BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 TINJAUAN PUSTAKA

Penelitian ini menggunakan referensi dari penelitian sebelumnya yang relevan sebagai pedoman dalam penyusunan, bertujuan untuk memahami dan menemukan relevansi dan menghindari kesamaan penelitian-penelitian sebelumnya dengan penelitian yang akan dilakukan.

Penelitian Arif Rahman Hakim pada tahun 2017, berjudul "PENERAPAN *LOAD BALANCING* PADA ROUTER PFSENSE BERBASIS FREEBSD". Penelitian ini bertujuan untuk mengimplementasikan teknik *load balancing* agar pembagian *bandwidth* pada pengguna mendapatkan layanan yang sama, penelitian ini diimplementasikan pada router pfSense. Penelitian ini tidak dilakukan analisis performa ataupun kinerja dari *load balancing*. Hasil dari penelitian ini adalah menggabungkan *bandwidth* dari dua ISP dan konfigurasi IP Address dari masing-masing ISP[12].

Penelitian yang dilakukan oleh Rahmad Ilahi, Muhammad Arif Fadly R., dan Erwin Setyo Nugroho pada tahun 2018 berjudul "Analisis Perbandingan Multi WAN Connection Menggunakan Cisco, Mikrotik dan pfSense". Tujuan penelitian ini adalah untuk menganalisis perbandingan performa dari masing-masing router dengan menerapkan *load balancing*. Pengujian dilakukan dengan meng-*capture* packet data pada interface 30 *client* yang melakukan video streaming phpMotion, kemudian dilakukan pengamatan pada parameter QoS terkait *throughput*, *packet loss*, *delay*, *jitter*, dan response *time*. Hasil pengujian menunjukkan bahwa perangkat Mikrotik dan pfSense memiliki performa yang lebih baik dalam parameter *throughput*, *packet loss*, *delay*, dan *jitter* dibandingkan dengan perangkat Cisco. Hasil pengujian juga menunjukkan bahwa respon request paling bagus terdapat pada router Mikrotik. Berdasarkan standar TIPHON, ketiga router

termasuk dalam kategori "sangat bagus" atau "bagus" pada parameter QoS yang diuji[13].

Penelitian Rahmat Fauzi, Yuliadi pada tahun 2019, berjudul “Penerapan *Load balancing* Pada pfSense berbasis FreeBSD”. Penelitian ini bertujuan untuk mengatur dan menggabungkan *bandwidth* dari dua ISP menggunakan teknik *Load balancing* agar mendapatkan pembagian *bandwidth* yang sama. Hasil dari penelitian didapat kecepatan koneksi internet dan akses menjadi lebih baik dan optimal dibanding hanya menggunakan satu ISP saja dan masing-masing perangkat yang terhubung mendapatkan secara otomatis penggabungan *bandwidth* dari kedua ISP. Penelitian ini hanya menerapkan *load balancing* pada pfSense dan tidak dilakukan analisis performa sesuai QoS[14].

Praja Risnaldy melakukan penelitian berjudul "ANALISA QOS (QUALITY OF SERVICE) ZEROSHELL PADA MEKANISME *LOAD BALANCING* DAN *FAILOVER*" pada tahun 2020 [15]. bertujuan untuk mengevaluasi bagaimana mekanisme *load balancing* dan *failover* pada router Zeroshell berfungsi dengan baik. Hal ini dilakukan dengan menggunakan metode weight round-robin. Hasil pengujian menunjukkan bahwa mekanisme *load balancing* memiliki kemampuan untuk meningkatkan nilai *throughput* dan diberi kategori "sangat bagus" pada parameter QOS seperti *throughput*, *packet loss*, dan *delay/latency* dengan indeks 4, sementara mekanisme *failover* memiliki nilai TIPHON dengan indeks 1[15].

