Syntax Literate: Jurnal Ilmiah Indonesia p–ISSN: 2541-0849
e-ISSN: 2548-1398
Vol. 9, No. 6, Juni 2024

APPLICATION OF THE MULTI-THREADING METHOD AND PYTHON SCRIPT FOR THE NETWORK AUTOMATION

Deny Wicaksono[1], Benfano Soewito[2]
Computer Science Department, BINUS Graduate Program, Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480[1,2]
Email: deny.wicaksono@gmail.com[1], bsoewito@binus.edu[2]

Abstract

In recent times, there has been a noticeable surge in the inclination towards the implementation of network automation solutions. This trend is driven by the objective of optimizing network availability within the context of hybrid data center networks, which are becoming increasingly prevalent. Reliability, performance, scalability, and minimal resource overhead are crucial solution design characteristics that significantly impact the decision-making process regarding adoption. Recent research has shown that 90% of network outages happen because of human factors and 69% of respondents manage their network with manual method. That high human error of up to 90%, occurs when configuring network devices using manual method. These problems had a negative impact on the functioning of the organization and were harmful to the user. In this research, we investigate solutions to reduce network outage that caused human error using network automation using the Python programming language and multi-threading method. This network automation will reduce configuration time, eliminate human error, and significantly increase efficiency. The aim of this research is to investigate the efficacy of network automation using Python and multi-threading to reduce network outages.The method used in this research is a parallel or multi-tread execution process method using GNS3 as network simulator. Based on experimental results, the multi-thread automation approach is significantly faster than both the serial automation method (67 seconds) and the manual method (248 seconds), this method requiring only 41 seconds to configure all Cisco router devices. If you're looking for speed, look no farther than the multi-thread automation approach, which is 6 times quicker than the manual method and 3.7 times quicker than the serial automation approach. The findings of this study have substantial and immediate implications for the ability of network engineers to speed up the configuration of networks and lower the rate of human error in doing so.

Keywords: Network Automation; Multi-Threading; Network Programming; Network Efficiency; Software Defined Network

Introduction

In 2016, an extensive network outage occurred, affecting millions of customers in important cities throughout Australia, including Sydney, Brisbane, Melbourne, Adelaide, and Perth. Telstra, the largest telecommunications provider in Australia, was the company that experienced the outage. Specifically, a typographical error in a network setup instruction was the cause of the disruption, which was caused by a technical issue. As a consequence of the typographical error, a routing loop was created, which created congestion on the network and finally resulted in the network being down. At around ten o'clock in the morning, Australian Eastern Standard Time, the power outage began and continued for a number of hours (Smolaks, 2016).

The research conducted by Dimensional Research, titled "Network Complexity, Change, and Human Factors Are Failing the Business," encompassed a sample size of 315 network specialists. The results showed that 97% of participants acknowledged that human factors are responsible for network disruptions, while 74% of respondents indicated that network changes can impact their business operations multiple times per year or more. Manual operational methods exacerbate issues as they give rise to complications. The research indicates a positive correlation between network complexity and the occurrence of network disruptions. This sentiment has been conveyed by 59% of the participants in the poll. Furthermore, 69% employ manual operational methods, such as verifying device setups through the command line interface (CLI) and manually conducting traceroute activities. In general, network administration may have inefficiencies as a result of the frequent need for manual checks and monitoring (Veriflow, 2016).

Following the completion of a case study at an e-commerce company, the author made the discovery that the organization suffered four instances of service interruptions between the years 2019 and 2023. These outages were caused by human error in the process of setting up network equipment. During the course of the examination, it was discovered that the firm had four instances of service interruptions that were caused by human error in the configuration of network equipment. When these disruptions occurred, they had a negative impact on the operations of the firm and caused clients to experience financial struggles. Devices such as distribution switches and firewalls, which are necessary for connecting business service server devices to the global internet network, are among the equipment that are responsible for interference. The approach that is currently used for network configuration is a manual method, which requires users to manually input commands or copy and paste commands from a text editor.

One potential solution that may be given is the utilization of a network automation system in order to reduce the possibility of network outages that are caused by human activities inside the corporate network. The findings of this study suggest the development of a network automation system that takes the Python programming language as its foundation and makes use of the Netmiko library to establish a remote connection to the device. The utilization of multi-threading or parallel process is suggested as a means of enhancing the speed of the device and providing a way for the execution of commands. When it comes to the software that simulates networks, it makes use of GNS3, which is both open source and free.

There are a variety of approaches that may be followed when it comes to automating networks and systems, and among these many alternatives, Python programming techniques are becoming an increasingly frequent practice. According to the results of a research that was conducted to create and test a network automation tool, it is feasible to develop a tool that is based on Python (Larsson, 2020). In the event that there are several suppliers involved, this is the situation that arises. There are plans in place to combine virtual network operations and automate services in order to provide end-to-end quantum encryption (Aguado et al., 2018). These plans pertain to the provision of quantum encryption. Over the course of the past several years, there have been significant shifts in both the automation of systems and the distribution of network services. There are a substantial number of components of the infrastructure, including security, that need to be automated in order to facilitate their capacity to rapidly adapt to changing circumstances (Arifin et al., 2019).

This research takes use of automation and the Python programming language in order to offer unique approaches to the setup of network devices. The objectives of this research are mentioned above. The amount of time that is spent preparing and maintaining the equipment is significantly cut down as a result of this (Web Pages and Mobile Apps, 2019). Python programming is employed for the goal of automating and abstracting network operations. The standard techniques of configuring network equipment are no longer a feasible alternative since they require a lot of manual labour and are prone to making mistakes. The expense of employing additional staff to finish the task is prohibitively high for major firms, which is another reason why the work cannot be completed. It is anticipated that the current trend of quickly growing the number of devices that are linked to a network will continue in the foreseeable future. Consequently, as a result of this, a rising number of firms are turning to automation since it helps them to achieve the right levels of speed, agility, consistency, and efficiency (Mihăilă et al., 2017). This is an immediate result of the circumstances that have arisen.

The implementation of network automation through the usage of Python scripts makes it feasible to configure network devices, most notably Cisco routers. This is made possible by the development of network automation. When compared to the use of human procedures, the results reveal that the amount of time required for configuration may be reduced by as much as 99% when automation is utilized. This is a significant reduction. Furthermore, automation reduces the risk of human mistake, which results in setup that is both speedier and more precise. This is a significant advantage (Bouhouras et al., 2010).

