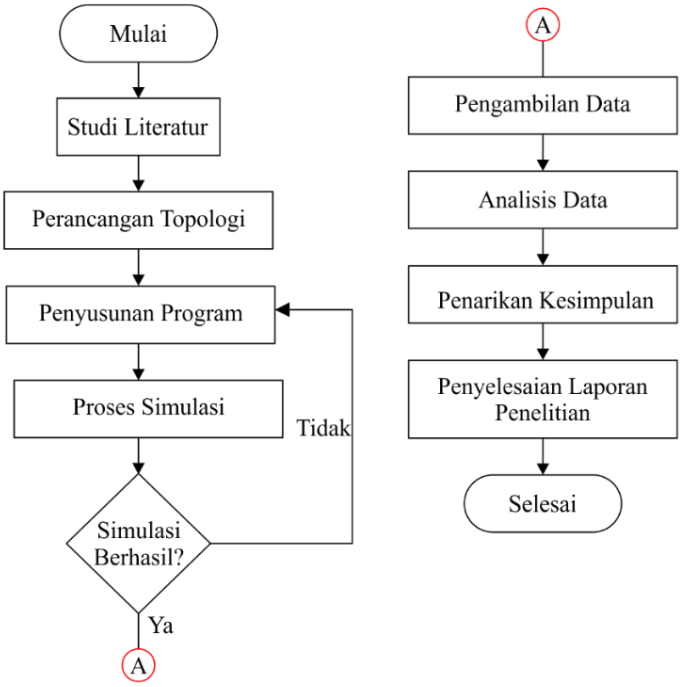
BAB III
METODE PENELITIAN

Pada metodologi penelitian ini mencakup deskripsi diagram alur penelitian, diagram simulasi yang akan dijabarkan melalui *flowchart*, perangkat yang digunakan, rancangan topologi jaringan, konfigurasi perangkat jaringan, proses pengumpulan data, dan skenario pengujian. Diagram alur penelitian menjelaskan langkah-langkah penelitian yang akan dilakukan. Selanjutnya, diagram alur simulasi menggambarkan proses simulasi jaringan dalam penelitian. Dalam proses simulasi, diperlukan perangkat tambahan yang berperan dalam mendukung jalannya riset. penelitian ini juga memerlukan topologi jaringan yang digunakan sebagai objek pengumpulan data untuk analisis penelitian.

3.1 DIAGRAM ALUR PENELITIAN

Proses penelitian akan dilaksanakan melalui serangkaian langkah yang dijelaskan dalam Gambar 3.1. Penelitian ini bertujuan untuk menyimulasikan otomasi jaringan OSPF dan EIGRP berbasis web dengan menggunakan bahasa pemrograman Python. Data yang didapat dari hasil simulasi akan dianalisis dan dibandingkan.



Gambar 3. 1 Diagram Alur Penelitian

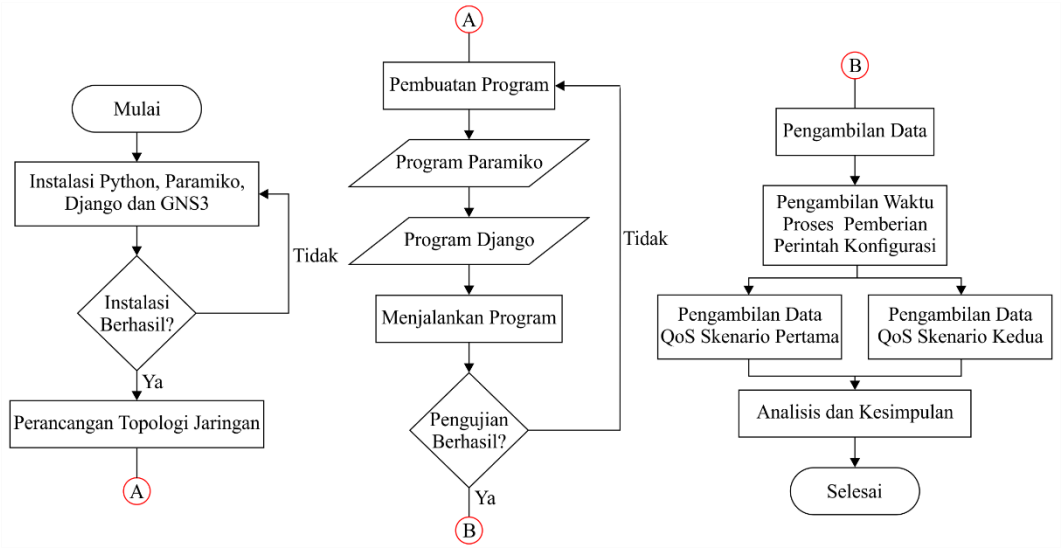
Gambar 3.1 menggambarkan urutan langkah dalam proses penelitian. Langkah awal adalah memahami penelitian sebelumnya sebagai dasar untuk tinjauan pustaka, yang membantu memahami konsep dasar dan menjadi referensi utama penelitian ini. Langkah berikutnya adalah merancang topologi jaringan yang akan digunakan. Dalam topologi ini akan mencakup berbagai perangkat seperti server, router, *switch*, dan *host*. Setelah perencanaan topologi selesai, langkah berikutnya adalah mengembangkan program yang memungkinkan konfigurasi routing OSPF dan EIGRP pada router. Program ini akan ditulis menggunakan bahasa pemrograman Python. Dalam penelitian ini, Python digunakan bersama dengan *framework* Django dan *library* Paramiko. Django digunakan untuk menampilkan halaman web yang dinamis, sedangkan Paramiko digunakan untuk menghubungkan server dengan perangkat jaringan melalui protokol SSH. Proses konfigurasi akan dilakukan di dalam server. Selanjutnya, proses simulasi akan dilakukan menggunakan perangkat lunak pendukung seperti GNS3.

Pengambilan data dilakukan setelah simulasi berhasil dieksekusi. Data akan diperoleh melalui aplikasi Wireshark, yang berfungsi untuk menangkap lalu lintas data saat program dikirim ke router. Selanjutnya, analisis terhadap *throughput*, *packet loss*, *delay*, dan *jitter* akan dilakukan menggunakan *library* Paramiko dan *framework* Django. Hasil analisis tersebut akan dibandingkan. Waktu yang diperlukan oleh program untuk memberikan perintah konfigurasi OSPF dan EIGRP ke setiap router, serta nilai *throughput*, *packet loss*, *delay*, dan *jitter* setelah program dikirim oleh server ke router, akan dicatat untuk menarik kesimpulan. Setelah kesimpulan diperoleh, laporan akan disusun untuk dokumentasi penelitian.

3.2 DIAGRAM ALUR SIMULASI

Diagram alur simulasi dalam penelitian ini melibatkan beberapa tahap utama yaitu instalasi perangkat lunak dan *framework* seperti Python, Paramiko, Django, dan GNS3, perancangan topologi jaringan dalam GNS3 dengan dua skenario topologi yang mencakup routing OSPF dan EIGRP, pengembangan program otomatisasi menggunakan Python yang mengintegrasikan Django dan Paramiko untuk menghubungkan server dengan perangkat jaringan melalui SSH, pelaksanaan simulasi menggunakan GNS3 dengan pengambilan data melalui Wireshark; analisis data untuk mengukur *throughput, packet loss, delay,* dan *jitter*,

serta hasil pengambilan data untuk menyimpulkan performa konfigurasi otomasi OSPF dan EIGRP. Skema penelitian akan diuraikan secara teknis mengenai langkah-langkah simulasi melalui diagram flowchart pada gambar 3.2.



Gambar 3. 2 Diagram Alur Simulasi

Tahapan pertama yang dilakukan yaitu melakukan instalisasi python, paramiko, Django dan GNS3. Bahasa pemograman python digunakan dalam penelitian ini dikarenakan bersifat *open source* dan memiliki banyak *library* yang bisa digunakan dalam keperluan pengembangan, selain itu python juga memiliki sintaks yang sederhana dan cukup mudah untuk dipahami. *Library* paramiko disini berfungsi untuk dapat menerapkan otomasi jaringan pada router menggunakan koneksi SSH. Instalasi Django diperlukan sebagai pengembangan otomasi jaringan berbasis web menggunakan bahasa pemograman python sehingga dalam proses otomasi akan lebih dinamis. Aplikasi GNS3 adalah emulator jaringan yang dipilih untuk digunakan dalam proses simulasi pada penelitian ini. Proses perancangan topologi akan dilakukan didalam *emulator* GNS3. Setelah proses instalasi selesai. Selanjutnya adalah membuat topologi jaringan pada GNS3, terdiri dari dua skenario topologi, yakni topologi router Cisco yang menggunakan routing OSPF dan EIGRP. Setiap skenario topologi akan memiliki 1 server, 1 *switch*, 20 router, dan 4 *client*. Setelah topologi selesai dibangun di GNS3, selanjutnya 20 router akan dikonfigurasi untuk menjalankan otomasi jaringan menggunakan *library* Paramiko dan *framework* Django. Routing OSPF dan EIGRP akan diterapkan melalui konfigurasi secara otomasi melalui *library* paramiko dan Django. Pertama akan

dilakukan program otomasi OSPF dan EIGRP untuk menjalankan proses otomasi jaringan dengan menggunakan *library* paramiko. Setelah program konfigurasi berhasil maka akan diteruskan dengan otomasi routing OSPF dan EIGRP melalui web menggunakan Django.

