

Implementation of OSPF and Static Routing Using Python Scripts in Cisco Packet Tracer Network Simulation

Ananda Syafika*
State University of Medan
Medan, Indonesia
anandasafika402@gmail.com

Dedy Kiswanto
State University of Medan
Medan, Indonesia
dedykiswanto@unimed.ac.id

Benedict Sandi Pangestu Rosa
State University of Medan
Medan, Indonesia
benedictsandi28@gmail.com

Nayla Luthfiah Hanan
State University of Medan
Medan, Indonesia
naylaluthfiahh@gmail.com

Abstract— The increasing complexity of computer networks demands a faster and more accurate process of configuring network devices. Manually configuring routers is time-consuming and prone to configuration inconsistencies, requiring a more efficient approach. This study aims to implement static routing and dynamic routing configurations based on the OSPF (Open Shortest Path First) protocol using the Python programming language in a simulation of the Cisco Packet Tracer network. The network topology consists of three routers, three switches, and three PCs with predefined IP addressing. Python scripts are designed to generate router interface configurations, static routing, and OSPF based on topological data, as well as store the results into .txt files. The test is performed using ping commands, show ip route, and show ip ospf neighbor to verify inter-network connectivity. The test results showed that both routing methods successfully connected three different subnets with 0% packet loss under stable network conditions. Python-based scripts have been shown to speed up the configuration preparation process as well as produce more consistent and structured configurations compared to manual methods.

Keywords— *Static routing; Dynamic Routing; OSPF; Python*



© 2026 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution ShareAlike (CC BY SA) license (<https://creativecommons.org/licenses/by-sa/4.0/>).

I. INTRODUCTION

The development of modern network infrastructure demands solutions Routing that are able to adapt to increasingly complex network dynamics [1]. Routing is a fundamental mechanism in a computer network that functions to transmit data packets from one network to another through the router device in the third layer of the OSI model [2]. No configuration Routing To be precise, communication between different networks cannot take place efficiently.

In practice, there are two main methods in Routing, namely static routing and dynamic routing. Static routing is a method that is manually configured by the network administrator, where each route is explicitly defined so that the administrator has full control over the data transmission path [3]. This method is suitable for simple network

topologies that undergo little change, are small-scale, and do not require additional computing load on the router [4]. Instead, Dynamic Routing allows the router to automatically update route information based on ongoing network conditions without manual intervention [5]. One of the protocols Dynamic Routing widely used is OSPF (Open Shortest Path First), which works using the Dijkstra algorithm to determine the shortest path in data transmission and is able to converge quickly when topological changes occur [6].

On the other hand, manually configuring a router takes a considerable amount of time and has the potential to cause misconfigurations (Human Error), such as an IP Address assignment error, Routing, as well as basic security configurations that can cause data traffic disruptions on the network [7]. This condition is increasingly critical as the number of network devices that must be managed increases. Therefore, a script-based mechanism is needed for network configuration creation to make the administration process faster, more efficient, and more structured. Python is one of the relevant approaches because it has a simple and flexible syntax, and it supports the development of Cisco network device configurations [3].

Several studies have confirmed the effectiveness of this approach. Prasetyo et al. (2024) demonstrated that combining Ansible and Python for Cisco ACI configuration in a production network environment can reduce configuration time from 50 minutes to approximately 6 minutes for 100 configuration rules, achieving an improvement of 833.33% with 100% configuration accuracy [8]. Elezi and Karras (2023) further showed that Netmiko-based Python scripting enables automated SSH-based configuration of multi-vendor network devices, significantly reducing repetitive manual work and configuration errors across single and multiple device environments [9].

The growing scale of modern network infrastructure further highlights this challenge. As the number of managed devices increases, the time required for manual configuration grows proportionally, while the risk of inconsistency between device configurations also increases. A configuration error on one router can disrupt connectivity across multiple subnets, requiring administrators to trace and correct each affected device manually. This condition becomes increasingly unmanageable in enterprise environments where dozens or

even hundreds of routers must be configured within tight timeframes.

In the context of network education and simulation, this challenge is equally relevant. Students and practitioners learning network administration often spend significant time writing repetitive configuration commands manually, which detracts from focus on understanding the underlying routing concepts. A script-based approach to configuration generation can streamline this process, allowing practitioners to focus on topology design and verification rather than manual command entry. This study therefore proposes the use of Python scripting as a structured and reproducible method for generating routing configurations, validated through simulation in Cisco Packet Tracer as a proof of concept before deployment in real network environments. The novelty of this research lies in the simultaneous application of Python-based automated configuration generation for both static routing and OSPF dynamic routing within a simulation environment, a context that has not been addressed by prior automation studies focused on production networks [8], [9]. Unlike those works, this study specifically adapts the automation paradigm to an educational simulation setting, producing structured Cisco IOS configuration files exportable as .txt that can be applied without any manual CLI input.