Because of the availability of several libraries, such as Paramiko and Netmiko, Python is utilized in the field of network technology. This is because these libraries make it easy to automate network setups. To execute certain programmes, it is necessary to include Python libraries in the programme script and this is a mandatory need (Mazin et al., 2021). A programme written in Python known as the Paramiko module is capable of performing the functions of both a client and a server, which enables the establishment of a secure shell connection. Paramiko is able to assist the setting of some pieces of network equipment, such as routers and Cisco switches, by utilizing a Secure Shell (SSH) connection. Both of these devices are examples of network equipment. It is essential for Paramiko to possess the Internet Protocol (IP) address, the Username, and the Password in order for her to be able to get access to the network device. If all three credentials are genuine, the Paramiko is able to obtain access to a network device by utilizing the SSH Client (Nugroho & Pujiarto, 2022). This is done in order to acquire access to the device.

In Python, the Netmiko library is a library that enables SSH connections on many vendors of network devices, sometimes known as multivendor support. In addition to being a Python library, the Paramiko library serves as its foundation. On the other hand, Netmiko is not able to independently determine which device from a certain vendor is being used without the assistance of the seller. Because it has a function called Connect Handler, Netmiko is more user-friendly than Paramiko when it comes to connecting to devices that are connected to a network. Using this capability, the process of establishing connections between Netmiko and assorted devices that are linked to a network is simplified. It is required for Netmiko to possess information in order for it to be able to access a network device. This information includes an Internet Protocol (IP) address, the category of the device, a username, and a password associated with the device. Netmiko is unable to automatically detect the kind of network device; an example of a network device type is cisco_ios, which is a router network device that is made by Cisco vendors (Mauboy & Wellem, 2022). Netmiko lacks the capability to automatically determine the type of network device.

Multiple protocols exist for remote access to a router, including Telnet and Secure Socket Shell (SSH) (Waheed & Ali, 2018). Telnet is an unencrypted protocol that is focused on transmitting text and does not include any authentication mechanisms (Maurice et al., 2017). A secure remote login to a computer or server can be achieved through the use of a network protocol known as the Secure Shell (SSH) protocol (Ylonen et al., 2015). It functions on the standard User Control Protocol (TCP) port 22, which is referred to as port 22. The usage of popular cryptographic algorithms, which Port 22 is able to function in a secure manner, may be used to achieve the task of authenticating users and assuring the safety of communication sessions. Within the context of this specific scenario, the standard encryption process involves encrypting data before to its transmission and then decrypting the data once it has arrived at its intended location (Ayasso & Mohammad-Djafari, 2010). The aim of this research is to investigate the efficacy of network automation using Python and multi-threading to reduce network outages caused by human error, decrease configuration time, and enhance efficiency in hybrid data center networks.

Research Methods

Fig. 1, This diagram depicts the network topology of the automation server, which is necessary in order to achieve the goals of the study. It is of the utmost importance that the topology must not solely rely on conventional techniques of network design but rather should also make it possible to automate the network. It is necessary to configure and validate the topology of the network before carrying out network testing that is appropriate.

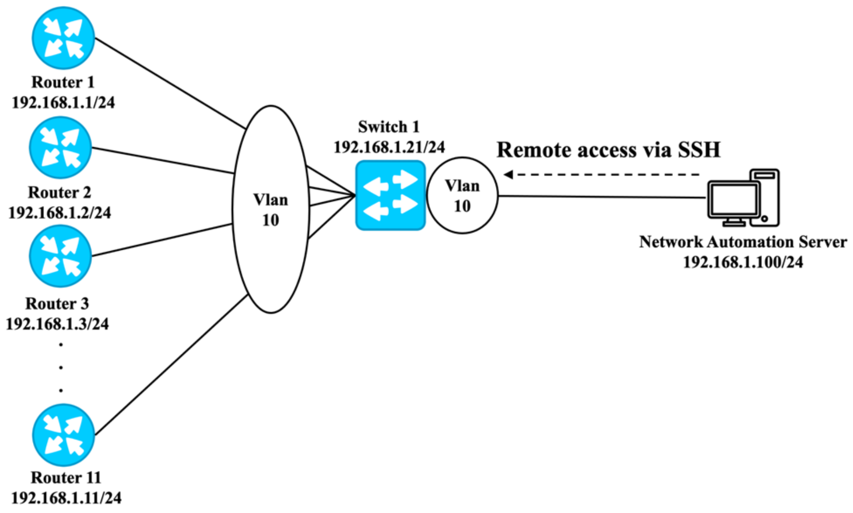


Figure 1. Server automation topology

The IP address scheme that will be used as the device management IP address that will be accessible by the network automation server is the private IP address 192.168.1.0/24. This is shown in the topology picture. A switch management device will be used to establish a connection between the network device and the network automation server, which is located on the right and has an IP address of 192.168.1.100. On the right, the switch management device will establish a connection between the network automation server and eleven routers from the Cisco 7200 series. These routers will have IP addresses ranging from 192.168.1.1 to 192.168.1.11. Vlan 10 will be used as a specific vlan for the purpose of managing devices that are connected to this connection. The network automation server will be able to establish a connection with the router device if this topology is utilized, as the router device is located inside the same IP address subnet. By utilizing the Python programming language in conjunction with the Netmiko library, the network automation server operates on the Linux Ubuntu operating system version 22.04. Through the use of the Secure Shell (SSH) protocol and the Command Line Interface (CLI) approach, the router devices will be able to be automatically configured from the network automation server. This will be accomplished through the use of remote access to the device. Developing a system for network automation through the use of the Python programming language is one way to accomplish this goal.

As part of this research, a number of different tools and pieces of software are utilized in order to successfully operate a network automation system. In addition, there is the software known as VMware Workstation, which serves as a supporter for virtualization; GNS3, which functions as a network simulator; Linux Ubuntu, which serves as the operating system of the network automation server; Jupyter notebook, which functions as an application for generating Python code and the Iperf3 application, which functions as a traffic generator in throughput mode (Taruk et al., 2018).