Tahap berikutnya, setelah program konfigurasi berhasil, adalah melakukan pengambilan data. Proses ini akan melibatkan perangkat lunak Wireshark untuk mengukur waktu yang diperlukan oleh otomasi jaringan dalam melakukan konfigurasi OSPF dan EIGRP sampai router dan klien terhubung. Selanjutnya, akan diambil nilai parameter QoS seperti *throughput*, *packet loss*, *delay*, dan *jitter* menggunakan *library* Paramiko dan *framework* Django. Perbandingan parameter QoS antara dua skenario yang berbeda juga akan dilakukan. Setelah semua proses pengambilan data selesai, kesimpulan dan analisis terhadap pengujian simulasi akan dilakukan.

Proses otomasi jaringan diawali dengan mengidentifikasi, yaitu konfigurasi router menggunakan protokol OSPF dan EIGRP. *Framework* yang digunakan yaitu Django sebagai *framework web* untuk antarmuka pengguna dan Paramiko sebagai *library* Python untuk mengelola koneksi SSH ke perangkat jaringan. Desain sistem otomasi mencakup pembuatan antarmuka web dengan Django yang memungkinkan pengguna mengelola konfigurasi router dengan mudah, serta penulisan skrip Python menggunakan Paramiko untuk mengirim perintah konfigurasi ke router melalui SSH. Pengujian dilakukan dengan mengukur waktu yang dibutuhkan untuk menjalankan perintah konfigurasi dan parameter QoS seperti *throughput, packet loss, delay,* dan *jitter*.

3.3 ALAT YANG DIGUNAKAN

Penelitian ini dirancang menggunakan pemodelan dan simulasi untuk menganalisis performa routing OSPF dan EIGRP dengan bantuan emulator jaringan serta bahasa pemrograman Python. Simulasi tersebut memerlukan berbagai perangkat yang mendukung pembuatan program, yang dibagi menjadi dua komponen utama: perangkat keras (*hardware*) dan perangkat lunak (*software*).

3.3.1 Perangkat Keras (*Hardware*)

Dalam perancangan sistem ini, diperlukan perangkat keras untuk merealisasikan sistem yang akan dibangun dalam penelitian dengan baik. Penelitian ini memanfaatkan satu unit PC/Server.

a. *Personal Computer* (PC) / Server

Dalam penelitian ini, digunakan satu unit PC dengan sistem operasi Windows 11 64-bit yang digunakan sebagai server. PC tersebut akan dipasangi perangkat lunak yang diperlukan untuk mendukung penelitian ini. Rincian spesifikasi lengkap dari PC yang digunakan dapat ditemukan dalam tabel 3.1.

Tabel 3. 1 Spesifikasi Server yang digunakan

Processor	AMD Ryzen 3 3250U with Radeon Graphics (4 CPUs), ~2.6GHz
RAM	20 GB
Memory	256 GB

3.3.2 Perangkat Lunak (*Software*)

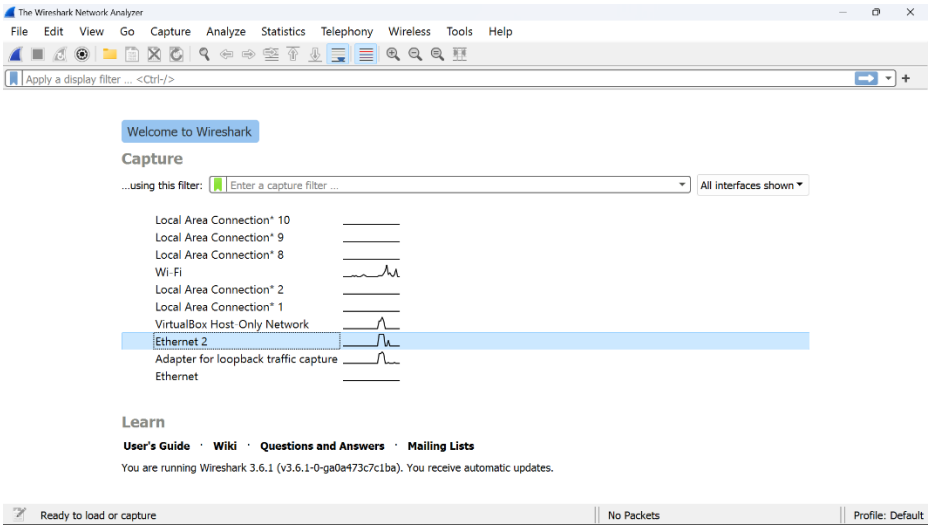
Selain *hardware*, *software* juga menjadi kebutuhan yang penting dan akan diinstal di PC yang digunakan dalam penelitian ini. Beberapa perangkat lunak yang akan digunakan meliputi:

a. GNS3

Untuk menjalankan virtualisasi perangkat jaringan dalam jumlah yang signifikan, diperlukan sumber daya yang besar. GNS3 digunakan sebagai alternatif untuk memungkinkan perangkat jaringan berfungsi secara optimal tanpa membebani CPU pada server fisik dan mengurangi risiko kehilangan *workspace*. Topologi yang digunakan dalam penelitian ini dibuat menggunakan GNS3, yang diakses langsung oleh komputer melalui antarmuka virtual. Hampir semua proses simulasi dilakukan menggunakan GNS3.

b. Wireshark

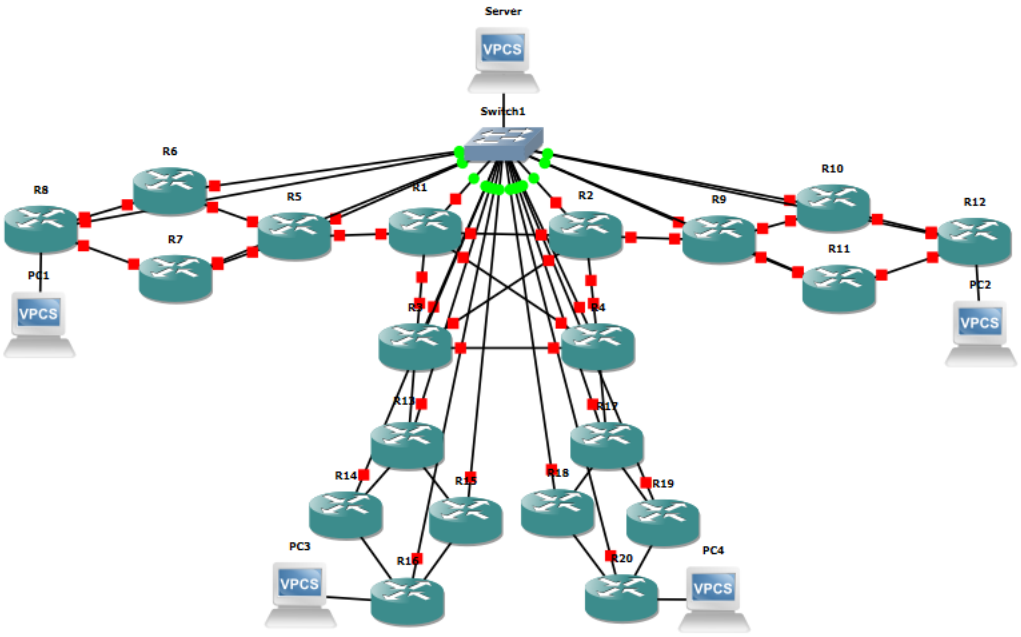
Wireshark merupakan sebuah aplikasi *open source* yang digunakan untuk memindai dan menangkap data di jaringan. Dalam penelitian ini, Wireshark menangkap data paket yang dikirim di jaringan untuk menganalisis performa protokol seperti OSPF dan EIGRP, mencatat *timestamp* dari paket pertama hingga terakhir saat konfigurasi untuk menghitung durasi, dan membantu menguji parameter QoS seperti *throughput*, *packet loss*, *delay*, dan *jitter*. Gambar 3.3 memperlihatkan tampilan awal aplikasi Wireshark.



Gambar 3. 3 Tampilan Awal Wireshark

3.4 RANCANGAN TOPOLOGI

Untuk membuat otomasi jaringan, diperlukan rancangan topologi yang tepat. Pada penelitian ini menggunakan sebuah model topologi *star*, dimana dalam topologi ini terdapat satu perangkat pusat (*switch*) yang terhubung ke beberapa perangkat router, topologi ini digunakan karena mudah dalam manajemen jaringan. Pada topologi ini akan diterapkan routing OSPF dan EIGRP dengan perangkat jaringan yang terdiri dari 20 router, 1 *switch*, 1 server, dan 4 *host*. Rancangan topologi ini dapat dilihat pada gambar 3.4.



