

BAB 3

METODOLOGI PENELITIAN

Metodologi Penelitian berisi uraian diagram alur penelitian. Diagram alur penelitian menjelaskan mengenai tahap-tahap penelitian. Dalam penelitian ini diperlukan pula alat pendukung untuk menunjang penelitian. Penelitian ini juga membutuhkan topologi yang berfungsi sebagai objek pengambilan data pada proses penelitian.

3.1 ALAT YANG DIGUNAKAN

3.1.1 Perangkat Keras

Perangkat keras yang akan digunakan pada penelitian ini menggunakan satu buah laptop dengan spesifikasi sebagaimana terdapat pada tabel 3.1

Tabel 3.1 Spesifikasi Perangkat Keras

OS	Windows 11
<i>Processor</i>	Intel i7-12700
<i>System Memori (RAM)</i>	16 GB
<i>Storage (HDD + SSD)</i>	1TB + 256 GB

3.1.2 Perangkat Lunak

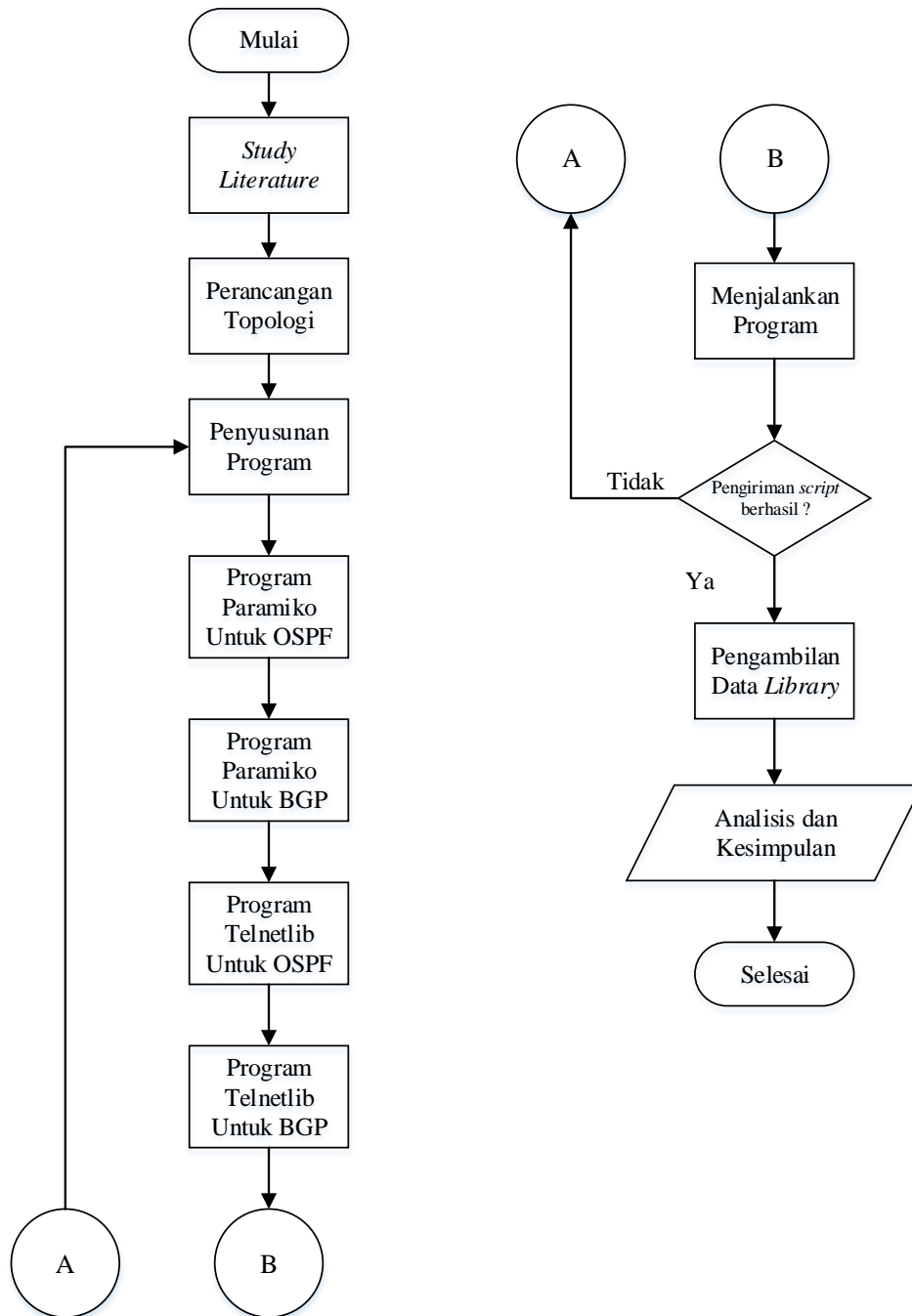
Perangkat lunak sebagai *tool* dan aplikasi yang digunakan pada penelitian ini dapat dilihat pada tabel 3.2

Tabel 3.2 Perangkat Lunak

No	<i>Software</i>	Versi	Fungsi
1	GNS3	2.2.34	Penyusunan Topologi dan Pengujian
2	VMWare Workstation	17.0.0	GNS3 VM
3	Wireshark	3.6.7	Pengambilan Data

3.2 ALUR PENELITIAN

Penelitian ini ditujukan untuk melakukan simulasi melalui software GNS3 untuk melakukan *NetworkAutomation* menggunakan *library* paramiko dan telnetlib pada jaringan IGP dan EGP. Penelitian dilakukan dalam beberapa tahap sesuai dengan diagram alur (*Flowchart*) yang ditunjukkan pada gambar 3.1



Gambar 3.1 Diagram Alur Penelitian

Gambar 3.1 menunjukkan diagram alur perancangan sistem dalam penelitian ini. Langkah pertama dalam penelitian yaitu melakukan studi literature beberapa penelitian terkait dengan *Networkautomation* untuk konfigurasi *routing protocol* serta materi lain yang berhubungan dengan penelitian ini. Dengan membandingkan beberapa jurnal terakut dan melakukan perbandingan untuk menentukan judul dan juga fokus dari penelitian ini. Selain membandingkan dan menentukan fokus atau judul penelitian, tahap ini juga berfungsi untuk memahami konsep dasar dari topik tersebut.

Selanjutnya adalah merancang topologi dimana topologi yang digunakan merupakan Topologi *Full Mesh* yang disusun menggunakan perangkat lunak GNS3 simulator. Sebelum melakukan penyusunan topologi, peneliti melakukan instalasi sistem *Networkautomation docker* yang sudah mendukung *networkautomation*. Topologi tersusun atas sebelas buah *router*, satu buah *switch* dan juga dua buah *client* dan sebuah sistem *Networkautomation* Ubuntu.