VMware Workstation

A virtual machine (VM) is a separate, self-contained computing environment where several operating systems can run concurrently on a physical computer by means of a software application known as a hypervisor. With virtual machines (VMs), users can run multiple operating systems and apps at once without interfering with one another. Virtual machines (VMs) function by assigning physical resources, such CPU, memory, and storage, to individual computers and maintaining total isolation between them. Virtualization software such as VMware Workstation Pro Virtual Machine enables users to run virtual operating systems on real computers. Any operating system, including Windows, Linux, and macOS, may be run concurrently on a single computer. Software for Virtual Machines in this study, VMware Workstation Pro version 17 is utilized to run the Linux operating system and GNS3 software in order to simulate the operation of the network automation system.

Simulation Topology

Fig. 2, demonstrates simulated network is divided into IDC1 on the left and IDC2 on the right data center locations. With simulations connected to external networks and the internet, the IP address scheme used is a public IP address. IDC1 utilizes the IP address block 124.158.1.0/24, while IDC2 uses the IP address block 124.158.2.0/24. Each IDC simulation involves the connection of four routers to four server machines. Within each IDC, there is a router device that serves as a distribution router. The distribution router is a bridge between the network in the IDC and the core gateway router device, where the core gateway router will be connected to the internet. Each IDC 1 and IDC 2 is comprised of five Cisco routers running dynamic OSPF routing with area 0. Using dynamic external BGP routing towards the IDC router and the internet, one of the routers serves as a gateway to the external network and the internet. Internet-connected routers that are tethered to a server on the internet. OSPF routing protocol, used to routing exchange between routers inside the IDC. Servers side used static routing and pointing the router IP address as the default gateway. The expected results after configuring this topology are from each server on IDC1 and IDC2, can ping the server on the internet.

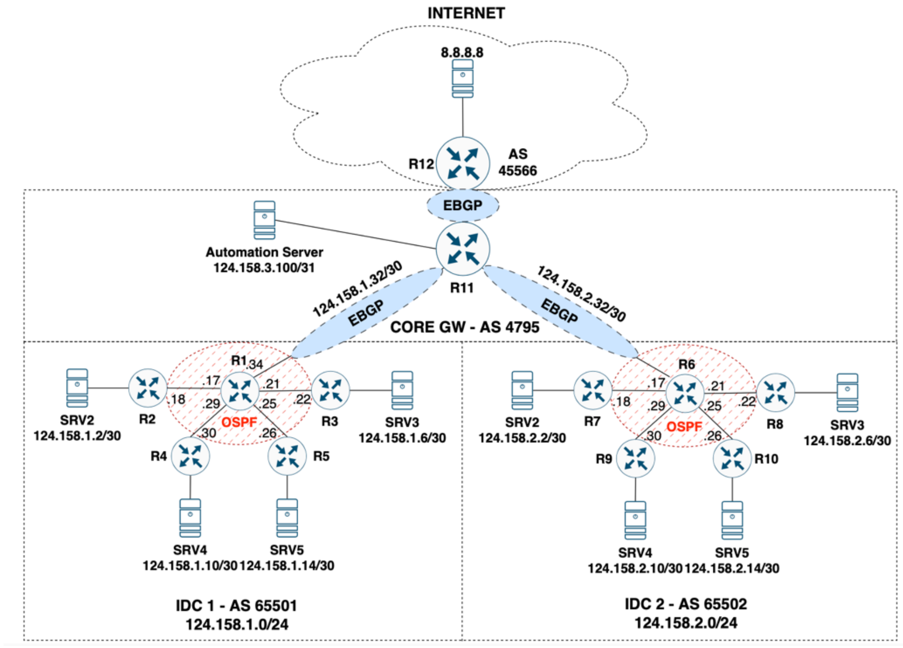


Figure 2. Logical simulation topology

Additionally, the pre-configuration is done in order to enable remote access to the router device from the network automation server by means of the SSH protocol. The setup that is required includes the setting of the username and password, the hostname, the management of IP addresses, the activation of the SSH protocol, and the activation of line vty.

GNS3 Network Simulator

Emulated GNS3 connects virtual devices to real and other virtual devices by providing them with the necessary tools to connect. Features of GNS3 have the potential to considerably facilitate areas such as utilization, reusability, manageability, interconnection, and dissemination. As a result, GNS3 will cut down on unnecessary expenses and delays. Using templates and libraries, users are able to design and configure network nodes with the help of GNS3, which is a clientless network emulator that has a user interface that is based on a browser. After downloading the software from the GNS3 website, the GNS3 network simulator may then be installed on your computer. A representation of the GNS3 network simulation may be seen in Fig 3.

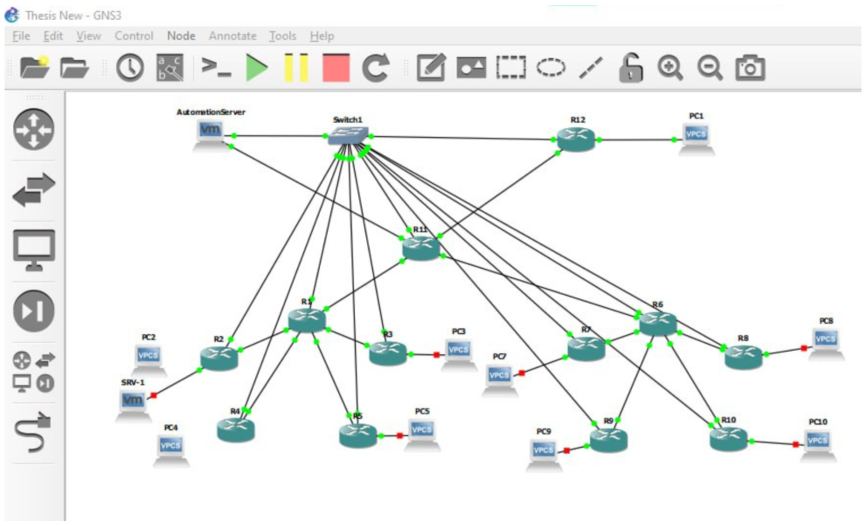


Figure 3. Simulation topology with GNS3

Prerequisites are the conditions that must be met in order to participate in concerned with this research. This system requires a central processing unit (CPU) that is manufactured by Intel and is compatible with the Intel® brand. The virtualization software that is required is VMware Workstation 15 or a later version, and the operating system that should be used is either Windows 10 or Windows 11. The hardware and software requirements for a laptop or PC are presented in the following Table 1.