Gambar 3. 4 Topologi Jaringan yang digunakan

Pada gambar 3.4, terdapat sebuah server (laptop) yang dihubungkan langsung ke *switch*, sehingga dapat terhubung dengan semua router di jaringan selama proses otomasi jaringan. Semua proses otomasi dilakukan didalam server, proses konfigurasi otomasi, instalasi python, *remote* router lewat SSH dan pengujian jaringan semua dilakukan pengontrolan terpusat pada server. Untuk dapat memastikan konfigurasi jaringan OSPF dan EIGRP berhasil atau tidak maka perlu dilakukan pengiriman paket ICMP yang dilakukan oleh masing-masing *host*. Setiap perangkat yang terhubung ke jaringan memerlukan konfigurasi alamat IP sebagai identitasnya. Detail konfigurasi alamat IP untuk setiap perangkat jaringan dapat dilihat pada tabel 3.2.

Tabel 3. 2 Alamat IP pada Router Cisco

Perangkat	***Interface***	**Alamat IP**	**Perangkat**	***Interface***	**Alamat IP**
R1	F0/0	10.10.10.2/24	R11	F0/0	10.10.10.12/24
	F1/0	20.20.20.1/29		F1/0	192.168.10.2/29
	F1/1	30.30.30.1/29		F1/1	192.168.12.1/29
	F2/0	60.60.60.1/29	R12	F0/0	10.10.10.13/24
	F2/1	192.168.1.1/29		F1/0	192.168.11.2/29
R2	F0/0	10.10.10.3/24		F1/1	192.168.12.2/29
	F1/0	20.20.20.2/29		F2/0	192.168.60.1/24
	F1/1	40.40.40.1/29	R13	F0/0	10.10.10.14/24
	F2/0	70.70.70.1/29		F1/0	192.168.3.2/29
	F2/1	192.168.2.1/29		F1/1	192.168.13.1/29
R3	F0/0	10.10.10.4/24		F2/0	192.168.14.1/29
	F1/0	30.30.30.2/29	R14	F0/0	10.10.10.15/24
	F1/1	50.50.50.1/29		F1/0	192.168.13.2/29
	F2/0	70.70.70.2/29		F1/1	192.168.15.1/29
	F2/1	192.168.3.1/29	R15	F0/0	10.10.10.16/24
R4	F0/0	10.10.10.5/24		F1/0	192.168.14.2/29
	F1/0	40.40.40.2/29		F1/1	192.168.16.1/29
	F1/1	50.50.50.2/29	R16	F0/0	10.10.10.17/24
	F2/0	60.60.60.2/29		F1/0	192.168.15.2/29
	F2/1	192.168.4.1/29		F1/1	192.168.16.2/29
R5	F0/0	10.10.10.6/24		F2/0	192.168.70.1/24
	F1/0	192.168.1.2/29	R17	F0/0	10.10.10.18/24
	F1/1	192.168.5.1/29		F1/0	192.168.4.2/29
	F2/0	192.168.6.1/29		F1/1	192.168.17.1/29
R6	F0/0	10.10.10.7/24		F2/0	192.168.18.1/29
	F1/0	192.168.5.2/29	R18	F0/0	10.10.10.19/24
	F1/1	192.168.7.1/29		F1/0	192.168.17.2/29
R7	F0/0	10.10.10.8/24		F1/1	192.168.19.1/29
	F1/0	192.168.6.2/29	R19	F0/0	10.10.10.20/24
	F1/1	192.168.8.1/29		F1/0	192.168.18.2/29
R8	F0/0	10.10.10.9/24		F1/1	192.168.20.1/29
	F1/0	192.168.7.2/29	R20	F0/0	10.10.10.21/24
	F1/1	192.168.8.2/29		F1/0	192.168.19.2/29
	F2/0	192.168.50.1/24		F1/1	192.168.20.2/29
R9	F0/0	10.10.10.10/24		F2/0	192.168.80.1/24
	F1/0	192.168.2.2/29	*Host* 1	E0	192.168.50.2/24

Perangkat	*Interface*	Alamat IP	Perangkat	*Interface*	Alamat IP
	F1/1	192.168.9.1/29	*Host* 2	E0	192.168.60.2/24
	F2/0	192.168.10.1/29	*Host* 3	E0	192.168.70.2/24
R10	F0/0	10.10.10.11/24	*Host* 4	E0	192.168.80.2/24
	F1/0	192.168.9.2/29	Server	F0	10.10.10.1/24
	F1/1	192.168.11.1/29			

Pada *interface* router yang terhubung dengan server otomasi jaringan, alamat IP harus berada dalam jaringan yang sama agar komunikasi dapat berlangsung tanpa perlu melakukan pemilihan rute. Prefix /29 dipilih untuk jaringan 10.10.10.0 karena terdapat 5 perangkat di dalam jaringan tersebut. Prefix /29 sering digunakan dalam jaringan kecil di mana hanya sedikit perangkat yang perlu dihubungkan dalam satu subnet. Hal ini juga dapat digunakan untuk menghemat ruang alamat IP dalam jaringan yang lebih besar dengan memecah jaringan menjadi subnet yang lebih kecil. Dengan menerapkan *subnetting*(proses membagi jaringan IP besar menjadi beberapa *subnet* yang lebih kecil), konfigurasi alamat IP dapat disederhanakan dan penggunaan IP dapat dioptimalkan untuk efisiensi maksimal.

3.5 KONFIGURASI PERANGKAT

Konfigurasi perangkat dalam penelitian ini melibatkan beberapa tahap penting untuk memastikan semua komponen jaringan berfungsi dan berkomunikasi dengan baik. Tahap pertama adalah konfigurasi pada server, termasuk penetapan alamat IP statis dan pembuatan *interface virtual* untuk menghubungkan server dengan router dalam GNS3.

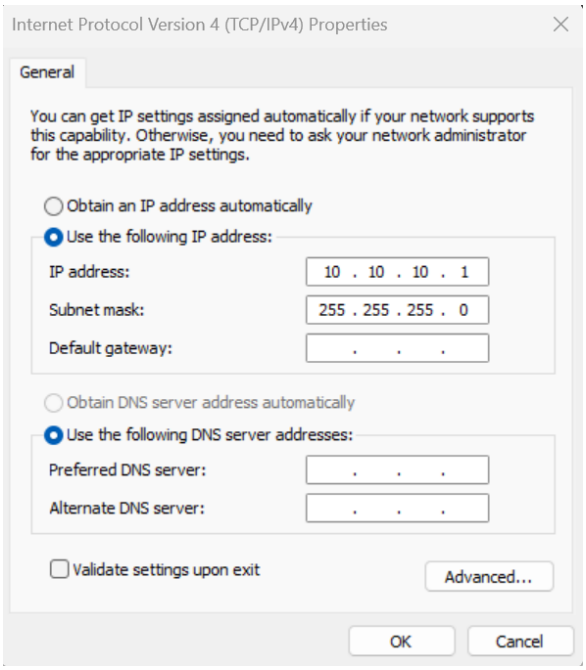
Tahap kedua adalah konfigurasi SSH pada router, yang memungkinkan komunikasi aman antara server dan router, yang melibatkan pengaturan alamat IP, pembuatan *username* dan *password* untuk akses *remote*, pengaturan *domain name*, dan aktivasi SSH.

Tahap terakhir adalah konfigurasi *library* dan *framework* yang digunakan, seperti Paramiko untuk mengirim perintah SSH dari server ke router dan Django untuk membangun aplikasi web yang memudahkan pengelolaan dan pemantauan jaringan. Wireshark juga digunakan untuk menangkap dan menganalisis paket jaringan, membantu mengukur kinerja jaringan seperti *throughput, packet loss, delay,* dan *jitter.*

3.5.1 Konfigurasi Pada Server

Untuk dapat menghubungkan server dengan router yang berada di GNS3 diperlukan *interface virtual* (*interface* jaringan yang digunakan oleh perangkat *virtual* yang diinstal) dikarenakan proses konfigurasi perangkat router diakses pada GNS3. Pengecekan paket ICMP ke masing-masing router yang berada pada GNS3 diperlukan untuk mengetahui semua *host* sudah terhubung ke server.

Gambar 3.5 adalah konfigurasi alamat IP pada server otomasi jaringan melalui *virtual interface* yang dikonfigurasi menggunakan IP *static*. Semua konfigurasi pemberian proses routing, konfigurasi SSH pada router, konfigurasi program paramiko, konfigurasi program Django serta pengambilan data akan diproses melalui server.



Gambar 3. 5 Konfigurasi Alamat IP pada Server

3.5.2 Konfigurasi SSH pada Router

Konfigurasi SSH diterapkan pada dua router Cisco dalam jaringan setelah alamat IP dikonfigurasi pada *interface* yang terhubung dengan server. Selain itu, *username* dan *password* juga dikonfigurasi untuk memungkinkan *remote* akses ke router melalui server. Pada router mikrotik SSH sudah aktif secara *default* sehingga tidak perlu melakukan konfigurasi apapun. Konfigurasi SSH dilakukan untuk memungkinkan program yang menggunakan Paramiko mengirimkan perintah ke setiap router yang terhubung dengan server.

```
R1#enable
R1#configure terminal
R1(config)#interface fa0/0
R1(config-if)#ip address 10.10.10.2 255.255.255.248
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#username yopi password skripsiyopi
R1(config)#username yopi privilege 15
```

Gambar 3. 6 Konfigurasi Alamat IP Router

Pada gambar 3.6 menjelaskan konfigurasi yang diterapkan pada dua router Cisco. Ini mencakup pemberian alamat *IP Address* pada antarmuka yang sesuai, pengaturan *username* dan *password*, serta konfigurasi *privilege 15* pada *username* untuk memungkinkan akses langsung ke *mode privileged* router.

```
R1(config)#ip domain-name ittelkom.local
R1(config)#crypto key generate rsa modulus 1024
R1(config)#line vty 0 15
R1(config-line)#login local
```

Gambar 3. 7 Konfigurasi SSH pada Router

Pada gambar 3.7 dilakukan konfigurasi SSH pada router Cisco untuk memberikan akses kepada *library* Paramiko yang terintegrasi langsung melalui protokol SSH. Hal ini memungkinkan untuk melakukan konfigurasi router sesuai dengan perintah yang diberikan. Konfigurasi *ip domain-name* digunakan untuk menetapkan *DNS domain name*. Sedangkan, konfigurasi *crypto key generate rsa modulus 1024* berfungsi untuk menghasilkan kunci publik dan privat dengan panjang 1024 bit. Konfigurasi *line vty 0 15* digunakan untuk mengaktifkan antarmuka *virtual* sehingga perangkat dapat diakses secara *remote* melalui jaringan.

