BAB II
DASAR TEORI

2.1 KAJIAN PUSTAKA

Penelitian 1[1] 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 2[2] 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 3[3] membahas perbedaan kinerja antara *Library* Paramiko dan Netmiko dalam pembuatan sistem otomasi jaringan dengan menggunakan metode analisis sistem. Penelitian ini bertujuan untuk mengetahui perbedaan performasi 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 4[4] 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.

Tabel 2. 1 Perbandingan Penelitian Sebelumnya

Aspek	**Penelitian 1**	**Penelitian 2**	**Penelitian 3**	**Penelitian 4**
Judul	Pengembangan Aplikasi Otomasi Administrasi Jaringan Berbasis Website Menggunakan Bahasa Pemrograman Python	Pengujian Konfigurasi Otomatis Penambahan *Gateway* Pada *Virtual Router* Menggunakan Aplikasi Otomasi Jaringan Berbasis Web	Perbandingan Kinerja *Library* Paramiko dan Netmiko Dalam Proses Otomasi Jaringan	Implementasi *Network Automation* Untuk Konfigurasi Jaringan Baru Dengan Netmiko
Tujuan	Menilai kinerja aplikasi otomatisasi jaringan berbasis web menggunakan pengujian *black-box*	Mengimplementasikan konfigurasi otomatis *gateway* pada router menggunakan Django dan Paramiko	Membandingkan kinerja *library* Paramiko dan Netmiko dalam otomasi jaringan	Menerapkan otomasi jaringan untuk konfigurasi jaringan baru menggunakan Netmiko
Metodologi	Pengujian *black-box* terhadap berbagai fitur aplikasi	Menggunakan Python, Django, dan *library* Paramiko; pengujian dengan metode *black-box testing*	Menggunakan Python dengan *library* Paramiko dan Netmiko, pengujian melalui *software* emulasi GNS3	Metode *Network Development Life Cycle* (NDLC), menggunakan Python dan Netmiko, pengujian dengan *black-box* dan kuesioner
Alat & Bahan	Ubuntu, Python 3.6.7, *Library* Paramiko 2.4, Django dan GNS3 2,1	Laptop, Python, Django, GNS3, Google Chrome, Putty	Komputer Docker, GNS3, Python, *library* Paramiko dan Netmiko	Python, Netmiko
Pengujian	*Black-box testing* terhadap fitur aplikasi (*setting*,	Pengujian koneksi dan fungsionalitas melalui *black-box*	Uji waktu konfigurasi, waktu konvergensi	Pengujian *black-box* dan kuesioner untuk

Aspek	Penelitian 1	Penelitian 2	Penelitian 3	Penelitian 4
	konfigurasi routing, VLAN, *backup, restore*)	*testing* pada GNS3	jaringan, *throughput,* dan *delay*	evaluasi performa dan efektivitas sistem
Hasil	Hasil *black-box testing* menunjukkan semua fitur sistem aplikasi berhasil berfungsi dengan baik	Hasil pengujian *black-box* menunjukkan konfigurasi otomatis *gateway* berhasil dilakukan pada router yang disimulasikan	Analisis performansi antara Paramiko dan Netmiko menunjukkan perbedaan dalam parameter waktu konvergensi dan *throughput*	Instalasi jaringan baru menjadi lebih mudah dan cepat, mengurangi risiko kesalahan manusia dan penggunaan sumber daya yang rendah

Penelitian 1 berfokus pada analisis perbandingan performansi antara dua *library* Python, Paramiko dan Netmiko, dalam otomasi jaringan dengan mengukur waktu konfigurasi, waktu Skonvergensi, *throughput*, dan *delay* menggunakan GNS3 dan Python, yang menunjukkan perbedaan signifikan dalam performansi antara keduanya. Penelitian 2 mengimplementasikan dan menguji konfigurasi otomatis *gateway* pada router menggunakan Django dan Paramiko, dengan hasil bahwa konfigurasi otomatis *gateway* berhasil diterapkan dan diuji pada lingkungan simulasi. Penelitian 3 menilai kinerja aplikasi otomatisasi jaringan berbasis web menggunakan *black-box testing* pada berbagai fitur aplikasi termasuk konfigurasi routing, VLAN, *backup,* dan *restore*, yang menunjukkan bahwa semua fitur sistem aplikasi berhasil diuji dan berfungsi dengan baik. Penelitian 4 menerapkan otomasi jaringan untuk konfigurasi jaringan baru menggunakan *library* Netmiko dan metode NDLC, dengan hasil bahwa instalasi jaringan baru menjadi lebih mudah dan cepat, mengurangi risiko kesalahan manusia, dan memanfaatkan sumber daya yang lebih efisien, dievaluasi melalui pengujian *black-box* dan kuesioner.

Perbandingan penelitian ini dengan penelitian sebelumnya menunjukkan bahwa penelitian ini fokus pada analisis performansi protokol routing OSPF dan EIGRP dalam konteks otomasi jaringan berbasis web, sementara penelitian 1 lebih fokus pada pengembangan aplikasi dengan fitur-fitur administratif. Penelitian ini serupa dengan penelitian 2 dalam penggunaan bahasa pemrograman Python dan *library* Paramiko untuk otomasi jaringan, tetapi penelitian ini lebih spesifik dalam menganalisis performa protokol routing OSPF dan EIGRP, sedangkan penelitian 2 fokus pada otomasi konfigurasi *gateway*. Selain itu, penelitian ini menggunakan *library* Paramiko untuk otomasi jaringan tanpa membandingkannya dengan

Netmiko, berbeda dari penelitian 3 yang membandingkan dua *library* otomasi. Penelitian ini juga menggunakan *library* Paramiko dan *framework* Django, berbeda dengan penelitian 4 yang menggunakan Netmiko dan berfokus pada implementasi sistem otomasi, sementara penelitian ini menitikberatkan pada analisis performa routing OSPF dan EIGRP.