Penelitian Daud Muhammad Tulloh, dkk. Pada tahun 2020 berjudul "ANALISIS JARINGAN INTERNET MENGGUNAKAN MIKROTIK DENGAN OPTIMALISASI *LOAD BALANCING* MENGGUNAKAN PARAMETER QoS"[16]. Tujuannya adalah mengukur kualitas jaringan internet dengan QoS pada router mikrotik yang diterapkan dengan *load balancing*. Parameter penelitian dengan *throughput*, *jitter*, *packet loss*, dan *delay* dengan mengirimkan data dan streaming video, serta membandingkan topologi yang sudah berjalan dengan topologi baru yang menerapkan teknik *load balancing*. Hasil penelitian menunjukkan nilai QoS pada topologi yang sudah berjalan kurang memuaskan

dengan rata-rata *delay* 157 ms yang termasuk kategori "bagus", sedangkan pada topologi baru, nilai *delay* yang didapat adalah 109,8 ms yang termasuk kategori "sangat bagus". Pada parameter *throughput*, baik topologi baru maupun topologi yang sudah berjalan memiliki nilai yang rendah karena kualitas saluran yang buruk, adanya interferensi, dan saluran yang *overload*[16].

Penelitian Firman Dwi Anggoro pada tahun 2021, berjudul "IMPLEMENTASI *LOAD BALANCING* MENGGUNAKAN RASPBERRY PI DENGAN ROUTEROS OPENWRT". Penelitian ini dilakukan dengan tujuan untuk mengimplementasi teknik *load balancing* dengan metode PPDIOO diimplementasikan pada router Openwrt. Penelitian ini dilakukan akibat koneksi internet yang tidak stabil. Penelitian ini juga melakukan analisis kualitas *load balancing* dengan pengujian ping ke google.com dan pengujian pada internet speed tester. Dengan menerapkan *load balancing*, *bandwidth* kedua ISP dapat bekerja dengan optimal dan efektif dibanding sebelum menggunakan *load balancing*, dengan menggunakan *load balancing* kendala koneksi *down* dapat teratasi[17].

Pada tahun 2022, Bayu Prasetyo melakukan penelitian yang berjudul "Analisis Perbandingan Quality Of Service Routeros Mikrotik Dengan Openwrt Menggunakan Metode *Load balancing*"[18]. Penelitian ini bertujuan untuk membandingkan kinerja dua router, yaitu mikrotik dan Openwrt, dengan menerapkan teknik *load balancing*. Penelitian ini menggunakan metode *load balancing* PCC, ECMP, dan Nth, serta menguji parameter *delay*, *packet* loss, *throughput*, dan *jitter* berdasarkan QOS dengan standar TIPHON. Hasil penelitian menunjukkan bahwa *load balancing* metode PCC pada Openwrt memiliki nilai *packet loss* dan *delay* yang termasuk dalam kategori "sangat bagus" dengan indeks 4. Sementara itu, *load balancing* metode Nth pada MikroTik memiliki nilai *throughput* yang termasuk dalam kategori "sangat bagus" dengan indeks 4. *Load balancing* pada metode PCC dan ECMP pada MikroTik serta *load balancing* metode PCC pada Openwrt memiliki nilai *throughput* yang termasuk dalam kategori "bagus" dengan indeks 3[18].

Tabel 2.1 Literature review dari penelitian terdahulu

No	Judul	Penulis	Tahun	Perbedaan	Hasil
1	PENERAPAN *LOAD BALANCING* PADA ROUTER PFSENSE BERBASIS FREEBSD	Arif Rahman Hakim[12]	2017	Tidak dilakukan analisis perbandingan dengan router lain	*Load balancing* bekerja dengan sendirinya pada router yang telah dikonfigurasi. Penggabungan ISP memberikan efektifitas kinerja yang selalu terhubung ke internet.
2	Analisis Perbandingan Multi WAN *Connection* Menggunakan Cisco, Mikrotik dan pfSense	Rahmad Ilahi[13]	2018	Analisis *load balancing* dilakukan pada router Cisco, Mikrotik dan pfSense	Perangkat mikrotik memiliki *load balancing* yang paling rata dalam membagi beban *traffic*. Dari segi QoS, perangkat mikrotik adalah yang terbaik untuk menangani multi WAN karena memiliki fitur yang *powerfull* seperti marking packet dan marking connection.
3	Penerapan *Load balancing* pada Router pfSense berbasis FreeBSD	Rahmat Fauzi, Yuliadi[19]	2019	Penelitian dilakukan hanya pada router pfSense dan tidak melakukan analisis menggunakan QoS	Perangkat mikrotik memiliki *load balancing* yang paling rata dalam membagi beban *traffic*. Dari segi QoS, perangkat mikrotik adalah yang terbaik untuk menangani multi WAN karena memiliki fitur yang *powerfull*