3.5.3 Konfigurasi Program Paramiko

Konfigurasi program Paramiko disiapkan di dalam server otomasi jaringan menggunakan antarmuka *virtual* pada PC. Program tersebut mencakup konfigurasi alamat IP yang sudah diberi *username* dan *password* oleh admin, serta konfigurasi routing EIGRP yang disimpan dalam *format file Python* (".py"). *Username* dan *password* ini digunakan untuk menjalankan SSH ke router sehingga memungkinkan akses untuk konfigurasi setiap router dalam jaringan.

Konfigurasi yang dijalankan pada program paramiko yaitu pemberian alamat IP untuk setiap *interface* yang terhubung dengan router dan konfigurasi routing EIGRP untuk setiap router. Untuk dapat masuk kedalam router, paramiko

membutuhkan 3 parameter penting agar router tersebut dapat diakses menggunakan SSH yaitu alamat IP, *username* dan *password* pada masing-masing router. Pada router cisco terdapat perintah *time.sleep(1)* yang berfungsi untuk memberikan waktu jeda setiap selesai melakukan perintah. Konfigurasi *script* routing EIGRP dilakukan setelah semua *interface* router sudah diberikan alamat IP masing-masing.

```
import paramiko
import time

ip_address = '10.10.10.2'
username = 'yopi'
password = 'skripsiyopi'

ssh_client = paramiko.SSHClient()
ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
ssh_client.connect(hostname=ip_address,username=username,password=p
assword)

print("Success login to {}".format(ip_address))
conn = ssh_client.invoke_shell()

conn.send("conf t\n")
conn.send("int f1/0\n")
conn.send("ip add 20.20.20.1 255.255.255.248\n")
conn.send("no sh\n")
conn.send("int f1/1\n")
conn.send("ip add 30.30.30.1 255.255.255.248\n")
conn.send("no sh\n")
conn.send("int f2/0\n")
conn.send("ip add 60.60.60.1 255.255.255.248\n")
conn.send("no sh\n")
conn.send("int f2/1\n")
conn.send("ip add 192.168.1.1 255.255.255.0\n")
conn.send("no sh\n")
                                    time.sleep(1)

conn.send("router eigrp 10\n")
conn.send("network 20.20.20.0 0.0.0.7\n")
conn.send("network 30.30.30.0 0.0.0.7\n")
conn.send("network 60.60.60.0 0.0.0.7\n")
conn.send("network 192.168.1.0 0.0.0.7\n")
time.sleep(1)

output = conn.recv(65535)
print(output.decode())

ssh_client.close()
```

3.5.4 Konfigurasi Program Django

Konfigurasi program Django akan diinstal didalam sebuah server otomasi yang dibuat menggunakan bahasa pemograman python, nantinya otomasi jaringan akan dikembangkan yang tadinya hanya menggunakan *library* paramiko sekarang akan berbasis web dengan Django. Pada program Django semua router akan di *remote* menggunakan koneksi SSH, *library* paramiko juga diperlukan untuk dapat me*remote* koneksi SSH pada setiap router. Program Django dapat diakses secara langsung dari server. Untuk dapat membuat *script Network Automation* menggunakan bahasa pemograman python diperlukan sebuah *software code editor* yang dapat berjalan di server, salah satu *code editor* yang dapat digunakan adalah *Visual Studio Code*. *Visual Studio Code* kompatibel dengan banyak bahasa pemograman dan *runtime environment* lain seperti PHP, Python, Java dan .NET.

Untuk dapat memulai projek Django diperlukan *command* agar bisa menjalankan projek otomasi jaringan. Ketika memulai projek pada Django maka Django akan otomatis membuat *directory* dengan nama projek yang diinginkan yaitu skripsi_network_automation. Proses membuat *directory* Django dapat dilihat pada gambar 3.8.

```
D:\#SKRIPSI>django-admin startproject skripsi_network_automation
```

Gambar 3. 8 Membuat *Directory* Django

Setelah *directory* berhasil dibuat, diharuskan untuk masuk ke dalam *directory* skripsi_network_automation yang berhasil dibuat. Selanjutnya akan dibuat *app* pada Django dengan nama network_automation. Django akan secara otomatis membuat *directory app* sesuai instruksi yang diberikan. Gambar 3.10 merupakan proses membuat *directory app* pada Django. Untuk dapat melihat *directory* yang berhasil dibuat oleh Django dapat dibuka menggunakan *Visual Studio Code* untuk proses membuat aplikasi otomasi jaringan. Untuk dapat menjalankan projek Django, perlu dijalankan terlebih dahulu, dapat dilihat pada gambar 3.9.

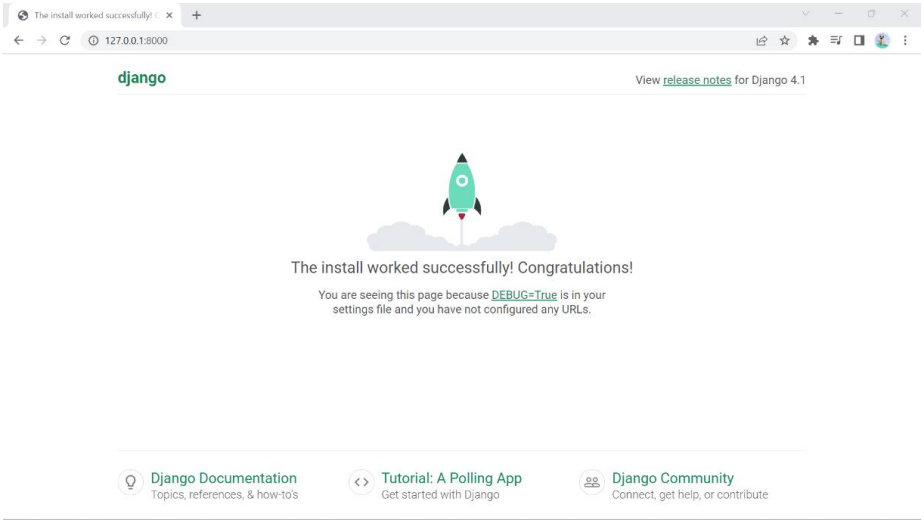
```
D:\#SKRIPSI>cd skripsi_network_automation
D:\#SKRIPSI\skripsi_network_automation>python manage.py startapp
network_automation
```

Gambar 3. 9 Membuat *App* pada Django

```
D:\#SKRIPSI\skripsi_network_automation>python manage.py runserver
```

Gambar 3. 10 Menjalankan *Project* Django

Aplikasi otomasi jaringan memiliki tampilan *frontend* yang dibangun menggunakan *framework* CSS *Bootstrap*. Penggunaan *Bootstrap* akan mempermudah pembuatan tampilan aplikasi Skripsi *Network Automation* agar lebih responsif, dengan mengintegrasikan teknologi HTML, JavaScript, dan CSS.



