JURNAL LITEK : Jurnal Listrik Telekomunikasi Elektronika, Vol. 21, No. 2, September 2024, pp. 59~68
pISSN: 1693-8097; eISSN: 2549-8762

ANALISIS PERFORMANSI OTOMASI JARINGAN PADA *ROUTING* OSPF DAN EIGRP BERBASIS WEB MENGGUNAKAN BAHASA PEMOGRAMAN PYTHON

Yopi Hermawan[1], Eka Wahyudi[2], Jafaruddin Gusti Amri Ginting[3]

[1,2,3)] Prodi S1 Teknik Telekomunikasi Institut Teknologi Telkom Purwokerto
Email: 18101035@ittelkom-pwt.ac.id, ekawahyudi@ittelkom-pwt.ac.id, jafaruddin@ittelkom-pwt.ac.id

Abstrak – Perkembangan perangkat jaringan yang pesat menghadirkan tantangan dalam konfigurasi router. Konfigurasi perangkat jaringan secara manual memerlukan waktu lama dan rentan kesalahan, terutama dalam skala besar. Penelitian ini mengusulkan solusi otomasi jaringan berbasis web menggunakan *framework* Django dan *library* Paramiko, Paramiko digunakan untuk menghubungkan server dengan perangkat jaringan melalui protokol SSH, sementara Django menampilkan halaman web yang dinamis. Penelitian ini mengukur pemberian waktu perintah konfigurasi dan menganalisis parameter *Quality of Service* untuk kedua protokol dalam dua skenario: kondisi jaringan normal dan kondisi gangguan dengan pemutusan tiga jalur *interface* utama pada router. Hasil dari pemberian konfigurasi OSPF dan EIGRP ke router memiliki perbedaan waktu. Konfigurasi EIGRP memerlukan rata-rata 3,438 detik, sedangkan OSPF rata-rata 4,037 detik. Perbedaan waktu ini disebabkan oleh jumlah data konfigurasi OSPF yang lebih besar dibandingkan EIGRP. Hasil dari pengukuran nilai QoS menunjukkan kedua protokol memberikan kinerja yang sangat baik. Pada skenario normal, EIGRP memiliki *throughput* 4,995 Mbps dan OSPF 3,932 Mbps; keduanya memiliki *packet loss* 0,106%; *delay* EIGRP 1,536 ms dan OSPF 1,916 ms; jitter EIGRP 0,001 ms dan OSPF 0,002 ms. Pada skenario gangguan, *throughput* EIGRP turun menjadi 3,148 Mbps dan OSPF 2,37 Mbps; *packet loss* EIGRP 0,106% dan OSPF 0,104%; *delay* EIGRP 2,345 ms dan OSPF 3,280 ms; *jitter* EIGRP 0,001 ms dan OSPF 0,004 ms. Hasil ini menunjukkan bahwa kedua protokol mampu menjaga performa yang sangat baik dalam kondisi gangguan. Secara keseluruhan, EIGRP dan OSPF mampu memberikan kinerja memuaskan dalam mengelola jaringan baik pada kondisi normal maupun saat terjadi gangguan.

Kata-kata kunci: EIGRP, Django, OSPF, Paramiko, SSH.

Abstract *– The rapid development of network devices presents challenges in router configuration. Manual configuration of network devices is time-consuming and prone to errors, especially on a large scale. This research proposes a web-based network automation solution using the Django framework and the Paramiko library. Paramiko is used to connect the server with network devices through the SSH protocol, while Django displays dynamic web pages. This study measures the configuration command execution time and analyzes the Quality of Service parameters for both protocols in two scenarios: normal network conditions and disruption conditions with the disconnection of three main router interfaces. The configuration execution time for OSPF and EIGRP on routers differs. EIGRP configuration requires an average of 3.438 seconds, while OSPF requires an average of 4.037 seconds. This time difference is due to the larger configuration data size for OSPF compared to EIGRP. The QoS measurement results show that both protocols provide excellent performance. In normal scenarios, EIGRP has a throughput of 4.995 Mbps and OSPF 3.932 Mbps; both have a packet loss of 0.106%; EIGRP delay is 1.536 ms and OSPF 1.916 ms; EIGRP jitter is 0.001 ms and OSPF 0.002 ms. In disruption scenarios, EIGRP throughput drops to 3.148 Mbps and OSPF 2.37 Mbps; EIGRP packet loss is 0.106% and OSPF 0.104%; EIGRP delay is 2.345 ms and OSPF 3.280 ms; EIGRP jitter is 0.001 ms and OSPF 0.004 ms. These results indicate that both protocols can maintain excellent performance under disruption conditions. Overall, EIGRP and OSPF can provide satisfactory performance in managing networks under both normal and disruptive conditions.*

Keywords: EIGRP, Django, OSPF, Paramiko, SSH.

