KESATRIA: Jurnal Penerapan Sistem Informasi (Komputer & Manajemen)
Terakreditasi Nomor 204/E/KPT/2022 | Vol. 6, No. 2, April (2025), pp. 593-607

Perbandingan Performa Algoritma Load Balancing Least Connection Dan Source IP Hash Pada Web Server Berbasis Software Defined Network

Gihon Mahendra[1], Abd Hadi[2]
[1,2]Institut Teknologi dan Bisnis Asia Malang, Indonesia
E-mail: gihonmahendra@gmail.com[1], hadi@asia.ac.id[2]

Abstract

The increasing number of internet users demands more efficient server infrastructure, particularly in handling high client request volumes. One common issue faced by web servers is overload, which can lead to server unresponsiveness or even downtime. A solution to address this problem is the implementation of load balancing techniques. This study compares the performance of two commonly used load balancing algorithms, namely Least Connection and Source IP Hash, within a Software Defined Network-based web server architecture. The implementation was conducted using Mininet as a network simulator and POX as the controller. The testing environment consisted of three servers and three clients, with simulation scenarios involving 3,000 requests at three different connection rates: 75, 150, and 300 connections per second. The results indicate that the Least Connection algorithm is more adaptive to increasing loads, as it distributes requests based on the number of active connections on each server. This leads to more stable server responses and reduces the likelihood of packet loss. On the other hand, the Source IP Hash algorithm uses a hashing technique based on the user's IP address to direct requests to consistent servers, effectively maintaining session stability. However, it poses the risk of uneven load distribution when traffic dynamically changes. This algorithm also demonstrated better packet delivery stability (jitter), particularly under high load scenarios. Based on the analysis, it can be concluded that the Least Connection algorithm is more suitable for high and dynamic load network environments, while the Source IP Hash algorithm is better suited for systems requiring consistent client-server connection sessions.

***Keywords**: Performance, Least Connection Load Balancing Algorithm, Source IP Hash, Web Server, Software Defined Network*

Abstrak

Meningkatnya jumlah pengguna internet menuntut infrastruktur server lebih efisien, khususnya dalam menangani jumlah request yang tinggi dari client. Salah satu permasalahan umum yang dihadapi web server adalah terjadinya overload yang dapat menyebabkan server tidak responsif atau bahkan mengalami downtime. Salah satu solusi untuk mengatasi masalah ini adalah dengan menerapkan teknik load balancing. Penelitian ini membandingkan performa dua algoritma load balancing yang umum digunakan, yaitu Least Connection dan Source IP Hash, pada arsitektur web server berbasis Software Defined Network. Implementasi dilakukan menggunakan Mininet sebagai simulator jaringan dan POX sebagai controller. Lingkungan pengujian terdiri dari tiga server dan tiga client, dengan skenario simulasi yang melibatkan 3.000 request pada tiga variasi rate koneksi: 75, 150, dan 300 koneksi per detik. Hasil pengujian menunjukkan bahwa algoritma Least Connection lebih adaptif dalam menangani beban yang meningkat, karena mampu mendistribusikan permintaan berdasarkan jumlah koneksi aktif pada masing-masing server. Hal ini menghasilkan respons server yang lebih stabil serta menekan kemungkinan terjadinya packet loss. Sementara itu, algoritma Source IP Hash menggunakan teknik hash berdasarkan alamat IP pengguna untuk

ISSN: 2720-992X

mengarahkan permintaan ke server yang konsisten, yang efektif dalam menjaga kestabilan sesi koneksi, namun berisiko menyebabkan distribusi beban yang tidak merata saat trafik berubah secara dinamis. Algoritma ini juga menunjukkan kestabilan waktu pengiriman paket (jitter) yang lebih baik, terutama pada skenario beban tinggi. Berdasarkan hasil analisis, dapat disimpulkan bahwa algoritma Least Connection lebih sesuai untuk lingkungan jaringan dengan beban tinggi dan dinamis, sedangkan algoritma Source IP Hash lebih cocok digunakan pada sistem yang memerlukan konsistensi sesi koneksi antar client dan server.