Based on these problems, this study aims to apply static routing and OSPF-based dynamic routing configurations using the Python programming language to the Cisco Packet Tracer network simulation [10], [11]. The implementation is done by creating a Python script that generates a configuration of the Cisco IOS router based on the topology data and addressing plan and then saving it into a configuration file formatted .txt. This approach is expected to speed up the configuration process, reduce the risk of manual errors, and provide a comparison of route management efficiency between the two routing methods based on network needs and scale.

II. LITERATURE REVIEW

A. Static Routing

Routing It is a mechanism on a computer network that functions to send data packets from one network to another through a router device. In the process Routing, router requires a table Routing which contains information about the destination network and the path that must be taken to reach it [7]. Sugiyatno (2024) defines Static Routing as a method Routing which is done by configuring the table Routing manually by the network administrator [13]. In contrast to Dynamic Routing that are able to update routing tables automatically, Static Routing requires administrators to explicitly define each route on each router. In the event of a change in the network topology, the administrator must update the configuration Routing manually on each affected router [3].

According to Jorepalli (2024), Static Routing It is generally used on simple networks that rarely undergo topological changes. This method is protocol-dependent Routing dynamic so that it does not require information exchange between routers, which means that the network load is lighter compared to the use of the protocol Dynamic Routing such as OSPF or RIP. Static routing It has several advantages, including simple configuration, no additional routing protocols, minimal use of CPU and memory

resources, and data delivery paths that can be predicted with certainty by administrators. In addition, Static Routing more secure because routing information is not spread across the network [3]. On the other hand, manual configuration is required on each Router becomes inefficient as the network evolves, as any topology changes must be updated manually thereby increasing the risk of misconfigurations (Human Error) which may result in communication interruptions between networks [4].

On Cisco devices, Static Routing configured using the IP Route command followed by the destination network address, Subnet Mask, and address Snoop Dogg or Interface output, in the following format [10]:

- Router(config)# ip route [network] [subnet-mask] [next-hop/exit-interface]
- The command defines a route to a specific destination network through an address Snoop Dogg which is the IP address of the next router to be passed through [10]

B. Dynamic Routing and OSPF

Dynamic routing is a routing method that is able to choose the best path and create and update tables Routing automatically based on network conditions and topological changes that occur [2], [5]. In contrast to manually configured static routing, dynamic routing allows the entire router to automatically obtain network change information without administrator intervention [5], [6]. In the process, routers require information about the directly connected network as well as the remote network, which is obtained through the protocol Routing by exchanging information between routers and storing it in a table Routing [6]. Dynamic routing is divided into two main categories, namely Interior Gateway Protocol (IGP) and Exterior Gateway Protocol (EGP). IGP is used for routing in one Autonomous System (AS), with examples of protocols such as RIP, OSPF, and EIGRP, while EGP is used for routing between Autonomous System with an example of the BGP protocol [2].

Static routing is a manually configured routing method and is commonly used on simple networks that rarely undergo topological changes [2]. In the event of a network change, the configuration Routing must be manually updated on each affected router [6]. In contrast, dynamic routing can automatically adjust the data transmission path so that it is more efficient and reduces the possibility of errors in table filling Routing [5]. In addition, OSPF as one of the dynamic routing protocols is able to determine the shortest path using the Dijkstra algorithm so as to increase the speed and efficiency of network communication [6].

Sugiyatno (2024) states that Open Shortest Path First (OSPF) is a dynamic routing protocol that belongs to the category of IGP and uses algorithms Link-state [12]. OSPF works by building a network topology map through information exchange Link State Advertisement (LSA) between routers, then determine the shortest path using the Dijkstra algorithm [6]. As an open standard protocol, OSPF can be used on devices from various network vendors. According to Maulana (2024), OSPF is widely applied to small to large-scale networks because it is able to manage data routes quickly and efficiently, and is designed hierarchically by dividing large networks into several areas to make management easier Routing [6].

