BAB 2
DASAR TEORI

2.1 KAJIAN PUSTAKA

Sebuah penelitian telah dilakukan pada penelitian [1] yang bertujuan untuk melakukan otomasi konfigurasi *routing* OSPF yang diimplementasi oleh Ansible dengan menggunakan metode penelitian *Research and Development* (R&D). Otomasi konfigurasi diimplementasikan melalui skrip pada Ansible yang dihubungkan dengan router melalui *Secure Shell* (SSH) sehingga konfigurasi dapat dimasukkan kedalam router. Router yang digunakan pada penelitian ini merupakan router virtual yang dibangun menggunakan GNS3. Hasil dari penelitian ini Ansible telah berhasil digunakan untuk mengotomasi konfigurasi router. Berapapun router yang akan dipasang pada jaringan, router tetap dapat di otomasi hanya dengan menggunakan skrip yang telah dibuat pada Ansible. Penerapan otomasi jaringan ini dapat memudahkan untuk konfigurasi perangkat jaringan dengan skala besar secara otomatis serta dapat meminimalisir human error.

Sebuah penelitian telah dilakukan untuk menerapkan otomasi jaringan menggunakan Ansible dan membandingkannya dengan metode konvensional pada jaringan EIGRP [6]. Pengujian ini dilakukan untuk mengevaluasi keakuratan Ansible dalam mengkonfigurasi protokol *routing* EIGRP dan membandingkan efektivitas otomasi Ansible dengan metode konvensional melalui empat skenario, yaitu konfigurasi alamat IP ke antarmuka, protokol *routing* EIGRP, rute statis *default*, dan konfigurasi EIGRP lanjutan. Pengujian dilakukan pada tiga router, satu *switch*, dan satu *Docker Network Automation* dengan menggunakan perangkat lunak GNS3. Hasil pengujian menunjukkan bahwa Ansible berhasil mengotomatisasi skrip dan menerapkan konfigurasi secara akurat dan efisien. Dalam skenario pertama, skrip *Playbook* berhasil dijalankan di *docker* otomasi jaringan untuk memverifikasi efisiensi tugas seperti alamat IP, alamat *Loopback*, dan EIGRP dasar. Skenario kedua menguji operasi EIGRP untuk memverifikasi konfigurasi EIGRP termasuk

tetangga EIGRP, informasi protokol *routing*, dan tabel *routing* dengan menggunakan perintah *ping* dan perintah *show run*. Skenario ketiga menguji rute statis *default* dengan menggunakan perintah *show* untuk melihat tugas menyebarkan skrip rute statis *default* pada Router. Skenario keempat adalah tes *Fine-Tune* EIGRP yang menguji utilisasi *bandwidth*, *hello* interval, dan hold timer. Dalam hasil dan analisis, konfigurasi EIGRP menggunakan skrip otomatis terbukti diverifikasi dan akurat serta lebih efisien dibandingkan dengan metode konvensional.

Terdapat penelitian yang menganalisis perbandingan performansi jaringan di Universitas Tanri Abeng dengan menggunakan protokol *routing* Statis, OSPF, dan BGP berdasarkan parameter QoS [8]. Penelitian ini menggunakan delapan skenario pengujian yang diulang untuk *set*iap parameter, dengan tujuan untuk menentukan metode *routing* yang lebih optimal. Untuk *set*iap skenario melakukan *ping* sebanyak 100 kali dengan ukuran paket data 32 *byte set*iap sekali *ping*. Dari empat parameter yang diuji dalam delapan skenario pengujian untuk *routing* statis, OSPF, dan BGP, dapat disimpulkan bahwa *routing* BGP memberikan hasil yang lebih baik. Dalam pengujian parameter *throughput*, *routing* BGP mencapai nilai 50bpms, yang masuk dalam kategori indeks parameter sedang berdasarkan standarisasi TIPHON (*Telecommunications and Internet Pro*to*col Harmonization Over Network*). Sedangkan dalam pengujian parameter *packet loss*, *routing* BGP mendapatkan nilai 0,25% yang masuk dalam kategori indeks parameter sangat bagus berdasarkan standarisasi TIPHON (*Telecommunications and Internet Pro*to*col Harmonization Over Network*).