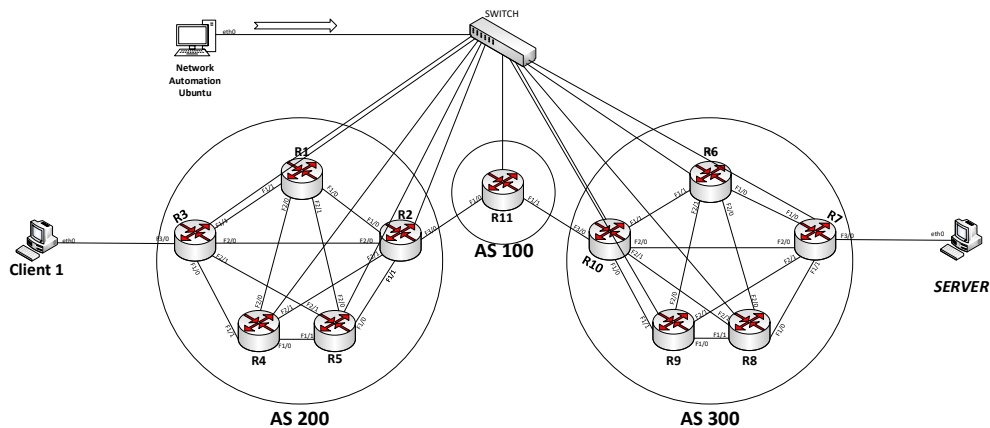
Setelah melakukan penyusunan topologi langkah selanjutnya adalah menyusun *script* atau baris program menggunakan bahasa pemrograman Python. Program yang akan dibuat total ada empat *script* yang akan digunakan yaitu, *library* paramiko untuk *routing* IGP dan EGP, serta *library* telnetlib untuk *routing* IGP dan EGP. Program python dibuat di sistem *NetworkAutomation* Ubuntu dengan membuat *file* dengan perintah "*nano*" supaya dapat masuk ke *file* yang telah dibuat. Setelah program dibuat, langkah selanjutnya adalah menjalankan masing-masing program dari empat program yang telah dibuat, apakah terdapat error atau tidak. Menjalankan program dilakukan pada sistem *NetworkAutomation* Ubuntu dengan perintah "*python3*". Jika terdapat *error* maka perlu kembali ke proses penyusunan program untuk memperbaiki *error* pada program yang telah dibuat, jika tidak terdapat *error* berarti pengujian berhasil

Ketika pengujian program berhasil dijalankan tanpa ada error maka langkah selanjutnya adalah pengambilan data. Data yang akan diambil berupa waktu yang diperlukan *NetworkAutomation* ubuntu pemberian konfigurasi ke *router*, waktu konvergensi IGP dan EGP, serta pengambilan data *Quality of Service* berupa *Delay* dan *Throughput*. Proses pengambilan ke tiga data dilakukan pada *software* Wireshark. Setelah semua data terkumpul dilanjutkan dengan langkah melakukan

analisis. Analisis dilakukan dengan cara membandingkan data yang diperoleh dari *library* paramiko dan telnetlib dalam bentuk grafik, kesimpulan diambil ketika proses analisis selesai dilakukan.

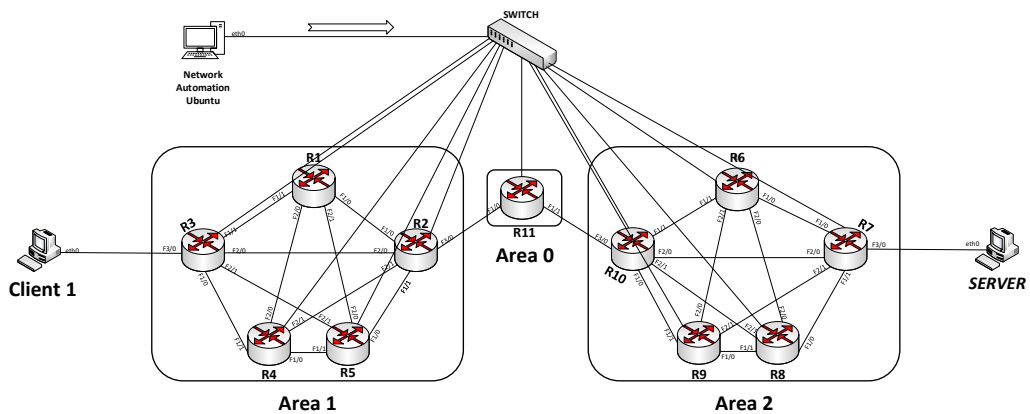
3.3 RANCANGAN TOPOLOGI

Penelitian ini menggunakan topologi jaringan untuk otomasi jaringan pada *routing protocol* IGP dan EGP. Topologi dapat dilihat pada gambar 3.2, dimana topologi yang digunakan merupakan topologi *Full Mesh*. Topologi tersebut tersusun atas sebelas buah *router* cisco 7200, satu buah *switch*, dua buah *pc client* dan satu buah sistem *NetworkAutomation* Ubuntu. Sistem *NetworkAutomation* Ubuntu berfungsi sebagai tempat untuk membuat *script* pemrograman python untuk *library* paramiko dan telnetlib. Sistem ini juga digunakan untuk mengirimkan *script* tersebut ke semua *router* melalui perantara dari perangkat *switch*. Selain berfungsi untuk menghubungkan sistem *networkautomation* dengan *router*, *switch* juga berfungsi dalam pengambilan data untuk mengetahui waktu yang dibutuhkan dalam memberikan konfigurasi.



Gambar 3.2 Topologi Jaringan IGP

Gambar 3.2 merupakan topologi yang akan digunakan untuk pengujian *library* paramiko dan telnetlib pada *routing* IGP yaitu *routing protocol* OSPF. Dalam penggunaan OSPF jenis area yang digunakan merupakan *multi-area*, dimana terdapat 3 area diantaranya area 0 (*Backbone*), area 1 dan area 2. *Node* R11 menuju R2 dan R10 berada dalam area 0 (*Backbone*). untuk R1,R2,R3,R4 dan R5 berada dalam area 1. Sedangkan untuk R6,R7,R8,R9,R10 berada dalam area 2.



Gambar 3.3 Topologi Jaringan EGP

Gambar 3.3 merupakan topologi yang akan digunakan dalam pengujian *library* paramiko dan telnetlib pada *routing* EGP. Dalam topologi ini tersusun atas 3 buah AS yaitu AS 100, AS 200 dan AS 300. Untuk AS 100 berisi R11, AS 200 berisi R1,R2,R3,R4 dan R5. Untuk AS 300 berisi R6,R7,R8,R9 dan R10.

Sebelas *router* disini digunakan untuk menerima konfigurasi *routing* IGP dan EGP yang diberikan oleh *NetworkAutomation* ubuntu. Dengan pemberian konfigurasi *routing* tersebut, dapat diperoleh hasil kinerja dari *library* paramiko dan telnetlib dalam penggunaan *NetworkAutomation*. PC *client* 1 dan 2 berfungsi untuk melakukan pengiriman data. Pengiriman data ini berupa paket ICMP yang berfungsi untuk mengetahui apakah PC *client* 1 dan 2 terhubung melalui konfigurasi IGP dan EGP. Dalam topologi ini di perlukan konfiggurasi IP untuk memberi identitas pada setiap *interface* pada Pc *client*, *router* dan *NetworkAutomation* Ubuntu. Untuk ip yang digunakan dapat dilihat pada tabel 3.3.

Tabel 3.3 Alamat IP Perangkat Jaringan

No	Device	Interface	IP Address	No	Device	Interface	IP Address
1.	R1	F0/0	20.20.20.2/27	8.	R4	F0/0	20.20.20.5/27
		F1/0	192.168.10.1/24			F1/0	192.168.55.1/24
		F1/1	192.168.15.1/24			F1/1	192.168.45.2/24
		F2/0	192.168.20.1/24			F2/0	192.168.20.2/24
		F2/1	192.168.25.1/24			F2/1	192.168.40.2/24
2.	R2	F0/0	20.20.20.3/27	9.	R7	F0/0	20.20.20.8/27
		F1/0	192.168.10.2/24			F1/0	172.16.10.2/24