Table 1. Hardware & Software Requirements

Items	Requirements
CPU	Intel i5 / i7 (4 Logical processors)
RAM	Min. 16 Gb
HDD Space	Min. 40 Gb
Network	LAN / WLAN
Operating System	Windows 10 or 11
Virtualization Software	Min. VMware Workstation Version 15

System Testing & Measurement

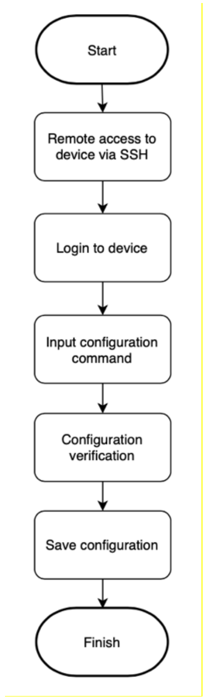

Start
Remote access to device via SSH
Login to device
Input configuration command
Configuration verification
Save configuration
Finish

Figure 4. Flow chart for automation process

Fig. 4 illustrates the complete process of testing simulation which includes the automation phase. Through the use of the Secure Shell (SSH) protocol and the establishment of a connection to the particular IP address that has been allocated to the device, the automation server will create a remote connection to the targeted device. After that, the system will begin the process of logging in by establishing the settings for the username and password that have been specified. In the following step, the server will proceed to input configuration commands by utilizing the Command Line Interface (CLI) of the device. The results of the configuration that were received will be subjected to a verification procedure in order to determine whether or not the device configuration was accurate and to locate any potential errors in the direction of the configuration commands. As soon as the verification procedure of the configuration command has been finished without any errors, the configuration that is now being used on the device will be stored. After then, the process of configuring the gadget has been completed in its entirety.

Users' perceptions of delay values, which are regarded to be an important factor, have a significant impact on the evaluation of the quality of the network service (Tolly, 2021). Relatively significant changes in the delay value from the beginning to the end can be detrimental to the effectiveness of communication attempts. In order to guarantee the maintenance of high-quality real-time traffic, the International Telecommunication Union (ITU) G.114 standard recommends that the maximum allowable end-to-end delay in one direction should not exceed 150 milliseconds (ms) (Taruk et al., 2018). Table 2 shows the ITU-T G.114 for delay standard.

Table 2. ITU-T G.114 Delay Standard

Category	Value
Good	0 – 150 ms
Medium	150 – 400 ms
Poor	> 400 ms

The packet loss is an important aspect that should be considered when evaluating the performance of a service since it has an effect on a wide variety of applications. When the rate of packet loss exceeds a particular threshold, the performance of the system is greatly impacted, and even when the threshold is at its highest, the system may become completely unusable. Issues such as congestion and packet loss have a negative impact on the routing of packets and their effective delivery to the intended destinations (Oleiwi et al., 2022). Table 3 shows the ITU-T G.114 for packet loss standard.

Table 3. ITU-T G.114 Delay Standard

Category	Value
Very Good	0 %
Good	3 %
Medium	15 %
Poor	25%

The Ping application is initiated from the network automation server in order to carry out measurements of delay and packet loss. During the Ping test, a total of 100 packets are continuously transmitted over a span of approximately 1 minutes and 40 seconds to selected pair of servers inside the GNS3 simulated topology. Every Ping packet consists of a 64-byte Internet Control Message Protocol (ICMP) echo, which is transmitted at regular intervals of one second. This study focuses on the measurement of the mean delay and percent of packet loss seen in the final Ping test. The measurement of throughput was conducted by transmitting synthetic network traffic, which was generated using the Iperf3 application, from the network automation server to two representative servers inside the GNS3 simulation architecture together with the ping test. The server responsible for network automation will function as the primary server, while the remaining two servers will operate as clients. The test employs TCP packets, with a test length of around 100 seconds. The evaluation of network performance occurs during four specific time periods throughout the day: late night (00:00 to 06:00), early morning (06:00 to 12:00), late afternoon (12:00 to 18:00), and early evening (18:00 to 24:00).

Throughput, a quantitative measure of bandwidth at a specific moment and under specific network conditions, represents the efficiency of data transfer. Throughput is a measure of the amount of data that can be transmitted over a network during a given time period, and it is directly related to the capacity needed to transfer a file of a specific size. The system throughput is the aggregate data transfer rate over the whole network terminal (Taruk et al., 2018). Table 4 shows the ITU-T G.114 for throughput standard.

Table 4. ITU-T G.114 Throughput Standard

Category	Value
Very Good	100%
Good	75%
Medium	50%
Poor	25%

Throughput testing at network of the e-commerce company for study case, requires a reference value to measure throughput in network simulations using the GNS3 simulator. The reference value is based on the production network traffic utilization currently running at the site. The data is taken from the NMS server within a month, with the maximum traffic utilization value taken at four times. Based on the average value of the traffic, the percentage of traffic utilization to network capacity that can be used as a reference for throughput testing, namely at 00.00 - 06.00 is 25%, at 06.00 - 12.00 is 27%, at 12.00 - 18.00 is 32% and at 18.00 - 24.00 is 36%.