Kata Kunci: *Performa, Algoritma Load Balancing Least Connection, Source IP Hash, Web Server, Software Definded Network*

1. Pendahuluan

Saat ini teknologi berkembang secara pesat dan berkesinambungan dari zaman ke zaman, terutama perkembangan teknologi pada era saat ini menyebabkan seluruh bidang yang ada menjadi lebih maju dari sebelum-sebelumnya. Hal ini dirasakan juga pada perkembangan jaringan, khususnya jaringan internet. Hal ini dapat dilihat dari semakin banyaknya pengguna yang terhubung ke jaringan internet di Indonesia berdasarkan hasil survei Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) bahwa pengguna internet pada periode 2022-2023 saat ini mencapai 215,63 juta orang, jumlah tersebut meningkat sebanyak 2,67% dibandingkan pada periode sebelumnya yang sebanyak 210,03 juta pengguna.

Perkembangan pengguna internet yang semakin meningkat khususnya di Indonesia, jika tidak diimbangi dengan infrastruktur yang memadai pastinya akan membawa dampak yang sangat signifikan (Narassati et al., 2023). Salah satu dampak yang pasti dialami adalah dengan banyaknya request yang dilakukan oleh pengguna internet dapat membuat sebuah web server (server yang berisikan data yang akan diakses oleh pengunjung website) mengalami overload dan pada akhirnya server tersebut akan down. Permasalahan tersebut yang saat ini sering terjadi pada sebuah web server diakibatkan oleh adanya request yang terjadi dalam satu waktu dengan jumlah yang banyak dan request tersebut hanya ditangani oleh satu web server saja (Karim et al., 2019).

Terdapat beberapa hal yang dapat digunakan untuk meminimalisir kegagalan dan sekaligus mengoptimalkan kinerja jaringan untuk mencapai proses distribusi beban yang merata agar sumber daya yang tersedia dapat dimanfaatkan secara maksimal, response time dapat meminimalkan, dan resiko overload pada jaringan dapat dicegah diantaranya yaitu meningkatkan spesifikasi web server, rutin melakukan pengecekan atau maintenance terhadap web server tersebut dan menggunakan konsep load balancing dengan arsitektur SDN (Rahmika, 2023). Didalam load balancing terdapat beberapa jenis algoritma yaitu ip hash, source ip hash, least connection, weight least connection, dan lain sebagainya.

Source IP Hash adalah algoritma load balancing yang mengambil source IP dari client untuk menghasilkan kunci hash yang unik. Kunci ini digunakan untuk mengalokasikan client ke server tertentu (Sumiati et al., 2020). Metode penyeimbangan beban ini dapat memastikan bahwa client diarahkan ke server yang sama dengan yang digunakan sebelumnya. Saat melakukan pengiriman Source IP Hash, fungsi Hash diterapkan pada alamat IP sumber dari permintaan yang masuk. Hal tersebut memastikan bahwa ketika suatu koneksi melewati perangkat, selama itu menggunakan alamat IP sumber yang sama, koneksi akan tetap persisten (Luthfi et al., 2023). Algoritma least connection melakukan pembagian beban berdasarkan banyaknya koneksi yang sedang dilayani oleh sebuah server (Hakim et al., 2019). Pada server yang memiliki kemampuan pemrosesan yang sama, algoritma penjadwalan least connection akan mendistribusikan beban permintaan

dengan baik karena permintaan yang panjang tidak akan disalurkan ke sebuah server (Nugroho et al., 2017).