I. PENDAHULUAN

Saat ini perkembangan perangkat jaringan terus berkembang secara pesat, sehingga untuk dapat melakukan konfigurasi router akan menjadi sebuah tantangan. Perangkat jaringan dikonfigurasi agar dapat saling terkoneksi satu sama lain menggunakan protokol *routing*, dengan menggunakan protokol *routing* sebuah router dapat menentukan jalur data yang akan dilewati agar sampai ke tujuan. Proses konfigurasi protokol *routing* ini dapat dilakukan secara langsung melalui *port console* atau dapat dilakukan secara jarak jauh (*remote*).

Apabila jumlah perangkat yang akan dikonfigurasi banyak, maka kedua cara tersebut tidak efisien dan membutuhkan waktu yang relatif lama bagi seorang administrator jaringan untuk dapat mengkonfigurasi perangkat.

Dalam konfigurasi secara manual dengan jumlah perangkat yang banyak seringkali terjadinya konfigurasi *error* yang disebabkan oleh kesalahan administrator jaringan dan memerlukan koordinasi yang rumit, sehingga dapat dinilai kurang efektif. Masalah tersebut dapat diatasi dengan diterapkannya otomasi jaringan. Dengan adanya otomasi jaringan seorang administrator jaringan dapat mengerjakan pekerjaan yang rumit tersebut dengan jauh lebih cepat dan terpusat.

Otomasi jaringan merupakan sebuah sistem yang dapat melakukan otomasi konfigurasi terhadap router yang terhubung atau dengan mengenali alamat IP router yang terdaftar pada sistem dengan menggunakan protokol SSH. Untuk dapat merancang otomasi jaringan diperlukan bahasa pemograman tertentu, salah satu bahasa pemograman yang digunakan adalah python. Bahasa pemograman python memiliki sintaks yang cukup sederhana sehingga cukup mudah untuk dimengerti. Untuk dapat menjalankan program tersebut dibutuhkan sebuah komputer yang difungsikan sebagai server untuk dapat mengkonfigurasi semua perangkat yang terhubung ke dalam jaringan. Pada penelitian ini ditujukan untuk dapat mengkonfigurasi router secara otomasi menggunakan *script* python. Untuk dapat melakukan proses otomasi jaringan pada router diperlukan *routing* protokol agar antar router dengan router lainnya dapat terhubung satu sama lain. Penelitian ini menggunakan protokol *routing* OSPF (*Open Shortest Path First*) dan EIGRP (*Enhanced Interior Gateway Routing Protocol*). Proses otomasi jaringan tidak dilakukan secara langsung oleh router, tetapi oleh sebuah server yang menjalankan program Python. Program tersebut bertugas melakukan konfigurasi ke semua perangkat router dalam jaringan. Python memiliki berbagai *library* yang mendukung implementasi otomasi jaringan, salah satunya adalah paramiko. *Library* paramiko memungkinkan untuk menjalankan skrip konfigurasi perangkat jaringan secara langsung, dan skrip tersebut dikirimkan ke perangkat yang akan dikonfigurasi menggunakan protokol SSH.

Selain itu, untuk dapat membuat otomasi jaringan berbasis web diperlukan *web framework* yang dapat bekerja pada bahasa pemograman pyhton. Django adalah sebuah *framework full-stack* yang dirancang untuk membangun aplikasi web. Penggunaan *framework* ini dapat mempercepat proses pembuatan situs web. Dalam Django, terdapat dua bagian utama, yaitu *front-end* dan *back-end. Front-end* merujuk pada bagian dari situs web yang terlihat oleh pengguna, sementara *back-end* mencakup aspek yang terkait dengan *database* dan logika bisnis. Sehingga Django dapat menjadi solusi dalam implementasi otomasi jaringan berbasis web karena dapat dikembangkan menggunakan bahasa pemograman python, selain itu framework Django juga sudah mendukung banyak *library*, selain itu Django dikenal sebagai *web framework* yang cukup popular dikalangan *high-level framework* dikarenakan memiliki banyak *library* yang telah siap digunakan. Dengan menggunakan *framework* dapat membantu membuat otomasi jaringan berbasis web yang lebih dinamis.

II. TINJAUAN PUSTAKA

Penelitian oleh Rheza Adhyatmaka Wiryawan dan Nur Rohman Rosyid[1] dalam jurnal mereka yang berjudul "Pengembangan Aplikasi Otomasi Administrasi Jaringan Berbasis Website Menggunakan Bahasa Pemrograman Python" mengeksplorasi proses pengembangan aplikasi web untuk otomasi administrasi jaringan. Aplikasi ini dibangun menggunakan bahasa pemrograman Python dan memanfaatkan *library* Paramiko. Penelitian ini menerapkan metode *Rapid Application Development* (RAD), yang meliputi empat tahapan utama: identifikasi, perancangan, pembuatan, serta implementasi aplikasi. Pada tahap identifikasi, perangkat keras yang digunakan adalah router Mikrotik dan Cisco, sementara perangkat lunaknya mencakup Ubuntu, Python 3.6.7, *library* Paramiko 2.4, Django, dan GNS3 2.1. Tahap perancangan meliputi desain topologi, model, dan tampilan. Tahap pembuatan aplikasi dilakukan dengan memanfaatkan *web framework* Django. Bagian *backend* sistem dikembangkan dengan Python, sedangkan bagian *frontend* menggunakan HTML, CSS, dan JavaScript. Tahap akhir adalah implementasi. Sebelum diimplementasikan secara nyata, dilakukan simulasi menggunakan GNS3. Dari keempat tahap tersebut, dihasilkan berbagai fitur termasuk konfigurasi *routing*, *restore, backup, setting*, dan VLAN. Pengujian terhadap lima fitur ini dilakukan menggunakan metode *black box testing.*