2.2 DASAR TEORI

2.2.1 Jaringan Komputer

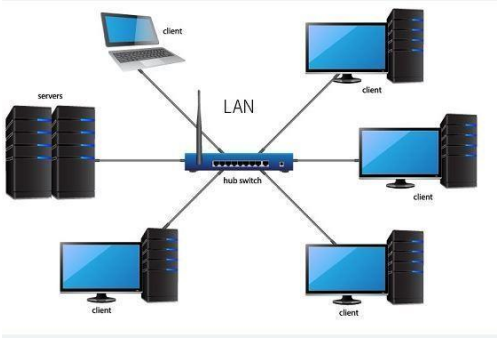
Jaringan komputer adalah infrastruktur telekomunikasi yang memungkinkan komputer-komputer untuk berkomunikasi dan bertukar data satu sama lain. Pembentukan jaringan komputer dilakukan dengan menggunakan kombinasi perangkat keras dan perangkat lunak. Bagian jaringan komputer yang menerima atau meminta layanan agar jaringan tersebut dapat saling berkomunikasi dan bertukar data disebut dengan *client*[5]. Bagian lain yang dapat memberikan atau mengirimkan data disebut dengan server.

Komputer yang terhubung dalam jaringan harus dilengkapi dengan setidaknya satu kartu jaringan (*Network Interface Card*) yang kemudian dihubungkan menggunakan koneksi kabel atau nirkabel sebagai sarana untuk mentransmisikan data. Selain itu, sebuah sistem operasi jaringan perlu diinstal di komputer untuk mengatur jaringan komputer yang sederhana[13]. Perangkat seperti hub, *switch*, dan router dapat dimanfaatkan untuk memperluas cakupan jaringan komputer yang dibentuk.

Menurut cakupan areanya, jenis jaringan komputer dapat di klasifikasikan sebagai berikut :

a. LAN (*Local Area Network*)

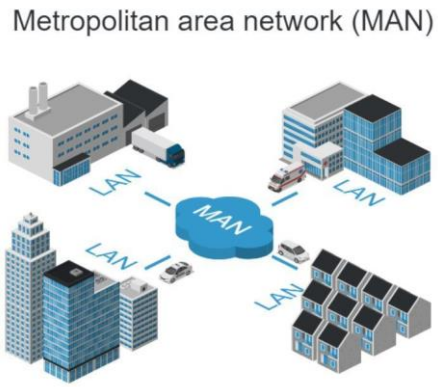
Konsep *Local Area Network* atau LAN menggabungkan perangkat jaringan dalam jarak yang terbatas, biasanya digunakan di lingkungan seperti perkantoran, lembaga pendidikan, rumah, dan sebagainya. Jaringan LAN memanfaatkan jenis konektivitas khusus. Implementasi jaringan LAN diilustrasikan dalam gambar 2.1. Meskipun LAN umumnya mengandalkan penggunaan kabel, terdapat juga teknologi nirkabel yang dikenal sebagai *Wireless Local Area Network* (WLAN).



Gambar 2. 1 Jaringan LAN

b. MAN (*Metropolitan Area Network*)

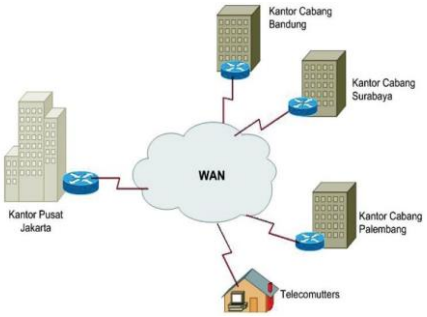
Metropolitan Area Network atau MAN adalah konsep yang menghubungkan perangkat jaringan dari area kota ke area kota lainnya. Berbeda dengan LAN, struktur LAN tidak memfasilitasi pembangunan jaringan MAN. Pada gambar 2.2 terlihat bahwa jaringan MAN memanfaatkan perangkat khusus dan membutuhkan operator telekomunikasi yang berperan sebagai penghubung antara jaringan komputer.



Gambar 2. 2 Jaringan MAN

c. WAN (*Wide Area Network*)

Wide Area Network (WAN) adalah konsep yang mengintegrasikan perangkat jaringan komputer dalam wilayah yang sangat luas, sering kali memanfaatkan teknologi tingkat lanjut. Jaringan WAN umumnya digunakan untuk menghubungkan jaringan antar negara, bahkan lintas benua[7]. Contoh teknologi yang digunakan adalah serat optik, di mana pemasangan serat optik dapat dilakukan di bawah laut atau di dalam tanah. Ilustrasi jaringan WAN terlihat pada gambar 2.3 yang menunjukkan hubungan antara kantor pusat dan kantor cabang yang berjarak sangat jauh.



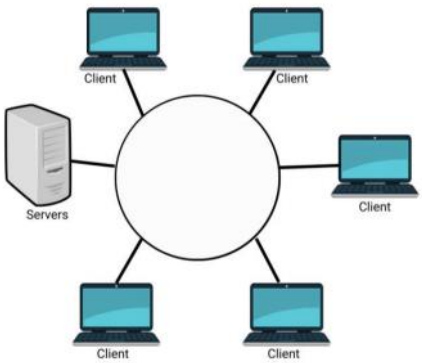
Gambar 2. 3 Jaringan WAN

2.2.2 Topologi Jaringan

Topologi jaringan adalah metode menghubungkan komputer satu sama lain untuk membentuk suatu jaringan. Pemilihan topologi jaringan komputer akan mempengaruhi kecepatan komunikasi. Oleh karena itu, memahami kelebihan dan kekurangan dari setiap topologi berdasarkan karakteristiknya menjadi sangat penting. Topologi jaringan merupakan representasi grafis dari struktur jaringan komputer. Terdapat dua jenis topologi jaringan, yaitu topologi logis (*logical topology* dan topologi fisik (*physical topology*)[17]. Topologi dapat dibagi menjadi beberapa jenis, yaitu :

a. Topologi Ring

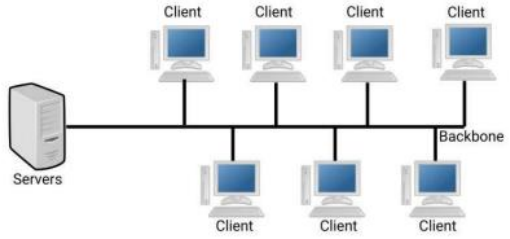
Topologi Ring adalah struktur jaringan di mana setiap komputer terhubung dengan yang lain dalam bentuk lingkaran. Umumnya diterapkan di lingkungan kantor atau perusahaan. Karakteristik khasnya meliputi penggunaan kabel UTP dan *patch cable* yang membentuk jaringan melingkar dengan *node* yang disusun secara seri, seperti yang terlihat pada gambar 2.4. Untuk memastikan bahwa setiap perangkat dapat saling memperkuat sinyal, penguat sinyal akan ditempatkan di masing-masing *node* pada sentral dengan cara yang sesuai.