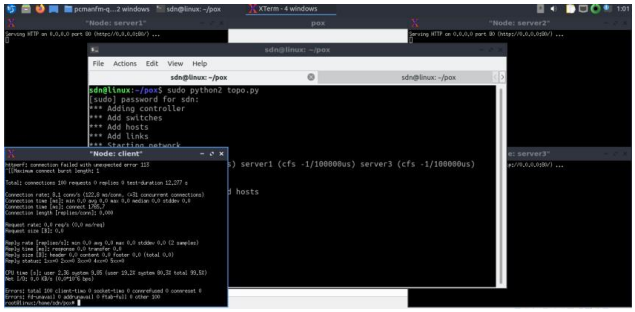
2. Metodologi Penelitian

Upaya untuk mendukung penelitian ini menggunakan metodologi, yaitu studi pustaka. Studi pustaka adalah proses pengumpulan data dan informasi melalui pembacaan literatur dan sumber-sumber tertulis seperti buku, jurnal, artikel, dan lain-lain yang relevan dengan topik atau masalah penelitian (Adlini et al., 2022). Studi pustaka dilakukan dengan cara mempelajari buku-buku referensi dan jurnal yang berkaitan dengan permasalahan penelitian sesuai dengan judul yang diangkat. teori yang dipelajari yaitu: load balancing, web server, least connection, source ip hash, software defined network dan sebagainya. Kemudian, melakukan analisis kebutuhan berkaitan dengan komponen yang dibutuhkan, seperti software dan hardware untuk kebutuhan analisis performa web server, melakukan perancangan topologi dan kebutuhan yang akan digunakan dalam melakukan analisis performa web server pada saat menggunakan teknik load balancing dengan algoritma Least Connection dan Source IP Hash, membuat atau menerapkan infrastruktur sesuai dengan topologi yang telah dirancang sebelumnya, pengujian dan evaluasi dilakukan untuk melihat performa dari web server dengan teknik load balancing menggunakan algoritma Least Connection dan Source IP Hash pada *software defined* network.

3. Hasil dan Pembahasan

3.1. Metode Load Balancing Dalam Penanganan Overload Web Server

Kondisi overload terjadi terutama karena beban berlebih pada CPU yang disebabkan oleh banyaknya client (Pradana et al., 2019). Seperti halnya pada laptop atau komputer, ketika banyak pengguna mengakses data atau program, CPU harus bekerja lebih keras. Jika kemampuan server tidak memadai untuk menangani beban ini, terjadilah overload. Saat web server mengalami overload, server menjadi lambat dalam merespons permintaan dari pengunjung, menyebabkan halaman web mungkin tidak dimuat dengan benar atau muncul kesalahan. Selain itu, overload dapat menyebabkan server tidak dapat diakses sama sekali. Kondisi ini ditunjukkan pada Gambar 1.



Gambar 1. *Error Server*

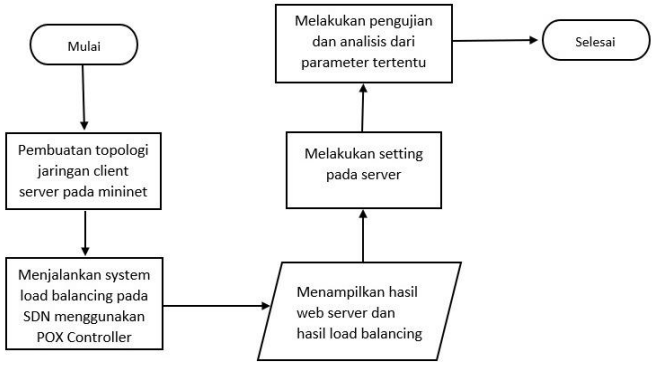
Jika kondisi tersebut berlangsung terus-menerus, bisa menyebabkan kerusakan pada hardware. Ada banyak cara untuk mengatasi masalah ini, salah satunya adalah menggunakan metode load balancing. Load balancing adalah teknik untuk mendistribusikan lalu lintas ke beberapa server, sehingga tidak hanya satu server yang menanggung beban. Dengan begitu, beban kerja server menjadi lebih seimbang.

3.2. Deskripsi Umum Sistem

Tahap awal melibatkan pembuatan topologi jaringan client-server pada simulator Mininet, dengan tiga server dan 1 client. Proses berikutnya melibatkan eksekusi web server menggunakan modul Python, sambil menjalankan sistem load balancing.

Penerapan load balancing menggunakan algoritma least connection dan source ip hash diatur melalui controller POX, yang berperan dalam mengelola distribusi trafik ke server.