Terdapat penelitian yang melakukan studi perbandingan *Library* Untuk Implementasi *Network Au*to*mation* Menggunakan Paramiko Dan Netmiko Pada Router Mikrotik [5]. Pengujian ini dilaksanakan untuk mengevaluasi rata-rata waktu eksekusi dari empat skenario yang berbeda, yaitu konfigurasi *Static Routing*, *Dynamic Routing* menggunakan RIPv2, *Firewall* dengan NAT (*Network Address Translation*), dan SNMP (*Simple Network Management Pro*to*col*). Dua topologi yang berbeda digunakan dalam pengujian, yakni topologi pertama menggunakan dua perangkat

MikroTik, sedangkan topologi kedua menggunakan tiga perangkat MikroTik. Pengambilan data dilakukan dengan menggunakan perangkat lunak GNS3 dan data yang diper*oleh* berupa eksekusi *script* Python. Hasil pengujian menunjukkan bahwa *library* Paramiko lebih unggul dalam waktu eksekusi *script* dibandingkan dengan Netmiko, dengan selisih rata-rata waktu sebesar 3,66 detik.

Terdapat penelitian yang melakukan studi perbandingan kinerja *Network Automation* Python menggunakan *Library* Paramiko dan Netmiko untuk mengaktifkan proses *routing* dinamis pada perangkat router Cisco dimana protokol *routing* yang digunakan adalah OSPF [7]. Skenario dilakukan pada topologi jaringan *partial-mesh* yang berisi empat buah router Cisco seri 7200 , satu *switch*, dan satu *Docker Network Automation* dengan scenario pengujian mengukur waktu pemberian *script* konfigurasi, waktu konvergensi jaringan, serta nilai *throughput* dan *delay* dari kedua *Library Network Automation* yaitu Netmiko dan Paramiko. waktu proses pemberian *script* konfigurasi ke perangkat router dengan menggunakan *library* Netmiko 4,14 kali lebih lambat dibandingkan jika menggunakan Paramiko. Dengan waktu kirim data lebih lambat, maka penggunaan *library* Netmiko menghasilkan waktu konvergensi jaringan yang juga lebih lambat dibandingkan Paramiko. Dengan waktu konvergensi lebih lambat, maka nilai *delay* yang dihasilkan oleh *library* Netmiko juga lebih lambat dan nilai *throughput* yang dihasilkan juga lebih kecil.

Tabel 2.1 Kajian Penelitian Sebelumnya

Year	*Author*	*Objective*	*Configuration*	*Automation*	*Result*
2020	Ega Restu Gumelar , Esa Noer Fadhila , Herdi Rizky Pratama, Galura Muhammad Suranegara	Dalam penelitian ini, dilakukan konfigurasi *routing* protokol OSPF menggunakan Ansible pada 3 router yang disimulasikan dilingkungan GNS 3	*Network Automation* pada *Routing Protocol* OSPF	Ansible	Ansible telah berhasil digunakan untuk mengotomasi konfigurasi *routing* protokol OSPF pada GNS 3.
2021	Mohd Faris Fuzi, Khai*run*nisa Abdullah, Imam Hazwam Abd Halim, Rafiza Ruslan	Pada penelitian ini, dilakukan penerapan otomasi jaringan menggunakan Ansible dan dibandingkan dengan metode konvensional untuk memverifikasi keakuratan konfigurasi EIGRP.	*Network Automation* pada EIGRP *Network*	Ansible	Penerapan Ansible telah berhasil dan memberikan hasil yang akurat, serta terbukti lebih efisien dibandingkan dengan metode konvensional.
2021	Sahril Amuda, Muhamad Femy Mulya, Felix Indra Kurniadi	Perbandingan performansi performansi jaringan di Universitas Tanri Abeng dengan menggunakan protokol *routing* OSPF, dan BGP berdasarkan parameter QoS.	*Routing Protocol* OSPF, dan BGP	-	Hasil pengujian menunjukkan bahwa perutean BGP untuk parameter *throughput* dan *packet loss* lebih unggul daripada *routing* protokol OSPF.