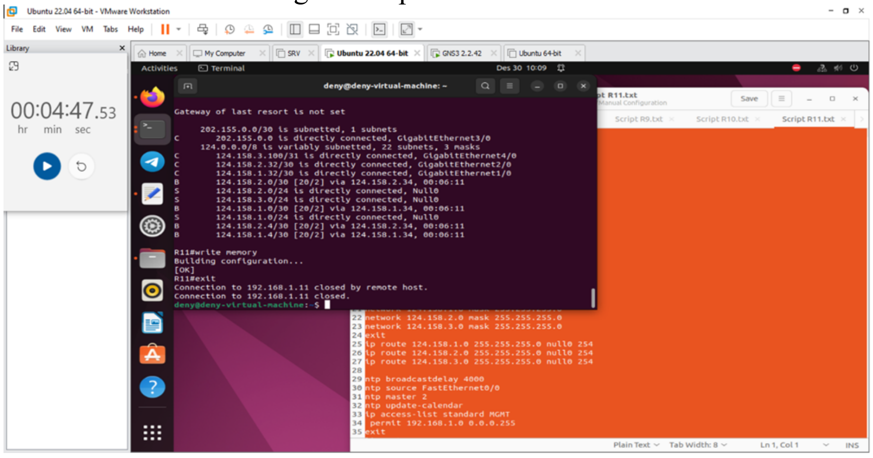
The GNS3 application, which functions as a network simulator, has restrictions when it comes to conducting throughput testing. The utilized maximum throughput does not align with the capacity of the router interface employed for simulation testing. Iperf3 utilizes a client-server framework to gauge the highest throughput of User Datagram Protocol, TCP, and Stream Control Transmission Protocol between client and server stations (Tolly, 2021). Iperf3 is intended to measure the maximum achievable bandwidth on Internet Protocol-based networks (Kopeć, 2022). The author performed throughput testing by utilizing the Iperf3 tool to assess the highest possible capacity of the GNS3 simulator network. This was done by employing a network automation server and one sample server behind R1 router on the topology. The author discovered that Iperf3 does not impose any restrictions on bandwidth when it comes to achieving maximum throughput. The average throughput number obtained from these tests reflects the highest network capacity that can be achieved using the GNS3 simulator. According to the test results, the highest possible network capacity of the GNS3 simulator we utilize is 12.9 Mbps. This figure will serve as the benchmark for conducting throughput testing on the GNS3 network simulation.

Result and Discussion

The Cisco devices were setup to conduct a comparative analysis between manual and automated operations, specifically focusing on the time required for execution and the Quality of Service (QoS) metrics such as delay, packet loss, and throughput. A total of 11 Cisco devices were configured by three different methods: manual configuration, serial automatic configuration, and multi-thread automatic configuration, facilitated by a Python script software. This approach is implemented to ensure that the outcomes accurately represent the environmental and network conditions inside an authentic organizational setting. The measurement of time commences upon initiating the SSH login process to the initial Cisco device and concludes at the completion of configuring the final Cisco device. The approach is effectively employed for the manual configuration process. The automatic configuration process, whether serial or parallel, involves the recording of the configuration process time through the implementation of a Python program.

Manual Configuration

The manual technique necessitates first gaining remote access to the device in order to make advantage of the SSH protocol. Then, configure the router device by manually inputting instructions directly by copying and pasting the text editor application into the Linux terminal program that is linked to the device. This will begin the configuration process. Utilizing the stopwatch feature of a computer, the duration is calculated by determining the amount of time that has passed between the beginning and the conclusion of the setup. Fig. 5 shows the time calculation and manual device configuration process.



```
Gateway of last resort is not set

      202.155.0.0/30 is subnetted, 1 subnets
C        202.155.0.0 is directly connected, GigabitEthernet3/0
      124.0.0.0/8 is variably subnetted, 22 subnets, 3 masks
C        124.158.3.100/31 is directly connected, GigabitEthernet4/0
C        124.158.2.32/30 is directly connected, GigabitEthernet2/0
C        124.158.1.32/30 is directly connected, GigabitEthernet1/0
B        124.158.2.0/30 [20/2] via 124.158.2.34, 00:06:11
S        124.158.2.0/24 is directly connected, Null0
S        124.158.3.0/24 is directly connected, Null0
B        124.158.1.0/30 [20/2] via 124.158.1.34, 00:06:11
S        124.158.1.0/24 is directly connected, Null0
B        124.158.2.4/30 [20/2] via 124.158.2.34, 00:06:11
B        124.158.1.4/30 [20/2] via 124.158.1.34, 00:06:11

R11#write memory
Building configuration...
[OK]
R11#exit
Connection to 192.168.1.11 closed by remote host.
Connection to 192.168.1.11 closed.
deny@deny-virtual-machine:~$
```

```
network 124.158.2.0 mask 255.255.255.0
network 124.158.3.0 mask 255.255.255.0
exit
ip route 124.158.1.0 255.255.255.0 null0 254
ip route 124.158.2.0 255.255.255.0 null0 254
ip route 124.158.3.0 255.255.255.0 null0 254

ntp broadcastdelay 4000
ntp source FastEthernet0/0
ntp master 2
ntp update-calendar
ip access-list standard MGMT
 permit 192.168.1.0 0.0.0.255
exit
```

Figure 5. Manual configuration process

Two hundred and forty-two seconds, which is equivalent to four minutes, was the total amount of time that was necessary for the configuration of Cisco equipment. Using these statistics, it was determined that the average amount of time required to configure a single piece of Cisco equipment was 22.6 seconds. The time necessary for manual configuration was presented in Table 5.

Table 5. Duration of manual configuration

Process	Time Duration (second)
Router 1 - 11	248.20
Average per router	22.56

Serial Automation

A sequential procedure was carried out with the assistance of automation, and it was carried out in accordance with a topology that had been specified beforehand. 66.65 seconds was the total length of time that included the entirety of the procedure, beginning with its beginning and ending with its conclusion. Because of this, it was found that the typical amount of time required to configure a single piece of Cisco hardware was just 6.06 of a second. The configuration operations on several router devices were carried out in a sequential manner by the Python script programme. The programme would finish the configuration on one device before moving on to the next gadget. This procedure was carried out several times until a total of eleven routers had been correctly configured. Table 6 presented comprehensive data regarding the procedure for documenting the commencement time, conclusion time, and duration of every router device configuration.

Table 6. Duration of serial automation

Process	Time Duration (second)
Router 1 - 11	66.65
Average per router	6.06

Multi-thread automation

In order to carry out the configuration process in parallel, the multi-thread automation method that is commonly referred to as the parallel process approach required the utilization of a Python script programme. Due to the fact that this technique makes use of the Threading library, it was necessary to carry out the configuration process concurrently on all of the routers that were contained inside the specified setup. Therefore, the Python script required a time period of just 41.20 seconds, which was less than one minute, in order to set up a total of eleven Cisco routers. Using these statistics, it was determined that the average amount of time required to configure a single piece of Cisco equipment was 3.75 seconds. The findings of the parallel automated experiment were presented in Table 7.