Langkah selanjutnya mencakup pengaturan pada client yang akan diuji. Selama pengujian, setiap client berkomunikasi dengan web server yang telah dikonfigurasi sebelumnya. Setelah permintaan dari client kepada web server dilakukan, hasil dari perbandingan antara sistem load balancing dengan algoritma least connection dan algoritma source ip hash akan ditampilkan pada masing-masing client. Tahap terakhir melibatkan analisis hasil permintaan dari client, dengan mempertimbangkan parameter seperti transaction rate, throughput dan CPU usage. Analisis ini memberikan Gambaran komprehensif terkait kinerja load balancing dalam konteks penggunaan algoritma least connection dan source ip hash pada arsitektur SDN.



Gambar 2. Diagram Alur

Diagram alir Gambar 2 menunjukkan langkah-langkah implementasi dan pengujian sistem load balancing pada jaringan SDN menggunakan POX Controller sebagai controller, dimulai dari pembuatan topologi jaringan client-server pada Mininet, menjalankan sistem load balancing, menampilkan hasil web server dan load balancing, melakukan pengaturan pada server, serta menguji dan menganalisis parameter tertentu, sebelum akhirnya menyelesaikan proses. Diagram ini memberikan panduan sistematis untuk memastikan distribusi beban server yang efisien dan optimal dalam jaringan SDN.

3.3. Analisis Kebutuhan

Analisis kebutuhan sistem dibagi menjadi dua yaitu kebutuhan fungsional dan kebutuhan non-fungsional. Kebutuhan fungsional sistem yaitu controller yang dapat menjalankan *Load balancing* menggunakan POX Controller, switch dapat menerima paket dari client, web server dapat memberikan respons terhadap request dari client, client dapat mengirimkan request terhadap web server. Kebutuhan non-fungsional meliputi kebutuhan perangkat lunak dan kebutuhan perangkat keras. Perangkat lunak yang digunakan pada penelitian antara lain:

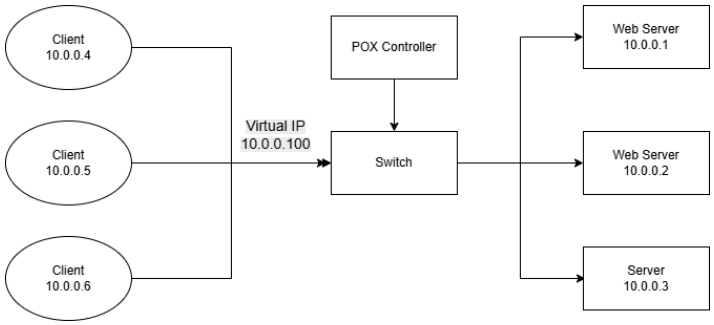
1. Sistem operasi Lubuntu 20.04.
2. Mininet sebagai emulator.
3. Iperf sebagai alat benchmarking.
4. Menggunakan POX Controller sebagai controller
5. Controller dapat menjalankan modul load balancing menggunakan POX Controller.
6. Switch dapat menerima paket dari client.
7. Web server dapat memberikan respon terhadap request dari client.
8. Client dapat mengirimkan request terhadap web server.

Kebutuhan perangkat keras yaitu 1 buah laptop dengan spesifikasi:

1. Laptop dengan processor AMD Ryzen 9 6900hx @4.9GHz (4 CPUs).
2. RAM 16 GB.
3. SSD dengan kapasitas 1024 GB.

3.4. Perancangan Sistem

Perancangan sistem jaringan dimulai dengan mendefinisikan topologi yang akan digunakan dalam penelitian. Topologi yang dirancang terdiri dari tiga client terhubung ke satu switch, yang selanjutnya dihubungkan ke satu controller dan tiga server. Dalam penelitian ini menerapkan dua load balancing, yaitu algoritma *Least Connection* dan *Source IP Hash.* Algoritma *Least Connection* mendistribusikan beban berdasarkan jumlah koneksi aktif pada setiap server, sehingga traffic diarahkan ke server yang sedang memiliki beban lebih ringan. Di sisi lain algoritma *Source IP Hash* melakukan pemetaan *traffic* secara konsisten dengan mengarahkan permintaan dari setiap alamat IP ke server tertentu.