2020	Luis Geraldo Mauboy, Theophilus Wellem	Melakukan studi Perbandingan *Library* Menggunakan Paramiko Dan Netmiko Pada Router Mikrotik	*Network Automation* pada *Static routing*, *Dynamic routing* menggunakan RIPv2, NAT, dan SNMP	Paramiko	Rata-rata selisih waktu eksekusi antara Netmiko dan Paramiko adalah 3.66 detik
2021	Kukuh Nugroho, Anggi Dzikri Abrariansyah, Syariful Ikhwan	melakukan studi perbandingan Library menggunakan Paramiko dan Netmiko pada perangkat router Cisco	OSPF	Python	Menggunakan Paramiko 4,14 kali lebih cepat dibandingkan menggunakan Netmiko
2023	Fadhel Anugrah Ananda	Otomasi jaringan menggunakan ansible pada protokol *routing Border Gateway Protokol* (BGP)	*Network Automation* pada *Routing* Protokol BGP	Ansible dan Paramiko	

2.2 DASAR TEORI

2.2.1 *Network Automation*

Network Automation adalah proses mengotomatisasi konfigurasi, manajemen, dan operasi dari sebuah jaringan komputer. Tugas-tugas yang biasanya dilakukan oleh administrator jaringan atau sistem dapat diotomatisasi menggunakan berbagai alat dan teknologi. Seperti yang kita ketahui, kesalahan manusia adalah alasan utama dari sebagian besar masalah, termasuk ketidaktersediaan, waktu henti, keamanan, dan sebagainya dalam lingkungan jaringan. Automasi yang tepat akan mengeliminasi kesalahan manusia dan juga mempercepat operasi, sehingga menghemat waktu dan biaya. Automasi jaringan diimplementasikan melalui kombinasi solusi berbasis perangkat keras dan perangkat lunak yang secara otomatis menjalankan tugas-tugas berulang di lingkungan jaringan [10].

Dalam konteks jaringan, automasi jaringan adalah proses mengotomatisasi konfigurasi, manajemen, pengujian, penyebaran, dan operasi perangkat fisik dan berbasis perangkat lunak dalam suatu jaringan. Tugas-tugas yang biasanya dilakukan oleh administrator jaringan atau sistem dapat diotomatisasi menggunakan berbagai alat dan teknologi [11]. *Set*iap jenis jaringan dapat menggunakan automasi jaringan, termasuk pusat data, penyedia layanan, dan kampus. Automasi jaringan menghilangkan langkah-langkah manual yang diperlukan untuk mengelola jaringan, seperti masuk ke router, *switch*, *load balance*r, dan *firewall* untuk mengubah konfigurasi secara manual sebelum keluar. Automasi jaringan mengandalkan skrip yang terhubung satu sama lain dan diprogram pada antarmuka baris perintah (CLI) dari sistem operasi (OS) atau perangkat lunak automasi yang telah dipaketkan sebelumnya.

2.2.2 Ansible

Ansible adalah perangkat lunak komputer yang membantu seorang administrator sistem dalam melakukan berbagai tugas, seperti instalasi aplikasi, *deployment*, dan pembaruan server. Ansible juga dapat digunakan untuk melakukan otomatisasi pada server lainnya. Ansible adalah salah satu jenis *Configuration*

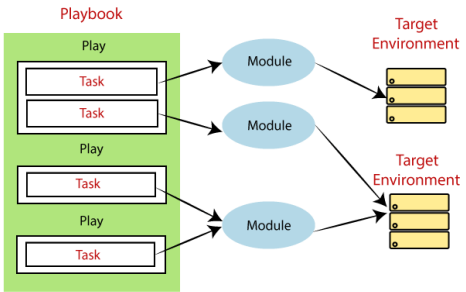
Management To*ols* yang dapat mengubah proses manajemen infrastruktur dan *deployment* dari manual menjadi otomatis [12].

Berikut merupakan tujuan penggunaan Ansible [13]:

1) *Clear*—Ansible menggunakan sintak sederhana (YAML) dan membuatnya mudah bagi orang-orang tertentu (seperti *developers*, *sysadmin*, dan manajer) untuk memahami dan membaca struktur kodenya.
2) *Fast* - Mudah untuk mengkonfigurasi karena tidak memerlukan instalasi aplikasi atau daemon tambahan pada server.
3) *Efficient* - Tidak ada perangkat lunak tambahan pada server, jadi semua yang dibutuhkan adalah sambungan, dan modul Ansible berkomunikasi dengan JSON untuk server.
4) *Secure* - Untuk terhubung ke server, Ansible menggunakan SSH tanpa perlu mengubah port atau aplikasi yang mengurangi potensi akses ke server.