Penelitian oleh Elin Sylvania Ginting, Suroso, dan Irawan Hadi[2] dalam jurnal yang berjudul "Pengujian Konfigurasi Otomatis Penambahan *Gateway* Pada Virtual Router Menggunakan Aplikasi Otomasi Jaringan Berbasis Web" membahas penerapan otomasi jaringan yang dikembangkan menggunakan bahasa pemrograman Python dan *library* Paramiko untuk keperluan otomasi, serta Django sebagai *framework full-stack*. Penelitian ini bertujuan untuk mengotomatisasi konfigurasi penambahan *gateway* pada perangkat jaringan secara menyeluruh. Otomasi konfigurasi penambahan *gateway* diterapkan pada beberapa virtual router, dan koneksi jaringan diuji menggunakan aplikasi PuTTY.

Penelitian yang dilakukan oleh Kukuh Nugroho, Anggi Dzikri Abrariansyah, dan Syariful Ikhwan[3] dengan judul "Perbandingan Kinerja *Library* Paramiko dan Netmiko Dalam Proses Otomasi Jaringan" membahas perbedaan kinerja antara *Library* Paramiko dan Netmiko dalam pembuatan sistem otomasi jaringan dengan menggunakan metode analisis sistem. Penelitian ini bertujuan untuk mengetahui perbedaan performansi antara *Library* Paramiko dan Netmiko dalam melakukan

konfigurasi. Topologi yang digunakan terdiri dari empat router Cisco yang saling terhubung menggunakan protokol *routing* OSPF, membentuk topologi *partial-mesh* untuk menyediakan mekanisme jalur *backup* dalam pertukaran lalu lintas data. Pengujian dilakukan terhadap empat parameter, yaitu waktu yang dibutuhkan untuk memberikan perintah konfigurasi ke router, waktu konvergensi jaringan OSPF, serta pengukuran *throughput* dan *delay*.

Penelitian yang dilakukan oleh Donny Rahardika dan Niki Ratama[4] mengenai "Implementasi *Network Automation* Untuk Konfigurasi Jaringan Baru Dengan Netmiko" menggunakan metode pengembangan *Network Development Life Cycle* (NDLC) untuk merancang sistem otomasi jaringan. Sistem ini bertujuan untuk membantu *Network Operation Center* dalam melakukan konfigurasi perangkat jaringan baru. Pembuatan otomasi jaringan dalam penelitian tersebut terbagi menjadi dua tahap yaitu persiapan skema jaringan dan persiapan sistem *Network Automation*. Hasil analisis dan perancangan sistem menghasilkan skema jaringan dengan topologi bintang (*star*). Topologi ini dipilih karena memungkinkan konvergensi dari *node* pusat ke setiap *node*. Sistem otomasi jaringan dikembangkan menggunakan *library* Netmiko.

OSPF adalah protokol *routing* yang mendukung kedua jaringan IPv4 dan IPv6, yang menerapkan algoritma *link-state*. OSPF menentukan rute terbaiknya dengan mempertimbangkan status setiap *node* yang menghubungkan sumber danatujuan, termasuk informasi seperti *prefix IP*, *network mask*, jenis jaringan, dan perangkat yang terkoneksi dalam jaringan. Data ini disebarkan melalui *link-state advertisement* (LSA) yang dikumpulkan dalam *link-state database*.

EIGRP merupakan sebuah *routing* protokol yang dikembangkan dan dirancang oleh Cisco Systems, Inc. *Routing* Protokol EIGRP diperkenalkan pada tahun 1993 untuk menggantikan *routing* protokol sebelumnya yaitu IGRP. Algoritma yang digunakan dalam EIGRP disebut *Diffusing Update Algorithm* (DUAL), yang bertujuan untuk menyatukan *control plane* ke dalam set jalur *loop-free*. DUAL memastikan bahwa tidak ada *loop* dalam setiap perhitungan rute pada setiap saat. DUAL memungkinkan semua router yang terlibat dalam perubahan topologi untuk menyinkronkan informasi secara bersamaan.

Throughput merupakan jumlah total kedatangan paket yang berhasil diamati ditujuan selama interval waktu tertentu, yang kemudian dibagi oleh durasi interval waktu tersebut. Walaupun satuan *throughput* sama dengan *bandwidth*, yaitu bps (*bit per second*), *throughput* lebih merepresentasikan *bandwidth* yang sebenarnya pada suatu waktu dan dalam kondisi jaringan tertentu.