Gambar 3. Topologi Jaringan

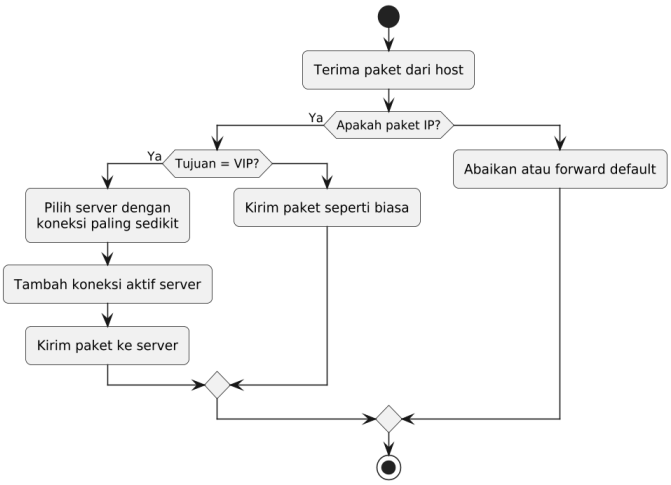
Pada Gambar 3 merupakan topologi yang akan dijalankan dalam mininet, sistem jaringan yang terdiri dari tiga klien, satu switch, satu controller, dan tiga server, yang mana konfigurasi ini mendukung pengujian parameter load balancing menggunakan algoritma Least Connection dan Source IP Hash. Topologi tersebut mengilustrasikan bagaimana ketiga klien terhubung ke switch, yang berfungsi mengatur lalu lintas dan mengarahkan koneksi ke server-server melalui controller, sehingga mendemonstrasikan pembagian beban antara server berdasarkan jumlah koneksi aktif (Least Connection) dan konsistensi routing berdasarkan alamat IP (Source IP Hash). Konsep topologi ini memberikan arahan yang jelas serta struktur yang terintegrasi, memungkinkan simulasi performa jaringan secara nyata di Mininet.

3.5. Perancangan Algoritma

Perancangan algoritma dilakukan setelah perancangan sistem selesai. Algoritma load balancing yang digunakan yaitu Least Connection dan Source IP Hash yang dimana kedua algoritma ini bekerja dengan menentukan bobot kinerja dari setiap server. Algoritma ini akan dijalankan menggunakan POX controller.

3.5.1. Perancangan Algoritma *Least Connection*

Perancangan Algoritma Least Connection dirancang dengan mendistribusikan permintaan client ke server secara efisien (Erkamim et al., 2024). Algoritma bekerja dengan cara memonitor secara real-time jumlah koneksi aktif pada tiap server, sehingga setiap kali ada permintaan baru, sistem secara otomatis memilih server yang memiliki koneksi paling sedikit. Berikut merupakan flowchart *load balancing* dengan menerapkan algoritma *least connection:*

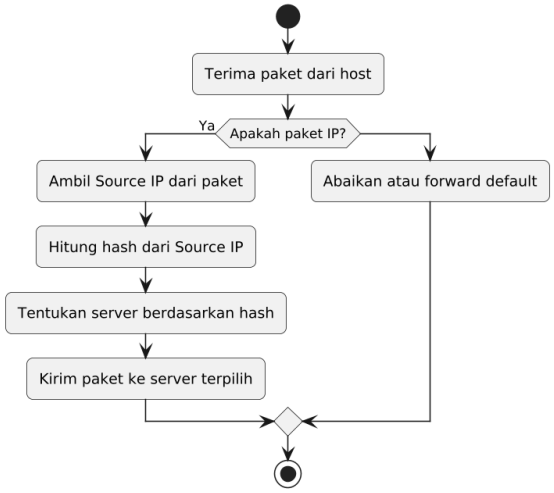


Gambar 4. Flowchart Algoritma Least Connection

Pada Gambar 4 mengGambarkan cara pengelolaan paket jaringan berdasarkan IP sumber, yang sangat penting dalam distribusi beban dan manajemen jaringan.