A. *Playbook* Ansible

Playbook ditulis dalam format YAML, yang berisi model konfigurasi atau proses, berkembang dari bahasa pemprograman atau skript [13]. Tugas (*tasks*), modul (*module*s), dan *File* semuanya merupakan komponen yang terdapat dalam struktur peran (*role*). Peran terdiri dari direktori dengan subdirektori, dan *set*iap subdirektori memiliki *File main*.yml yang menentukan urutan operasi yang harus dilakukan. Modul merupakan program kecil yang melakukan tugas-tugas khusus pada sistem. Modul dapat digunakan secara mandiri atau sebagai bagian dari skrip yang lebih besar yang disebut *Playbook*. Gambar 2.1 menunjukkan struktur *Playbook*.



Gambar 2.1 Struktur *Playbook*

*Set*iap *Playbook* adalah kumpulan dari satu atau lebih play. *Playbook* terstruktur dengan menggunakan *Plays*. Dalam satu *Playbook* dapat terdapat lebih

dari satu *play*. Fungsi dari *play* adalah untuk memetakan sekelompok instruksi yang didefinisikan untuk *Host* tertentu. Dalam *Playbook* dapat menjalankan beberapa *tasks* secara langsung dan berulang kali. Saat Ansible menjalankan *tasks*, sebenarnya Ansible sedang menjalankan sebuah skrip sesuai dengan tugas yang dijalankan. Skrip ini disebut sebagai *module* [14].

File YAML memiliki beberapa aturan yang harus kita patuhi, di antaranya sebagai berikut:

1) Spasi digunakan untuk menandai struktur penulisan
2) Tabulasi dilarang dan tidak diizinkan seperti pada bahasa Python
3) Penggunaan tanda — ditulis pada baris pertama
4) Penggunaan tanda ... digunakan jika kita ingin mengakhiri dokumen dalam *File* yang sama
5) Menggunakan tanda # (pagar) untuk memberikan komentar pada baris tertentu
6) Tanda - (strip) menunjukkan anggota daftar pada *set*iap subdivisi objek
7) Menggunakan tanda : (titik dua) untuk memberikan nilai pada *set*iap kunci atau *variable*
8) Menggunakan tanda ; (titik koma) untuk membuat *array*

B. Manfaat Ansible

Ansible memberikan beberapa keuntungan dibandingkan dengan alat pengelolaan konfigurasi dan otomatisasi lainnya [15], antara lain:

1) Kemudahan: Ansible menggunakan format YAML dan memiliki sintaks yang sederhana, sehingga mudah dipelajari dan digunakan, bahkan oleh pengguna *non*-teknis.
2) Tanpa agen: Ansible tidak memerlukan instalasi agen tambahan pada *node* yang dikelola. Ansible menggunakan SSH atau WinRM untuk berkomunikasi dengan mesin target, sehingga lebih ringan dan mudah dikonfigurasi.
3) Orkestrasi *multi-node*: Ansible dapat mengelola tugas pada banyak node secara bersamaan, sehingga cocok untuk implementasi skala besar.
4) Ekosistem yang luas: Ansible memiliki komunitas yang besar dan aktif yang telah mengembangkan berbagai modul dan peran, sehingga mudah menemukan solusi yang sudah ada untuk tugas umum.

5) *Cross-platform*: Ansible dapat digunakan dengan berbagai sistem dan teknologi, termasuk Linux, Windows, dan perangkat jaringan, sehingga fleksibel dalam mengelola infrastruktur yang beragam.
6) Modularitas: Ansible memungkinkan organisasi *Playbook* dalam peran, yang memudahkan penggunaan dan berbagi *Playbook* di berbagai proyek.
7) Fleksibilitas: Ansible dapat digunakan untuk berbagai kasus penggunaan, termasuk penyediaan, pengelolaan konfigurasi, implementasi aplikasi, dan orkestrasi *multi-node*.
8) Integrasi yang baik: Ansible dapat diintegrasikan dengan alat lain, seperti penyedia awan (*cloud provider*), alat pemantauan (*moni*to*ring* to*ols*), dan lainnya, sehingga memberikan nilai tambah bagi *set* alat yang digunakan oleh suatu organisasi.
9) Dukungan yang kuat: Ansible didukung oleh komunitas yang besar dan merupakan alat *open-source*, sehingga tersedia banyak sumber daya, tutorial, dan dukungan bagi para pengguna.