No	Judul	Penulis	Tahun	Perbedaan	Hasil
					seperti marking packet dan marking connection.
4	ANALISA QOS (QUALITY OF SERVICE) ZEROSHELL PADA MEKANISME *LOAD BALANCING* DAN *FAILOVER*	Praja Risnaldy[15]	2020	Penelitian dilakukan pada routerOS Zeroshell dan tidak dilakukan analisis perbandingan	Pada zeroshell, *load balancing* dan *failover* dapat berfungsi; menggunakan *load balancing* memiliki nilai *throughput* lebih tinggi dibandingkan menggunakan dua ISP tanpa *load balancing*. Nilai parameter *throughput*, *packet loss*, dan *delay* termasuk dalam kategori "sangat bagus", dan nilai *jitter* QoS termasuk dalam kategori "bagus". Nilai *delay* sistem *failover* termasuk dalam kategori "jelek".
5	ANALISIS JARINGAN AKSES INTERNET MENGGUNAKAN MIKROTIK ROUTER OS DI SMK TUNAS HARAPAN DENGAN OPTIMALISASI	Daud Muhammad Tulloh, M. Ficky Duskarnaen, ST, M.Sc, Hamidillah Ajie, S.Si., MT[16]	2020	Analisis dan penerapan *load balancing* dilakukan pada router Mikrotik	Perangkat mikrotik memiliki *load balancing* yang paling rata dalam membagi beban *traffic*. Dari segi QoS, perangkat mikrotik adalah yang terbaik untuk menangani multi WAN karena memiliki fitur yang *powerfull*

No	Judul	Penulis	Tahun	Perbedaan	Hasil
	LOAD BALANCING MENGGUNAKAN PARAMETER QoS				seperti marking packet dan marking connection.
6	IMPLEMENTASI *LOAD BALANCING* MENGGUNAKAN RASPBERRY PI DENGAN ROUTER OS OPENWRT	Firman Dwi Anggoro[17]	2021	Tidak dilakukan analisis perbandingan dengan router lain hanya menggunakan router Openwrt	Dengan diterapkannya *load balancing* kendala *Downtime* dapat teratasi. Dengan menggunakan *load balancing* dapat koneksi dapat memanfaatkan *bandwidth* dengan optimal.
7	Analisis Perbandingan Quality Of Service Routeros Mikrotik Dengan Openwrt Menggunakan Metode *Load balancing*	Bayu Prasetyo[18]	2022	Analisa dilakukan pada router mikrotik dan Openwrt	Nilai *packet loss* dan *delay* yang termasuk dalam kategori "sangat bagus" menggunakan *load balancing* menggunakan metode PCC, ECMP, dan Nth pada MikroTik dan Openwrt, sedangkan metode PCC dan ECMP pada MikroTik dan Openwrt menampilkan nilai *throughput* yang termasuk dalam kategori "bagus".

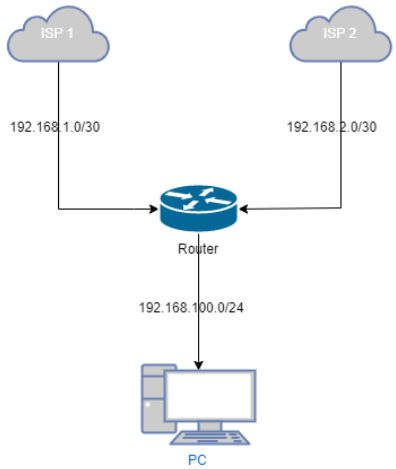
Referensi rujukan utama:

a. Analisis Perbandingan Multi WAN *Connection* Menggunakan Cisco, Mikrotik, dan pfSense[13].
b. Analisis Perbandingan Quality Of Service Routeros Mikrotik Dengan Openwrt Menggunakan Metode *Load balancing*[18].