Gambar 2. 4 Topologi Ring

b. Topologi Bus

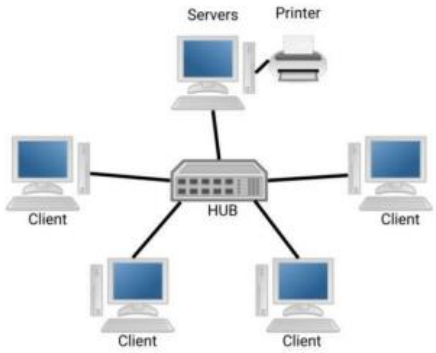
Topologi Bus adalah jenis struktur jaringan yang lebih sederhana dan sering digunakan dalam instalasi jaringan yang menggunakan kabel koaksial. Topologi ini umumnya diaplikasikan pada jaringan perkantoran berskala kecil. Karakteristik utamanya adalah penggunaan satu kabel tunggal yang menghubungkan seluruh jaringan. Seperti yang terlihat pada gambar 2.5 setiap komputer yang terhubung ke kabel utama memungkinkan pengiriman dan penerimaan paket data. Proses pengiriman paket data antar komputer terjadi ketika kabel utama tidak sedang digunakan oleh komputer lain, yakni pada saat tidak ada pertukaran data yang sedang berlangsung.



Gambar 2. 5 Topologi Bus

c. Topologi *Star*

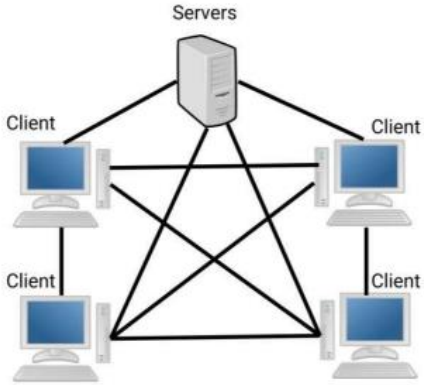
Topologi *Star* adalah jenis struktur jaringan yang membentuk pola seperti bintang dan menggunakan perangkat *hub* atau *switch* untuk menghubungkan antara klien. Topologi ini sering diimplementasikan dalam lingkungan perkantoran dalam skala kecil hingga menengah. Terlihat pada gambar 2.6 terdapat satu perangkat pusat yang mengatur semua aktivitas jaringan. Setiap komputer *host* terhubung langsung ke perangkat pusat, baik itu *hub* atau *switch*, melalui kabel sendiri dalam sistem *point-to-point*[18].



Gambar 2. 6 Topologi *Star*

d. Topologi *Mesh*

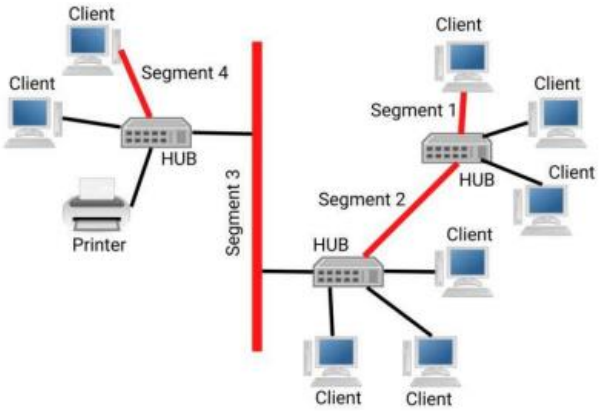
Topologi *Mesh* adalah jenis struktur jaringan di mana setiap perangkat komputer memiliki koneksi langsung dengan perangkat lainnya. Topologi ini digunakan dalam jaringan dengan banyak rute, memanfaatkan kabel tunggal untuk mempercepat proses pengiriman data tanpa memerlukan *hub* atau *switch*. Dikarenakan adanya koneksi langsung antara komputer-komputer, kecepatan transfer data meningkat karena tidak ada perantara. Ilustrasi pada gambar 2.7 menunjukkan bahwa proses transfer data antar komputer berlangsung lebih cepat karena setiap perangkat terhubung secara langsung melalui kabel.



Gambar 2. 7 Topologi *Mesh*

e. Topologi *Tree*

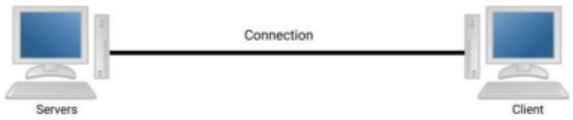
Topologi *Tree*, yang juga dikenal sebagai topologi pohon, merupakan perpaduan antara topologi bus dan topologi *star*. Ciri khas topologi *tree*, ditunjukkan pada gambar 2.8 yaitu adanya kabel utama yang menghubungkan beberapa *hub* dalam jaringan *star*. Topologi ini dibangun dengan struktur hierarkis di mana perangkat pusat, seperti *hub*, bertindak sebagai server yang mengatur aliran data di dalam jaringan.



Gambar 2. 8 Topologi *Tree*

f. Topologi *Peer to Peer*

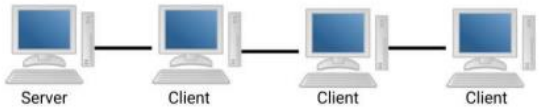
Topologi *Peer to Peer* adalah jenis jaringan yang simpel karena hanya melibatkan dua komputer yang terhubung menggunakan satu kabel.Topologi ini tidak memerlukan perangkat tambahan seperti *hub* atau *switch*, sebagaimana terlihat pada gambar 2.9. Kedua komputer terhubung langsung satu sama lain melalui satu kabel, memungkinkan komunikasi langsung di antara keduanya.