2.3 Python Paramiko

Paramiko adalah *library* Python yang memiliki antarmuka yang mudah digunakan dan dokumentasi yang lengkap, yang memungkinkan pengguna untuk menggunakannya dengan mudah. Paramiko memiliki berbagai fitur, seperti eksekusi perintah, transfer *file*, dan tunel jaringan, yang memungkinkan pengguna untuk melakukan otomatisasi pada perangkat jaringan. Selain itu, *library* ini memiliki fitur keamanan yang kuat, seperti enkripsi AES, autentikasi dengan kunci publik, dan proteksi terhadap serangan kekerasan [16].

Paramiko dapat digunakan untuk mengotomatisasi tugas-tugas seperti konfigurasi perangkat jaringan, *backup* konfigurasi, dan pemantauan jaringan. Selain itu, Paramiko dapat digunakan untuk mengotomatisasi transfer *file* antara perangkat jaringan dan server. Selain itu, Anda dapat menggunakan Paramiko untuk mengontrol perangkat yang berjalan pada Unix, Linux, atau macOS [16].

2.3.1 *Secure Shell* (SSH)

Secure Shell (SSH) adalah pendekatan perangkat lunak yang populer dan kuat untuk keamanan jaringan. SSH secara otomatis mengenkripsi dan membuka kunci data *set*iap kali komputer mengirimkannya ke jaringan. Hasilnya adalah enkripsi transparan: pengguna dapat bekerja secara normal, tanpa menyadari bahwa komunikasi mereka aman terenkripsi di jaringan. Selain itu, SSH menggunakan algoritma enkripsi modern dan cukup efektif sehingga digunakan dalam aplikasi yang kritis pada perusahaan-perusahaan besar [17].

2.3.2 *Routing* Protocol

Routing adalah suatu proses menentukan jalur atau rute yang digunakan untuk mengirimkan paket data dari source node ke *destination* node [18]. *Routing Protocol* merupakan serangkaian aturan yang digunakan untuk bertukar informasi mengenai rute dan menghasilkan tabel *routing*. Tujuan utama dari protokol *routing* adalah untuk memperjelas pengalamatan pada paket data yang akan dikirim. Protokol *routing* juga digunakan untuk mencari rute terpendek atau terbaik agar paket data dapat dikirimkan dengan efisien ke alamat tujuan yang dituju. Dengan adanya protokol *routing*, jaringan dapat mengatur lalu lintas data secara efektif dan mengoptimalkan pengiriman paket melalui jalur yang paling sesuai [19]. Ada beberapa jenis protokol *routing* yang berbeda, seperti OSPF, EIGRP, RIP, IS-IS, BGP, dan lain-lain. *Set*iap jenis protokol memiliki kelebihan dan kekurangan dalam mengelola lalu lintas jaringan. Protokol *routing* dapat dikelompokkan menjadi dua jenis:

A. *Interior Gateway Pro*to*col* (IGP) adalah protokol *routing* yang digunakan untuk *routing* di dalam satu *Au*to*nomous System* (AS). IGP juga dikenal sebagai *routing* Intra-AS, yang berarti proses *routing* terjadi di dalam AS tersebut. Protokol *routing* IGP dapat dibagi menjadi dua kategori:
    1) *Distance-vec*to*r Routing pro*to*col*s adalah protokol yang menghitung rute berdasarkan jarak dan arah. Jarak dihitung menggunakan metrik tertentu, sementara arah menunjukkan langkah berikutnya dalam jalur. Contoh dari protokol *routing* distance-vector adalah RIPv1, IGRP, RIPv2, dan EIGRP.

2) *Link-state Routing protocol*, juga dikenal sebagai protokol *Shortest Path First* (SPF), membangun sebuah database topologi sendiri di *set*iap router. *Set*iap router menerima peta (*map*) dari router tetangganya. Contoh dari protokol *routing link-state* adalah OSPF dan IS-IS.