No	Device	Interface	IP Address	No	Device	Interface	IP Address
		F1/1	192.168.30.1/24			F1/1	172.16.30.1/24
		F2/0	192.168.35.1/24			F2/0	172.16.35.1/24
		F2/1	192.168.40.1/24			F2/1	172.16.40.1/24
		F3/0	10.10.10.1/27			F3/0	172.16.60.2/24
3.	R3	F0/0	20.20.20.4/27	10.	R10	F0/0	20.20.20.11/27
		F1/0	192.168.45.1/24			F1/0	172.16.55.2/24
		F1/1	192.168.15.2/24			F1/1	172.16.15.2/24
		F2/0	192.168.35.2/24			F2/0	172.16.35.2/24
		F2/1	192.168.50.1/24			F2/1	172.16.50.2/24
		F3/0	192.168.60.2/24			F3/0	30.30.30.1/27
4.	R5	F0/0	20.20.20.6/27	11.	R8	F0/0	20.20.20.9/27
		F1/0	192.168.30.2/24			F1/0	172.16.30.2/24
		F1/1	192.168.55.2/24			F1/1	172.16.45.1/24
		F2/0	192.168.25.2/24			F2/0	172.16.20.2/24
		F2/1	192.168.50.2/24			F2/1	172.16.50.1/24
5.	R6	F0/0	20.20.20.7/27	12.	R9	F0/0	20.20.20.10/27
		F1/0	172.16.10.1/24			F1/0	172.16.45.2/24
		F1/1	172.16.15.1/24			F1/1	172.16.55.1/24
		F2/0	172.16.20.1/24			F2/0	172.16.25.2/24
		F2/1	172.16.25.1/24			F2/1	172.16.40.2/24
6.	R11	F0/0	20.20.20.12/27	13.	<i>Client 1</i>	Eth0	192.168.60.1/24
		F1/0	10.10.10.2/27	14.	<i>Server</i>	Eth0	172.16.60.1/24
		F1/1	30.30.30.2/27	15.	<i>NetworkAutomation</i>	Eth0	20.20.20.1/27

Tabel 3.3 berisi mengenai identitas berupa alamat ip dari masing-masing perangkat *router*, *client 1*, *server 2* serta ip dari sistem *networkautomation* ubuntu pada jaringan yang digunakan. Setiap interface diberikan ip supaya setiap perangkat jaringan berupa PC Client 1, Server, Network automation dan setiap router dapat saling komunikasi satu sama lainnya.

3.4 KONFIGURASI PERANGKAT

3.4.1 Konfigurasi SSH Dan Telnet

3.4.1.1 *Secure Shell (SSH)*

Konfigurasi SSH (*Secure shell*) dilakukan untuk setiap *router* yang ada pada topologi, supaya *NetworkAutomation* Ubuntu dapat berkomunikasi dengan semua *router*. Dalam konfigurasi ini yang dilakukan adalah pemberian alamat ip, pemberian *username* dan *password* dan konfigurasi SSH pada *interface router* yang terhubung dengan *NetworkAutomation* Ubuntu. Konfigurasi SSH dapat dilihat di bawah:

```
R1(config)#int f0/0
R1(config-if)#ip add 20.20.20.1 255.255.255.224
R1(config-if)#no sh
R1(config-if)#ex
R1(config)#username yuansa secret coba123
R1(config)#username yuansa privilege 15
R1(config)#ip domain-name ittelkom.com
R1(config)#crypto key generate rsa modulus 1024
R1(config)#line vty 0 4
R1(config)# transport input ssh
R1(config)#login local
```

Konfigurasi SSH di atas berfungsi agar *NetworkAutomation* Ubuntu dapat mengirimkan *script* python paramiko dan telnetlib ke setiap *router* yang terhubung. Pemberian konfigurasi alamat ip tersebut dilakukan ke setiap *router* dengan alamat ip sesuai dengan tabel 3.3. Selain pemberian alamat ip dilakukan konfigurasi lain berupa pemberian *username*, *password* dan konfigurasi SSH. Pemberian konfigurasi SSH dilakukan agar *NetworkAutomation* Ubuntu dapat meremote atau mengakses setiap *router*. Hal tersebut dilakukan agar *NetworkAutomation* Ubuntu dapat memberikan *script* konfigurasi paramiko dan telnetlib kesetiap *router*.

Pemberian konfigurasi “*ip domain-name ittelkom.com*” digunakan untuk membuat DNS *domain name* pada *router*. Konfigurasi “*crypto key generate rsa modulus 1024*” Pembuatan pasangan kunci RSA untuk *router* dengan nilai ukuran modulus minimum 1024. Untuk perintah “*line vty 0 4*” berfungsi untuk membatasi perangkat yang mengakses ssh tersebut (empat perangkat). Perintah

“*Transport input ssh*” berfungsi sebagai perintah untuk menentukan perintah *protocol* remote berupa *ssh*. Dan yang terakhir perintah “*login local*” berfungsi untuk mengautentikasi *username* dan *password*.



Gambar 3.4 Akses SSH ke Router1

Gambar 3.4 merupakan salah satu hasil dari konfigurasi SSH pada *router*. Perintah “*ssh yuansa@20.20.20.2*” mengakses *router* 1 yang sudah terkonfigurasi dengan SSH. Pengaksesan SSH pada setiap *router* memiliki perintah yang sama dengan *router* 1 hanya berbeda pada ip yang digunakan. Setelah perintah tersebut diberikan maka akan muncul autentikasi berupa memasukan *password* yang sudah dibuat sebelumnya.

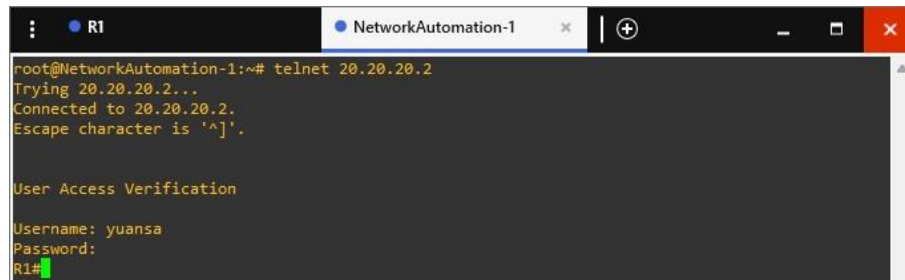
3.4.1.2 Telnet

Konfigurasi *protocol* telnet dilakukan agar sistem *networkautomation* Ubuntu dapat mengakses atau melakukan remote pada setiap *router* yang ada pada topologi. Dalam konfigurasi ini yang dilakukan adalah pemberian ip, *username* dan *password* untuk protokol telnet pada *interface router* yang terhubung dengan sistem *NetworkAutomation* Ubuntu. Konfigurasi telnet dapat dilihat di bawah:

```
R1(config)#int f0/0
R1(config-if)#ip add 20.20.20.2 255.255.255.240
R1(config-if)#no sh
R1(config-if)#ex
R1(config)#username yuansa secret coba123
R1(config)#username yuansa privilege 15
R1(config)#line vty 0 4
R1(config)#transport input telnet
R1(config)#login local
```

Pemberian konfigurasi Telnet hampir sama dengan konfigurasi SSH. Hanya saja dalam konfigurasi telnet tidak ada perintah pemberian nama domain ke *router* dan tidak ada kunci atau keamanan berupa *crypto* dalam komunikasi

antara *client* (*telnet*) dan *server*(*telnetd*). Hal tersebut membuat perbedaan antara *protocol* Telnet dan SSH yaitu terletak pada keamanan dalam berkomunikasi.



```
root@R1:~# telnet 20.20.20.2
Trying 20.20.20.2...
Connected to 20.20.20.2.
Escape character is '^]'.

User Access Verification

Username: yuansa
Password:
R1#
```

Gambar 3.5 Akses Telnet ke Router 1

Gambar 3.6 merupakan hasil dari konfigurasi Telnet pada Router 1. Perintah untuk dapat mengakses atau meremote router 1 menggunakan “*telnet 20.20.20.2*”. selain router 1 perintah yang digunakan sama hanya berbeda pada ip saja. Setelah memasukan perintah tersebut maka akan muncul “*User Access Verification*” yang berisi verifikasi *username* dan *password* untuk dapat mengakses router1.