Gambar 2. 9 Topologi *Peer to Peer*

g. Topologi *Linier*

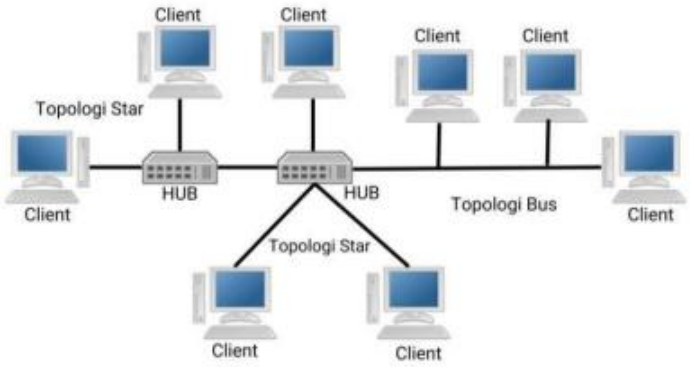
Topologi *Linier*, juga dikenal sebagai topologi bus berurutan, adalah jenis struktur jaringan yang menggunakan satu kabel utama sebagai penghubung untuk setiap titik sambungan di setiap komputer. Pada gambar 2.10 terlihat penggunaan konektor BNC dan kabel RJ 58, yang membentuk skema jaringan serupa dengan topologi bus[19].



Gambar 2. 10 Topologi *Linier*

h. Topologi *Hybrid*

Topologi *Hybrid* adalah struktur jaringan yang memadukan beberapa jenis topologi yang berbeda untuk membentuk jaringan yang baru. Seperti yang ditunjukkan pada gambar 2.11 topologi *hybrid* mengintegrasikan dua atau lebih jenis topologi jaringan yang berbeda sehingga membentuk satu sistem jaringan.



Gambar 2. 11 Topologi *Hybrid*

2.2.3 *Web Application*

Web Application adalah perangkat lunak yang dikembangkan melalui *coding* menggunakan bahasa pemrograman yang mendukung *platform* web seperti CSS, HTML, JavaScript, Java, Python dan bahasa pemrograman lain[16]. *Web Application* ini cenderung lebih hemat biaya dan terjangkau dibandingkan dengan aplikasi berbasis *desktop*. Dari perspektif fungsionalitas, *Web Application* memiliki beragam tujuan dan manfaat yang beragam. Banyak perusahaan yang berkembang menggunakan *Web Application* untuk mengelola data dan informasi sebagai bagian dari perencanaan sumber daya mereka[9].

a. Cara Kerja *Web Application*

Web Application membutuhkan server web untuk mengelola permintaan dari klien. Server bertugas menjalankan fungsi yang diminta dan memanfaatkan *database* untuk menyimpan informasi yang diperlukan. Secara umum, alur kerja dan prinsip operasi aplikasi web dapat dijelaskan sebagai berikut:

1. Pengguna: Proses dimulai saat klien mengirimkan permintaan ke server melalui internet, baik melalui web browser atau antarmuka pengguna aplikasi.
2. Web Server: Permintaan tersebut diteruskan oleh server web ke server aplikasi web yang relevan.
3. Proses: Server aplikasi web mengeksekusi tugas yang diminta, seperti pemrosesan data, dan kemudian menghasilkan output dari data tersebut.
4. Hasil: Server aplikasi web mengirimkan kembali informasi yang telah diproses atau diminta kepada pengguna.
5. Respon: Web server kemudian merespons klien dengan mengirimkan informasi yang diminta.

b. Kelebihan *Web Application*

Beberapa kelebihan menggunakan *Web Application* antara lain:

1. *Web Application* dapat diakses melalui web browser, sehingga tidak memerlukan proses instalasi yang rumit.
2. *Web Application* hanya membutuhkan sedikit ruang penyipanan pada perangkat *client*.

3. *Web Application* mengatasi masalah kompatibilitas (Windows, Mac, Linux), karena pengguna hanya memerlukan browser.

c. Kekurangan *Web Application*

Beberapa kekurangan menggunakan *Web Application* antara lain:

1. *Web Application* memerlukan pengkodean, sehingga mereka harus mengikuti standar tertentu agar dapat dijalankan oleh berbagai browser yang juga mengikuti standar tersebut.
2. *Web Application* bergantung pada koneksi ke server tempat aplikasi tersebut dijalankan, yang berarti koneksi internet harus tersedia setiap saat.
3. Banyak *Web Application* bergantung pada server tempat aplikasi tersebut dihosting.

2.2.4 Bahasa Pemograman

Bahasa pemrograman adalah kumpulan aturan sintaks dan semantik yang digunakan untuk merancang program komputer. Dengan bahasa pemrograman, seorang pengembang dapat secara spesifik menentukan data mana yang akan diolah oleh komputer.

Bahasa pemrograman terbagi menjadi beberapa jenis yang digunakan untuk berbagai tujuan, seperti pengembangan website, pembuatan *game*, dan pengembangan sistem operasi. Berikut adalah beberapa jenis bahasa pemrograman yang umum digunakan:

a. Bahasa Pemograman Python

Python adalah jenis bahasa pemrograman tingkat tinggi yang bersifat interpretatif dan berorientasi objek. Dikenal dengan sintaks yang sederhana dan mudah dipelajari, Python menekankan kemudahan dalam membaca kode serta mengurangi biaya perbaikan program. Python memiliki keunggulan sebagai bahasa pemrograman dinamis dengan manajemen memori yang otomatis[11]. Selain itu, Python dapat diterapkan untuk berbagai keperluan pengembangan perangkat lunak dan kompatibel dengan berbagai *platform* sistem operasi. Kelebihan lainnya adalah Python bersifat *open source*, dengan standar *library*-nya tersedia secara gratis[10].