2.2 LANDASAN TEORI

2.2.1 *Load balancing*

Load balancing adalah teknik untuk mendistribusikan beban *traffic* pada dua jalur atau lebih untuk diperoleh koneksi yang seimbang, *traffic* yang lebih optimal, meningkatkan *throughput data*, meminimalisir *delay*, dan agar tidak terjadi *overload*[20]. Dengan menerapkan *load balancing* diharapkan mampu menambah kinerja server agar lebih optimal. Metode *load balancing* dapat digunakan dalam membagi lalu lintas beban yang masuk ke jaringan melalui beberapa *gateway* jaringan yang tersedia sehingga tidak terpusat pada satu penyedia layanan internet (ISP)[5]. Setiap koneksi WAN(*gateway*) membagi bagian dari lalu lintas *traffic* dan membagi sama beban *bandwidth*. Topologi *Load balancing* seperti pada gambar 2.1.



Gambar 2.1 Topologi *Load balancing* multi WAN[12]

2.2.2 *Failover*

Failover merupakan suatu mekanisme dalam sistem jaringan untuk berpindah jalur *gateway* jika salah satu jalur mengalami kegagalan sehingga sistem membackup agar koneksi tidak terputus. *Failover* agar bisa berjalan dibutuhkan lebih dari satu jalur *gateway* yang digunakan sebagai cadangan. *Failover* secara otomatis langsung memastikan akses internet berjalan dengan baik melalui jalur *backup* jika jalur utama mengalami kendala[13].

2.2.3 *Gateway groups*

Gateway groups memungkinkan untuk pengelompokan *gateway* untuk tujuan *load balancing* ataupun *failover*. Pada *load balancing* pengelompokan beberapa *gateway* dengan beban*(weight)* yang ditentukan memungkinkan pemanfaatan *bandwidth* dengan seimbang dan meningkatkan kinerja dengan membagi beban lalu lintas(*gateway*) dengan *(Tier 1)* semua. Dalam kasus *failover* pengelompokan dengan prioritas yang ditentukan dapat mempengaruhi *gateway* ketika *gateway* utama *(Tier 1)* tidak tersedia sehingga *failover* mampu mengalihkan secara otomatis ke *gateway* alternatif *(Tier 2)* yang tersedia dalam *groups*. *Trigger level* merupakan pemicu perubahan *gateway* dalam grup *gateway*, dalam *gateway* Groups terdapat 4 jenis *Trigger level* yaitu:

1. *Packet loss*: Ketika paket hilang melebihi batas yang ditentukan maka lalu lintas akan beralih ke *gateway* alternatif yang tersedia.
2. *Latency*: Ketika waktu tunda melebihi batas yang ditentukan maka lalu lintas akan beralih ke *gateway* alternatif yang tersedia.
3. *Member Down*: Ketika sistem mendeteksi salah satu *gateway* tidak tersedia maka lalu akan beralih ke *gateway* alternatif yang tersedia.
4. *Packet loss and Latency*: gabungan dari *Trigger Packet loss* dan *latency*.

Pada umumnya konfigurasi grup *failover* menggunakan “Member *down*” dam untuk *load balancing* menggunakan *Trigger level “Packet loss” “Latency* atau *“Packet loss dan Latency”* sesuai kondisi yang ada[8].

2.2.4 *Multi WAN*

Multi WAN adalah teknik penggabungan koneksi internet dari dua atau lebih layanan ISP untuk meningkatkan kapasitas akses internet dan keandalan server. Skenario multi WAN dapat digunakan untuk *load balancing* atau *failover*[20].

2.2.5 *Bandwidth*

Bandwidth adalah besaran yang menunjukkan jarak total atau kisaran antara sinyal tinggi dan rendah pada saluran komunikasi. Selain itu, *bandwidth* menunjukkan jumlah data yang melewati jaringan.

Secara umum, *bandwidth* diukur dalam bps, atau bits per *second*. Semakin besar *bandwidth*, semakin baik koneksi jaringan[21].

2.2.6 PfSense

Gambar 2.2 Logo pfSense[8]

PfSense adalah sistem operasi *open source* berbasis linux turunan FreeBSD berdasarkan *packet filtering* dari *open*BSD yang sebelumnya dirancang sebagai IPFilter. PfSense dirancang sebagai *firewall* dan router pada jaringan komputer yang dapat dikelola dengan antarmuka web[8]. pfSense dapat digunakan untuk mengontrol lalu lintas jaringan dan dapat sebagai *firewall*, server DHCP, dan membuat VLAN, menginstal paket, dan bahkan menjalankan *Wire Guard* atau *OpenVPN*. pfSense juga dapat diintegrasikan dengan aplikasi tambahan seperti paket pemantauan sehingga dapat mengakses jaringan dimana saja[22].