B. *Exterior Gateway Protocol* (EGP) adalah protokol *routing* yang digunakan untuk *routing* antara *Autonomous Systems* (AS). EGP juga dikenal sebagai *routing Inter*-AS, yang berarti proses *routing* antara AS. EGP hanya terdiri dari satu jenis, yaitu protokol *Path-Vector Routing*, yang dalam hal ini adalah BGP.

2.3.3 *Border Gateway Protocol* (BGP)

Border Gateway Protocol (BGP) merupakan protokol yang digunakan untuk pertukaran rute antar *Autonomous System* (AS) atau domain di Internet. Informasi yang ditukar meliputi informasi jumlah AS dalam jalur pengiriman informasi [20]. Dengan adanya informasi ini, dapat dibentuk grafik dari jalur AS yang saling terhubung untuk menghindari terjadinya *loop routing*. Selain itu, BGP juga digunakan untuk menerapkan kebijakan *routing* di tingkat interdomain. BGP memiliki tabel *routing* khusus yang berisi kumpulan prefiks *routing* yang diterima dari router BGP lainnya.

BGP tergolong dalam keluarga protokol *routing Path Vector*. BGP juga dikenal sebagai iBGP (*Interior Border Gateway Protocol*) ketika digunakan dalam AS yang sama, sedangkan eBGP (*Exterior Border Gateway Protocol*) digunakan dalam AS yang berbeda. BGP mengambil keputusan *routing* berdasarkan atribut-atribut jalur. Atribut-atribut tersebut meliputi *weight*, *local preference*, AS *path*, Origin, dan MED (*Multi-Exit Discriminator*) [8].

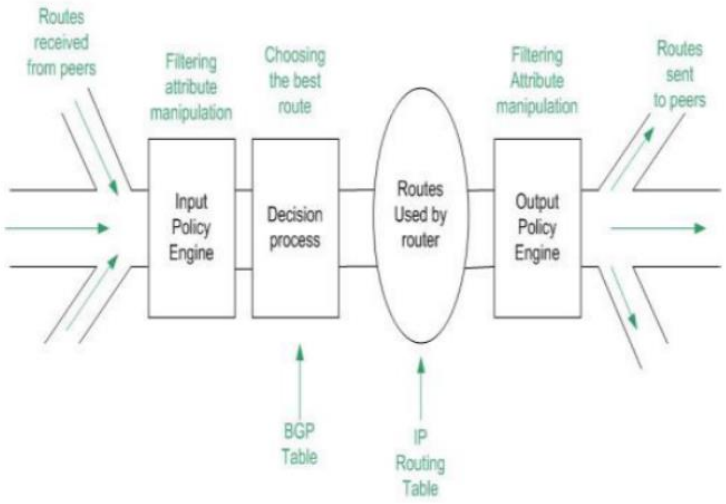
2.3.3.1 Operasi BGP

Ketika kedua router terhubung dengan TCP (*Transmision Control Protocol*) dan dapat mengaktifkan BGP, maka kedua router tersebut dinamakan *neighbours* dan *peers* sedangkan yang mengaktifkan BGP dinamakan BGP *speaker*. Beberapa BGP *speaker* bisa ditempatkan pada AS

yang sama ataupun pada AS yang berbeda. Pada *set*iap bagian AS, BGP *speaker* akan berkomunikasi satu dengan yang lainnya, dengan tujuan untuk melakukan pertukaran informasi perihal tabel *routing* [8].

2.3.3.2 Proses *Routing* BGP

Terdapat beberapa tahap yang dilakukan BGP ketika melakukan proses *routing* yang terdapat pada gambar 2.2. Tahap-tahap tersebut dijelaskan sebagai berikut [8].



Gambar 2.2 Diagram Proses *Routing* BGP

1. ***Routes* Received from *Peers***

   Pada tahapan ini, router BGP akan menerima informasi terkait *routing* dari perangkat router pasangannya, baik pada router *internal* maupun router eksternal. Tahapan ini akan tergantung dari kebijakan yang telah diterapkan atau diimplementasikan oleh router BGP. Selanjutnya, sebagian atau semua informasi *routing* yang diperoleh akan dimasukkan ke dalam tabel *routing*.