Salah satu penerapan python dalam dunia jaringan yaitu penerapan otomasi jaringan untuk dapat melakukan konfigurasi perangkat secara otomasi. Otomasi jaringan dapat diimplementasikan sesuai dengan perintah yang ada pada program

python dengan didukung oleh *library* yang telah disediakan[3]. Fungsi *library* ini untuk memudahkan dalam membuat suatu program tanpa harus menulis banyak kode. Salah satu *library* yang mendukung dalam otomasi jaringan yaitu Paramiko.

```
#python
print "Hello, World!"
```

b. Bahasa Pemograman C

Bahasa Pemrograman C adalah salah satu bahasa pemrograman yang digunakan secara luas dalam pengembangan berbagai jenis aplikasi, termasuk sistem operasi, perangkat lunak antivirus, pengolah gambar, dan *compiler* untuk bahasa pemrograman lainnya. Meskipun dapat digunakan untuk membuat berbagai macam aplikasi, Bahasa Pemrograman C lebih sering digunakan untuk merancang aplikasi yang memiliki interaksi langsung dengan sistem operasi dan perangkat keras.

```
// C
#include <iostream>
using namespace std;
int main() {
  cout << "Hello, World";
  return 0;
}
```

c. Bahasa Pemograman C++

Bahasa Pemrograman C++ adalah bahasa pemrograman komputer yang merupakan pengembangan dari bahasa C yang dikembangkan oleh Bjarne Stroustrup. Proses pengembangan C++ dimulai di Bell Labs pada awal tahun 1970-an oleh Dennis Ritchie. C++ dibuat untuk menghadirkan beragam fitur yang tidak ada dalam C, dan juga untuk memberikan efisiensi serta mendukung pemrograman tingkat rendah. Salah satu peningkatan utama dalam C++ adalah adopsi konsep pemrograman berorientasi objek, seperti penggunaan kelas dengan fitur-fiturnya seperti *inheritance* dan *overloading*.

```
// C++
#include <iostream>
using namespace std;
int main() {
  cout << "Hello, World";
  return 0;
}
```

d. Bahasa Pemograman PHP

Hypertext Preprocessor (PHP) adalah jenis bahasa pemrograman yang beroperasi di sisi server, karena dijalankan pada komputer server. PHP dapat digunakan untuk membuat halaman web dinamis. Seiring berjalannya waktu, PHP telah berkembang menjadi salah satu bahasa pemrograman yang sangat diminati dan sering digunakan oleh situs web terkemuka seperti Facebook, Wikipedia, Joomla, WordPress, dan lainnya. Kemudahan dalam mempelajari dan menerapkan PHP membuatnya sangat diminati oleh banyak orang.

```php
<?php
echo "Hello World!";
?>
```

e. Bahasa Pemograman Java

Java merupakan bahasa pemrograman yang bersifat *multi-platform* dan dapat dijalankan di berbagai perangkat. Program Java dikompilasi menjadi *bytecode* yang dapat dieksekusi di dalam *Java Virtual Machine* (JVM). Kelebihan Java terletak pada kemampuannya untuk beroperasi pada berbagai sistem operasi karena sifatnya yang generik dan tidak terikat pada *platform* tertentu. Java adalah bahasa pemrograman yang berbasis objek, di mana program dibangun melalui pembuatan objek yang mewakili atribut dan perilaku khusus untuk menyelesaikan permasalahan yang diberikan. Java juga memiliki kemampuan dinamis dalam proses *linking*.

```java
// java
class HelloWorld
{
    public static void main(String args[])
    {
        System.out.println("Hello, World");
    }
}
```

2.2.5 *Secure Shell* (SSH)

Secure Shell (SSH) adalah sebuah protokol jaringan yang berjalan di lapisan aplikasi dalam model protokol TCP/IP. Untuk dapat menggunakan SSH perlu adanya autentikasi *user* berupa *username* dan *password* yang sudah terenkripsi[12]. Pada umumnya, dalam sistem komunikasi yang aman berbasis arsitektur *client-server*, enkripsi dan dekripsi otomatis dalam koneksi digunakan untuk menjamin kerahasiaan dan integritas data. SSH memungkinkan penggunaan komputer secara

remote dan pembuatan terowongan yang terenkripsi (*port forwarding*/ *tunneling*)[6]. *Port forwarding* memungkinkan pengalihan koneksi TCP yang tidak aman ke dalam koneksi SSH yang aman, sehingga koneksi dapat diarahkan dari satu IP ke IP lain seolah-olah klien terhubung langsung ke IP tujuan. Dengan SSH, sambungan aman dapat dibuat antara komputer lokal dan *remote*, memanfaatkan layanan yang tersedia.

Untuk dapat menggunakan SSH tersedia secara murah dan bahkan gratis untuk penggunaan non-komersial. SSH mempunyai dua versi yaitu SSHv1 dan SSHv2. SSHv2 merupakan versi terbaru dan paling aman dibanding dengan SSHv1.

Manfaat dari penggunaan SSH yaitu bisa digunakan untuk terowongan berbasis TCP aplikasi melalui SSH, misalnya email protokol, *tools* pemograman bahkan aplikasi bisnis seperti *oracle*. Terdapat keterbatasan dalam menggunakan SSH yaitu SSH tidak dirancang untuk dimasukkan ke dalam *gateway* jaringan seperti router maupun *firewall* sebagai solusi lengkap VPN.

a. SSH Server

SSH Server dapat memungkinkan sebuah SSH *client* dapat terkoneksi secara aman dan terenkripsi ke dalam perangkat jaringan. Fungsi dari SSH server yaitu untuk menerima permintaan (*request*) dari *client* dan mengeksekusi sebuah perintah yang diinstruksikan oleh *client*. Jika *client* meminta layanan (*request*). Maka server akan langsung menjalankan perintah yang diminta oleh *client*.

b. SSH *Client*

SSH *Client* dapat memungkinkan sebuah perangkat atau *host* dapat saling terkoneksi secara aman dan terenkripsi ke sebuah perangkat yang menjalankan SSH server. SSH *client* dapat meminta instruksi ke sebuah server dan dapat menjalankan perintah-perintah kepada SSH server untuk segera dieksekusi.