Gambar 3. 11 Tampilan Awal *Dashboard* Django

Pada gambar 3.11 menampilkan tampilan awal halaman *dashboard* dari aplikasi Django yang berfungsi sebagai antarmuka pengguna yang menyediakan berbagai fitur dan informasi penting untuk mengelola jaringan. Halaman *dashboard* tersebut dibuat menggunakan *template* html. Untuk mengakses halaman *dashboard*, gunakan alamat 127.0.0.1:8000 pada server. Halaman *dashboard* akan menampilkan total perangkat router, jumlah perangkat router Cisco, dan jumlah perangkat router MikroTik yang sebelumnya telah ditambahkan melalui admin. Selain itu, pada halaman *dashboard* juga terdapat bagian "*Last Event*" yang menampilkan sepuluh aktivitas terakhir yang dilakukan oleh admin.

```
{% extends "base.html" %}
{% block content %}
   <h2 class="mt-4">Device List</h2>
   <table class="table">
      <tr>
         <th>IP Address</th>
         <th>Hostname</th>
         <th>Vendor</th>
      </tr>
      {% for device in all_device %}
      <tr>
            <td>{{ device.ip_address }}</td>
            <td>{{ device.hostname }}</td>
            <td>{{ device.vendor }}</td>
```

```
            </tr>
        {% endfor %}
    </table>
{% endblock content %}
```

Pada penulisan *script* terdapat konfigurasi menu "*Device List*" yang akan menampilkan semua perangkat jaringan, seperti router Cisco dan MikroTik, yang telah ditambahkan oleh admin melalui *backend* . Menu "*Device List*" menampilkan informasi perangkat jaringan berupa alamat *IP Address*, *Host*name, dan *Vendor*.

```
{% extends "base.html" %}
{% block content %}
    <h1>{{mode}}</h1>
    <h1></h1>
    <form method="POST">
        {% csrf_token %}
        <h4 class="mt-3">Chose Target:</h4>
        {% for device in devices %}
            <input type="checkbox" name="device" value="{{ device.id
}}">{{device.ip_address}} - {{device.vendor}}<br>
        {% endfor %}
        <h4 class="mt-3">Mikrotik Command</h4>
        <textarea class="form-control" rows="5"
name="mikrotik_command"></textarea>
        <h4 class="mt-3">Cisco Command</h4>
        <textarea class="form-control" rows="5"
name="cisco_command"></textarea>
        <button type="submit" class="mt-3 btn btn-primary">Submit</button>
    </form>
{% endblock content %}
```

Pada penulisan *script* berikutnya terdapat konfigurasi menu "*Configure*" yang memberikan akses kepada admin untuk memilih sejumlah router yang ingin dikonfigurasi. Menu ini memiliki dua isian *text box* yang mewakili setiap router, yaitu Mikrotik dan Cisco *command*. Admin dapat mengisi konfigurasi pada kedua *text box* tersebut secara bersamaan, dan perintah konfigurasi akan otomatis dikirimkan ke perangkat yang dipilih oleh admin. Hasil dari *input* pada menu "*Configure*" akan langsung ditampilkan pada menu "*Log*" dengan hasil *output* yang dapat berupa *success* atau *error*.

```
{% extends "base.html" %}
{% block content %}
    <h1 class="mt-4">Verify Result</h1>
    <pre>       {{result}}
    </pre>
{% endblock content %}
```

Pada penulisan *script* selanjutnya terdapat konfigurasi menu "*Verify Config*" yang memiliki tampilan mirip dengan menu "*Configure*". Menu ini juga menawarkan opsi untuk memilih router yang akan dikonfigurasi dan memiliki *text box* untuk masing-masing perangkat jaringan, baik mikrotik maupun cisco. Namun, berbeda dengan menu "*Configure*" yang hanya menampilkan aktivitas *log*, menu "*Verify Config*" akan menampilkan semua hasil seperti yang ditampilkan di konsol pada masing-masing router. Fungsi dari menu *Verify Config* untuk memastikan bahwa konfigurasi jaringan yang telah diterapkan sesuai dengan yang diharapkan.

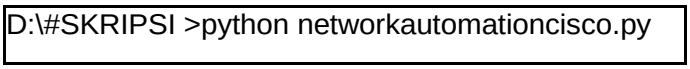
```
{% extends "base.html" %}
{% block content %}
    <h2 class="mt-4">Log</h2>
    <table class="mt-3 table">
        <tr>
            <th>Target</th>
            <th>Action</th>
            <th>Status</th>
            <th>Time</th>
            <th>Messages</th>
        </tr>
        {% for log in logs %}
            <tr>
                <td>{{ log.target }}</td>
                <td>{{ log.action }}</td>
                <td>{{ log.status }}</td>
                <td>{{ log.time }}</td>
                <td>{{ log.messages }}</td>
            </tr>
        {% endfor %}
        </table>
{% endblock content %}
```

Pada penulisan *script* terakhir terdapat konfigurasi menu "*Log*" yang menampilkan semua aktivitas yang dilakukan oleh admin, baik itu *Configure* maupun *Verify Config*. Menu "*Log*" ini juga menampilkan status dari *Configure* maupun *Verify Config*, seperti *success* atau *error*, serta menunjukkan waktu eksekusi.

3.6 PENGUJIAN PARAMIKO

Langkah berikutnya setelah membuat topologi dan mengonfigurasi program adalah menjalankan atau mengeksekusi program tersebut. Pengujian topologi dilakukan menggunakan bahasa pemrograman Python dengan *file* berformat ".py". Pengujian ini bertujuan untuk memastikan *library* paramiko dapat berfungsi dengan

baik untuk melakukan koneksi SSH. Proses pengujian dilakukan secara langsung didalam server otomasi. Dalam penelitian ini, program otomasi jaringan dibangun dengan menggunakan kombinasi *framework* web Django dan *library* Paramiko. Jenis router yang akan digunakan yaitu cisco. Pada router tersebut nantinya akan dibuat program yang bisa mengeksekusi sekaligus dalam satu buah program. Program yang menggunakan router cisco disimpan dengan nama "networkautomationcisco.py". Program dijalankan dengan menggunakan bahasa pemograman Python pada PC server yang dapat dilihat pada gambar 3.12.



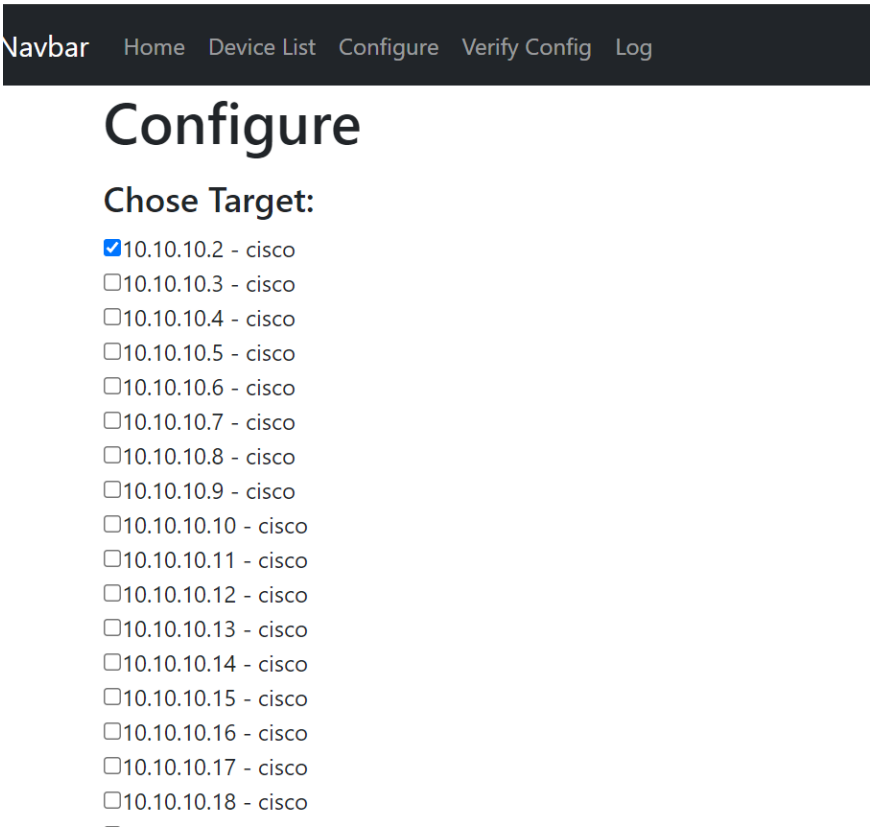
Gambar 3. 12 Menjalankan Program Otomasi pada Cisco

Sebelum dapat melakukan perintah pada program, perlu dilakukan pengecekan router secara manual pada semua router yang saling terhubung. Pengecekan dapat dilakukan pada masing-masing router menggunakan perintah PING. Apabila pengecekan secara manual sudah berhasil melakukan PING, dapat dilanjutkan dengan menggunakan program Python.

Tujuan pengujian *library* Paramiko adalah untuk memverifikasi bahwa Paramiko dapat berfungsi dengan baik dan andal dalam melakukan operasi koneksi SSH, pertukaran data, dan menjalankan perintah pada perangkat jaringan yang terhubung. Selain itu, pengujian juga bertujuan untuk memastikan keamanan protokol SSH yang diimplementasikan oleh Paramiko, kemampuan penanganan kesalahan yang memadai, kinerja yang efisien, kompatibilitas dengan berbagai sistem operasi dan versi Python, serta mampu mengatasi berbagai skenario penggunaan yang kompleks.