OSPF works through several main stages. First, the

router sends packets Hello. periodically to find and build relationships Adjacency with a neighboring router. Once the relationship is formed, routers exchange topological information in the form of LSA which contains information about the links and neighbors of each router [6]. All LSA information is stored in Link State Database (LSDB), and based on that database each router runs the Dijkstra algorithm to calculate the shortest path and enter it into the routing table. In the event of a topology change, the router deploys the latest LSA so that the entire router can recalculate its best path through the convergence process [6]. According to Shah & Nevrisa (2024), OSPF uses the principle of Link-state with Area 0 as Backbone that connects the entire area, and each network segment must be registered in the OSPF configuration for routers to recognize each other and exchange routing information [13]. This has been proven through the implementation of simulations Cisco Packet Tracer [14] and Mikrotik CHR-based virtual environments [1].

OSPF has rapid convergence, use Bandwidth efficient, as well as the ability to manage small to large-scale networks through the concept of area [13]. Mechanism Updated that OSPF uses allows routing information updates to be performed only when there is a topological change so that the use of Bandwidth Become more efficient [6]. OSPF has also proven to be able to maintain packet loss 0% on most tests in topology Ring [1], and enable the division of the network into segments that remain interconnected to support better network management and security [2], [13].

C. Creating Network Configurations with Python

Script-based network configuration creation is the process of generating configuration, managing, and operating network devices using specific scripts or software [4]. This approach aims to improve network configuration efficiency, speed up the network device administration process, and reduce the likelihood of manual misconfiguration. Prasetyo et al. (2024) state that script-based configuration creation allows the management of multiple devices simultaneously using the Python programming language, so that network administrators can configure tens or even hundreds of devices more easily and quickly compared to manual methods that are prone to human error [15]. Beyond SSH-based device management, similar automation principles have also been applied using the Netmiko library for direct SSH-based configuration push [16] and using Ansible combined with Python for template-driven configuration of SDN controllers such as Cisco ACI [17], both demonstrating that Python-based scripting can substantially reduce configuration time compared to manual methods. In its implementation, the script-based approach is widely applied using the Python programming language because it has high flexibility and supports various modern network administration needs [10].

Python is a high-level programming language that is widely used in the development of network configurations due to its simple and easy-to-understand syntax [16]. Andjarwirawan (2023) explained that development using Python can be found in many systems such as web applications, mobile, machine learning, to IoT equipment, and in the context of computer networks, Python can help network administrators from monitoring to configuration of several devices at once, including in the calculation of IP

address subnetting [17]. In the field of computer networking, Python can be used to generate router configurations, monitor network devices, and back up device configurations. In this study, Python was used to generate static and OSPF routing configurations based on predefined network topology data [10].

III. RESEARCH METHODS

The research stage is carried out systematically to produce script-based routing configurations that can help the network administration process become more efficient. The research flow can be seen in the research methodology flow chart in Fig. 1.

Flowchart Metodologi Penelitian



Fig. 1. Research methodology flowchart

The stage of research carried out are as follows:

1. Network Topology Design: The initial stage is done by designing a network topology using Cisco Packet Tracer which consists of 3 routers, 3 switches, and 3 PCs.
2. Routing Plan Drafting: At this stage, the IP address is assigned to each router and PC device interface so that all devices can connect to each other.
3. Routing Configuration: Routing configuration is done using two methods, namely static routing and dynamic routing using the OSPF protocol.
4. Python Script Design: Python scripts are designed to generate Cisco IOS router configurations based on predefined topology and routing data.
5. Configuration Creation: Python scripts process router, interface, IP address, and routing data to generate Cisco IOS configurations.
6. Export Configuration to TXT File: The configuration results generated by Python are saved into the .txt file as documentation and reference for router configuration.
7. Implementation on Cisco Packet Tracer: The resulting configuration is then applied to the Cisco router in Cisco Packet Tracer.
8. Network Testing: The test is performed using ping commands, show ip route, and show ip ospf neighbor to ensure connectivity between networks is running smoothly.

A. Network Topology

The network topology used in the implementation of static routing consists of three routers (Router0, Router1, Router2) that are serially connected to each other to form a linear path. Each router is connected to one switch and one PC on its local network. Router1 acts as a middle router that connects Router0 and Router2. The implementation was carried out using the Cisco Packet Tracer simulator. This research is limited to a simple topology simulation as a proof-of-concept stage. This approach was chosen to validate the script framework before being applied to more complex and larger-scale network topologies. Here is the IP address configuration used on each device in the topology:

TABLE 1. IP CONFIGURATION ON PC AND ROUTER

Devices	Interface	IP Address	Subnet Mask
PC 0	Fa0	192.168.1.2	255.255.255.0
PC 1	Fa0	192.168.2.2	255.255.255.0
PC 2	Fa0	192.168.3.2	255.255.255.0
Router 0	Gig0/0	192.168.1.1	255.255.255.0
Router 0	Gig0/1	10.0.1.1	255.255.255.252
Router 1	Gig0/0	192.168.2.1	255.255.255.0
Router 1	Gig0/1	10.0.1.2	255.255.255.252
Router 1	Gig0/2	10.0.12.1	255.255.255.252
Router 2	Gig0/0	192.168.3.1	255.255.255.0
Router 2	Gig0/1	10.0.12.2	255.255.255.252

B. Static Routing Configuration

Fig. 2 displays the configuration process on Router0 via the Cisco Packet Tracer CLI, including assigning IP addresses to each interface, defining static routes using the ip route* command, and verifying the routing table using the show ip route command* which displays a route entry marked with the letter S as confirmation of the success of the static routing configuration.