TABEL I
Standarisasi *Throughput* menurut TIPHON[20]

Kategori *Throughput*	*Throughput* (bps)	Indeks
Sangat Baik	100	4
Baik	75	3
Cukup	50	2
Buruk	<25	1

Packet loss merupakan keadaan ketika paket-paket data yang dikirim melalui jaringan tidak mencapai tujuan mereka dan hilang selama proses pengiriman. Penyebabnya bisa bermacam-macam, seperti masalah pada koneksi jaringan, kesulitan router atau perangkat jaringan dalam mengelola lalu lintas, atau masalah pada peralatan jaringan.

TABEL II
Standarisasi *Packet Loss* menurut TIPHON[20]

Kategori *Packet loss*	*Packet loss* (%)	Indeks
Sangat Baik	0	4
Baik	3	3
Cukup	15	2
Buruk	25	1

Delay adalah waktu yang diperlukan bagi sebuah paket yang dikirim dari sumber ke tujuan. Dalam proses transmisi paket, delay dapat timbul karena adanya antrian yang panjang atau pemilihan rute alternatif untuk menghindari kemacetan pada jalur.

TABEL III
Standarisasi *Delay* menurut TIPHON[20]

Kategori *Delay*	*Delay* (ms)	Indeks
Sangat baik	<150 ms	4
Baik	150 – 300 ms	3
Cukup	300 – 450 ms	2
Buruk	>450 ms	1

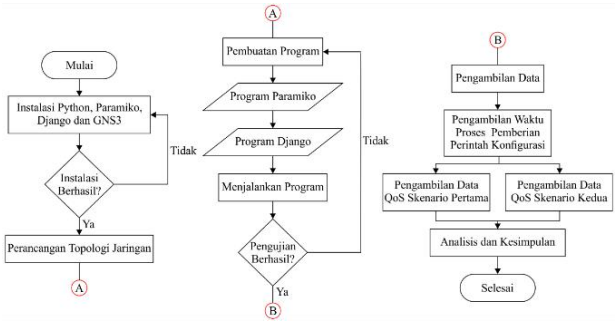
Jitter adalah variasi waktu yang tidak teratur antara kedatangan paket data yang sukses di jaringan. Dalam konteks jaringan komunikasi, *jitter* mengacu pada fluktuasi atau variasi yang tidak diinginkan dalam waktu tiba paket data yang dikirim melalui jaringan. *Jitter* terjadi ketika paket-paket data mengalami penundaan yang bervariasi dalam perjalanan mereka melalui jaringan.

TABEL IV
Standarisasi *Jitter* menurut TIPHON[20]

Kategori *Jitter*	*Jitter* (ms)	Indeks
Sangat baik	0 ms	4
Baik	0 ms s/d 75 ms	3
Cukup	75 ms s/d 125 ms	2
Buruk	125 ms s/d 225 ms	1

III. METODOLOGI

Analisis performansi otomasi jaringan pada *routing* OSPF dan EIGRP berbasis web menggunakan django dapat diuraikan secara teknis mengenai langkah-langkah simulasi pada Gbr. 1. Kedua *routing* tersebut akan dijalankan pada komputer server, sehingga proses konfigurasi dilakukan pada satu komputer secara terpusat. Semua perangkat yang terhubung yang digunakan untuk proses pengambilan data dilakukan pada *software* GNS3.



Gbr. 1 Diagram Alur Simulasi

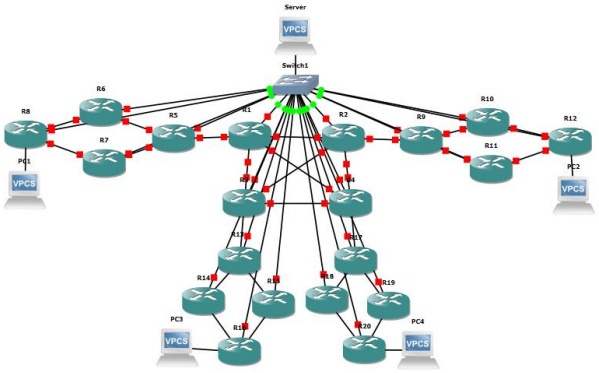
Tahapan pertama yang dilakukan yaitu melakukan instalisasi python, paramiko, Django dan GNS3. Bahasa pemograman python digunakan dalam penelitian ini dikarenakan bersifat *open source* dan memiliki banyak *library* yang bisa digunakan dalam keperluan pengembangan, selain itu python juga memiliki sintaks 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 th*roughput, packet loss, delay*, dan *jitter* menggunakan *library* Paramiko dan *framework* Django. Perbandingan parameter QoS antara dua skenario yang berbeda juga 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*.

A. Rancangan Topologi Jaringan

Untuk membuat otomasi jaringan, diperlukan rancangan topologi yang tepat. Penelitian ini menggunakan *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 Gbr. 2.



Gbr. 2 Rancangan Topologi Jaringan

Pada Gbr. 2 terdapat sebuah server 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 5.

