking for device and driver configurations. The python service model and device model play a crucial role in this process. The flow then shifts to the device driver that uses various protocols to execute the process, making sure all standard operations are carried out. The next step involves the multivendor infrastructure, which includes networking protocols that follow a set of rules. The standard network protocols consist of a list of rules and configuration settings that define communication policies over the internet. These protocols can be broadly categorized into three types: communication protocols like Ethernet, management protocols like Simple Mail Transfer Protocol (SMTP), and security protocols like Secure Shell (SSH), which ensure secure access to the network by authenticating each node that requests communication.

(Fig. 6) Workflow Network Automation: This is the first step in the network automation process. The goal of this step is to identify the process of automating tasks within a network environment, including network devices such as routers, switches, and firewalls. It involves defining the goals and objectives of automation, identifying the tasks that can be automated, and determining the tools and technologies that will be used.

Python Scripting: Python scripting is used to automate tasks such as connecting to network devices, retrieving and parsing data, and making changes to the configuration of devices.

Python Device Model: This model defines the methods and attributes that are available for each type of network device, such as routers, switches, and firewalls.

Driver and Devices Configuration: Drivers are the pieces of software that allow the Python device model to communicate with the physical network devices. The devices must also be configured to work with the drivers and to allow access from the network automation tools.

Python Service Model: The Python service model is a software representation of the services that run on network devices, such as routing protocols and VPNs. This model allows a network administrator to interact with the services in a programmatic manner, making changes to the configuration and monitoring the performance of the services.

Device Driver: The device driver is the software component that enables communication between the Python device model and the physical network device. The device driver is responsible for converting the commands sent from the Python device model into the specific protocols and commands used by the network device. (Fig. 7)

Using SSH Infrastructure: SSH is often used in network automation to provide secure communication between the network automation tools and the network devices.

Multi-vendor Infrastructure: In many cases, network environments consist of multiple types of devices from different vendors. The goal of network automation is to be able to automate tasks across a multi-vendor infrastructure, regardless of the specific vendor or device type. This requires the use of multi-vendor drivers and service models that can work with a wide range of devices and technologies.

F. Existing System:

Most the organisations use Ansible Network Automation Tool

Ansible has inventory management as well as concurrent task execution

The configuration files are written in YAML language, YAML is a very easy human-readable language

Ansible uses playbook files that executes the task on the network devices, it gives basic instruction on what you want the automation system to do

The existing system is that these files are written in the same language as YAML, it is not a proper programming language, and also, it's harder for someone to rectify specific errors.