Table 7. Duration of multi-thread automation

Process	Time Duration (second)
Router 1 - 11	41.20
Average per router	3.75

Comparison of Manual and Automatic Results

The utilization of automation strategies, more especially the utilization of a multi-thread process using a Python programmed, came about as a consequence of a large reduction in the amount of time that was required for the configuration of eleven Cisco devices. A time savings of 207 seconds was achieved through the utilization of this technique, which can be regarded as a small 17% reduction in comparison to the amount of time required for the manual setup of the devices. Fig. 6 illustrated the comparison of the time required to configure 11 Cisco devices among the manual approach, serial automation, and parallel automation.

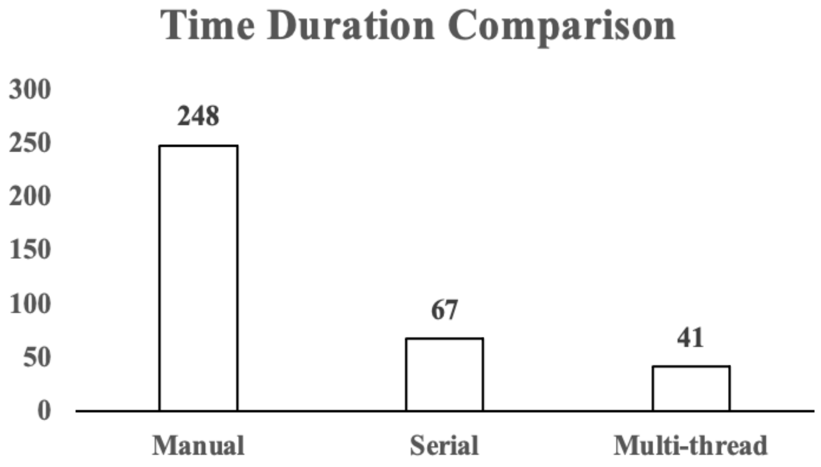


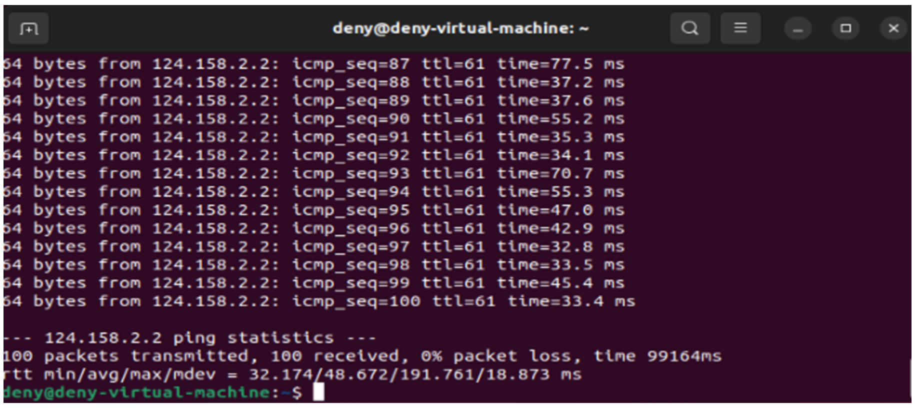
Figure 6. Time Duration Comparison

In accordance with Cisco's definition, a network of medium size is distinguished by its capability to provide services for a number of devices ranging from 200 to 1,000 (Academy, 2014). Therefore, in the case of a network of moderate size consisting of one thousand devices, the process of manually configuring all of the devices takes roughly three hundred and seventy-six minutes, which is equivalent to a period of six and a half hours. Alternately, the target might be completed by the use of multi-thread automation, which would result in a time period of 62.4 minutes, which is nearly equivalent to 1.04 hours. As a consequence, the installation of automation might have led to a reduction of 5.23 man-hours, which is comparable to an 83% decrease in the amount of time spent.

Modern IT infrastructure management relies on network automation to boost efficiency and effectiveness. These procedures are usually faster and more successful depending on how they are executed. Sequential network automation processes can take a long time, especially in complicated network systems. A proposed switch to multi-thread processing is advised to overcome this issue and speed up implementation. In this case, Python script change is crucial. Network engineers can utilise threading libraries like the popular one to turn automated operations into multi-threaded activities. This method allows splitting jobs onto many threads, making concurrent execution easier and allowing automation operations to be handled simultaneously. Because of this, network administrators may boost velocity and efficacy, which can reduce the time needed to perform key network management tasks. This ensures that network automation meets current IT environments' changing needs.

Quality of Service (QoS) Measurements

This study investigated the assessment of delay and evaluation of data transfer capacity in the context of network automation. The assessment of delay was conducted using the Ping application from Linux terminal. Ping was sent from the automation server to the personal computers located in IDC1 and IDC2. Fig. 7 shows ping result during testing.



```
deny@deny-virtual-machine: ~
64 bytes from 124.158.2.2: icmp_seq=87 ttl=61 time=77.5 ms
64 bytes from 124.158.2.2: icmp_seq=88 ttl=61 time=37.2 ms
64 bytes from 124.158.2.2: icmp_seq=89 ttl=61 time=37.6 ms
64 bytes from 124.158.2.2: icmp_seq=90 ttl=61 time=55.2 ms
64 bytes from 124.158.2.2: icmp_seq=91 ttl=61 time=35.3 ms
64 bytes from 124.158.2.2: icmp_seq=92 ttl=61 time=34.1 ms
64 bytes from 124.158.2.2: icmp_seq=93 ttl=61 time=70.7 ms
64 bytes from 124.158.2.2: icmp_seq=94 ttl=61 time=55.3 ms
64 bytes from 124.158.2.2: icmp_seq=95 ttl=61 time=47.0 ms
64 bytes from 124.158.2.2: icmp_seq=96 ttl=61 time=42.9 ms
64 bytes from 124.158.2.2: icmp_seq=97 ttl=61 time=32.8 ms
64 bytes from 124.158.2.2: icmp_seq=98 ttl=61 time=33.5 ms
64 bytes from 124.158.2.2: icmp_seq=99 ttl=61 time=45.4 ms
64 bytes from 124.158.2.2: icmp_seq=100 ttl=61 time=33.4 ms

--- 124.158.2.2 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 99164ms
rtt min/avg/max/mdev = 32.174/48.672/191.761/18.873 ms
deny@deny-virtual-machine:~$
```