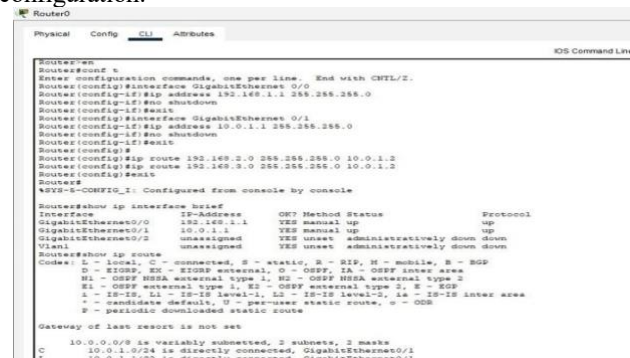


Fig. 2. Configuration of each router.

C. Ping Results Between PCs

Fig. 3 shows the equations are an exception to the prescribed shows the results of connectivity tests using ping commands from PC0 to PC2 (192.168.3.2), where the entire packet was successfully received with a packet loss of 0%.

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edi
	Successful	PC0	PC1	ICMP		0.000	N	3	(ec
	Successful	PC1	PC2	ICMP		0.000	N	4	(ec
	Successful	PC0	PC2	ICMP		0.000	N	5	(ec

Fig. 3. Ping test

D. Dynamic Routing

OSPF Implementation:

1) OSPF Configuration on the Router

Router0 is configured using OSPF with Process ID 1 and Area 0 which can be seen in Fig. 4. Local network 192.168.1.0/24 as well as inter-router network 10.0.1.0/30 are added to the OSPF process via the following command:

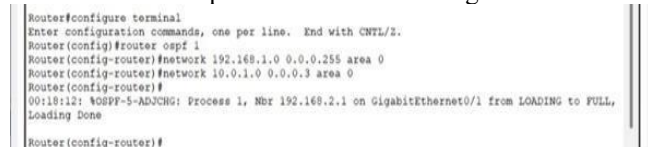


Fig. 4. OSPF configuration on router 0

2) Process ID explained

Process ID is used to identify OSPF processes running on a router. This value is local and does not need to be the same between routers, but in this study all routers used Process ID 1 to simplify configuration and facilitate network management.

3) Wildcard Mask

Wildcard masks are used in OSPF configurations to specify the range of network addresses to be advertised. Wildcard masks are the opposite of subnet masks, where a value of 0 means bits must match and a value of 255 means bits are ignored. The following wildcard masks were used in this study can be seen in TABLE 2. LAN networks use the wildcard mask 0.0.0.255 because it includes 256 host addresses, while inter-router point-to-point networks use 0.0.0.3 because it only requires 4 addresses.

TABLE 2. WILDCARD MASK USED

Subnet Mask	Wildcard Mask	Remarks
255.255.255.0	0.0.0.255	Used for /24 (LAN) networks
255.255.255.252	0.0.0.3	Used for /30 networks (inter-router)

4) Area OSPF

The entire router is placed in Area 0 (backbone area). The use of this single area was chosen to simplify the implementation of OSPF given the small scale of the network, while ensuring that all routers can exchange routing information directly with each other without the need for an Area Border Router (ABR).

5) Routing Testing

Based on the results in Fig. 5, the show ip ospf neighbor command on Router0, there is one neighbor with Neighbor ID 192.168.2.1 in FULL/BDR status through the GigabitEthernet0/1 interface. The FULL status on all neighbors indicates that the adjacency relationship between

the routers is perfectly established.



Fig. 5. Show ip ospf neighbor router 0

The routing table on Router0 shows the routes learned through OSPF, namely the 10.0.12.0/30, 192.168.2.0/24, and 192.168.3.0/24 networks, which are marked with the letter O. The letter O in the routing table which can be seen in Fig. 6 indicates that the routes are learned dynamically through the OSPF protocol, rather than being configured manually.