3.4.2 Konfigurasi Program Paramiko

Program Python *library* paramiko yang digunakan berisi konfigurasi-konfigurasi alamat IP setiap *interface* router, *routing* IGP dan EGP. Program tersebut dibuat di dalam *NetworkAutomation* Ubuntu yang tersedia di perangkat lunak GNS3. Perintah pembuatan *file* paramiko pada *routing* IGP:

```
root@NetworkAutomation-1:~#nano paramikoospf.py
```

Pembuatan *file* python dilakukan di *NetworkAutomation* dengan menggunakan perintah seperti pada konfigurasi di atas. *File* program tersebut disimpan dengan nama dan format “*paramikoospf.py*”. *Script* dibawah merupakan salah satu *sample* pemrograman Python dengan *library* Paramiko untuk konfigurasi *Routing protocol* IGP. Untuk Program IGP paramiko ini akan dijalankan pada di topologi pada gambar 3.2. Pada *script* di bawah terdapat perintah *import* yang digunakan untuk memasukan modul yang akan digunakan. Pada program paramiko untuk modul yang digunakan adalah modul “*paramiko*” sendiri dan juga modul “*time*”.

```

import paramiko
import time
def configure_router(host, username, password, commands):
    ssh_client = paramiko.SSHClient()
    ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    ssh_client.connect(hostname=ip, username=username,
password=password)
    print ("Berhasil Login ke {0}".format(ip))
    conn = ssh_client.invoke_shell()
    for command in commands:
        conn.send(command + "\n")
def main():
    # Router 1 configuration
    host1 = "20.20.20.2"
    username1 = "yuansa"
    password1 = "coba123"
    commands1 = [
        "conf t",
        "int fa1/0",
        "ip add 192.168.10.1 255.255.255.0",
        "no sh",
        "int fa1/1",
        "ip add 192.168.15.1 255.255.255.0",
        "no sh",
        "int fa2/0",
        "ip add 192.168.20.1 255.255.255.0",
        "no sh",
        "int fa2/1",
        "ip add 192.168.25.1 255.255.255.0",
        "no sh",
        "router ospf 10 ",
        "network192.168.10.0 0.0.0.255 area 1",
        "network192.168.15.0 0.0.0.255 area 1",
        "network192.168.20.0 0.0.0.255 area 1",
        "network192.168.25.0 0.0.0.255 area 1"
    ]
    print(" Konfigurasi Router 1...")
    output1 = configure_router(host1, username1, password1,
commands1)
    print(output1)

```

Kemudian terdapat deklarasi fungsi *configure_router* dengan memanggil variabel yaitu variabel “*host, username, password* dan *commands*”. Variabel *username* dan *password* berfungsi untuk menyimpan *username* dan *password* SSH, supaya paramiko dapat mengakses perangkat yang *teremote* menggunakan SSH. Perintah selanjutnya digunakan untuk memanggil fungsi SSH *client* dari paramiko dan juga penambahan key policy secara otomatis. *Command* “*conn*” berfungsi untuk memberikan perintah dari *script* python ke *router*. “*time.sleep(1)*”

merupakan perintah yang diberikan oleh *library* paramiko untuk memberikan 1 detik jeda untuk menunggu *router* selesai memberikan *command*. Kemudian terdapat variabel *commands* yang berisi konfigurasi ip address dan *routing* OSPF. Terdapat fungsi *main()* digunakan untuk menyimpan isi variabel *host*, *username*, *password* dan *commands*, dimana variabel tersebut nantinya akan dipanggil oleh fungsi *configure_router* untuk dapat dibaca oleh *router* dengan protokol SSH.

3.4.3 Konfigurasi Program Telnetlib

Pembuatan program python telnetlib hampir sama dengan *library* paramiko, hanya berbeda pada perintah pengiriman *command* ke *router*. Program Python *library* telnetlib yang digunakan berisi konfigurasi-konfigurasi alamat IP setiap *interface router*, *routing* OSPF dan BGP. Perintah pembuatan *file library* telnetlib pada *routing* IGP:

```
root@NetworkAutomation-1:~#nano telnetlibospf.py
```

pembuatan *file* python dilakukan di *NetworkAutomation* dengan menggunakan perintah di atas. *File* program tersebut disimpan dengan nama dan format “*telnetlibospf.py*”. *Script* di bawah merupakan salah satu sample *script* program python pada *library* telnetlib untuk *routing* IGP. Program tersebut dijalankan di topologi pada gambar 3.2. Pada *script* telnetlib modul yang diimpor merupakan modul “telnetlib” sendiri dan juga modul “time” yang berfungsi memberi waktu jeda kepada *router* untuk ke konfigurasi selanjutnya.

```
import telnetlib
import time
def configure_router(host, username, password, commands):
    tn = telnetlib.Telnet(host)
    tn.read_until(b"Username: ")
    tn.write(username.encode('ascii') + b"\n")
    if password:
        tn.read_until(b"Password: ")
        tn.write(password.encode('ascii') + b"\n")
    for command in commands:
        tn.write(command.encode('ascii') + b"\n")
        time.sleep(1)
    tn.write(b"exit\n")
    tn.read_all().decode('ascii')
    tn.close()
    return output
```

```

def main():
    # Router 1 configuration
    host1 = "20.20.20.2"
    username1 = "yuansa"
    password1 = "coba123"
    commands1 = [
        "conf t",
        "int fa1/0",
        "ip add 192.168.10.1 255.255.255.0",
        "no sh",
        "int fa1/1",
        "ip add 192.168.15.1 255.255.255.0",
        "no sh",
        "int fa2/0",
        "ip add 192.168.20.1 255.255.255.0",
        "no sh",
        "int fa2/1",
        "ip add 192.168.25.1 255.255.255.0",
        "no sh",
        "router ospf 10 ",
        "network192.168.10.0 0.0.0.255 area 1",
        "network192.168.15.0 0.0.0.255 area 1",
        "network192.168.20.0 0.0.0.255 area 1",
        "network192.168.25.0 0.0.0.255 area 1"
    ]
    # Perform configuration for Router 1
    print(" Konfigurasi Router 1...")
    output1 = configure_router(host1, username1, password1,
    commands1)
    print(output1)
    print("Router 1 Selesai.\n")

```

Kemudian terdapat deklarasi fungsi `configure_router` dengan memanggil variabel yaitu variabel “host, username, password dan commands”. Variabel username dan password berfungsi untuk menyimpan username dan password Telnet, serta variabel commands yang berisi konfigurasi ip address dan *routing* OSPF. Pengkodean “ascii” digunakan untuk mengubah string menjadi byte, karena *protocol* telnet menggunakan byte sebagai unit komunikasi.

3.5 PENGUJIAN KINERJA *LIBRARY*

Pengujian kinerja library merupakan proses selanjut dalam melakukan otomasi jaringan. Pengujian ini dilakukan menggunakan Topologi *Full Mesh* yang sudah tersusun oleh beberapa perangkat dan sistem *NetworkAutomation* Ubuntu. Proses pengujian dilakukan dengan cara mengeksekusi *script* atau program python

yang telah dibuat. Dimana program tersebut berisi merupakan konfigurasi *Ip Address* dan juga *routing protocol* IGP dan EGP menggunakan *library* Paramiko dan Telnetlib dengan format “.py”. Program tersebut akan dieksekusi pada topologi yang telah dibuat di aplikasi GNS3, tepatnya pada sistem *NetworkAutomation* Ubuntu melalui *solar-putty* sebagai *remote access* untuk menjalankan perintah. Perintah untuk menjalankan program paramiko pada *routing* OSPF :

```
root@NetworkAutomation-1:~#python3 paramikoospf.py
```

Total program yang diuji sebanyak 4 *script* python yang terdiri dari *Library* Paramiko untuk *routing* IGP dan EGP, serta *Library* Telnetlib untuk *routing* IGP dan EGP. Program akan dirun melalui *mode root* pada sistem *NetworkAutomation* Ubuntu dengan mengetik perintah untuk Program *library* Paramiko untuk *routing* IGP “*python paramikoospf.py*” dan untuk EGP “*python paramikobgp.py*”. Kemudian perintah untuk menjalankan program telnetlib untuk *routing* IGP “*python telnetlibospf.py*” dan untuk EGP “*python telnetlibbgp.py*”. Untuk program OSPF Paramiko dan telnetlib akan dijalankan pada topologi untuk IGP yaitu menggunakan topologi pada gambar 3.2, sedangkan untuk program EGP paramiko dan telnetlib dijalankan pada topologi yang ada pada gambar 3.3

3.6 HASIL PEMBERIAN KONFIGURASI KE *ROUTER*

Pemberian konfigurasi ke router dilakukan oleh *networkautomation* dengan mengirimkan *script* python ke setiap router. Hasil pemberian konfigurasi ke *router* dilakukan menggunakan kedua *library* yaitu *library* paramiko dan telnetlib. Konfigurasi yang diberikan ke *router* merupakan konfigurasi *IP Address* dan *Routing protocol* IGP berupa OSPF dan EGP berupa BGP. Gambar dibawah ini menunjukkan hasil verifikasi bahwa *script* konfigurasi yang dikirim menggunakan *library* paramiko dan telnetlib berhasil diterima oleh *router*.