2.2.7 OPNsense

Gambar 2.3 Logo OPNsense[9]

OPNsense adalah *virtual firewall* untuk platform *routing* dan *firewall berbasis open source* berdasarkan FreeBSD. OPNsense merupakan turunan dari pfSense dan proyek penggabungan dengan *monowall* yang dirilis resmi pada Januari 2015 dengan memiliki keunggulan lebih baik dari pfSense. OPNsense dapat digunakan seperti *firewall*, VPN, untuk mengintrol lalu lintas jaringan dan dapat digunakan untuk *load balancing*. OPNsense dapat berjalan pada hardware fisik, virtual mesin dan dapat berjalan pada *cloud*[9].

2.2.8 Bitbox *Open Network Appliance*

Bitbox adalah perangkat jaringan terbuka yang dapat diimplementasikan dengan bermacam skenario untuk menjalankan fungsi *firewall*, *routing*, *load balancer*, maupun sistem operasi untuk server seperti Windows, Linux dan FreeBSD perangkat ini juga mendukung teknologi virtualisasi seperti teknologi *Intel VT-x* sehingga kompatibel dengan *KVM*, *XEN*, *Hyper-V* dan *VMware* serta dapat digunakan sebagai *openstack* dan *containerization*[11].

2.2.9 Wireshark

Wireshark adalah perangkat lunak *packet-sniffing* yang dibuat oleh Gerald Combs pada tahun 1998 yang digunakan untuk menganalisis dan memecahkan masalah jaringan, dengan menggunakan *wireshark* memungkinkan untuk memantau lalu lintas jaringan. Wireshark memiliki beberapa fitur seperti inspeksi protokol, pengambilan langsung dan analisis *offline*, *multiplatform*[23].

2.2.10 *Quality of Service (QOS*)

Quality of Service merupakan teknik pengukuran yang menentukan tingkat kualitas layanan teknologi dapat beroperasi dan berjalan sesuai dengan standar kualitas layanan yang telah ditetapkan[24]. Menurut ITU-T E.800, QoS adalah sekumpulan efek kinerja yang menentukan tingkat kepuasan pengguna terhadap layanan yang diberikan oleh jaringan[25]. QoS mengacu pada pengelolaan lalu lintas data untuk mengurangi *packet loss*, *delay/latency* dan *jitter*. Parameter untuk mengukur QoS dalam jaringan yaitu: *bandwidth*, *throughput*, *packet loss*, *jitter* dan *delay*[26].

Parameter QoS digunakan untuk menilai tingkat kualitas penyedia layanan dan jaringan (seperti operator telekomunikasi), penting dipahami oleh pelanggan untuk membandingkan kualitas antara penyedia layanan. Kualitas jaringan internet dapat ditentukan berdasarkan nilai QoS sesuai yang ditetapkan oleh TIPHON. *Telecommunications and Internet Protocol Harmonization over Networks* (TIPHON) merupakan sebuah standarisasi yang dikeluarkan oleh *European Telecommunication Standards Institute* (ETSI) yang dijadikan sebagai standar untuk penilaian *Quality of Service*[27].

Tabel 2.2 Kategori QoS standar TIPHON

Kategori	Parameter				Indeks
	Throughput (%)	*Delay* (ms)	*Packet loss* (%)	*Jitter* (ms)	
Sangat Baik	100	< 150	0-2	0	4
Baik	75	< 250	3-14	1 s/d 75	3
Cukup Baik	50	< 350	15-24	76 s/d 125	2
Jelek	<25	< 450	> 24	126 d 225	1

2.2.11 *Throughput*

Throughput adalah jumlah total kedatangan paket yang berhasil yang diamati dengan sengaja selama interval waktu tertentu dibagi dengan durasi interval. *Throughput* merupakan kecepatan transfer data yang efektif dalam periode waktu tertentu, yang diukur dalam *bit per second* (bps). Semakin besar nilai *throughput* maka semakin baik kualitas jaringan tersebut[25]. *Throughput* dapat dipengaruhi oleh perilaku pengguna akhir, potensi pemrosesan komponen sistem, keterbatasan media fisik. *Throughput* dapat dinyatakan pada persamaan berikut:

$$Throughput = \frac{data\ paket\ yang\ dikirim(bit)}{durasi\ pengiriman\ data(second)} \quad (2.1)$$