2.2.6 Otomasi Jaringan

Otomasi jaringan merupakan sebuah proses untuk mengotomasi konfigurasi, manajemen, pengujian dan operasi perangkat fisik maupun *virtual* dalam jaringan yang dikemas dengan bahasa pemograman tertentu. Dengan adanya otomasi jaringan dapat meminimalisir seorang *administrator* jaringan dalam membuat kesalahan konfigurasi, sehingga dapat meningkatkan efisiensi dan biaya

operasinya[13]. Untuk dapat menjalankan otomasi jaringan perlu dikemas dengan menggunakan sistem operasi yang dapat mendukung pengeksekusian bahasa pemograman tertentu. Dengan otomatisasi jaringan, seorang administrator tidak perlu lagi mengkonfigurasi perangkat secara manual. melainkan dengan membuat sebuah program saja dengan bantuan bahasa pemograman konfigurasi dapat diselesaikan dengan mudah dan jauh lebih cepat. Program tersebut dapat berisi alamat IP perangkat yang akan dikonfigurasi dan juga perintah-perintah tambahan yang ada dalam sebuah jaringan sesuai kebutuhan. Otomasi jaringan dapat diimplementasikan ke perangkat yang mendukung protokol SSH dan protokol *remote login* sehingga pekerjaan dapat terselesaikan lebih efisien dan lebih mudah untuk diimplementasikan kedalam jaringan yang berskala besar[14].

Otomasi jaringan dapat dijalankan dengan bahasa pemograman python dengan beberapa *library*-nya sehingga memungkinkan komunikasi yang disederhanakan dengan perangkat jaringan. Terdapat tiga *library* yang sering digunakan dalam otomasi jaringan yaitu Paramiko, Netmiko dan NAPALM.

a. Paramiko

Paramiko merupakan *library* python yang dapat digunakan dalam otomasi jaringan dengan menerapkan koneksi SSH yang dapat digunakan sebagai SSH *client* maupun server. Paramiko dapat melakukan otomasi konfigurasi pada perangkat jaringan seperti router maupun *multi-layer switch* yang terhubung dengan menggunakan koneksi SSH.

Library Paramiko terletak di *header* program seperti kebanyakan *library* pada umumnya, dengan menggunakan perintah "*import paramiko*" maka *library* paramiko dapat mengkompilasi fungsi yang ada pada program tersebut[15]. Untuk dapat masuk kedalam suatu perangkat jaringan, paramiko memerlukan informasi seperti alamat IP, *username* dan *password* sebagai proses pengenalan identitas.

b. Netmiko

Netmiko merupakan *library* python yang didasari oleh *library* paramiko yang mendukung koneksi SSH diberbagai *vendor* perangkat jaringan, tetapi netmiko tidak dapat mendeteksi perangkat *vendor* secara otomatis. *Library* netmiko memiliki fungsi yang sama dengan paramiko dalam proses otomasi perangkat jaringan.

c. NAPALM

NAPALM (*Network Automation and Programmability Abstraction Layer with Multi-Vendor Support*) merupakan suatu *library* yang dapat dimanfaatkan untuk otomasi jaringan dan pemograman. Salah satu fitur dari NAPALM adalah kemampuannya untuk mengelola konfigurasi dan mengumpulkan data dari perangkat jaringan. NAPALM sering digunakan untuk mendapatkan informasi seperti versi sistem operasi, *host name,* dan *list interface*. Selain itu, NAPALM memfasilitasi pengembalian perubahan dan penerapan konfigurasi yang berbeda untuk mendukung manajemen konfigurasi jaringan.

2.2.7 *Framework* Python

Framework Python adalah kumpulan modul dan paket yang membantu pengembang dalam menulis kode program menggunakan bahasa pemrograman Python. *Framework* membantu dalam pengembangan struktur program dengan menyediakan model penataan kode aplikasi *default* yang cepat, konsisten, dan mudah dikelola. Dengan menyediakan kumpulan urutan kode sumber tingkat tinggi, *framework* Python berguna dalam pengembangan sistem yang kompleks, serta menyediakan berbagai pustaka untuk membuat fitur yang unik. Penggunaan *framework* juga membantu pengembang menghemat waktu karena memungkinkan manajemen kode yang lebih konsisten dan mudah.

a. Django

Django merupakan sebuah *web framework* yang dapat digunakan untuk pengembangan website dengan menggunakan bahasa pemograman Python[8]. Dengan menggunakan *framework* dapat membantu membuat aplikasi web yang dinamis dan lebih efisien dibandingkan harus menulis kode dari awal. Django menggunakan model MTV yaitu *model*, *template* dan *view*. *Model* dapat diartikan sebagai *layer* yang digunakan untuk berinterkasi dengan *database*, sedangkan *template* diartikan sebagai *layer* presentasi yang digunakan pada bagian HTML, XML dan lainnya. *View* disini berisikan sebuah data dari *model* yang akan dikirimkan ke *template*. Django dikenal sebagai salah satu *web framework* yang sangat populer di kalangan *high-level framework*. Hal ini disebabkan oleh penggunaannya dalam bahasa pemrograman Python yang memiliki banyak *library* yang siap digunakan. *Library-library* tersebut dapat digunakan sesuai kebutuhan

pengembangan. Tujuan utama dari Django adalah untuk menyederhanakan pengembangan situs web serta manajemen basis data yang rumit[1]. Salah satu keunggulan utama dari Django adalah adopsi ORM (*Object Relational Mapper*) yang memungkinkan pengembang untuk mengelola *database* tanpa perlu menyesuaikan *query* secara manual jika terjadi perubahan pada struktur *database*.

b. Flash

Flash merupakan *framework* yang memfasilitasi *developer* untuk pengembangan proyek perangkat lunak web dan aplikasi dengan mudah dan cepat. Dikarenakan flash mempunyai *template script* siap pakai dan desain *modular* yang membuatnya dapat beradaptasi untuk berbagai macam proses pengembangan. Flash memiliki fitur yang cukup lengkap, termasuk *Built-in development server*, *Debugger* yang cepat dan dukungan terintegrasi untuk pengujian unit. Semua fitur tersebut dapat menunjang *developer* dalam meningkatkan kualitas web. Namun, Flash tidak mencakup fungsi standar seperti *database* dan validasi formulir. Sehingga, para pengembang perlu menggunakan fungsi yang disediakan pihak ketiga.

c. Tornado

Tornado merupakan *framework* dengan *library* jaringan asinkron, dengan menggunakan I/O jaringan *non-blocking*. Tornado dapat menjalankan hingga puluhan ribu koneksi. Dengan kata lain, tornado merupakan *framework* yang hadir dengan dukungan bawaan untuk aspek pengembangan web. Contohnya adalah *user authentication*, *web templates*, *localization* dan lainnya. Hal menarik dari tornado yaitu dapat digunakan bersama dengan *framework* lain.

d. Web2Py

Web2Py merupakan *full-stack framework* yang mendukung dalam pengembangan proyek web dengan cepat. *Framework* python ini dapat menyederhanakan prosedur pengembangan melalui server web, *database* SQL dan online UI. Web2Py memiliki sifat *cross-platform*, yang memungkinkannya digunakan di berbagai sistem operasi. Selain itu, Web2Py menyediakan fitur *Integrated Development Environment* (IDE) berbasis web yang dilengkapi dengan editor kode, *debugger*, dan kemudahan *deployment* dengan sekali klik untuk menyebarkan situs web dengan mudah.