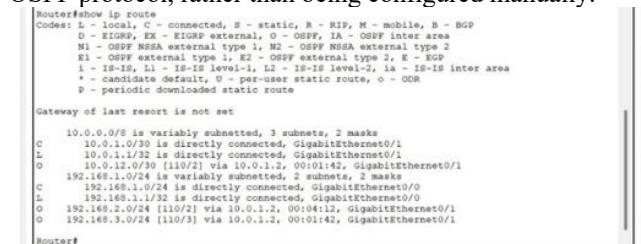


Fig. 6. Show ip route on router 0

Connectivity testing was conducted from PC0 to PC2 (192.168.3.2). In the first attempt of PC0, the first packet experienced a timed-out request that is common during the ARP process, but the next three packets were successfully accepted. This result which can be seen in Fig. 7 and Fig. 8 is also confirmed by the PDU List which shows the status of Successful for all internet delivery scenarios.

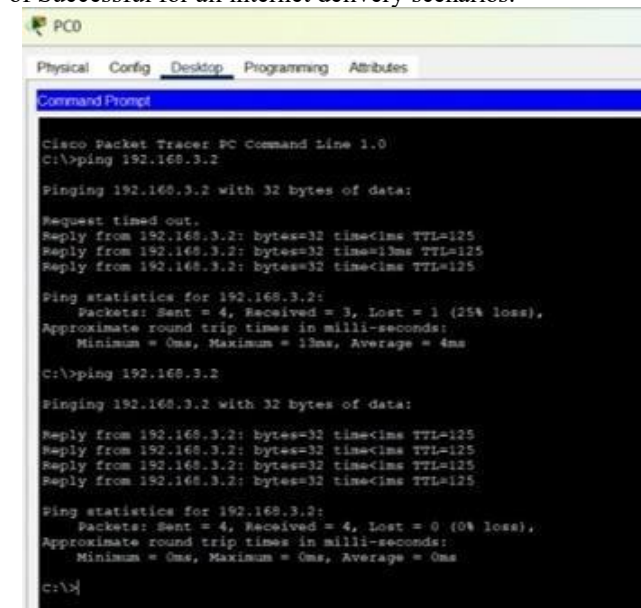


Fig. 7. Ping test PC 0

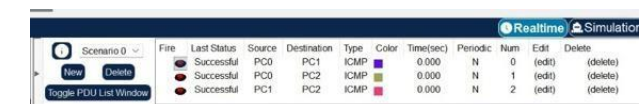


Fig. 8. Inter-PC ping test

6) Python Script Planning

Network configuration in this study was made using the Python programming language. Python scripts are designed to generate the following configurations:

- Router interface,
- static routing,
- OSPF configuration,

- as well as export the configuration to .txt file.

Router configuration data is stored using a dictionary data structure that contains hostname, interface, IP address, subnet mask, and routing configuration information.

Here's an example of the router's data structure used in a Python script:

```
routers = {
    "Router0"
: {
    "interfaces": [
        ("gig0/0", "192.168.1.1", "255.255.255.0"),
        ("gig0/1", "10.0.1.1", "255.255.255.252")
    ],
    "routes": [
        ("192.168.2.0", "255.255.255.0", "10.0.1.2"),
        ("192.168.3.0", "255.255.255.0", "10.0.1.2")
    ]
}
```

Python scripts use looping to automatically generate Cisco IOS configurations on each router.

7) Python-Based Static Routing Configuration Implementation

In static routing implementations, the Python script generates routing configurations based on the destination network, subnet mask, and next-hop gateway predefined in the topology design. The configuration results are then stored in a .txt file and used as a router configuration reference on Cisco Packet Tracer.

8) Python-Based OSPF Configuration Implementation

In dynamic routing implementations, Python scripts are used to automatically generate OSPF configurations based on process IDs, network addresses, wildcard masks, and OSPF areas. The OSPF configuration is generated using network data that has been adjusted to the network topology used in the research.

9) Testing Techniques

Testing is performed to ensure that the routing configuration generated by Python can run properly on Cisco Packet Tracer network simulations. The testing technique is carried out using:

- ping commands to test connectivity between networks,
- show IP route to view the routing table,
- as well as show ip ospf neighbor to ensure that OSPF neighbor relationships are successfully formed.

IV. RESULTS AND DISCUSSION

A. Static Routing Analysis

Based on the test results, the implementation of static routing on a three-router network went well. All routers successfully find routes to networks that are not directly connected through manually configured static route entries. Ping testing between subnets shows a 100% communication success rate after the first ARP process is completed. This implementation also shows the main characteristics of static

routing, which is a fixed route that does not change automatically according to network conditions. This is both an advantage and a limitation, because even if the data transmission is predictable, the topology change will require a manual configuration update on each affected router.