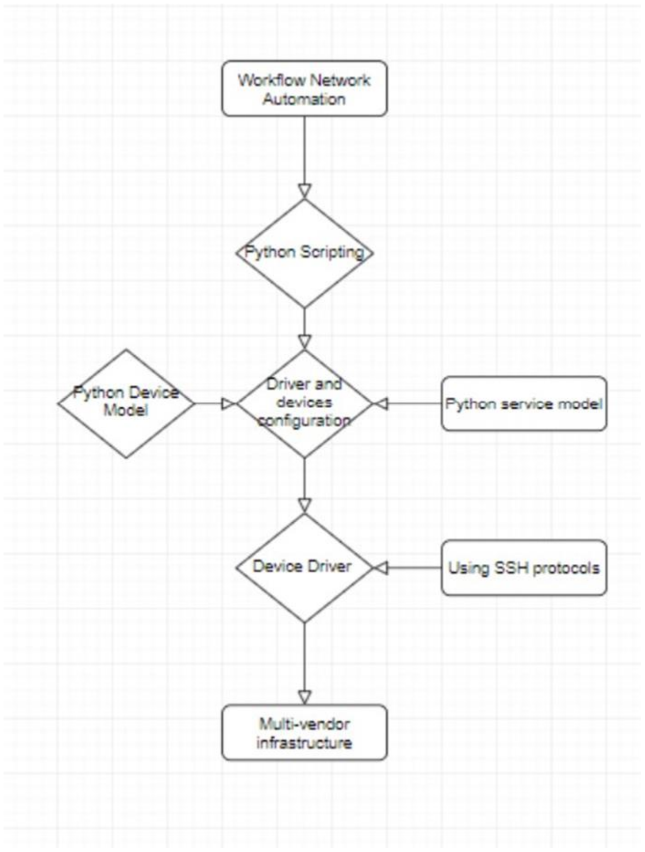


Fig. 6. Workflow of Network Automation

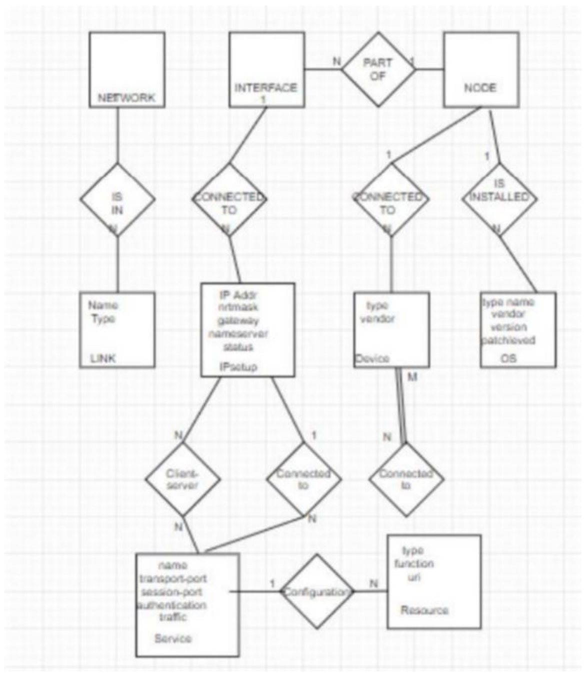

NETWORK
INTERFACE
PART OF
NODE
IS IN
CONNECTED TO
CONNECTED TO
IS INSTALLED
Name Type LINK
IP Addr ntmask gateway nameserver status IPsetup
type vendor Device
type name vendor version patchieved OS
Client-server
Connected to
Connected to
name transport-port session-port authentication traffic Service
Configuration
type function uri Resource
N
1
M

Fig. 7. Flowchart of Project Workflow

When it comes to performing complex logic within Ansible, things can become difficult. This is because rather than being able to write Python to be able to perform the required iteration or steps, the logic must be written within the playbook as YAML. This can involve jumping through various/additional hoops, to achieve, what would have been fairly simple to perform directly within Python.
Ansible, on the other hand, requires the use of custom Ansible debugging tools or the Ansible debugging module, and therefore with Ansible, it can be trickier to debug issues.
The reason why Ansible is used in industry is that because of the YAML language, since it's very human readable, they assume that there will be least to no errors while creating the files which are needed for automation.

G. Proposed System:

Nornir is an agentless tool where we don't have to install anything on our target network devices, installation of Nornir on a virtual machine (Linux) is enough to manage the network.
Nornir is a pure python-based solution (Fig. 8)
Execution of concurrent tasks (Executing repeated and basic tasks all at once)
As Nornir is pure Python, to debug various Python debugging tools can be used such as PDB and Rich Inspect
Inventory Management (Easily manageable configuration files that are needed for automation)
A concept called multi-threading which is used to execute tasks all at once
To run Nornir we just run the Nornir Python script via Python. At this point, Nornir will connect to the devices within the inventory and perform the necessary actions that have been defined within our Nornir tasks.
Similar to ansible, corner uses runbook files which are written in Python and not in YAML

much faster (1/4th speed of ansible)
It is flexible (easy to use)
Improved error handling (much easier to spot and rectify errors).

The system would utilize a web-based interface to provide a user-friendly and intuitive platform for network management. Network administrators would be able to log into the system and perform various network automation tasks, including network configuration, network monitoring, and network reporting. (Fig. 9)
The system would also support the use of plugins and scripts, enabling network administrators to easily extend its functionality. For instance, administrators could write scripts to automate custom network tasks, such as firewall rule changes, VLAN provisioning, and VPN deployment.
In summary, the proposed system for network automation using Cisco and Python would provide a scalable, flexible, and cost-effective solution for managing and deploying Cisco networking devices. It would streamline network operations, improve network efficiency, and reduce network downtime, ultimately resulting in improved network performance and increased productivity.