Gambar 3.6 menunjukkan bahwa ketika *library* paramiko dan telnetlib mengirimkan konfigurasi IGP berupa OSPF ke *router* maka akan diperoleh hasil bahwa setiap *router* sudah terkonfigurasi *routing* IGP. Gambar 3.7 merupakan hasil konfigurasi *routing* IGP, dimana semua alamat *network* yang tidak terhubung ke R11 sudah tertampil pada menu “*show ip route ospf*”.

```

Aug 14 06:05:49.695: NLINK-5-CHANGED: Interface FastEthernet1/0, changed state to administratively down
Aug 14 06:05:49.699: NLINK-5-CHANGED: Interface FastEthernet1/0, changed state to administratively down
Aug 14 06:05:49.783: NLINK-5-CHANGED: Interface FastEthernet4/0, changed state to administratively down
Aug 14 06:05:49.787: NLINK-5-CHANGED: Interface FastEthernet4/0, changed state to administratively down
Aug 14 06:05:50.695: NLINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet1/0, changed state to down
Aug 14 06:05:50.697: MCRPTD-6-15AMP_ON_OFF: 15AMP is OFF
Aug 14 06:05:50.851: MCRPTD-6-001_ON_OFF: 0001 is OFF
Aug 14 06:05:50.431: NLINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet1/1, changed state to down
Aug 14 06:05:50.697: NLINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet2/0, changed state to down
Aug 14 06:05:50.698: NLINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet2/1, changed state to down
Aug 14 06:05:50.699: NLINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet3/0, changed state to down
Aug 14 06:05:50.699: NLINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet3/1, changed state to down
Aug 14 06:05:50.783: NLINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet4/1, changed state to down
Aug 14 06:05:50.787: NLINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet4/1, changed state to down
R11#sh ip route ospf
Aug 14 06:07:56.151: NLINK-3-UPDOWN: Interface FastEthernet1/0, changed state to up
Aug 14 06:07:56.281: NLINK-3-UPDOWN: Interface FastEthernet1/1, changed state to up
Aug 14 06:07:57.151: NLINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet1/0, changed state to up
R11#sh ip route ospf
Aug 14 06:07:57.563: NDIS-5-COMP16.1: Configured from console by yuansu on vty0 (20.20.20.1)
Aug 14 06:07:57.743: NLINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet1/1, changed state to up
R11#sh ip route ospf
Aug 14 06:08:04.643: NSPPF-5-ADJCHG: Process 18, Nbr 192.168.40.1 on FastEthernet1/0 from LOADING to FULL, Loading Done
R11#sh ip route ospf
Aug 14 06:08:04.131: NSPPF-5-ADJCHG: Process 18, Nbr 172.16.55.2 on FastEthernet1/1 from LOADING to FULL, Loading Done
R11#sh ip route ospf
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
NI - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
I - IS-IS, IA - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, I - ISIS
+ - replicated route, % - next hop override
Gateway of last resort is not set
0.0.0.0 is subnetted, 11 subnets
O IA 172.16.18.0 [110/2] via 30.30.30.1, 00:02:35, FastEthernet1/1
O IA 172.16.15.0 [110/2] via 30.30.30.1, 00:02:35, FastEthernet1/1
O IA 172.16.20.0 [110/3] via 30.30.30.1, 00:02:35, FastEthernet1/1
O IA 172.16.25.0 [110/3] via 30.30.30.1, 00:02:35, FastEthernet1/1
O IA 172.16.30.0 [110/3] via 30.30.30.1, 00:02:35, FastEthernet1/1
O IA 172.16.35.0 [110/2] via 30.30.30.1, 00:02:35, FastEthernet1/1
O IA 172.16.40.0 [110/3] via 30.30.30.1, 00:02:35, FastEthernet1/1
O IA 172.16.45.0 [110/3] via 30.30.30.1, 00:02:35, FastEthernet1/1
O IA 172.16.50.0 [110/2] via 30.30.30.1, 00:02:35, FastEthernet1/1
O IA 172.16.55.0 [110/2] via 30.30.30.1, 00:02:35, FastEthernet1/1
O IA 172.16.60.0 [110/3] via 30.30.30.1, 00:02:35, FastEthernet1/1
O IA 192.168.0/24 [110/2] via 10.10.10.1, 00:03:10, FastEthernet1/0
O IA 192.168.10.0/24 [110/3] via 10.10.10.1, 00:03:10, FastEthernet1/0
O IA 192.168.20.0/24 [110/3] via 10.10.10.1, 00:03:10, FastEthernet1/0
O IA 192.168.30.0/24 [110/3] via 10.10.10.1, 00:03:10, FastEthernet1/0
O IA 192.168.40.0/24 [110/2] via 10.10.10.1, 00:03:10, FastEthernet1/0
O IA 192.168.50.0/24 [110/3] via 10.10.10.1, 00:03:10, FastEthernet1/0
O IA 192.168.60.0/24 [110/3] via 10.10.10.1, 00:03:10, FastEthernet1/0
R11#

```

Gambar 3. 6 Hasil konfigurasi protokol routing IGP pada R11

Dalam gambar terlihat bahwa terdapat IP yang tidak terhubung dengan router 11. Dan terdapat variabel “O” yang menunjukkan bahwa IP tersebut merupakan IP milik tetangga dan router yang mengaktifkan protokol routing OSPF. Hal tersebut menandakan bahwa konfigurasi routing protocol IGP berupa OSPF berhasil dilakukan.

```

+ - replicated route, % - next hop override
Gateway of last resort is not set
R11#sh ip route bgp
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
NI - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2
I - IS-IS, IA - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
ia - IS-IS inter area, * - candidate default, U - per-user static route
o - ODR, P - periodic downloaded static route, H - NHRP, I - ISIS
+ - replicated route, % - next hop override
Gateway of last resort is not set
0.0.0.0 is subnetted, 1 subnets
O 1.1.1.1 [200] via 10.10.10.1, 00:02:52
O 2.0.0.0 [200] via 10.10.10.1, 00:02:52
O 3.0.0.0 [200] via 10.10.10.1, 00:02:52
O 4.0.0.0 [200] via 10.10.10.1, 00:02:52
O 5.0.0.0 [200] via 10.10.10.1, 00:02:52
O 6.0.0.0 [200] via 10.10.10.1, 00:02:52
O 7.0.0.0 [200] via 10.10.10.1, 00:02:52
O 8.0.0.0 [200] via 10.10.10.1, 00:02:52
O 9.0.0.0 [200] via 10.10.10.1, 00:02:52
O 10.0.0.0 is verypduy subnotted, 3 subnets, 2 masks
O 50.10.10.1/32 [200] via 30.30.30.1, 00:02:52
O 172.16.0/24 [200] via 30.30.30.1, 00:02:52
O 172.16.15.0 [200] via 30.30.30.1, 00:02:52
O 172.16.20.0 [200] via 30.30.30.1, 00:02:52
O 172.16.25.0 [200] via 30.30.30.1, 00:02:52
O 172.16.30.0 [200] via 30.30.30.1, 00:02:52
O 172.16.35.0 [200] via 30.30.30.1, 00:02:52
O 172.16.40.0 [200] via 30.30.30.1, 00:02:52
O 172.16.45.0 [200] via 30.30.30.1, 00:02:52
O 172.16.50.0 [200] via 30.30.30.1, 00:02:52
O 172.16.55.0 [200] via 30.30.30.1, 00:02:52
O 192.168.0/24 [200] via 10.10.10.1, 00:02:52
O 192.168.10.0/24 [200] via 10.10.10.1, 00:02:52
O 192.168.20.0/24 [200] via 10.10.10.1, 00:02:52
O 192.168.30.0/24 [200] via 10.10.10.1, 00:02:52
O 192.168.40.0/24 [200] via 10.10.10.1, 00:02:52
O 192.168.50.0/24 [200] via 10.10.10.1, 00:02:52
O 192.168.60.0/24 [200] via 10.10.10.1, 00:02:52
R11#
R11#