B. Dynamic Routing Analytics

Based on the test results, the implementation of OSPF on a three-router network went well. The show ip ospf neighbor command proves that the entire router successfully forms an adjacency relationship with the state of FULL, which means that the network topology information exchange process is completely complete. The routing table obtained via show ip routes shows that each router has learned the other network automatically without a static route configuration. Ping testing between subnets shows a communication success rate of up to 100% in stable experiments, with packet loss occurring only on the first packet due to the ARP resolution process. Overall, OSPF successfully connects three different subnets—192.168.1.0/24, 192.168.2.0/24, and 192.168.3.0/24—dynamically and efficiently.

C. Implementation of Python-Based Configuration Generation

In this study, python was used to assist the preparation of Cisco router configurations based on the designed network topology. The developed Python scripts are capable of generating:

- Router interface configuration,
- static routing configuration,
- OSPF configuration,
- as well as export the configuration to .txt file.

The generated configuration is based on a pre-designed network topology using Cisco Packet Tracer which can be seen in Fig. 9. The network topology consists of:

- 3 Cisco routers,
- 3 switch,
- 3 pcs,
- connection between routers using a point-to-point network, as well as the implementation of static routing and OSPF.

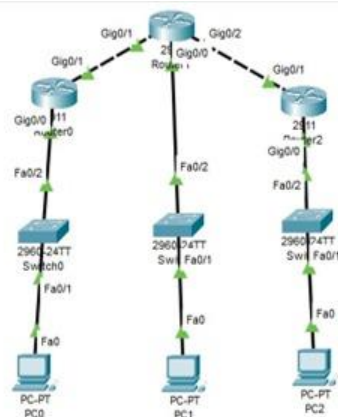


Fig. 9. Cisco packet tracer network

D. Python Static Routing Script Implementation

Python static routing scripts are created to automatically generate routing configurations based on predefined router data. Configurations are generated without the need to perform manual inputs on the router one by one.

Router data is stored using a data dictionary structure that contains interface information, IP addresses, subnet masks, and routing destinations.

Here's an example of a router's data structure in a Python script:

```

routers = {
    "Router0": {
        "interfaces": [
            ("gig0/0", "192.168.1.1", "255.255.255.0"),
            ("gig0/1", "10.0.1.1", "255.255.255.252")
        ],
        "routes": [
            ("192.168.2.0", "255.255.255.0", "10.0.1.2"),
            ("192.168.3.0", "255.255.255.0", "10.0.1.2")
        ]
    }
}

```

The Python script then generates the Cisco IOS configuration automatically and saves it into a .txt file which can be seen in Fig. 10.

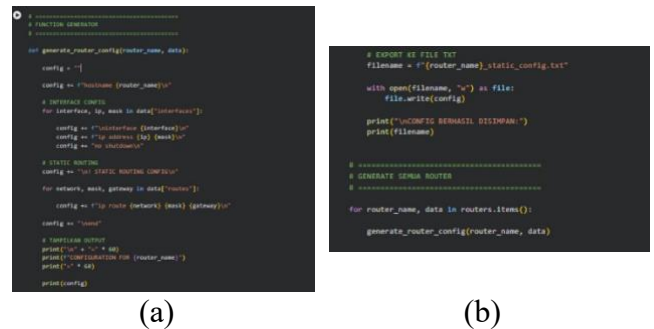


Fig. 10. Script python static routing generator. (a) Generate router config function; (b) Export configuration and generate all routers

Fig. 11 is the examples of the resulting configuration results are as follows:

hostname router0

Interface gig0/0
IP address 192.168.1.1 255.255.255.0
and shutdown

Interface gig0/1
IP address 10.0.1.1 255.255.255.252
and shutdown

IP route 192.168.2.0 255.255.255.0 10.0.1.2
IP Route 192.168.3.0 255.255.255.0 10.0.1.2

end

E. OSPF Python Script Implementation

In addition to static routing, Python scripts are created to generate OSPF configurations based on predefined process IDs, network addresses, wildcard masks, and OSPF areas as can be seen in Fig. 12. OSPF data is stored in the dictionary and processed using looping to generate configurations automatically. Here's an example of the OSPF configuration generated by the Python script:

OSPF Router 1
network 192.168.1.0 0.0.0.255 area 0

network 10.0.1.0 0.0.0.3 area 0.