2. **Input *Policy Engine***

   Pada tahapan ini, router BGP dapat melakukan proses *filter*ing terhadap semua informasi yang diperoleh pada tahapan sebelumnya. Kemudian informasi yang sudah dilakukan proses seleksi, selanjutnya akan

menuju ke tahapan berikutnya, yaitu proses penentuan atau pengambilan keputusan berupa rute atau jalur terbaik yang akan dilalui.

3. **The Decision Process**

   Pada tahapan ini, akan dilakukan proses penentuan atau pengambilan keputusan untuk semua jalur atau rute yang ada. Selanjutnya, pada proses penentuan atau pengambilan keputusan ini, diperoleh berdasarkan informasi dari tahapan sebelumnya. Pada tahapan ini, bertujuan untuk memperoleh lebih dari satu rute atau jalur untuk mencapai tujuan ke *network* yang telah ditentukan.

4. ***Routes Used By User***

   *Set*elah memperoleh rute atau jalur yang dapat ditempuh untuk sampai ke *network* tujuan, selanjutnya pada tahapan ini, informasi itu akan disusun untuk memperoleh rute atau jalur terbaik. Kemudian informasi yang telah diperoleh, lalu akan dimasukkan ke dalam tabel *routing*. Selanjutnya, informasi hasil seleksi ini nantinya akan diberikan ke BGP pasangannya, *set*elah melalui atau melewati tahapan berikutnya.

5. ***Output Policy Engine***

   Pada tahapan ini, seluruh informasi pada *routing* yang telah diperoleh akan diteruskan atau dilanjutkan ke BGP pasangannya. Lalu akan dilakukan proses seleksi dengan membedakan antara BGP pasangan yang *internal* maupun yang eksternal. Pada jalur atau rute *routing* yang berasal dari BGP *internal*, AS tidak diperkenankan untuk melanjutkan ke pasangan BGP *internal*. Ini dapat mencegah *routing* pada *loop* terjadi lagi.

6. ***Routes Advertised* to *Peers***

   Pada tahapan akhir ini, seluruh informasi pada *routing* yang telah diproses, selanjutnya akan diteruskan atau dilanjutkan ke *set*iap pasangan BGP, baik AS BGP eksternal maupun *internal*.

2.3.4 Waktu pengiriman *Script*

Waktu pengiriman *script* adalah waktu yang dibutuhkan oleh program untuk memberikan perintah konfigurasi BGP ke *set*iap router pada jaringan dan memanfaatkan *traffic* protokol SSH yang ditangkap oleh

Wireshark dari jalur *Docker Network Automation* menuju *switch*. *Filter*isasi protokol SSH dilakukan menggunakan hasil yang dihasilkan oleh Wireshark, yang mengurangi waktu akhir SSH dan mengurangi waktu awal akses. [21].

2.3.5 QoS (*Quality of Service*)

QoS (*Quality of Service*) adalah istilah yang digunakan untuk menggambarkan kemampuan suatu jaringan dalam menyediakan tingkat layanan yang berbeda-beda dengan jaminan kualitas tertentu [21].

A. *Delay* (*Latency*)

Delay adalah waktu yang diperlukan bagi sebuah paket untuk mencapai tujuannya akibat proses transmisi dari satu titik ke titik lain. Satuan yang digunakan dalam perhitungan *Delay* adalah milidetik (ms) [21]. Persamaan yang digunakan untuk menghitung *Delay* dapat dilihat pada persamaan 2.1:

$$Delay = \frac{jumlah\ waktu\ pengiriman\ data\ (sec)}{jumlah\ paket} \text{ [21]}$$

Persamaan 2.1 Rumus Nilai *Delay*

Kategori nilai *Delay* suatu jaringan dapat dikategorikan berdasarkan standar TIPHON05001 sebagaimana terlihat pada Gambar 2.3.:

Kategori	Besar *Delay*
Sangat Bagus	<150 ms
Bagus	150 s/d 300 ms
Sedang	300 s/d 450 ms
Buruk	>450 ms

Gambar 2.3 Kategori jaringan berdasarkan nilai *Delay*(versiTIPHON-05001)

B. *Throughput*

Throughput merujuk pada kapasitas atau kemampuan jaringan dalam mengirimkan data. Satuan yang digunakan dalam perhitungan *throughput* adalah bit per detik (bps) [21]. Persamaan yang digunakan untuk menghitung *throughput* dapat dilihat pada persamaan 2.2:

$$Throughput = \frac{jumlah\ data\ yang\ dikirim\ (bits)}{jumlah\ waktu\ pengiriman\ data}$$ [21]

Persamaan 2.2 Rumus Nilai *Throughput*

Nilai *throughput* suatu jaringan dapat dikategorikan berdasarkan standar TIPHON05001, sebagaimana terlihat pada Gambar 2.4.