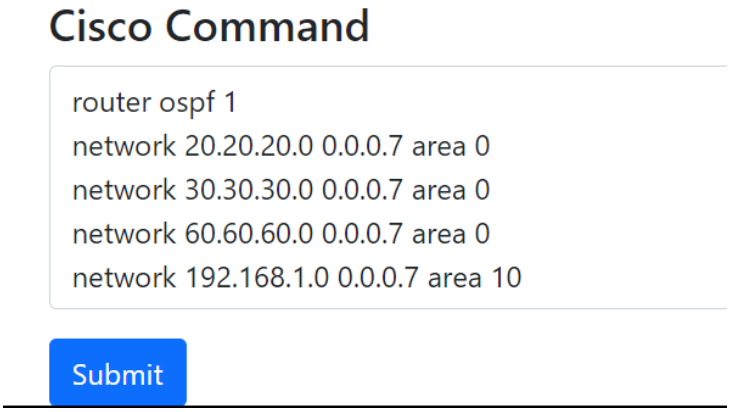
3.7 PEMBERIAN KONFIGURASI ROUTING PADA DJANGO

Untuk dapat melakukan pengambilan data, semua router harus di konfigurasi terlebih menggunakan routing EIGRP maupun OSPF. Proses konfigurasi dimulai dengan memilih router yang akan dikonfigurasi. Admin harus mengetahui alamat IP dari masing-masing router untuk memastikan konfigurasi dilakukan pada perangkat yang tepat. Menu konfigurasi menampilkan daftar router yang tersedia beserta alamat IP mereka. Gambar 3.13 merupakan tampilan dari menu *configure.*



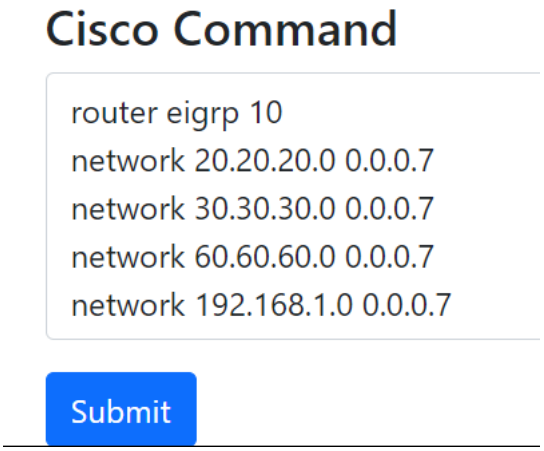
Gambar 3. 13 Tampilan Pada Menu *Configure*

Tahap berikutnya yaitu memberikan konfigurasi terhadap router yang dipilih, pada bagian ini terdapat *text box* untuk memasukkan konfigurasi terhadap router. Perintah konfigurasi tersebut akan secara otomatis dikirimkan ke perangkat yang dipilih melalui protokol SSH yang sebelumnya sudah aktif pada masing-masing router. Gambar 3.14 menunjukkan tentang pemberian konfigurasi routing OSPF pada menu *configure*.



Gambar 3. 14 *Command* Pada Menu *Configure* Untuk Routing OSPF

Gambar 3.15 merupakan tampilan antarmuka web yang digunakan untuk memberikan konfigurasi routing EIGRP pada perangkat jaringan Cisco. Pada tampilan web terdapat *text area* yang digunakan untuk memasukkan perintah-perintah konfigurasi yang akan dikirimkan ke router, perintah yang dimasukkan adalah konfigurasi untuk routing EIGRP. Selain itu terdapat tombol submit yang digunakan untuk mengirim perintah konfigurasi yang telah dimasukkan ke dalam *text area* ke server, yang kemudian akan memproses dan mengirimkan perintah tersebut ke perangkat jaringan.



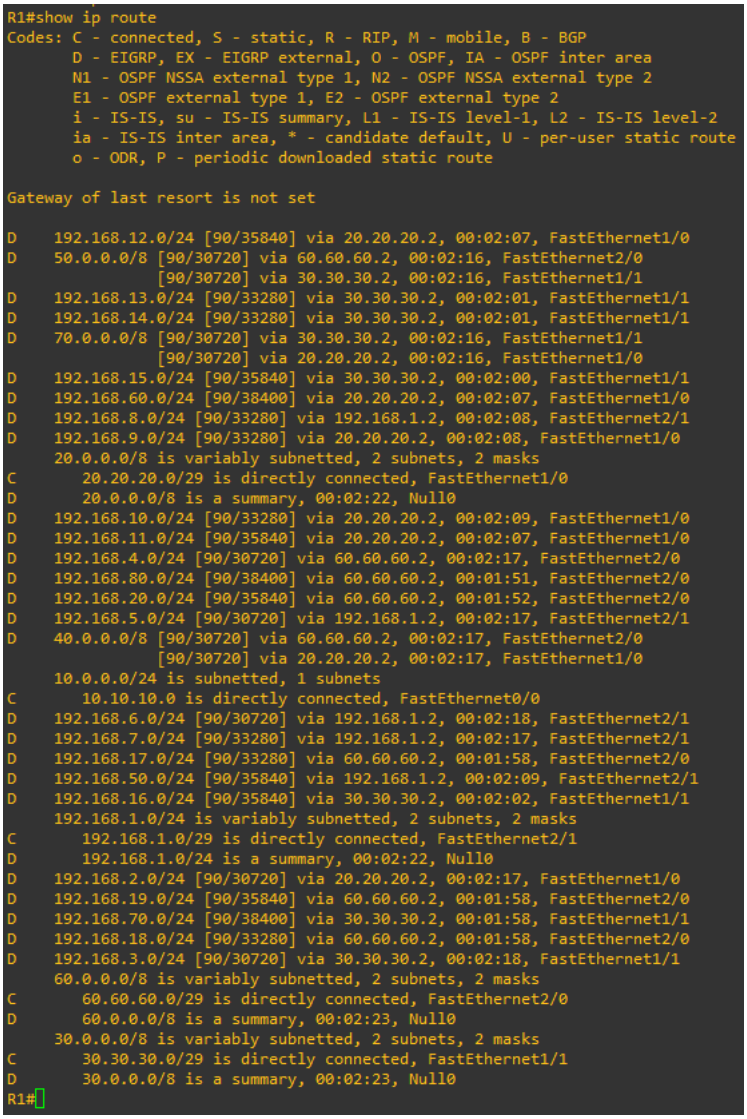
Gambar 3. 15 Command Pada Menu Configure Untuk Routing EIGRP

3.8 VERTIFIKASI ROUTING PROTOKOL

Setelah pemberian konfigurasi routing EIGRP dan OSPF ke masing-masing router berhasil maka diperlukan vertifikasi dengan melakukan perintah PING dari *host* 1 ke *host* lainnya. Tujuan dari verifikasi routing yaitu untuk memastikan bahwa perutean jaringan yang dikonfigurasi dalam sistem komunikasi berjalan dengan benar dan efisien. Verifikasi routing melibatkan pemeriksaan konfigurasi routing untuk memastikan bahwa lalu lintas data dapat diarahkan dengan tepat dari sumber ke tujuan yang diinginkan.

Selain menggunakan perintah PING, proses vertifikasi routing juga dapat dilakukan dengan perintah "*show ip route*" yang dapat dilihat pada gambar 3.16. Perintah ini digunakan untuk menampilkan tabel routing, yang berisi informasi tentang rute-rute yang diketahui oleh router dan cara router mengarahkan trafik ke tujuan tertentu. Berdasarkan gambar menunjukan terdapat dua jenis alamat IP yang terhubung yaitu kode huruf C dan huruf D. Kode yang bertanda C berarti alamat IP

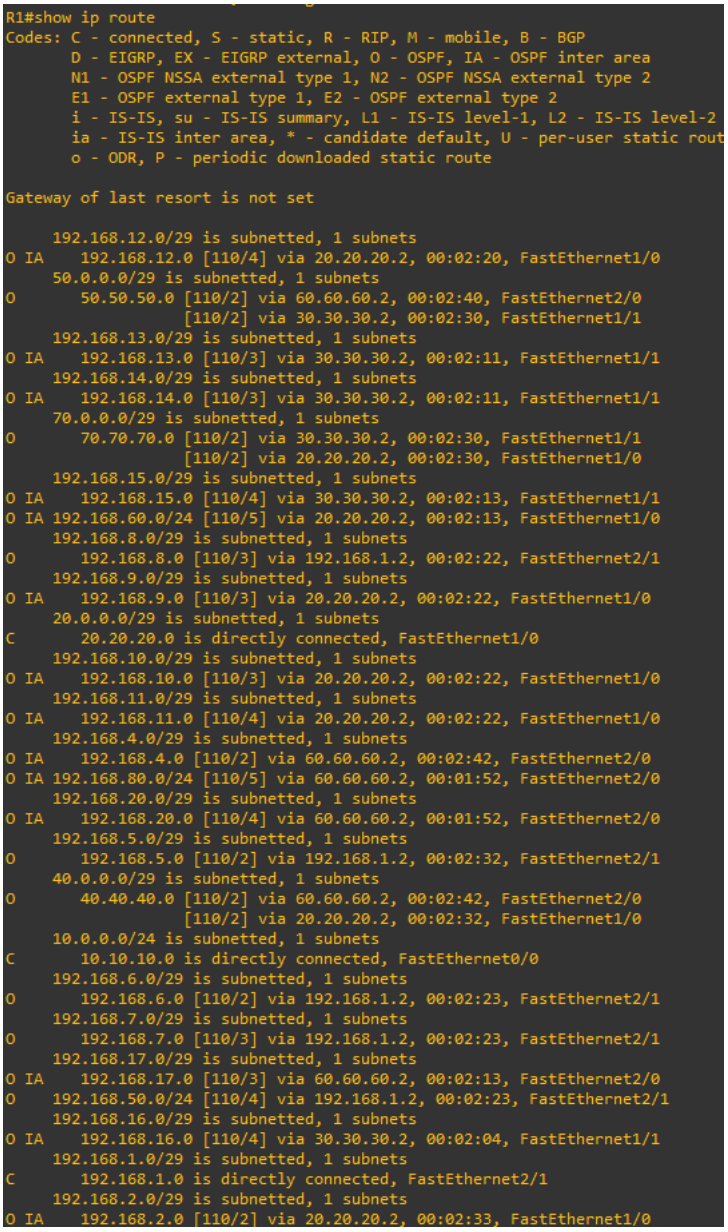
tersebut terhubung secara langsung dengan router, rute ini tercipta karena antarmuka router terhubung langsung ke jaringan tersebut, dan router secara otomatis menambahkan rute ini ke tabel routing sebagai rute yang terhubung (*connected route*). Sedangkan kode D menunjukkan bahwa rute yang didapatkan melalui protokol routing EIGRP.