2.2.8 Routing Protokol

Routing merupakan suatu proses yang dilakukan untuk meneruskan paket antar jaringan sehingga membentuk *rute* tertentu. Untuk dapat melakukan routing membutuhkan router, router dapat melanjutkan paket-paket data dari satu jaringan ke jaringan lainnya sehingga *host* antar *network* dapat saling berkomunikasi. Router juga melakukan analisis setiap paket data yang melewatinya, menentukan jalur terbaik untuk mencapai tujuan yang diinginkan, dan menyimpan informasi tersebut dalam sebuah tabel yang disebut *Routing Information Base* (RIB)[17]. Protokol routing memiliki kemampuan secara dinamis untuk mengumpulkan informasi dan memperbarui tabel routing saat terjadi perubahan dalam jaringan. Dalam konteks jaringan TCP/IP, routing dibagi menjadi dua jenis, yakni IGP (*Interior Gateway Protocol*) dan EGP (*Exterior Gateway Protocol*). IGP digunakan di dalam satu sistem jaringan independen atau yang sering disebut sebagai *autonomous system* (AS). Contoh dari routing protokol IGP adalah RIPv2, EIGRP, OSPFv3, dan IS-IS. EGP bekerja antara beberapa sistem jaringan independen atau AS. Contoh dari protokol routing EGP adalah EGP dan BGP. Algoritma routing dapat dibagi menjadi dua kategori, yaitu *Distance Vector* dan *Link State Protocol*. *Distance Vector* adalah protokol yang mempertimbangkan jarak dan arah untuk menentukan jalur yang akan diambil. Sedangkan *Link State Protocol*, seperti IS-IS dan OSPF, mengandalkan setiap router dalam jaringan untuk menyebarkan informasi tentang kondisi koneksi di setiap sambungan dalam AS.

AS merupakan kumpulan dari beberapa router yang beroperasi dalam satu sistem administrasi yang sama. Setiap AS memiliki nomor identifikasi yang unik, yang diberikan dan diatur oleh IANA (*Internet Assigned Numbers Authority*)[18]. Pemberian nomor identifikasi pada AS dimulai dari nomor 1 hingga 65.535. Rentang nomor identifikasi yang digunakan untuk AS privat berada antara nomor 64.512 hingga 65.535. Penting untuk diperhatikan bahwa dalam penggunaan nomor identifikasi AS privat, nomor AS tersebut tidak boleh tersebar ke luar jaringan AS. Hal ini karena dapat menyebabkan kekacauan dalam sistem pengalamatan AS.

a. *Enhanced Internet Gateway Routing Protocol* (EIGRP)

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. Algoritma ini bertujuan membangun jalur dengan biaya terendah ke setiap tujuan dalam jaringan, yang terdiri dari router (*node*) dan *link* (*edge*). Setiap jalur yang dibuat oleh algoritma DUAL adalah jalur bebas *loop*. Ini dilakukan oleh semua router yang terpengaruh oleh perubahan topologi, yang menghitung jalur terbaik yang baru dan memeriksa jalur menggunakan *Feasibility Condition* untuk memastikan tidak adanya *loop*. Dalam algoritma DUAL, terdapat beberapa jenis pesan, yaitu:

Dalam EIGRP, setiap rute menuju tujuan memiliki status, yakni PASSIVE atau ACTIVE. Status ini menunjukkan apakah rute telah diverifikasi bebas dari *loop* dan merupakan rute terpendek (PASSIVE), atau masih dalam proses verifikasi (ACTIVE).

Pada routing *Enhanced Interior Gateway Routing Protocol* (EIGRP), *metric* juga disebut sebagai *Feasible Distance* (FD). *Metric* ini digunakan oleh router EIGRP untuk menentukan kualitas atau biaya jalur menuju tujuan tertentu. EIGRP menggunakan beberapa parameter untuk menghitung *metric* FD, termasuk *bandwidth*, *delay* (tundaan), *reliability* (keandalan), dan *load* pada antarmuka jaringan yang terlibat.

Bandwidth adalah parameter yang menunjukkan kapasitas maksimum dari antarmuka jaringan. Semakin tinggi *bandwidth*, semakin baik kapasitasnya, dan akan menghasilkan nilai *metric* yang lebih rendah. *Delay* adalah parameter yang menunjukkan tundaan atau latensi yang dihadapi oleh antarmuka jaringan. Semakin rendah nilai *delay*, semakin cepat jalur tersebut, dan akan menghasilkan nilai *metric* yang lebih rendah. *Reliability* adalah parameter yang menunjukkan tingkat keandalan antarmuka jaringan. Semakin tinggi nilai *reliability*, semakin andal jalur tersebut, dan akan menghasilkan nilai *metric* yang lebih rendah. *Load* adalah parameter yang menunjukkan tingkat pemanfaatan atau beban pada antarmuka

jaringan. Semakin rendah nilai *load*, semakin sedikit beban yang dihadapi jalur tersebut, dan akan menghasilkan nilai *metric* yang lebih rendah.

Dalam EIGRP, *metric* FD dihitung dengan menggabungkan nilai-nilai parameter di atas menggunakan formula tertentu. Formula perhitungan *metric* FD EIGRP tidak disertakan di sini karena rumit dan dapat berbeda tergantung pada versi dan konfigurasi EIGRP yang digunakan.