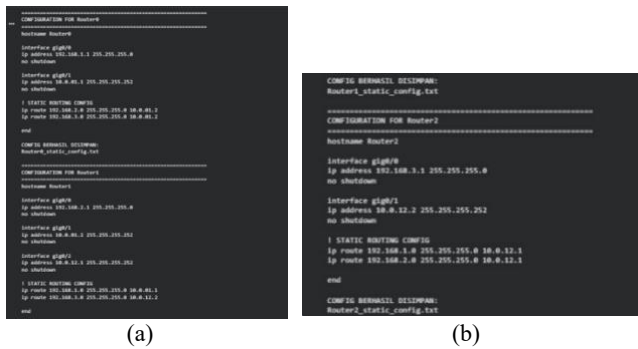


Fig. 11. Output python static routing. (a) Configuration output of router0 and router1; (b) Configuration output of router2



Fig. 12. Script python OSPF generator. (a) Router data structure; (b) Generate OSPF config function; (c) Export configuration and generate all routers

The configuration results are then automatically saved in a .txt file according to the name of the router used.

F. Export Configuration to TXT File

Fig. 13 shows the addition to generating router configurations automatically, Python scripts are also capable of saving the configuration results into .txt files. The configuration file is used as a documentation as well as a reference for the configuration of the router on Cisco Packet Tracer. Fig. 14 shows the configuration export process is done using the open() function in Python so that each router has its own configuration file.

G. Analysis of Python-Based Configuration Generation

Based on the implementation results, the use of Python has successfully assisted the network configuration preparation process to become more structured and efficient. Compared to manual configuration, the Python script provides several advantages, including accelerating configuration preparation, producing more consistent

configuration formats across multiple routers, and reducing the possibility of typing errors during configuration writing. In manual configuration, network administrators must enter configuration commands individually on each router. By using Python, configuration commands can be generated automatically from predefined topology and addressing data, then stored in a .txt file for documentation and implementation purposes. This approach also simplifies configuration management because all generated configurations are stored in a structured format and can be reused when network topology modifications are required. The implementation in this study is limited to generating Cisco router configuration files based on static routing and OSPF parameters within a Cisco Packet Tracer simulation environment. The generated configuration files demonstrate that Python can be utilized to support network configuration preparation and improve the consistency of the configuration development process.



Fig. 13. TXT file results of configuration static routing. (a) List of configuration files; (b) Router0 config file

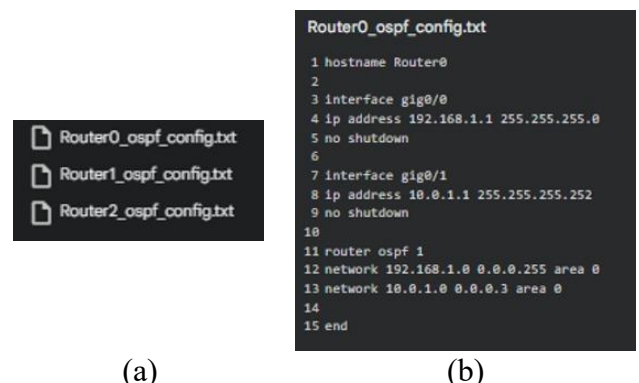


Fig. 14. Ospf configuration results txt file. (a) List of configuration files; (b) Router0 OSPF config file.

V. CONCLUSION

Based on the results of the implementation and tests that have been conducted, it can be concluded that static routing and dynamic routing configurations using OSPF were successfully implemented on the three-router network topology in Cisco Packet Tracer. The routing methods successfully connected three different subnets, namely 192.168.1.0/24, 192.168.2.0/24, and 192.168.3.0/24, with a 100% communication success rate and 0% packet loss under stable network conditions. In static routing, all routes are successfully recorded in a routing table with the letter S marker and connectivity between subnets can be established through a manually defined next-hop configuration. In dynamic routing, the show ip ospf neighbor command proves that the entire router successfully forms an adjacency

relationship with a FULL status, so that the exchange of topology information takes place automatically without administrator intervention. The use of Python in this study successfully generates router interface configurations, static routing configurations, and OSPF configurations based on predefined topology data. Compared to manual configuration, this approach reduces configuration time, minimizes the risk of configuration inconsistencies, and results in more consistent and well-documented configuration outputs. In addition, all generated configurations can be stored in .txt files, making them easier to manage, review, and reuse when network topology modifications are required. Although this study still focuses on configuration generation and Cisco Packet Tracer simulation, the results demonstrate that Python can support a more structured and efficient network configuration process. Future work may involve direct deployment to network devices using SSH-based communication and libraries such as Netmiko within environments such as GNS3 or real Cisco routers.