2.2.12 *Packet loss*

Packet loss adalah parameter yang menunjukkan banyaknya paket yang hilang atau tidak sesuai dengan tujuan pada saat pengiriman data dari sumber ke tempat tujuan. Faktor-faktor seperti *overload* jaringan, gangguan sinyal, atau masalah perangkat jaringan adalah beberapa penyebab hal ini[25]. Persamaan berikut dapat digunakan untuk menghitung nilai *Packet loss*:

$$Packet\ loss = \frac{\text{paket data yang dikirim} - \text{paket data yang diterima}}{paket\ data\ yang\ dikirim} \times 100\% \qquad (2.2)$$

2.2.13 *Delay*

Delay atau *Latency* adalah lama waktu jeda yang dibutuhkan sebuah jaringan untuk mengirimkan paket data dari sumber ke tujuan. *Delay* dapat dipengaruhi oleh jarak, media fisik, kemacetan atau juga waktu pemrosesan yang lama[27]. Persamaan berikut dapat digunakan untuk menghitung nilai *delay*:

$$Delay = \frac{\text{waktu penerimaan paket} - \text{waktu pengiriman paket}}{\text{Total paket yang diterima}} \qquad (2.3)$$

2.2.14 *Jitter*

Jitter merupakan variasi penundaan berkaitan dengan *delay*[27]. *Jitter* terjadi akibat terdapat panjang antrean pengelolaan data, dan juga perbedaan waktu kedatangan paket ke penerima [16]. Semakin besar beban *traffic* maka menyebabkan semakin besar nilai *jitter*. Untuk mengetahui nilai *Jitter* dapat menggunakan persamaan berikut[28]:

$$Jitter = \frac{\text{Total variasi delay}}{\text{Total paket yang diterima} - 1} \qquad (2.4)$$

2.2.15 *Grafana*

Grafana merupakan *software open source* yang digunakan untuk memvisualisasikan dan menganalisis data matrik dan log dari sumber data perangkat. *Grafana* dapat membuat panel dan *dashboard* yang interaktif untuk memantau kinerja sistem, memvisualisasikan data bisnis, analisis log, pemantauan infrastruktur yang dapat di*monitoring* secara *realtime* dari berbagai sumber data. *Grafana* memiliki beberapa fitur dan konsep sebagai berikut[29] :

1. *Data Sources* pada *grafana* mendukung berbagai jenis sumber data, termasuk database seperti *MySQL*, *PostgreSQ*L, *InfluxDB*, *Prometheus*, serta layanan *cloud* seperti *Amazon Web Services* (AWS) *CloudWatch* dan *Google Cloud Monitoring* dan lain-lain.
2. *Panels*: panel dapat berisi grafik, tabel, atau matrik lainnya. Pengguna dapat mengatur panel sesuai kebutuhan dan memilih jenis visualisasi yang sesuai, seperti grafik garis, grafik batang, grafik lingkaran, atau peta.
3. *Queries*: untuk mendapatkan data dari sumber data, pengguna dapat menggunakan bahasa kueri yang sesuai dengan tipe sumber data yang digunakan.
4. *Dashboards*: *Grafana* dapat membuat *dashboard* yang terdiri dari beberapa panel dan dapat mengatur tata letak panel, menambahkan judul, membuat filter interaktif, dan menyesuaikan tampilan *dashboard* sesuai preferensi yang dibutuhkan.
5. *Alerts*: *Grafana* menyediakan fitur notifikasi dan pemantauan yang dapat diatur berdasarkan aturan yang ditentukan. Pengguna dapat membuat aturan pemantauan untuk memperingatkan ketika kondisi tertentu terpenuhi, seperti batasan nilai tertentu tercapai atau penurunan tiba-tiba dalam data.
6. Integrasi: *Grafana* memiliki integrasi yang luas dengan berbagai alat dan platform lainnya, seperti sistem manajemen kontainer seperti *Kubernetes*, alat pemantauan seperti *Prometheus* dan *Elasticsearch*, serta layanan *cloud* seperti AWS, *Azure*, dan *Google Cloud*. Hal ini memungkinkan pengguna untuk menggabungkan data dari berbagai sumber dan menciptakan visualisasi yang komprehensif.