Penting untuk diingat bahwa *metric* FD digunakan untuk menentukan jalur terbaik dalam routing EIGRP. Jalur dengan *metric* FD yang lebih rendah akan dipilih sebagai jalur terpendek dan dianggap jalur terbaik untuk mencapai tujuan. Jalur dengan *metric* FD yang lebih tinggi akan dianggap sebagai jalur alternatif (*backup*) yang akan digunakan jika jalur terbaik mengalami gangguan atau kegagalan.

b. *Routing Information Protocol* (RIP)

RIP adalah protokol yang mengadopsi algoritma *distance-vector*. Algoritma *distance-vector* adalah jenis algoritma routing di mana setiap *node* membentuk *array* satu dimensi yang berisi jarak (*cost*) ke semua *node* dalam jaringan dan menyebarkan vektor ke *node* tetangga secara langsung. Dalam algoritma *distance-vector*, setiap *node* memiliki informasi tentang *cost* ke setiap *node* tetangga yang terhubung langsung. Dalam konteks routing RIP, router bertukar informasi untuk menghitung jalur terpendek dalam jaringan Internet Protokol (IP). Setiap router yang menggunakan RIP diasumsikan memiliki satu atau lebih *interface*, di samping itu dianggap bukan sebagai router. Tabel routing dalam RIP diperbarui secara berkala dan diteruskan ke router tetangga. Namun, RIP memiliki keterbatasan di mana jalur terpanjangnya hanya dapat mencapai 15 hop.

c. *Open Shortest Path First* (OSPF)

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 dan tujuan, 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*. Dengan menggunakan algoritma Dijkstra, data LSA dalam *database* dihitung untuk

menghasilkan tabel routing OSPF. Untuk OSPF versi IPv6 (OSPFv3), proses routing tidak perlu lagi didefinisikan secara eksplisit.

Pada routing OSPF, *metric* disebut sebagai "*cost*" atau biaya. *Metric* ini digunakan oleh router OSPF untuk menentukan kualitas atau biaya jalur menuju tujuan tertentu. Perhitungan *metric* (*cost*) OSPF didasarkan pada *bandwidth* antarmuka jaringan.

Secara lebih rinci, untuk menghitung *metric* (*cost*) OSPF pada sebuah *link* atau antarmuka jaringan, langkah-langkah berikut diikuti:

Bandwidth (BW):

Bandwidth adalah parameter yang menunjukkan kapasitas maksimum dari antarmuka jaringan. Nilai *bandwidth* diukur dalam bps (bit per detik) dan menunjukkan seberapa banyak data yang dapat dikirim melalui antarmuka dalam satu detik. Semakin besar nilai *bandwidth*, semakin baik kapasitas antarmuka, dan akan menghasilkan nilai *cost* yang lebih rendah.

Perhitungan *Cost*:

Cost atau biaya pada OSPF dihitung menggunakan *formula inverse proportion*, yang berarti semakin besar *bandwidth*, semakin kecil nilai *cost*-nya. Rumus perhitungan *cost* OSPF (biasanya dinyatakan dalam satuan) adalah:

Cost = Reference Bandwidth / Bandwidth

Referensi *Bandwidth* (*Reference Bandwidth*) adalah nilai yang ditetapkan oleh administrator jaringan dan digunakan untuk menghitung *cost*. Secara *default*, referensi *bandwidth* OSPF adalah 100 Mbps, tetapi dapat diubah oleh administrator sesuai kebutuhan.

Sebagai contoh, jika kita menggunakan referensi *bandwidth* 100 Mbps dan antarmuka memiliki *bandwidth* sebesar 10 Mbps, maka perhitungan *cost*-nya adalah:

Cost = 100 Mbps / 10 Mbps = 10

Jadi, *cost* pada antarmuka tersebut adalah 10.

Penting untuk diingat bahwa *cost* atau *metric* pada OSPF menentukan jalur terpendek, di mana jalur dengan *cost* yang lebih rendah akan dipilih sebagai jalur terbaik menuju tujuan. Jalur dengan *cost* yang lebih tinggi akan dianggap sebagai

jalur alternatif (*backup*) yang akan digunakan jika jalur terbaik mengalami gangguan atau kegagalan.

d. *Border Gateway Protocol* (BGP)

BGP adalah sebuah protokol routing yang digunakan untuk pertukaran tabel routing antar AS. BGP memiliki kemampuan untuk mengumpulkan rute, bertukar rute, dan menentukan rute terbaik ke lokasi tertentu dalam sebuah jaringan. BGP termasuk dalam jenis protokol routing yang disebut EGP[19]. Keunggulan dari routing BGP yaitu mampu untuk menghindari terjadinya *loop path selection*, di mana paket data dapat terus-menerus dikirimkan melalui rute yang berputar-putar. Dalam menentukan rute, BGP menggunakan kebijakan administrasi jaringan. Perkembangan BGP terus berlanjut, dan saat ini telah mencapai BGP4 yang mampu mengakomodasi *Classless Inter-Domain Routing* (CIDR), sehingga membuat proses routing menjadi lebih efisien.

2.2.9 *Throughput*

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. Untuk dapat menghitung nilai *throughput* dapat dilihat pada persamaan 2.1.

$$Throughput = \frac{jumlah\ data\ yang\ dikirim\ (bytes)}{waktu\ pengiriman\ data\ (s)} \quad (2.1)$$

Tabel 2. 2 Standarisasi *Throughput* menurut TIPHON[20]

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

2.2.10 *Packet loss*

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. Untuk dapat menghitung nilai *throughput* dapat dilihat pada persamaan 2.2.

$$Packet\ Loss = \frac{(Paket\ data\ dikirim - Paket\ data\ diterima) x\ 100\%}{Paket\ data\ yang\ dikirim} \quad (2.2)$$

Tabel 2. 3 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

2.2.11 *Delay*

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[20]. Untuk dapat menghitung rata-rata nilai *delay* dapat menggunakan rumus pada persamaan 2.3. Tabel 2.4 merupakan standarisasi *delay* menurut TIPHON.

$$Delay\ rata - rata = \frac{total\ delay}{total\ paket\ yang\ diterima} \quad (2.3)$$

Tabel 2. 4 Standarisasi *Delay* menurut TIPHON[20]

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

2.2.12 *Jitter*

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. Nilai *jitter* dapat dihitung melalui persamaan 2.4.

$$Jitter = \frac{Total\ variasi\ delay\ (s)}{Total\ paket\ yang\ diterima} \quad (2.4)$$

Tabel 2. 5 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