REFERENCES

- [1] F. G. Sembiring, S. Ashillah, A. K. Nasution, and D. Kiswanto, "Dynamic Routing Performance Analysis in Virtual Networks Using Mikrotik Chr," *J. Inform. and Tek. Electro Therapy.*, vol. 13, no. 3, 2025, doi: 10.23960/jitet.v13i3.6516.
- [2] R. Syah, Amri, and A. Nevrisa, "Comparative Analysis of the Performance of Static Routing Protocol and Dynamic Routing Protocol on Redistribution Networks," *JTIK (ICT Journal)*, vol. 4, no. 2, pp. 221–230, 2024.
- [3] S. Nugroho, B. Pujiarto, U. M. Magelang, and P. Correspondence, "Network Automation in Some Router Devices," *J. Techno. Inf. and Computing Science.*, vol. 9, no. 1, pp. 79–86, 2022, doi: 10.25126/jtiik.202293947.
- [4] S. Jorepalli, "Leveraging Network Automation with Python, Terraform, and Ansible to Enhance Security and Operational Efficiency in Large-Scale Networks," *Int. J. Intell. Syst. Appl. Eng.*, vol. 12, no. 23s, pp. 2009–2016, 2024, [Online]. Available: <https://ijisae.org/index.php/IJISAE/article/view/7222>
- [5] W. Agnestisya, "Comparative Analysis of Static and Dynamic Routing Performance with Techniques," *Digit. Transform. Technol.*, vol. 2, no. 2, pp. 31–34, 2022.
- [6] G. Purnama, "Performance Study of OSPF Dynamic Routing Protocol Using Opnet Simulator," *J. Inform. and Tek. Electro Therapy.*, vol. 11, no. 3s1, pp. 990–996, 2023, doi: 10.23960/jitet.v11i3s1.3431.
- [7] R. I. Napitupulu, J. Gunawan, F. Gianadevi, and T. Rumambi, "Network Automation Applications on Cisco Routers Using Python," *J. Echo Ekon.*, vol. 11, no. 6, p. 2022, 2022.
- [8] F. Osei-Wusu, W. Asiedu, K. A. Frimpong, D. Yeboah, E. A. Sarfo, and S. A. Muntaka, "Automating Network Programmability and Backup on Cisco Devices Using Python and Netmiko Library: A Case Study of Komfo Anokye Teaching Hospital LAN," *J. Comput. Res. Innov.*, vol. 10, no. 1, pp. 227–242, 2025.
- [9] N. E. Pasaribu et al., "Designing a Network by Implementing the Cisco Packet Tracer-Based Open Shortest Path First (OSPF) Dynamic Routing Protocol," *J. Authentic Res.*, vol. 4, no. 3341, p. 826, 2025, [Online]. Available: <https://journal-center.litpam.com/index.php/jar/index>
- [10] Sugiyatno, "Teaching Materials for Computer Network Routing Techniques," in *Bhayangkara University of Greater Jakarta*, 2024, pp. 7–43.
- [11] A. Maulana, "Application of OSPF Routing Protocol on Mikrotik Routers and Performance Analysis Using Traffic Generator Tool (Case Study: Munzir Kitchen Jakarta)," *CONTAIN COMPUT. Netw. Technol.*, vol. 4, no. 2, pp. 135–146, 2024, [Online]. Available: <http://jurnal.bsi.ac.id/index.php/conten>
- [12] A. Hardiansyah, M. Hilman, and H. Holilah, "Comparative Analysis of the Implementation of Open Shortest Path First (OSPF) and Border Gateway Protocol (BGP) Dynamic Routing Protocols in Computer Labs Using Cisco Packet Tracer," *J. Ilm. Science and Technology.*, vol. 9, no. 1, pp. 129–142, 2025, doi: 10.47080/saintek.v9i1.3904.
- [13] F. D. J. Prasetyo, H. Sutarjo, D. Triantoro, and D. V. S. Y. Sakti, "Backup Automation of Cisco Computer Network Device Configuration," *J. MIND J. | ISSN*, vol. 9, no. 1, pp. 99–112, 2024, [Online]. Available: <https://ejournal.itenas.ac.id/index.php/mindjournal/article/view/11204/3665>
- [14] J. Chen, W. Xiao, H. Zhang, J. Zuo, and X. Li, "Dynamic routing optimization in software-defined networking based on a metaheuristic algorithm," *J. Cloud Comput.*, vol. 13, no. 1, pp. 1–18, 2024, doi: 10.1186/s13677-024-00603-1.
- [15] J. Andjarwirawan, "Computer Network Programming with Python," in *Petra Christian University Research and Community Service Institute*, 2023, pp. 1–60.