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. Dengan menerapkan *subnetting*, konfigurasi alamat IP dapat disederhanakan dan penggunaan IP dapat dioptimalkan untuk efisiensi maksimal.

TABEL V
Alamat IP Pada Router

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
	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			

B. Konfigurasi SSH Pada Router

Konfigurasi SSH (*Secure Shell*) 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(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
```

Gbr. 3 Konfigurasi SSH Pada Router

Pada Gbr. 3 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.

C. 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.

```
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=password

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()
```

Gbr. 4 Konfigurasi IP dan *Routing* EIGRP

Gbr. 4 merupakan 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.

D. Konfigurasi Program Django

Konfigurasi program Django 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 meremote 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.

```
{% 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 %}
```

Gbr. 5 *Script* Django Pada Menu *Device List*

Pada Gbr. 5 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, Hostname,* 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 %}
```

Gbr. 6 *Script* Django Pada Menu *Configure*

Pada Gbr. 6 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 %}
```

Gbr. 7 *Script* Django Pada Menu *Verify Config*

Pada Gbr. 7 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.

```
{% 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 %}
```

Gbr. 8 *Script* Django Pada Menu *Log*

Pada Gbr. 8 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.

E. Pengambilan Waktu Proses Konfigurasi OSPF dan EIGRP

Proses pengambilan waktu dapat dilakukan saat memberikan perintah konfigurasi OSPF dan EIGRP keasetiap router pada jaringan dengan menghitung waktu yang tercatat pada aplikasi Wireshark pada jalur router yang menuju ke *switch*. Hasil yang diperoleh dari Wireshark akan difilter untuk protokol SSH karena konfigurasi dari Django ke router menggunakan protokol SSH. Selanjutnya, waktu akhir dikurangi dengan waktu awal saat akses. Perhitungan ini dilakukan untuk mendapatkan waktu yang dibutuhkan oleh Django dalam memberikan perintah konfigurasi OSPF dan EIGRP ke setiap router.

F. Pengukuran Nilai *Quality of Service* Skenario 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.

G. Pengukuran Nilai *Quality of Service* Skenario 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*.

IV. HASIL DAN PEMBAHASAN

Penelitian ini bertujuan untuk menganalisis kinerja otomasi jaringan OSPF dan EIGRP berbasis web dengan menggunakan Django. Aplikasi ini dikembangkan dengan dukungan *library* Python dan *framework* Django. Data yang dibutuhkan untuk mengukur kinerja meliputi waktu yang dibutuhkan untuk memberikan perintah konfigurasi OSPF dan EIGRP ke jaringan, serta pengukuran *throughput, packet loss, delay,* dan *jitter*. Pengambilan data dilakukan dengan menangkap lalu lintas jaringan menggunakan aplikasi Wireshark.

A. Hasil Pengukuran Waktu Pemberian Perintah Konfigurasi ke Router

Pengukuran waktu pemberian perintah konfigurasi ke setiap router dilakukan berdasarkan data yang diperoleh dari penangkapan *traffic* di server. Perhitungan ini dilakukan dengan mengukur selisih antara waktu akhir dan waktu awal komunikasi, yang diperoleh dari penangkapan *traffic* untuk menentukan waktu yangadibutuhkan program dalam memberikan perintah konfigurasi ke setiap router. Penangkapan *traffic* dilakukan pada jalur Ethernet 2 di server yang terhubung ke *switch*, dengan menerapkan filterisasi protokol SSH.

Hasil dari pemberian konfigurasi OSPF dan EIGRP ke router pada TABEL VI dan TABEL VII memiliki perbedaan waktu atau selisih. Pada *routing* EIGRP menghasilkan waktu dengan rata-rata 3,438 detik, sedangkan pada *routing* OSPF menghasilkan waktu rata-rata 4,037 detik. Pemberian konfigurasi dengan menggunakan *routing* EIGRP memperoleh waktu yang lebih cepat jika dibandingkan dengan pemberian perintah konfigurasi menggunakan *routing* OSPF. Perbedaan waktu ini dapat disebabkan oleh perbedaan dalam konfigurasi *routing* untuk kedua protokol. Pemberian perintah konfigurasi OSPF memiliki jumlah data konfigurasi yang lebih besar dibandingkan dengan konfigurasi EIGRP.