Figure 7. Ping result during testing

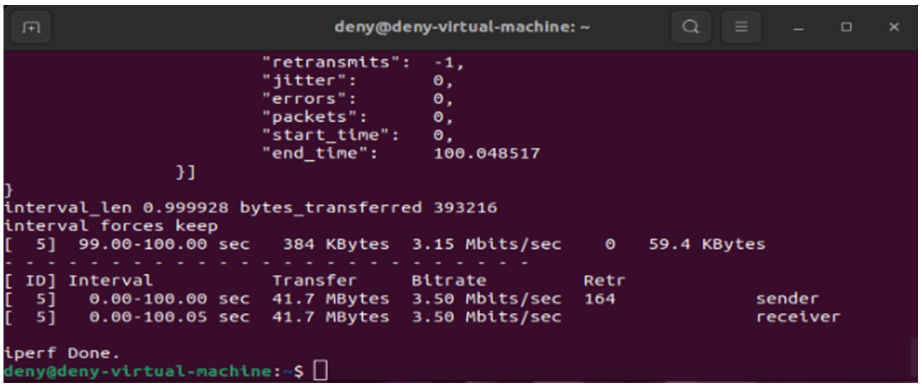
Based on the results of testing using the GNS3 simulator, the average test delay results show a negligible difference of less than 1 millisecond. This indicates that there is no major distinction between the manual, serial, and multi-thread techniques. The test results from two test servers, using three methodologies and four distinct testing times, consistently show a latency of less than 50 milli second. Although the results of packet loss tests are nearly same, there is a slight variance in the average packet loss, which is less than 1 percent. The average packet loss findings of the three approaches at four different periods range from 0.71% to 1.43%. This indicates that among 100 Ping tests, the number of request time out results on the ICMP packet transmitted is fewer than 2. Table 8 displays the outcomes of the experiments conducted to measure delay and packet loss.

Table 8. Delay and Packet loss measurements

Time	Server IP Address	Delay (milli second)			Packet Loss (%)		
		Manual	Serial	Multi-Thread	Manual	Serial	Multi-Thread
00:00 - 06:00	124.158.1.2	48,69	47,39	47,22	0,71	1,09	1,00
	124.158.2.2	47,66	47,01	47,07	0,83	1,03	0,80
06:00 - 12:00	124.158.1.2	48,14	48,55	47,68	1,00	0,94	1,23
	124.158.2.2	48,23	48,58	48,93	0,91	0,97	1,00
12:00 - 18:00	124.158.1.2	49,98	49,53	47,00	1,31	1,43	0,77
	124.158.2.2	49,65	49,90	48,61	1,26	1,29	1,03
18:00 - 24:00	124.158.1.2	49,88	49,73	49,34	1,43	1,23	1,34
	124.158.2.2	49,82	49,96	49,76	1,43	1,40	1,40

Based on the requirements set by ITU-T G.114 regarding delay and packet loss, it may be inferred that the test delay findings are classified as "good." This means that the average delay recorded is less than 150 milli second. Regarding the results of the packet loss test, it falls under the good category as the average packet loss is below 3%.

The Iperf3 traffic generator program was employed in throughput testing to transmit TCP packets from the network automation server to two sample servers within the GNS3 simulation architecture in IDC1 and IDC2. Fig. 8 illustrated the throughput test result using Iperf3 application.



deny@deny-virtual-machine: ~

```
                        "retransmits":  -1,
                        "jitter":       0,
                        "errors":       0,
                        "packets":      0,
                        "start_time":   0,
                        "end_time":     100.048517
                }]
}
interval_len 0.999928 bytes_transferred 393216
interval forces keep
[  5]  99.00-100.00 sec   384 KBytes  3.15 Mbits/sec    0   59.4 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-100.00 sec  41.7 MBytes  3.50 Mbits/sec  164             sender
[  5]   0.00-100.05 sec  41.7 MBytes  3.50 Mbits/sec                  receiver

iperf Done.
deny@deny-virtual-machine:~$ 
```

Figure 8. Throughput test result using Iperf3

According to the measurements, the manual, serial, and multi-threaded methods all produced the same throughput test results at four different times categories. Specifically, the throughput was 3.2 Mbps from 00:00 to 06:00, 3.5 Mbps from 06:00 to 12:00, 4.1 Mbps from 12:00 to 18:00, and 4.6 Mbps from 18:00 to 24:00. This demonstrates that while the GNS3 simulator has constraints on its maximum traffic capacity, it does not align with the interface on the router device. However, GNS3 can direct traffic depending on the percentage of use, as per the references from the production network at e-commerce company. The experimental results for measuring throughput of each configuration approach utilized were presented in **Table 9**.

Table 9. Throughput Measurement

Time	Server IP Address	Throughput Test					
		Manual		Serial		Multi-Thread	
		Reference (Mbps)	Result (Mbps)	Reference (Mbps)	Result (Mbps)	Reference (Mbps)	Result (Mbps)
00:00 - 06:00	124.158.1.2	3,2	3,2	3,2	3,2	3,2	3,2
	124.158.2.2	3,2	3,2	3,2	3,2	3,2	3,2
06:00 - 12:00	124.158.1.2	3,5	3,5	3,5	3,5	3,5	3,5
	124.158.2.2	3,5	3,5	3,5	3,5	3,5	3,5
12:00 - 18:00	124.158.1.2	4,1	4,1	4,1	4,1	4,1	4,1
	124.158.2.2	4,1	4,1	4,1	4,1	4,1	4,1
18:00 - 24:00	124.158.1.2	4,6	4,6	4,6	4,6	4,6	4,6
	124.158.2.2	4,6	4,6	4,6	4,6	4,6	4,6

On the basis of the delay and packet loss standards established by ITU G.114, we are able to draw the conclusion that the throughput test results are in the good category. This is because the throughput generated reaches 100 percent of the traffic that was transmitted by the Iperf3 server at four different test times.

The data obtained from tests measuring delay, packet loss and throughput in network configuration have shown several aspects that necessitate additional comprehension. Despite the absence of statistically significant disparities in characteristics such as delay, packet loss, and throughput among manual, serial automated, and multi-thread automated configuration approaches, it is crucial to acknowledge that these findings may have relevance within a particular experimental context based on the topology and network simulator employed, but their direct applicability to all network scenarios may be limited. Hence, it is imperative to conduct additional research and undertake a comprehensive analysis in order to comprehend the ramifications of these discoveries on wider network construction methodologies.