```

Gambar 3.7 Hasil konfigurasi protokol routing EGP pada R11

Gambar 3.7 menunjukkan bahwa ketika *library* paramiko dan telnetlib mengirimkan konfigurasi EGP berupa BGP ke *router* maka akan diperoleh hasil bahwa setiap *router* sudah terkonfigurasi *routing* EGP. Gambar 3.8 merupakan hasil konfigurasi *routing* EGP, dimana semua Alamat *network* yang tidak terhubung ke R11 atau tidak berada dalam satu area sudah tertampil pada menu “*show ip route bgp*”. Dalam gambar terlihat bahwa terdapat *IP* yang tidak terhubung dengan router 11. Dan terdapat variabel “O” yang menunjukkan bahwa *IP* tersebut merupakan *IP* milik *neighbor* dan router yang mengaktifkan protokol routing BGP. Hal tersebut menandakan bahwa konfigurasi *routing protocol* EGP berupa BGP berhasil dilakukan.

3.6.1 Hasil Pemberian Konfigurasi Menggunakan *Library* Paramiko

3.6.1.1 Internal Gateway Protocol (IGP)

Pemberian konfigurasi IGP ke *router* menggunakan *networkautomation* dilakukan dengan cara melakukan pengiriman *script* bahasa python ke *router*. *script* bahasa python ini dibuat dengan cara “*nano paramikoigp.py*”. *Script* python dikirim menggunakan perintah “*python3 paramikoigp.py*” maka program akan dikirimkan oleh *networkautomation* melalui *switch*.

```

File "paramikoigp.py", line 285, in main
  output = configure_router(local, username1, password1, commands)
File "paramikoigp.py", line 7, in configure_router
  ssh_client.connect(hostname=host, username=username, password=password)
File "/usr/local/lib/python3.8/dist-packages/paramiko/client.py", line 406, in connect
  t.start_client(timeout=timeout)
File "/usr/local/lib/python3.8/dist-packages/paramiko/transport.py", line 660, in start_client
  raise EOFError
File "/usr/local/lib/python3.8/dist-packages/paramiko/transport.py", line 2055, in run
  ptype = self.packetizer.read_message()
File "/usr/local/lib/python3.8/dist-packages/paramiko/packet.py", line 459, in read_message
  header = self.read_all(self._block_size_in, check_key=True)
File "/usr/local/lib/python3.8/dist-packages/paramiko/packet.py", line 303, in read_all
  raise EOFError()
EOFError
root@NetworkAutomation-1:~# python3 paramikoigp.py
Konfigurasi Router 1...
Berhasil Login ke 20.20.20.2
None
Router 1 Selesai.
Konfigurasi Router 2...
Berhasil Login ke 20.20.20.3
None
Router 2 Selesai.
Konfigurasi Router 3...
Berhasil Login ke 20.20.20.4
None
Router 3 Selesai.
Konfigurasi Router 4...
Berhasil Login ke 20.20.20.5
None
Router 4 Selesai.
Konfigurasi Router 5...
Berhasil Login ke 20.20.20.6
None
Router 5 Selesai.
Konfigurasi Router 6...
Berhasil Login ke 20.20.20.7
None
Router 6 Selesai.
Konfigurasi Router 7...
Berhasil Login ke 20.20.20.8
None
Router 7 Selesai.
Konfigurasi Router 8...
Berhasil Login ke 20.20.20.9
None
Router 8 Selesai.
Konfigurasi Router 9...
Berhasil Login ke 20.20.20.10
None
Router 9 Selesai.
Konfigurasi Router 10...
Berhasil Login ke 20.20.20.11
None
Router 10 Selesai.
Konfigurasi Router 11...
Berhasil Login ke 20.20.20.12
None
Router 11 Selesai.
root@NetworkAutomation-1:~#

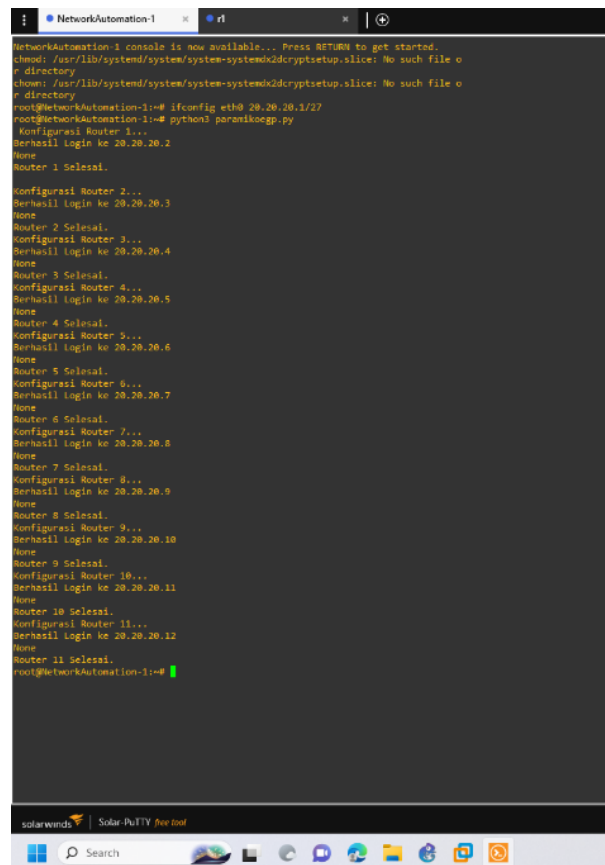
```

Gambar 3.8 Hasil Pemberian Perintah Konfigurasi IGP ke *Router Library* Paramiko

Gambar 3.8 merupakan hasil notifikasi bahwa *script* python untuk konfigurasi IGP ke *router* sudah dikirimkan dan diterima. Notifikasi tersebut berisi bahwa sudah bisa masuk *remote ssh* dengan keterangan notif “Berhasil Login ke 20.20.20.1”, melakukan konfigurasi dan notifikasi terakhir yaitu “Router 1 Selesai” yang berarti seluruh konfigurasi telah diterima.