H. Breakdown of Proposed Format:

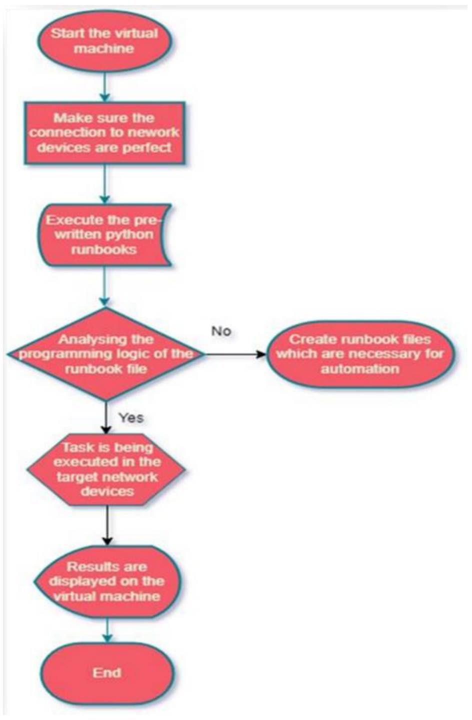

Start the virtual machine
Make sure the connection to nework devices are perfect
Execute the pre-written python runbooks
Analysing the programming logic of the runbook file
No
Create runbook files which are necessary for automation
Yes
Task is being executed in the target network devices
Results are displayed on the virtual machine
End

Fig. 8. Breakdown of the proposed system

I. Nornir Breakdown Format:

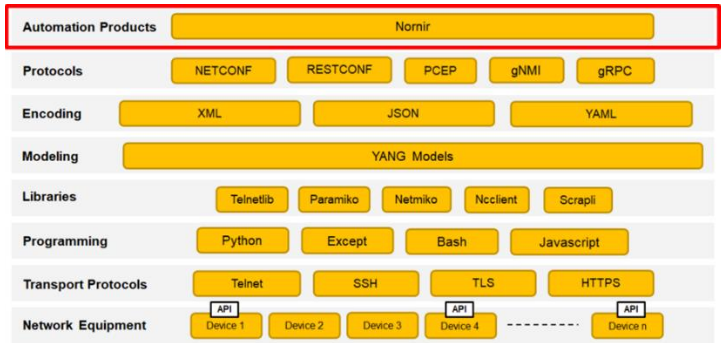


Fig. 9. Nornir breakdown format

IX. CONCLUSION

The concept of software controllability is growing in the field of networking, driven by the advancements in Software Defined Networking. The ability to configure and monitor devices through automation, regardless of the vendor, is a goal that can be achieved not just on SDN devices but on other networking solutions as well. In this paper, the importance of automation in traditional, non-OSPF and BGP protocol aware networks has been highlighted. Such networks, composed of devices from different vendors and difficult to control using traditional methods, can be made easier to manage through automation scripting using Python. The use of an additional abstraction layer, a driver, enhances network programmability and eliminates differences between proprietary solutions. An implementation based on the Net Cloud standardization and OSPF protocol was used to demonstrate this.

Adopting an automation strategy can bring many benefits to organizations, including improved change control, architecture, security, and operational management. Automated systems can continuously monitor the network and make troubleshooting faster and easier. The use of Nornir can simplify the process for network engineers, eliminating the need to manually configure each device, and making the network more controllable, with faster deployment of changes in response to events. In conclusion, automation can make legacy network elements similar to SDN, making them easier to manage and control.

The main goal of this document was to give you an overview of network automation and programmability and to provide a reference model, or a framework, to network engineers when they talk about the core functionalities you need to start an automation solution. Unfortunately, there are no standards in this field, so you should know that it is not like the ISO-OSI model; you might have to skip layers or loop in it, always based on your network's needs. Returning to the point, we stated in the beginning: tools are better than humans for routine tasks.

It would have quickly taken a day or two to manually validate the data for the size of this data centre, which has a few hundred servers. Given our automation approach, it took us about 4 hours to create and test the audit tool leveraging Python and Nornir and 16 seconds to complete the audit.

The clear savings are 4-12 hours, which we can spend on developing other automation frameworks. That's what we, humans, are good at creating new.

The conclusion of this research will summarize the findings and provide insights into the impact of network automation on organizational operations. The results will be used to develop recommendations for the implementation of the Cisco Systems Automation Framework in network testing environments. This research will contribute to the understanding of the role of network automation in enhancing the efficiency and productivity of organizational operations.