Gambar 3. 16 Proses Vertifikasi Protokol Routing EIGRP

Gambar 3.17 menunjukkan *tabel routing* OSPF dari perintah *show ip route* yang menampilkan rute yang terhubung secara langsung dengan kode huruf C dan rute OSPF dengan kode huruf O dan IA. Kode O IA menunjukkan bahwa alamat IP tersebut diperoleh melalui OSPF *inter-area*. Pada tabel routing terdapat *prefix* dan *mask* yang menunjukkan jaringan atau *subnet*, nilai *metric* pada tabel routing

digunakan untuk menentukan biaya atau jarak ke rute jaringan, serta *next hop* yang merupakan alamat IP dari router tujuan berikutnya dalam jalur menuju jaringan.



```
R1#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static rout
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

     192.168.12.0/29 is subnetted, 1 subnets
O IA    192.168.12.0 [110/4] via 20.20.20.2, 00:02:20, FastEthernet1/0
     50.0.0.0/29 is subnetted, 1 subnets
O       50.50.50.0 [110/2] via 60.60.60.2, 00:02:40, FastEthernet2/0
                   [110/2] via 30.30.30.2, 00:02:30, FastEthernet1/1
     192.168.13.0/29 is subnetted, 1 subnets
O IA    192.168.13.0 [110/3] via 30.30.30.2, 00:02:11, FastEthernet1/1
     192.168.14.0/29 is subnetted, 1 subnets
O IA    192.168.14.0 [110/3] via 30.30.30.2, 00:02:11, FastEthernet1/1
     70.0.0.0/29 is subnetted, 1 subnets
O       70.70.70.0 [110/2] via 30.30.30.2, 00:02:30, FastEthernet1/1
                   [110/2] via 20.20.20.2, 00:02:30, FastEthernet1/0
     192.168.15.0/29 is subnetted, 1 subnets
O IA    192.168.15.0 [110/4] via 30.30.30.2, 00:02:13, FastEthernet1/1
O IA 192.168.60.0/24 [110/5] via 20.20.20.2, 00:02:13, FastEthernet1/0
     192.168.8.0/29 is subnetted, 1 subnets
O       192.168.8.0 [110/3] via 192.168.1.2, 00:02:22, FastEthernet2/1
     192.168.9.0/29 is subnetted, 1 subnets
O IA    192.168.9.0 [110/3] via 20.20.20.2, 00:02:22, FastEthernet1/0
     20.0.0.0/29 is subnetted, 1 subnets
C       20.20.20.0 is directly connected, FastEthernet1/0
     192.168.10.0/29 is subnetted, 1 subnets
O IA    192.168.10.0 [110/3] via 20.20.20.2, 00:02:22, FastEthernet1/0
     192.168.11.0/29 is subnetted, 1 subnets
O IA    192.168.11.0 [110/4] via 20.20.20.2, 00:02:22, FastEthernet1/0
     192.168.4.0/29 is subnetted, 1 subnets
O IA    192.168.4.0 [110/2] via 60.60.60.2, 00:02:42, FastEthernet2/0
O IA 192.168.80.0/24 [110/5] via 60.60.60.2, 00:01:52, FastEthernet2/0
     192.168.20.0/29 is subnetted, 1 subnets
O IA    192.168.20.0 [110/4] via 60.60.60.2, 00:01:52, FastEthernet2/0
     192.168.5.0/29 is subnetted, 1 subnets
O       192.168.5.0 [110/2] via 192.168.1.2, 00:02:32, FastEthernet2/1
     40.0.0.0/29 is subnetted, 1 subnets
O       40.40.40.0 [110/2] via 60.60.60.2, 00:02:42, FastEthernet2/0
                   [110/2] via 20.20.20.2, 00:02:32, FastEthernet1/0
     10.0.0.0/24 is subnetted, 1 subnets
C       10.10.10.0 is directly connected, FastEthernet0/0
     192.168.6.0/29 is subnetted, 1 subnets
O       192.168.6.0 [110/2] via 192.168.1.2, 00:02:23, FastEthernet2/1
     192.168.7.0/29 is subnetted, 1 subnets
O       192.168.7.0 [110/3] via 192.168.1.2, 00:02:23, FastEthernet2/1
     192.168.17.0/29 is subnetted, 1 subnets
O IA    192.168.17.0 [110/3] via 60.60.60.2, 00:02:13, FastEthernet2/0
O    192.168.50.0/24 [110/4] via 192.168.1.2, 00:02:23, FastEthernet2/1
     192.168.16.0/29 is subnetted, 1 subnets
O IA    192.168.16.0 [110/4] via 30.30.30.2, 00:02:04, FastEthernet1/1
     192.168.1.0/29 is subnetted, 1 subnets
C       192.168.1.0 is directly connected, FastEthernet2/1
     192.168.2.0/29 is subnetted, 1 subnets
O IA    192.168.2.0 [110/2] via 20.20.20.2, 00:02:33, FastEthernet1/0
```

Gambar 3. 17 Proses Vertifikasi Protokol Routing OSPF

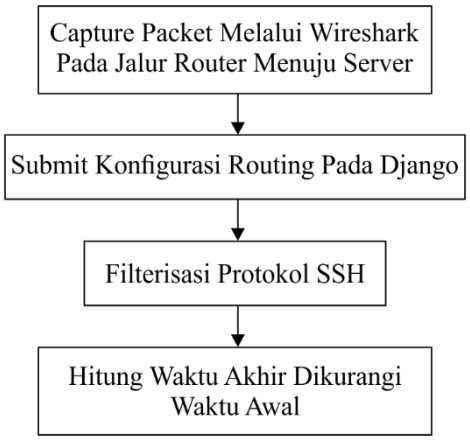
3.9 PENGAMBILAN DATA

Pada tahap pengambilan data ini terbagi menjadi dua bagian yaitu pengambilan waktu proses konfigurasi dan pengambilan data QoS. Pengambilan waktu proses konfigurasi dilakukan pada saat konfigurasi routing dilakukan pada web otomasi. Pengambilan data QoS dapat dihitung dengan cara menangkap lalu lintas pada wireshark, parameter QoS yang diuji dalam penelitian ini meliputi *throughput*, *packet loss*, *delay*, dan *jitter*. Data tersebut akan diperoleh dari setiap

pengujian topologi yang dilakukan, dan hasilnya akan dianalisis untuk perbandingan.

3.9.1 Pengambilan Waktu Perintah Konfigurasi OSPF dan EIGRP

Proses pengambilan waktu dapat dilakukan saat memberikan perintah konfigurasi OSPF dan EIGRP dari web otomasi pada saat proses *submit* dilakukan pada setiap router dalam jaringan dengan menghitung waktu yang tercatat pada aplikasi Wireshark pada jalur router yang menuju ke *switch*. Untuk dapat melihat waktu proses konfigurasi pastikan wireshark sudah berjalan dan tangkap lalu lintas jaringan pada *interface* yang terhubung langsung dengan server. Hasil yang diperoleh dari Wireshark akan difilter untuk protokol SSH (port 22) karena konfigurasi dari Django ke router menggunakan *library* paramiko melalui protokol SSH. Selanjutnya, waktu akhir dikurangi dengan waktu awal saat akses. Pada wireshark gunakan *timestamp* dari paket pertama yang dikirim saat memulai konfigurasi dan paket terakhir yang diterima saat konfigurasi selesai untuk menghitung durasi konfigurasi. Perhitungan ini dilakukan untuk mendapatkan waktu yang dibutuhkan oleh Django dalam memberikan perintah konfigurasi OSPF dan EIGRP ke setiap router. Skema pengambilan waktu proses pemberian konfigurasi dapat dilihat pada gambar 3.18.

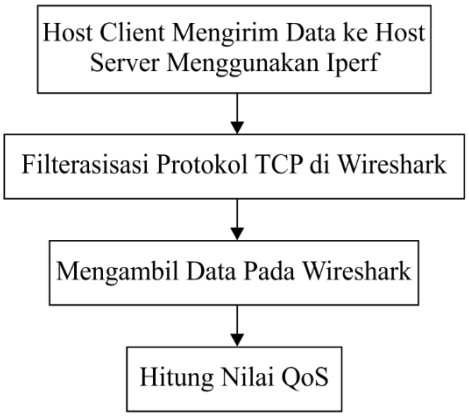


Gambar 3. 18 Alur Pengambilan Waktu Proses Konfigurasi

3.9.2 Pengambilan Data *Quality of Service* (QoS)

Hasil dari penangkapan waktu pada Wireshark yang menuju ke server akan digunakan sebagai perhitungan nilai *throughput*, *packet loss*, *delay* dan *jitter*.