TABEL VI
Waktu Pemberian Perintah Konfigurasi *Routing* EIGRP

***Routing* EIGRP**	**Waktu Awal (Detik)**	**Waktu Akhir (Detik)**	**Waktu Pemberian Perintah Konfigurasi (Detik)**
R1	0,035	4,659	4,623
R2	0,031	4,674	4,642
R3	0,031	4,649	4,617
R4	0,021	4,565	4,543
R5	0,010	3,700	3,689
R6	0,022	2,673	2,651
R7	0,032	2,692	2,659
R8	0,032	3,671	3,639
R9	0,021	3,637	3,616
R10	0,024	2,653	2,629
R11	0,044	2,658	2,613
R12	0,043	3,752	3,709
R13	0,042	3,671	3,628
R14	0,033	2,678	2,645
R15	0,032	2,657	2,624
R16	0,034	3,694	3,659
R17	0,032	3,667	3,635
R18	0,043	2,708	2,665
R19	0,032	2,647	2,615
R20	0,051	3,704	3,653
Total Waktu			68,764
Rata-rata			3,438

TABEL VII
Waktu Pemberian Perintah Konfigurasi *Routing* OSPF

***Routing* OSPF**	**Waktu Awal (Detik)**	**Waktu Akhir (Detik)**	**Waktu Pemberian Perintah Konfigurasi (Detik)**
R1	0,045	5,050	5,005
R2	0,087	5,199	5,112
R3	0,092	5,166	5,073
R4	0,072	5,013	4,941
R5	0,094	4,198	4,104
R6	0,052	3,081	3,028
R7	0,079	3,319	3,240
R8	0,031	4,145	4,114
R9	0,078	4,347	4,269
R10	0,057	3,262	3,204
R11	0,045	3,166	3,121
R12	0,063	4,116	4,052
R13	0,035	4,135	4,100
R14	0,068	3,465	3,396
R15	0,065	4,571	4,505
R16	0,086	4,441	4,354
R17	0,079	4,425	4,346
R18	0,109	3,379	3,269
R19	0,056	3,412	3,355
R20	0,109	4,251	4,142
Total Waktu			80,740
Rata-rata			4,037

B. Hasil Nilai *Quality of Service* Skenario Pertama

Perbandingan parameter QoS skenario pertama dapat dilihat pada TABEL VIII dengan pengiriman perintah konfigurasi routing EIGRP dan OSPF. 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.

TABEL VIII
Indeks Perbandingan Parameter QoS Skenario Pertama

No	**Parameter QoS**	***Routing* EIGRP**		***Routing* OSPF**	
		Nilai	**Kategori**	**Nilai**	**Kategori**
1.	*Throughput*	4,995 Mbps	Sangat Baik	3,932 Mbps	Sangat Baik
2.	*Packet loss*	0,068 %	Sangat Baik	0,068 %	Sangat Baik
3.	*Delay*	1,536 ms	Sangat Baik	1,916 ms	Sangat Baik
4.	*Jitter*	0,001 ms	Sangat Baik	0,002 ms	Sangat Baik

Pada skenario pertama penelitian ini, dilakukan analisis terhadap parameter *Quality of Service* (QoS) untuk dua protokol *routing*, yaitu EIGRP (*Enhanced Interior Gateway Routing Protocol*) dan OSPF (*Open Shortest Path First*). Pengujian dilakukan dalam kondisi jaringan normal, tanpa adanya gangguan atau pemutusan koneksi pada kedua protokol tersebut. Hasil pengujian memperlihatkan nilai th*roughput, packet loss, delay,* dan *jitter* untuk kedua protokol tersebut.

Dari hasil pengujian, didapatkan bahwa nilai *throughput* untuk *routing* EIGRP adalah sebesar 4,995 Mbps, sedangkan nilai *throughput* untuk *routing* OSPF adalah sebesar 3,932 Mbps. Menurut standar kategori TIPHON, kedua nilai throughput tersebut termasuk dalam kategori "Sangat Baik". Ini menunjukkan bahwa keduanya berhasil memberikan kinerja yang memuaskan dalam mengirimkan data melalui jaringan.

Selanjutnya, nilai *packet loss* yang diukur pada kedua protokol adalah sama, yaitu sebesar 0,106%. Standar kategori TIPHON juga menyatakan bahwa nilai ini termasuk dalam kategori "Sangat Baik". Hal ini menunjukkan bahwa baik *routing* EIGRP maupun OSPF berhasil menjaga keandalan transmisi data dengan jumlah paket yang hilang yang sangat rendah.

Kemudian, dilihat dari nilai *delay*, *routing* EIGRP mendapatkan hasil sebesar 1,536 ms, sementara OSPF memiliki nilai *delay* sebesar 1,916 ms. Kedua nilai ini masuk dalam kategori "Sangat Baik" menurut standar TIPHON. Ini berarti kedua protokol mampu mengirimkan data dengan waktu transmisi yang rendah, sehingga mengurangi waktu tunggu dalam pengiriman paket data.

Terakhir, nilai *jitter* untuk *routing* EIGRP adalah sebesar 0,001 ms, sementara untuk OSPF adalah sebesar 0,002 ms. Kedua nilai ini juga masuk dalam kategori "Sangat Baik" menurut standar TIPHON. Hal ini menunjukkan bahwa kedua protokol berhasil mencapai stabilitas waktu pengiriman data yang sangat baik, dengan variasi waktu pengiriman yang minimal.