3.6.1.2 Exterior Gateway Protocol (EGP)

Pemberian konfigurasi IGP ke *router* menggunakan *networkautomation* dilakukan dengan cara melakukan pengiriman *script* bahasa python ke *router*. *script* bahasa python ini dibuat dengan cara “*nano paramikoegp.py*”. *Script* python dikirim menggunakan perintah “*python3 paramikoegp.py*” maka program akan dikirimkan oleh *networkautomation* melalui switch.



```
NetworkAutomation-1 console is now available... Press RETURN to get started.
cloud: /usr/lib/systemd/system/system-systemdxdcryptsetup.slice: No such file or
r directory
cloud: /usr/lib/systemd/system/system-systemdxdcryptsetup.slice: No such file or
r directory
root@NetworkAutomation-1:~# ifconfig eth0 20.20.20.1/27
root@NetworkAutomation-1:~# python3 paramikoegp.py
Konfigurasi Router 1...
Berhasil login ke 20.20.20.2
None
Router 1 Selesai.

Konfigurasi Router 2...
Berhasil login ke 20.20.20.3
None
Router 2 Selesai.
Konfigurasi Router 3...
Berhasil login ke 20.20.20.4
None
Router 3 Selesai.
Konfigurasi Router 4...
Berhasil login ke 20.20.20.5
None
Router 4 Selesai.
Konfigurasi Router 5...
Berhasil login ke 20.20.20.6
None
Router 5 Selesai.
Konfigurasi Router 6...
Berhasil login ke 20.20.20.7
None
Router 6 Selesai.
Konfigurasi Router 7...
Berhasil login ke 20.20.20.8
None
Router 7 Selesai.
Konfigurasi Router 8...
Berhasil login ke 20.20.20.9
None
Router 8 Selesai.
Konfigurasi Router 9...
Berhasil login ke 20.20.20.10
None
Router 9 Selesai.
Konfigurasi Router 10...
Berhasil login ke 20.20.20.11
None
Router 10 Selesai.
Konfigurasi Router 11...
Berhasil login ke 20.20.20.12
None
Router 11 Selesai.
root@NetworkAutomation-1:~#
```

Gambar 3.9 Hasil Pemberian Perintah Konfigurasi EGP ke Router Library Paramiko

Gambar 3.9 merupakan hasil notifikasi bahwa *script* python untuk konfigurasi EGP ke *router* sudah dikirimkan dan diterima. Notifikasi tersebut berisi

bahwa sudah bisa masuk remotes ssh dengan keterangan notif “Berhasil Login ke 20.20.20.1 ”,, melakukan konfigurasi dan notifikasi terakhir yaitu “Router 1 Selesai” yang berarti seluruh konfigurasi telah diterima.

3.6.2 Hasil Pemberian Konfigurasi Menggunakan *Library* Telnetlib

3.6.2.1 *Internal Gateway Protocol (IGP)*

Pemberian konfigurasi IGP ke *router* menggunakan *networkautomation* dilakukan dengan cara melakukan pengiriman *script* bahasa python ke *router*. *script* bahasa python ini dibuat dengan cara “*nano telnetlibigp.py*” *Script* python dikirim menggunakan perintah “*python3 telnetlibigp.py*” maka program akan dikirimkan oleh *networkautomation* melalui *switch*.



```
root@NetworkAutomation-1:~# nano egg.py
root@NetworkAutomation-1:~# nano telnetlibigp.py
root@NetworkAutomation-1:~# python3 telnetlibigp.py
Configurasi Router 1...
None
Router 1 Selesai.
Configurasi Router 2...
None
Router 2 Selesai.
Configurasi Router 3...
None
Router 3 Selesai.
Configurasi Router 4...
None
Router 4 Selesai.
Configurasi Router 5...
None
Router 5 Selesai.
Configurasi Router 6...
None
Router 6 Selesai.
Configurasi Router 7...
None
Router 7 Selesai.
Configurasi Router 8...
None
Router 8 Selesai.
Configurasi Router 9...
None
Router 9 Selesai.
Configurasi Router 10...
None
Router 10 Selesai.
Configurasi Router 11...
None
Router 11 Selesai.
root@NetworkAutomation-1:~#
```

Gambar 3.10 Hasil Pemberian Perintah Konfigurasi IGP ke *Router Library* Telnetlib

Gambar 3.10 merupakan hasil notifikasi bahwa *script* python untuk konfigurasi IGP ke *router* sudah dikirimkan dan diterima. Notifikasi yang muncul berupa melakukan konfigurasi ke setiap *router* dan notifikasi terakhir yaitu “Router 1 Selesai” yang menandakan konfigurasi IGP telah diterima ke *router*.

3.6.2.2 *Exterior Gateway Protocol (EGP)*

Pemberian konfigurasi IGP ke *router* menggunakan *networkautomation* dilakukan dengan cara melakukan pengiriman *script* bahasa python ke *router*. *script*

bahasa python ini dibuat dengan cara “*nano telnetlibigp.py*” *Script* python dikirim menggunakan perintah “*python3 telnetlibigp.py*” maka program akan dikirimkan oleh *networkautomation* melalui *switch*.

```

root@NetworkAutomation-1:~# python3 egp.py
Konfigurasi Router 1...
None
Router 1 Selesai.
Konfigurasi Router 2...
None
Router 2 Selesai.
Konfigurasi Router 3...
None
Router 3 Selesai.
Konfigurasi Router 4...
None
Router 4 Selesai.
Konfigurasi Router 5...
None
Router 5 Selesai.
Konfigurasi Router 6...
None
Router 6 Selesai.
Konfigurasi Router 7...
None
Router 7 Selesai.
Konfigurasi Router 8...
None
Router 8 Selesai.
Konfigurasi Router 9...
None
Router 9 Selesai.
Konfigurasi Router 10...
None
Router 10 Selesai.
Konfigurasi Router 11...
None
Router 11 Selesai.
root@NetworkAutomation-1:~# nano egp.py

```

Gambar 3.11 Hasil Pemberian Perintah Konfigurasi EGP ke *Router Library* Telnetlib

Gambar 3.11 merupakan hasil notifikasi bahwa *script* python untuk konfigurasi EGP ke *router* sudah dikirimkan dan diterima. Notifikasi yang muncul berupa melakukan konfigurasi ke setiap *router* dan notifikasi terakhir yaitu “*Router 1 Selesai*” yang menandakan bahwa seluruh konfigurasi routing protocol EGP telah diterima.

3.7 PERHITUNGAN ARAH JALUR PADA *ROUTING* PROTOCOL

3.7.1 *Internal Gateway Protocol (IGP)*

Routing protocol IGP yang digunakan pada penelitian ini adalah OSPF. Penentuan arah jalur pengiriman data pada OSPF ditentukan menggunakan *cost*. Dimana nilai *cost* diperoleh dari rumus berikut :

$$cost = \frac{10^8}{Bw} \quad (3.1)$$

Keterangan :

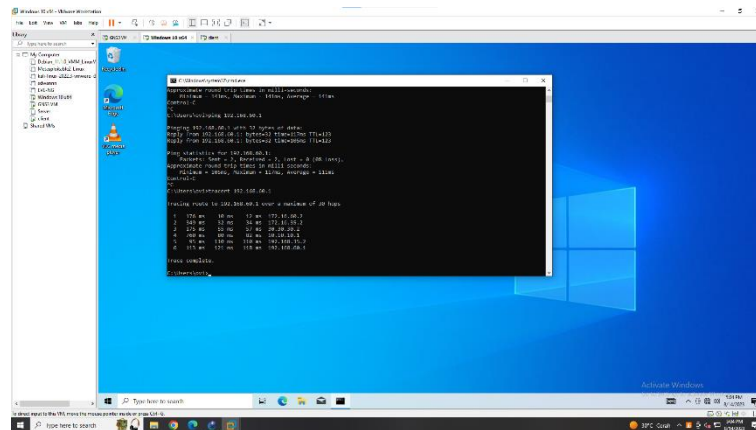
Bw = *Bandwidth* yang digunakan

Pada penelitian ini menggunakan port *FastEthernet* dengan nilai *bandwidth* sebesar 100 Mbps dan *Ethnet* pada pc *Client* sebesar 10 Mbps. Sehingga diperoleh nilai *cost* pada penelitian ini sebesar :

$$\text{cost FastEthernet} = \frac{10^8}{100000000} = 1 \quad (3.2)$$

$$\text{cost Ethernet} = \frac{10^8}{10000000} = 10 \quad (3.3)$$

Maka penggunaan *routing protocol* IGP diperoleh jalur terdekat yaitu menggunakan jalur *server* > R7 (*FastEthernet* 3/0) > R10 (*FastEthernet* 2/0) > R11 (*FastEthernet* 1/1) > R2 (*FastEthernet* 3/0) > R3 (*FastEthernet* 2/0) > *client* 1 (*Ethernet* 0) dengan nilai *cost* sebesar : 1 + 1 + 1 + 1 + 1 + 10 = 15 *cost*.