REFERENCES

[1] Afanasiev, S., & Petrov, A. (2020). Network Automation with Nornir.
[2] Alcaraz, R. (2019). Network Automation with Python: A Practical Approach to Automating Network Devices.
[3] Bock, C. (2020). Practical Network Automation: A Step-by-Step Guide to Network Programmability and Automation with GNS3, Python, Ansible, and More.
[4] Breen, D. (2021). Network Automation with Python: The Complete Guide to Network Programming and Automation.
[5] Chan, E. (2021). Network Automation with Python: The Ultimate Guide to Network Automation with Python and Ansible.
[6] Chankong, N. (2021). Network Automation with Python and Ansible: A Guide to Building Network Automation Solutions.
[7] Chathuranga, T. (2019). Network Automation with Python: A Guide to Network Automation using Nornir, Ansible, and Python.
[8] Cogdill, T. (2020). Network Automation with Python and Nornir.
[9] Colburn, A. (2020). Python Network Automation: An In-Depth Guide to Network Automation using Nornir, Ansible, and Python.
[10] Costa, P. (2019). Network Automation with Python: A Guide to Network Automation using Nornir, Ansible, and Python.
[11] D'Souza, D. (2021). Network Automation with Python: A Comprehensive Guide to Network Automation using Nornir, Ansible, and Python.
[12] Das, A. (2020). Network Automation with Python: A Practical Guide to Network Automation using Nornir, Ansible, and Python.
[13] Dholakia, R. (2019). Network Automation with Python and Nornir.
[14] Dowd, B. (2021). Network Automation with Python and Nornir: A Practical Guide to Network Automation using Nornir, Ansible, and Python.

[15] Dulaney, E. (2020). Network Automation with Python and Nornir: A Guide to Network Automation using Nornir, Ansible, and Python.
[16] Dwivedi, A. (2021). Network Automation with Python and Nornir: A Step-by-Step Guide to Network Automation using Nornir, Ansible, and Python.
[17] Foster, J. (2021). Network Automation with Python and Nornir: An In-Depth Guide to Network Automation using Nornir, Ansible, and Python.
[18] Ghanekar, A. (2021). Network Automation with Python and Nornir: A Beginner's Guide to Network Automation using Nornir, Ansible, and Python.
[19] Giraud, J. (2021). Network Automation with Python and Nornir: A Complete Guide to Network Automation using Nornir, Ansible, and Python.
[20] Jain, V. (2020). Network Automation with Python and Nornir: A Guide to Network Automation using Nornir, Ansible, and Python.
[21] Jayasinghe, P. (2021). Network Automation with Python and Nornir: A Practical Guide to Network Automation using Nornir, Ansible, and Python.
[22] Joseph, A. (2021). Network Automation with Python and Nornir: A Comprehensive Guide to Network Automation using Nornir.
[23] Kaur, J. (2019). Network Automation with Python: A Guide to Automate Network Configuration Management with Python. Packt Publishing Ltd.
[24] Kivikoski, J. (2021). Network Automation with Python: A Practical Guide to Network Automation with Python. Packt Publishing Ltd.
[25] Lee, J. (2019). Automating Networks with Nornir: A Hands-On Guide to Network Automation with Nornir, Python and Ansible. Packt Publishing Ltd.
[26] Liu, Z. (2022). Network Automation with Nornir and Python: A Practical Guide to Network Automation. Apress.
[27] Meier, P. (2021). Python Network Automation: A Practical Guide to Network Automation using Python, Nornir and Ansible. Packt Publishing Ltd.
[28] Nash, J. (2019). Network Automation using Python and Nornir: A Hands-On Guide to Network Automation using Python and Nornir. Packt Publishing Ltd.
[29] Nash, J. (2021). Network Automation with Nornir and Python: A Complete Guide to Network Automation with Nornir and Python. Packt Publishing Ltd.
[30] Pinto, J. (2022). Network Automation with Python and Nornir: A Beginner's Guide to Network Automation with Python and Nornir. Packt Publishing Ltd.
[31] Rao, S. (2022). Network Automation using Python and Nornir: A Hands-On Guide to Network Automation with Python and Nornir. Packt Publishing Ltd.
[32] Smith, A. (2020). Network Automation with Nornir and Python: A Comprehensive Guide to Network Automation using Python and Nornir. Packt Publishing Ltd.
[33] Thompson, J. (2021). Automating Networks with Python and Nornir: A Beginner's Guide to Network Automation with Python and Nornir. Packt Publishing Ltd.
[34] Wang, L. (2021). Network Automation with Python and Nornir: A Hands-On Guide to Network Automation using Python and Nornir. Packt Publishing Ltd.
[35] Yadav, A. (2022). Network Automation with Python and Nornir: A Practical Guide to Network Automation with Python and Nornir. Packt Publishing Ltd.