Secara keseluruhan, hasil analisis menunjukkan bahwa baik *routing* EIGRP maupun OSPF memberikan kinerja yang sangat baik dalam skenario pengujian normal ini. Kedua protokol berhasil mencapai nilai *throughput, packet loss, delay,* dan *jitter* yang memuaskan sesuai dengan standar kategori TIPHON.

C. Hasil Nilai *Quality of Service* Skenario Kedua

Perbandingan parameter QoS skenario pertama dapat dilihat pada TABEL IX dengan pengiriman perintah konfigurasi *routing* EIGRP dan OSPF.

TABEL IX
Indeks Perbandingan Parameter QoS Skenario Kedua

No	**Parameter QoS**	***Routing* EIGRP**		***Routing* OSPF**	
		Nilai	**Kategori**	**Nilai**	**Kategori**
1.	*Throughput*	3,148 Mbps	Sangat Baik	2,37 Mbps	Sangat Baik
2.	*Packet loss*	0,106 %	Sangat Baik	0,104 %	Sangat Baik
3.	*Delay*	2,345 ms	Sangat Baik	3,280 ms	Sangat Baik
4.	*Jitter*	0,001 ms	Sangat Baik	0,004 ms	Sangat Baik

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*.

Dalam skenario kedua penelitian ini, analisis dilakukan terhadap parameter *Quality of Service* (QoS) untuk dua protokol *routing*, yaitu EIGRP (*Enhanced Interior Gateway Routing Protocol*) dan OSPF (*Open Shortest Path First*). Pengujian dilakukan dengan mengirimkan perintah konfigurasi *routing* menggunakan kedua protokol tersebut, namun dengan adanya gangguan pada jaringan. Gangguan ini terjadi dengan memutuskan tiga jalur *interface* utama pada router selama pengiriman paket dari *host* pengirim ke *host* tujuan.

Hasil pengujian menunjukkan bahwa nilai *throughput* untuk *routing* EIGRP adalah sebesar 3,148 Mbps, sedangkan untuk *routing* OSPF adalah sebesar 2,37 Mbps. Kedua nilai *throughput* tersebut masuk dalam kategori "Sangat Baik" menurut standar TIPHON. Hal ini menunjukkan bahwa baik *routing* EIGRP maupun OSPF mampu menjaga kinerja yang baik dalam pengiriman data meskipun terdapat gangguan pada jaringan.

Selanjutnya, nilai *packet loss* untuk *routing* EIGRP adalah sebesar 0,106%, sementara untuk OSPF adalah sebesar 0,104%. Kedua nilai tersebut juga masuk dalam kategori "Sangat Baik" menurut standar TIPHON. Ini menunjukkan bahwa kedua protokol berhasil menjaga keandalan transmisi data dengan tingkat kehilangan paket yang sangat rendah, bahkan saat menghadapi gangguan pada jalur *interface* utama.

Kemudian, nilai *delay* yang diukur pada *routing* EIGRP adalah sebesar 2,345 ms, sedangkan untuk OSPF adalah sebesar 3,280 ms. Kedua nilai ini termasuk dalam kategori "Sangat Baik" menurut standar TIPHON. Hal ini menunjukkan bahwa kedua protokol mampu mengirimkan data dengan waktu transmisi yang rendah, meskipun menghadapi situasi gangguan pada jaringan.

Terakhir, nilai *jitter* untuk *routing* EIGRP adalah sebesar 0,001 ms, sementara untuk OSPF adalah sebesar 0,004 ms. Kedua nilai jitter ini juga masuk dalam kategori "Sangat Baik" menurut standar TIPHON. Ini menunjukkan bahwa kedua protokol berhasil mencapai stabilitas waktu pengiriman data yang sangat baik, meskipun dihadapkan pada gangguan pada jalur *interface* utama.

Secara keseluruhan, hasil analisis menunjukkan bahwa baik *routing* EIGRP maupun OSPF mampu menjaga performa yang sangat baik dalam mengelola jaringan, bahkan saat menghadapi gangguan pada beberapa jalur *interface* utama.

V. KESIMPULAN

Berdasarkan hasil analisis dan pembahasan data, penelitian dari analisis performansi otomasi jaringan pada *routing* OSPF dan EIGRP berbasis web menggunakan Django diperoleh kesimpulan sebagai berikut:

1. Waktu proses pemberian perintah konfigurasi EIGRP ke router memperoleh waktu dengan rata-rata sebesar 3,438 detik lebih cepat dibandingkan dengan konfigurasi *routing* OSPF yaitu sebesar 4,009 detik.
2. Pada kondisi normal, nilai *throughput* yang dihasilkan oleh *routing* EIGRP yaitu sebesar 4,995 Mbps lebih besar dibandingkan dengan *routing* OSPF yaitu sebesar 3,932 Mbps.
3. Pada kondisi normal, nilai *packet loss* yang dihasilkan oleh *routing* EIGRP dan *routing* OSPF sama yaitu 0,068 %.