Gambar 3.12 Hasil Tracer Router Server ke Client pada Routing protocol IGP

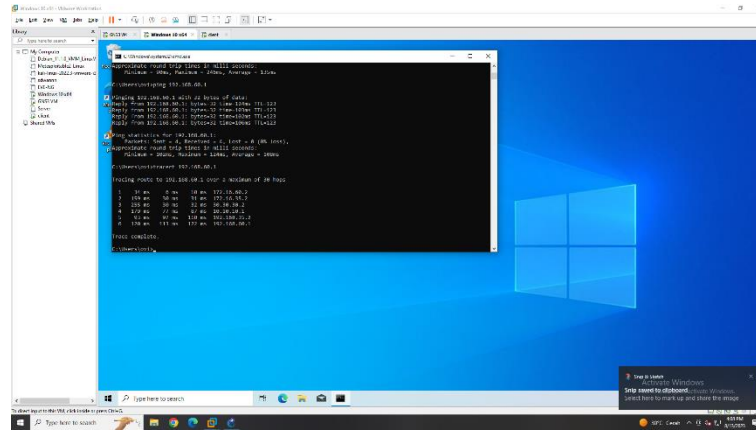
Gambar 3.12 merupakan hasil tracer *router* menggunakan *routing protocol* IGP dari Server ke *client*. Dari hasil tracer route tersebut diperoleh bahwa data dikirimkan melalui total 6 buah *interface*. *Interface* yang pertama melalui *interface* dengan IP 172.16.60.2 (R7), kedua 172.16.35.2 (R10), ketiga melewati 30.30.30.2 (R11), kemudian melewati 10.10.10.1 (R2), kelima melewati 192.168.35.2 (R3) dan yang terakhir yaitu *interface ethernet0* dari *client*. Hasil *trace route* ini diperoleh sama dengan perhitungan yang telah dilakukan.

3.7.2 Exterior Gateway Protocol (EGP)

Routing protocol EGP yang digunakan pada penelitian ini adalah BGP. Penentuan arah jalur pengiriman data pada BGP ditentukan menggunakan *metric* berupa jumlah *hop*. Jumlah *hop* disini yaitu jumlah wilayah AS yang dilewati. Pada

penelitian ini menggunakan tiga wilayah AS Number yaitu AS 100, AS 200 dan AS 300. Sehingga diperoleh jalur yang akan dilewati :

Server > AS 300 (R7 > R10) AS 100 (R11) dan AS 200 (R2 > R3) > Client



Gambar 3.13 Gambar Hasil Tracer Route pada *Routing protocol EGP*

Gambar 3.13 merupakan hasil tracer router menggunakan *routing protocol IGP* dari Server ke *client*. Dari hasil tracer route tersebut diperoleh bahwa data dikirimkan melalui total 6 buah *interface*. Namun untuk penentuan jalur terpendek itu menggunakan jumlah hop dari AS yang dilewati. Data akan melewati AS 300 melalui *interface* dengan IP 172.16.60.2 dan 172.16.35.2 atau R7 dan R10 yang berada pada AS 300. Yang kedua melewati *interface* dengan IP 30.30.30.2 atau R11 yang berada pada AS 100. Dan yang terakhir yaitu melalui *interface* dengan IP 10.10.10.1, 192.168.35.2 dan 192.168.60.1 atau R2, R3 dan *Client*.

3.8 SKENARIO PENGUJIAN DAN PENGAMBILAN DATA

Tahap selanjutnya adalah pengambilan data, dimana data yang dibutuhkan pada penelitian ini adalah waktu pengiriman konfigurasi ke *router*, waktu konvergensi IGP dan EGP, serta nilai QoS dari kedua *routing protocol IGP* dan EGP.

Tabel 3.4 Skenario Pengujian *Library NetworkAutomation*

<i>Library</i>	Skenario Pengujian	<i>Routing Protocol</i>
	Waktu Pemberian Perintah Konfigurasi dan Waktu Konvergensi	
Paramiko	20 Kali	IGP - OSPF

<i>Library</i>	Skenario Pengujian	<i>Routing Protocol</i>
	Waktu Pemberian Perintah Konfigurasi dan Waktu Konvergensi	
		EGP - BGP
Telnetlib	20 Kali	IGP - OSPF
		EGP - BGP

Tabel 3.5 Skenario Pengujian *Routing protocol*

<i>Routing Protocol</i>	Ukuran File	Pengujian <i>Quality of Service (QoS)</i>
IGP – OSPF	531 Mb	5 Kali
EGP – BGP		

3.8.1 Waktu Pemberian Konfigurasi Ke Router

Program *python* digunakan untuk memberikan perintah konfigurasi ke setiap *router*. Perintah konfigurasi yang diberikan adalah pemberian alamat IP pada *interface router* dan mengaktifkan protokol *routing* IGP dan EGP untuk proses *routing*. Lamanya waktu pengiriman *script* python dengan *library* paramiko dan telnetlib dari sistem *NetworkAutomation* Ubuntu menuju ke *router* akan dihitung. Data diperoleh dari menghitung waktu SSH dan Telnet yang diambil dari cuplikan Wireshark pada jalur dari *NetworkAutomation* Ubuntu ke perangkat *router*. Sehingga waktu akhir SSH dan Telnet dikurangi dengan waktu awal akses atau pengiriman SSH dan telnet pada masing-masing *router*. Dengan mengetahui lamanya waktu pengiriman *script* dan melakukan perhitungannya, sehingga dapat mengetahui *library* tercepat diantara paramiko dan telnetlib pada protokol *routing* IGP dan EGP . Untuk tabel skenario pengujian dapat dilihat pada tabel 3.4

3.8.2 Waktu Konvergensi IGP Dan EGP

Pengambilan data waktu konvergensi dilakukan karena menjadi faktor penentu suatu paket dapat diteruskan oleh *router* sampai ke tujuan. *Router* dapat melakukan proses *routing* atau meneruskan paket apabila sudah mengetahui informasi rute yang tersimpan dalam tabel *routing*. Jika semua informasi rute sudah

diketahui maka jaringan sudah masuk dalam kondisi konvergen dan *router* dapat meneruskan paket ke tujuan.

Penulis menggunakan perangkat laptop untuk menangkap *traffic* di jalur antara *Client* 1 dan *Router* 3 melalui software wireshark. Data waktu konvergensi IGP diperoleh saat *Client* 1 mengirim paket data ICMP atau melakukan “Ping” ke *Server* dengan kondisi awal berupa “Destination Host Unreachable”. Ketika kondisi sudah “reply” berarti kondisi sudah konvergen. Rentan antara kondisi DHU dan Reply merupakan hasil data untuk waktu konvergensi. Untuk tabel skenario pengujian dapat dilihat pada tabel 3.4

3.8.3 Data Quality of Service (QoS)

Proses pengambilan data yang terakhir adalah pengambilan data QoS. Pengukuran QoS dilakukan untuk mengetahui kinerja dari kedua *routing protocol* yaitu IGP dan EGP setelah diberikan konfigurasi oleh *networkautomation* ubuntu. Pengambilan dilakukan menggunakan laptop penulis dengan menggunakan software wireshark. Pengujian dilakukan dengan melakukan pengiriman data menggunakan FTP protocol dengan mengirimkan *file* film dengan format mkv dari sisi *server* ke *client* 1. Pengujian dilakukan sebanyak 5 kali dengan menggunakan *file* film berukuran 531 Mb. Pengambilan data dilakukan dengan cara melakukan filter *protocol* “*ftp-data*”. Pengukuran QoS ini dilakukan pada *interface client* 1 dan *client* 2. Untuk tabel skenario pengujian dapat dilihat pada tabel 3.5