Kategori	*Throughput*
Sangat Bagus	76 s/d 100 kbps
Bagus	51 s/d 75 kbps
Sedang	26 s/d 50 kbps
Buruk	<25 kbps

Gambar 2.4 Kategori jaringan berdasarkan nilai *throughput*

2.3.6 Waktu Konvergensi

Konvergensi adalah proses di mana semua perangkat jaringan dalam suatu sistem secara keseluruhan mengetahui dan memahami perubahan topologi jaringan yang terjadi. Ini berarti semua perangkat telah menyesuaikan tabel *routing* mereka dan memiliki pemahaman yang konsisten tentang bagaimana rute ke tujuan tertentu harus diambil.. Waktu konvergensi adalah waktu yang dibutuhkan oleh router untuk berbagi informasi atau mengirim informasi, menghitung jalur, dan memperbarui tabel peruteanya [5].

2.3.7 GNS3 (*Graphic Network Simulator* 3)

GNS3 adalah sebuah alat virtualisasi jaringan yang digunakan untuk memvirtualisasi perangkat-perangkat jaringan. GNS3 adalah alat virtualisasi sumber terbuka yang dilisensikan di bawah GNU GPL7 dan merupakan alat yang banyak digunakan untuk merancang, menguji, dan bereksperimen dengan topologi jaringan yang berbeda. GNS3 adalah salah satu dari sedikit alat virtualisasi yang mendukung implementasi perangkat jaringan dari berbagai vendor. GNS3 memvirtualisasi perangkat jaringan dengan memuat dan menjalankan Sistem Operasi (OS) yang sesungguhnya, sehingga semua fungsionalitas dan perintah yang terdapat pada perangkat

fisik dengan OS yang sama tersedia untuk digunakan pada perangkat jaringan yang divirtualisasi di GNS3. Dengan menggunakan GNS3 sebagai perangkat lunak virtualisasi, OS dari produsen perangkat jaringan dapat diunggah ke lingkungan virtual dan menciptakan topologi jaringan yang divirtualisasi. Hal ini memungkinkan kontrol yang lebih besar terhadap topologi jaringan dan jumlah perangkat dapat diperluas tanpa mengorbankan biaya ekonomi dibandingkan dengan pembuatan jaringan fisik. GNS3 menggunakan virtualisasi bertingkat untuk memvirtualisasi beberapa perangkat pada satu server yang sama. Namun, penggunaan beberapa perangkat pada satu server dapat mengakibatkan penurunan kinerja [22].

2.3.8 Wireshark

Wireshark merupakan sebuah perangkat lunak yang digunakan untuk menganalisis jaringan, yang memungkinkan pengguna untuk merekam dan menganalisis lalu lintas jaringan secara *real-time*. Fungsionalitas Wireshark meliputi pemecahan masalah jaringan, identifikasi gangguan, optimalisasi kinerja jaringan, dan penyelesaian masalah keamanan.

Wireshark bekerja dengan memulai proses perekaman data yang dikirim melalui jaringan, baik dalam bentuk paket jaringan maupun frame *Ethernet*. *Set*elah itu, Wireshark menganalisis data yang telah direkam dan menampilkan informasi detail tentang *set*iap paket yang melewati jaringan, termasuk jenis protokol yang digunakan, alamat sumber dan tujuan, isi paket, dan sebagainya.

Dengan informasi yang ditampilkan oleh Wireshark, pengguna dapat mengambil tindakan yang tepat, seperti memecahkan masalah jaringan, meningkatkan kinerja jaringan, atau mengidentifikasi potensi masalah keamanan. Wireshark memberikan pengguna kemampuan untuk menganalisis data secara mendalam dan mengambil keputusan yang sesuai berdasarkan informasi yang diperoleh [23].