Pengambilan data ini dilakukan dengan mengirimkan paket data TCP dari *host* 1 (pengirim) ke *host* 2 (penerima), wireshark pada server akan mencatat jumlah data yang dikirim dan waktu pengiriman pada saat proses mengirimkan paket data TCP. Dengan mengetahui data tersebut maka perhitungan *throughput*, *packet loss*, *delay* dan *jitter* dapat dilakukan. Untuk menghitung parameter *Quality of Service* (QoS) seperti *throughput, packet loss, delay*, dan *jitter*, dapat menggunakan alat seperti *iperf* untuk mengirim data dan Wireshark untuk menangkap lalu lintas jaringan, kemudian menganalisis data tersebut dengan menghitung *throughput* sebagai jumlah data yang diterima dibagi durasi pengujian, *packet loss* sebagai persentase paket yang hilang dari total yang dikirim, *delay* sebagai waktu pengiriman hingga penerimaan setiap paket, dan *jitter* sebagai variasi *delay* antar paket. Untuk dapat mempermudah perhitungan, data yang diambil dari Wireshark nantinya akan disimpan dalam *format file* CSV dan dapat diakses melalui Excel. Skema pengambilan data nilai *throughput* dan *delay* dapat dilihat pada gambar 3.19.



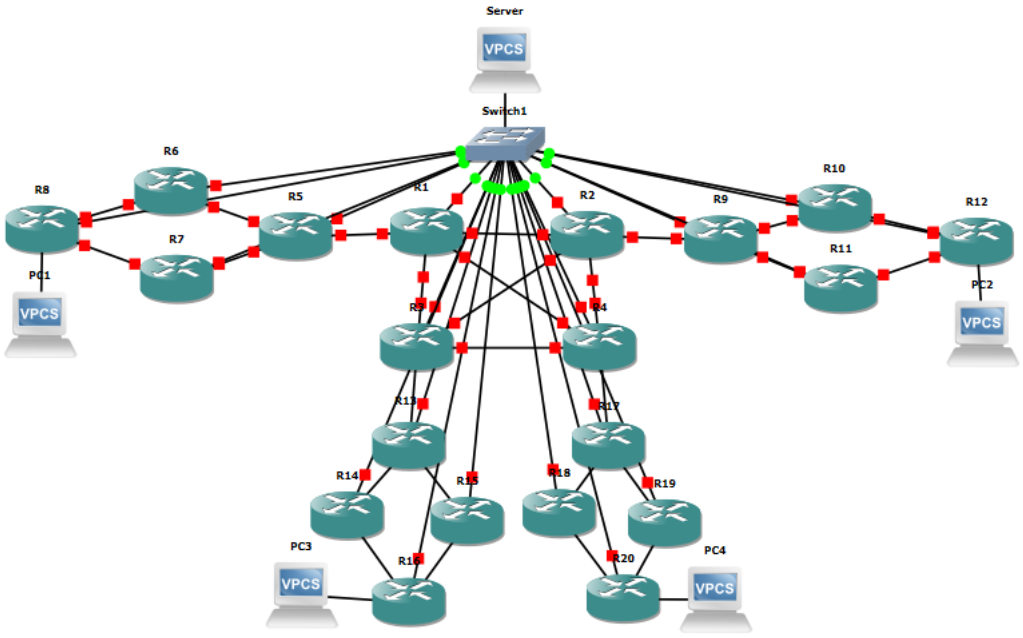
Gambar 3. 19 Alur Pengambilan Nilai QoS

3.10 SKENARIO PENGUJIAN

Pada penelitian ini, skenario pengujian digunakan sebagai acuan saat melakukan proses pengujian dan simulasi untuk mempelajari optimasi protokol routing EIGRP dan OSPF. Skenario pengujian yang disusun akan terdiri dari beberapa skenario menggunakan parameter QoS seperti *throughput*, *packet loss, delay* dan *jitter*. Berdasarkan parameter tersebut, maka akan dibuat dua skenario pengujian sebagai berikut:

3.10.1 Skenario Pengujian Pertama

Skenario pengujian pertama akan dilakukan dengan kondisi normal tanpa ada gangguan maupun pemutusan jaringan terhadap routing protokol EIGRP maupun OSPF. Pengujian dilakukan dengan mengirimkan paket data protokol TCP dari *host* pengirim ke *host* tujuan. Pengujian skenario pertama ini menggunakan perintah *iperf* dengan durasi pengujian selama 180 detik dan proses pengujian ini akan direkam oleh wireshark. Langkah pertama melakukan persiapan pengujian, termasuk instalasi *iperf* dan Wireshark pada *host* pengirim dan penerima, serta konfigurasi jaringan dengan EIGRP dan OSPF. Selanjutnya, *iperf server* dijalankan di *host* tujuan, dan *iperf client* dijalankan di *host* pengirim dengan durasi 180 detik. Wireshark digunakan untuk menangkap lalu lintas jaringan selama pengujian berlangsung. Setelah pengujian selesai, hasil tangkapan Wireshark dianalisis untuk mengukur performa jaringan, termasuk *throughput, packet loss, delay* dan *jitter* yang kemudian didokumentasikan untuk membandingkan performa antara protokol routing EIGRP dan OSPF. Gambar 3.20 merupakan topologi pada skenario pertama.

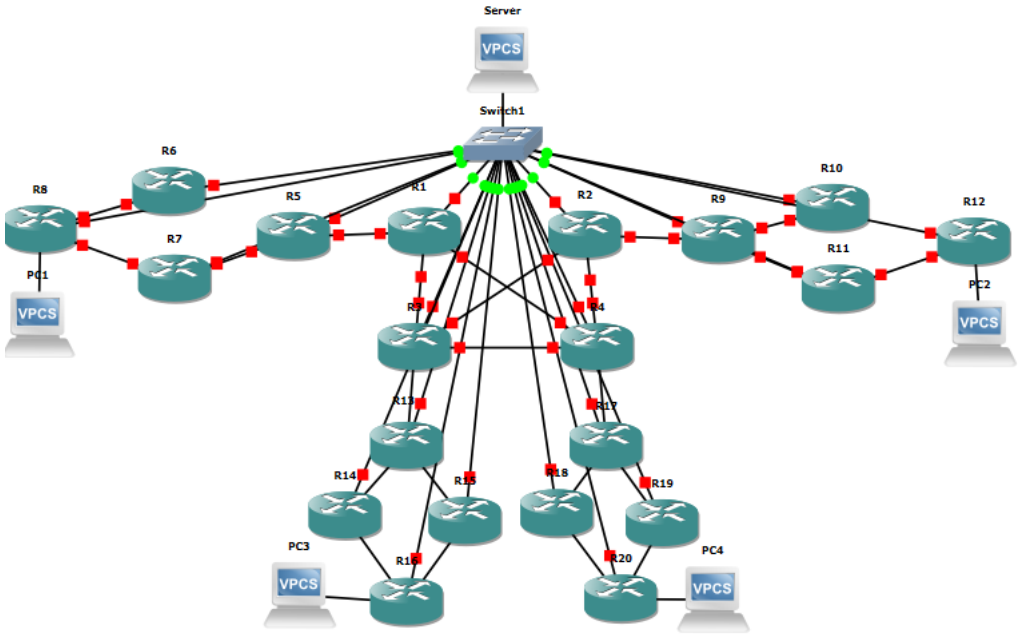


Gambar 3. 20 Topologi pada Skenario Pengujian Pertama

3.10.2 Skenario Pengujian Kedua

Skenario pengujian akan dilakukan dengan cara yang sama dengan skenario sebelumnya, namun pada pengujian kedua ini akan diberikan gangguan terhadap

jaringan EIGRP maupun OSPF pada saat pengiriman paket dari *host* pengirim ke *host* tujuan yaitu dengan memutuskan jalur *interface* utama pada router. Jumlah jalur *interface* yang akan diputus pada pengujian ini yaitu tiga *interface*. Tiga jalur *interface* yang diputus tersebut yaitu jalur R12 yang menuju ke R10, jalur R2 menuju R1 dan jalur R5 menuju R6. Fungsi dari pemutusan ketiga jalur *interface* ini yaitu untuk melihat seberapa cepat protokol routing EIGRP dan OSPF dapat menemukan rute alternatif ketika jalur utama terganggu serta melihat seberapa stabil dan efektif protokol routing dalam menghadapi gangguan. Langkah pertama dalam persiapan pengujian meliputi instalasi *iperf* dan Wireshark pada *host* pengirim dan penerima, serta konfigurasi jaringan dengan protokol EIGRP dan OSPF. Selanjutnya, *iperf server* dijalankan di *host* tujuan, sementara *iperf client* dijalankan di *host* pengirim dengan durasi pengujian selama 180 detik. Wireshark digunakan untuk menangkap lalu lintas jaringan selama pengujian berlangsung. Setelah pengujian selesai, hasil tangkapan dari Wireshark dianalisis untuk mengukur performa jaringan, termasuk *throughput*, *packet loss*, *delay*, dan *jitter*. Hasil analisis ini kemudian didokumentasikan untuk membandingkan performa antara protokol routing EIGRP dan OSPF setelah mengalami gangguan. Gambar 3.21 merupakan topologi pada skenario kedua.



Gambar 3. 21 Topologi pada Skenario Pengujian Kedua