4. Secara keseluruhan, *routing* EIGRP maupun OSPF mampu menjaga performa yang sangat baik dalam mengelola jaringan, bahkan saat menghadapi gangguan pada beberapa jalur interface utama.

REFERENSI

[1] Wiryawan, R. A., & Rosyid, N. R. 2019). Pengembangan Aplikasi Otomatisasi Administrasi Jaringan Berbasis Website menggunakan Bahasa Pemrograman Python. *Simetris: Jurnal Teknik Mesin, Elektro dan Ilmu Komputer*, *10*(2), 741-752.

[2] Ginting, E. S., & Hadi, I. (2020). Testing Automatic Configuration of Adding Gateways to a Virtual Router using Web-Based Network Automation Applications Elin. *J. Media Inform. Budidarma*, *4*(4), 1126-1131.

[3] Nugroho, K., Abrariansyah, A. D., & Ikhwan, S. (2020). Comparison of Paramiko and Netmiko Library Performance in Network Automation Process.

[4] Rahardika, D., & Ratama, N. (2021). Implementasi Network Automation untuk Konfigurasi Jaringan Baru Dengan Netmiko. *Vol*, *2*, 190-200.

[5] Prabowo, W. (2021). *Rancang Bangun Automasi Jaringan Komputer dengan Python.* Politeknik Negeri Malang: Skripsi Diploma IV, Teknologi Informasi.

[6] Nugroho, S., & Pujiarto, B. (2022). Network Automation pada beberapa Perangkat Router menggunakan Pemrograman Python. *Jurnal Teknologi Informasi dan Ilmu Komputer*, *9*(1), 79-86.

[7] Mauboy, L. G., & Wellem, T. (2022). Studi Perbandingan Library untuk Implementasi Network Automation Menggunakan Paramiko dan Netmiko Pada Router Mikrotik. *JURIKOM (Jurnal Riset Komputer)*, *9*(4), 790-799.

[8] Novendra, Y., Arta, Y., & Siswanto, A. (2018). Analisis Perbandingan Kinerja Routing OSPF dan EIGRP. *IT Journal Research and Development*, *2*(2), 97-106.

[9] Ramdhani, M. D., Sugiarto, B., & Rukmana, A. (2021). Simulasi Jaringan SDN menggunakan Controller RYU pada Mininet dengan 5 Topologi Jaringan. *Fuse-Teknik Elektro*, *1*(2), 101-110.

[10] Fauzi, S., Larasati, S., Putri, A. A., Restyasari, N., & Suranegara, G. M. (2021). Simulasi Multi-Topologi Jaringan Berbasis SDN dengan Controller POX. *Telecommunications, Networks, Electronics, and Computer Technologies (TELNECT)*, *1*(2), 77-84.

[11] Fahmi, M., Maisyaroh, M., Komarudin, I., Faizah, S., & Fadhilah, I. (2021). Otomatisasi Jaringan menggunakan Script Python untuk Penyediaan Konfigurasi Internet dan Manajemen Mikrotik. *Bina Insani ICT Journal*, *8*(1), 53-62.

[12] Amalia, R., Kalsum, T. U., & Riska, R. (2021). Analisis dan Implementasi Software Defined Networking (SDN) untuk Automasi Perangkat Jaringan. *Infotek: Jurnal Informatika dan Teknologi*, *4*(2), 312-322.

[13] Aziz, A. A., & Haerudin, H. (2022). Perancangan Otomatisasi Jaringan Berbasis Web dengan Django (Studi Kasus PT. PLATINUM CITRA INDONESIA). *OKTAL: Jurnal Ilmu Komputer dan Sains*, *1*(06), 583-592.

[14] Fadhila, E. N., Gumelar, E. R., Pratama, H. R., & Suranegara, G. M. (2021). Otomasi Konfigurasi Routing pada Router menggunakan Ansible. *Telecommunications, Networks, Electronics, and Computer Technologies (TELNECT)*, *1*(2), 93-98.

[15] Perkasa, K. D., Sudaryanto, A., & Hartono, E. D. (2021). Pengujian Bandwidth pada Sistem Setting Bonding Mikrotik Otomatis Menggunakan Library Paramiko. *Informatics, Electrical and Electronics Engineering (Infotron)*, *1*(1), 1-5.

[16] FAZA, M. Z. (2020). *Aplikasi Otomatisasi Jaringan menggunakan Python Berbasis Web.* Politeknik Negeri Sriwijaya.

[17] Saputra, R. I. (2022). *Analisis Kinerja Redistribution Routing Protokol OSPF, EIGRP, dan BGP*. Universitas Islam Riau: Skripsi).

[18] Nurhidayah, M. S., Pranindito, D., & Wahyuningrum, R. D. (2022). Analisis dan Simulasi Routing Border Gateway Protocol (BGP) antar Autonomous System menggunakan Free Range Routing (FRR). *Jurnal Litek: Jurnal Listrik Telekomunikasi Elektronika*, *19*(2), 48-56.

[19] Mukmin, C., & Negara, E. S. (2019). Analisis Kinerja Redistribusi Routing Protokol Dinamik (Studi Kasus: Rip, Eigrp, Is-Is). *Klik-Kumpul. J. Ilmu Komput*, *6*(3), 284.

[20] Utami, P. R. (2020). Analisis Perbandingan Quality of Service Jaringan Internet Berbasis Wireless Pada Layanan Internet Service Provider (Isp) Indihome Dan First Media. *Jurnal Ilmiah Teknologi dan Rekayasa*, *25*(2).