These tests offer intriguing first findings on network configuration and Quality of Service (QoS). However, there remain some elements that warrant further investigation. To arrive at more thorough results, it may be imperative to get a deeper comprehension of the various aspects that influence Quality of Service (QoS) and their potential interactions with network configuration. Additionally, it may contribute to the advancement of optimal network management and optimization strategies, particularly in intricate and high-capacity network environments.

Conclusion

First and foremost, the purpose of this investigation is to identify the most efficient approach to enhancing the effectiveness of script programming in the process of configuring network devices. In addition, the purpose of the research is to ascertain the amount of time that is necessary for the setup of network devices by employing three different automation methods: the manual method, the serial automation method, and the multi-thread automation method. In order to design the topology of the network, a total of eleven Cisco router devices were utilised in combination with the GNS3 network simulator. For the purpose of facilitating the actual application of automation, the topology was constructed in an appropriate manner. The particular objective was to reduce the amount of time required for Cisco devices to be configured and to reduce the number of errors that occurred. As a result of the findings of the study, it is clear that the use of the multi-thread strategy for the purpose of automating the configuration of network devices results in a considerable improvement in the effectiveness of the script, particularly with regard to the speed, when compared to the manual method. As a result of this performance, it is clear that automation is an effective method for configuring network equipment. Automation has the ability to cut the amount of time needed for setup by as much as 83 percent. When it comes to the results of speed calculations, the parallel automation strategy demonstrates a sixfold gain in efficiency in comparison to the manual method, and a one-and-a-half-fold rise in comparison to the serial automation method. Furthermore, it has been discovered that there is no statistically significant difference in the outcomes of testing packet loss, delay, and throughput with manual configuration, serial automation, and multi-thread automation. This is the conclusion that can be drawn from the findings. Incorporating network automation techniques for the aim of widening the scope of network setup with the intention of enhancing future research endeavors is a recommendation that should be taken into consideration.

BIBLIOGRAPHY

Academy, C. N. (2014). *Connecting networks companion guide*. WebEx Communications.

Aguado, A., Lopez, V., Martinez-Mateo, J., Peev, M., Lopez, D., & Martin, V. (2018). Virtual network function deployment and service automation to provide end-to-end quantum encryption. *Journal of Optical Communications and Networking*, *10*(4), 421–430.

Arifin, M. A. Z., Kassim, M., & Suliman, S. I. (2019). Automation security system with laser lights alarm on web pages and mobile apps. *2019 IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 287–292.

Ayasso, H., & Mohammad-Djafari, A. (2010). Joint NDT image restoration and segmentation using Gauss–Markov–Potts prior models and variational Bayesian computation. *IEEE Transactions on Image Processing*, *19*(9), 2265–2277.

Bouhouras, A. S., Andreou, G. T., Labridis, D. P., & Bakirtzis, A. G. (2010). Selective automation upgrade in distribution networks towards a smarter grid. *IEEE Transactions on Smart Grid*, *1*(3), 278–285.

Kopeć, J. (2022). Evaluating Methods of Transferring Large Datasets. *Asian Conference on Supercomputing Frontiers*, 102–120.

Larsson, J. (2020). *Network Automation in a Multi-vendor Environment (Dissertation)*. https://urn.kb.se/resolve?urn=urn:nbn:se:ltu:diva-80810

Mauboy, L. G., & Wellem, T. (2022). Studi Perbandingan Library Untuk Implementasi Network Automation Menggunakan Paramiko Dan Netmiko Pada Router Mikrotik. *JURIKOM (Jurnal Riset Komputer)*, *9*(4), 790–799.

Maurice, C., Weber, M., Schwarz, M., Giner, L., Gruss, D., Boano, C. A., Mangard, S., & Römer, K. (2017). Hello from the Other Side: SSH over Robust Cache Covert Channels in the Cloud. *NDSS*, *17*, 8–11.

Mazin, A. M., Ab Rahman, R., & Kassim, M. (2021). Performance analysis on network automation interaction with network devices using python. *2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 360–366.

Mihăilă, P., Bălan, T., Curpen, R., & Sandu, F. (2017). Network automation and abstraction using Python programming methods. *MACRo 2015*, *2*(1), 95–103.

Nugroho, S., & Pujiarto, B. (2022). Network Automation Pada Beberapa Perangkat Router Menggunakan Pemrograman Python. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, *9*(1), 79–86.

Oleiwi, S. S., Mohammed, G. N., & Al_barazanchi, I. (2022). Mitigation of packet loss with end-to-end delay in wireless body area network applications. *International Journal of Electrical and Computer Engineering*, *12*(1), 460.

Smolaks, M. (2016). *Telstra blames major network outage on human error*. https://www.datacenterdynamics.com/en/news/telstra-blames-major-network-outage-on-human-error/

Taruk, M., Budiman, E., Rustam, M. R., Azis, H., & Setyadi, H. J. (2018). Quality of service voice over internet protocol in mobile instant messaging. *2018 2nd East Indonesia Conference on Computer and Information Technology (EIConCIT)*, 285–288.

Tolly, K. (2021). *How to use iPerf3 to test network bandwidth*. Networking.

Veriflow. (2016). *Global Survey Reveals Complexity, Change and Human Factors Are Key Causes of Today's Network Outages and Vulnerabilities*. GlobeNewswire News Room. https://www.globenewswire.com/news-release/2016/11/15/1195128/0/en/Global-Survey-Reveals-Complexity-Change-and-Human-Factors-Are-Key-Causes-of-Today-s-Network-Outages-and-Vulnerabilities.html

Waheed, F., & Ali, M. (2018). Hardening CISCO Devices based on Cryptography and Security Protocols-Part II: Implementation and Evaluation. *Annals of Emerging Technologies in Computing (AETiC), Print ISSN*, 281–2516.

Web Pages and Mobile Apps. (2019). *2019 IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*. https://doi.org/10.1109/iscaie.2019.8743998

Ylonen, T., Turner, P., Scarfone, K., & Souppaya, M. (2015). Security of interactive and automated access management using Secure Shell (SSH). *NISTIR 7966, National Institute of Standards and Technology*.