3.5.2. Perancangan Algoritma Source IP Hash

Algoritma *Source IP Hash* atau *Hash IP* Sumber adalah algoritma penyeimbangan beban yang menggunakan alamat IP sumber dan tujuan dari permintaan klien untuk menghasilkan kunci hash unik. Kunci hash ini kemudian digunakan untuk mengalokasikan klien ke server tertentu. Berikut adalah langkah-langkah perancangan algoritma Source IP Hash, Berikut merupakan diagram alur dari algoritma *Source IP Hash:*



Gambar 5. Flowchart Source IP Hash

Pada Gambar 5 paket-paket yang relevan yang diproses lebih lanjut untuk mendukung efisiensi distribusi beban dan manajemen jaringan secara keseluruhan.

3.6. Skenario Pengujian Sistem Load Balancing

Skenario pengujian sistem *load balancing* dengan menerapkan algoritma *Source IP Hash* dilakukan setelah evaluasi kinerja sistem untuk memastikan kesesuaian dengan kebutuhan yang telah dianalisis. Pengujian dalam penelitian ini akan melibatkan beberapa aspek sebagai berikut:

(a) Memberikan jumlah request pada server sebanyak 3000 request dengan rate 300, 150, dan 75 *conn/sec*.

(b) parameter yang menjadi fokus uji melibatkan Throughput (ms), Avg. Latency (ms), Jitter (ms) dan Packet Loss (%) sebagai indikator performa yang diukur dan dievaluasi

3.7. Install dan Konfigurasi Mininet

Instalasi Mininet pada sistem operasi lubuntu memungkinkan pengujian lingkungan jaringan virtual yang realistis. Mininet adalah perangkat lunak open-source yang berguna untuk mensimulasikan jaringan, terutama dalam konteks SDN. Dengan Mininet, pengguna dapat membuat topologi jaringan yang terdiri dari host, switch, dan router, serta melakukan eksperimen pengujian.

3.8. Instalasi POX Controller

Instalasi POX membuat implementasi controller SDN yang fleksibel dan mudah menyesuaikan. POX adalah perangkat lunak open-source yang digunakan untuk mengembangkan aplikasi dan mengelola controller pada jaringan SDN. Dengan menggunakan POX, pengguna dapat mengatur dan mengendalikan aliran lalu lintas dalam jaringan yang disimulasikan menggunakan Mininet (Iryani et al., 2021; Abidin, 2021).

3.9. Sistem *Load Balancing*

Sistem *load balancing* diimplementasikan dengan mengintegrasikan dua algoritma, yaitu Least Connection dan Source IP Hash, untuk mengoptimalkan distribusi trafik dan menjaga kestabilan koneksi. Metode Least Connection mendistribusikan permintaan dengan mengarahkan trafik ke server yang memiliki jumlah koneksi aktif paling sedikit, sehingga beban dapat terbagi secara merata dan menghindari kelebihan beban pada satu server (Riskiono & Pasha, 2020).

Sementara itu, algoritma Source IP Hash menggunakan teknik hashing terhadap alamat IP pengguna untuk memastikan bahwa setiap permintaan dari sumber yang sama selalu diarahkan ke server yang konsisten, yang membantu menjaga stabilitas sesi koneksi. Kombinasi kedua algoritma ini memungkinkan sistem untuk merespons beban kerja secara efisien dan meningkatkan performa jaringan dalam lingkungan dinamis.

3.9.1. Sistem *Load Balancing* Menggunakan Algoritma *Least Connection*

Sistem load balancing dioperasikan dengan menerapkan algoritma *Least Connection*, yang secara dinamis mengalokasikan traffic ke server dengan jumlah koneksi aktif paling sedikit. Algoritma ini bekerja dengan cara memonitor beban kerja tiap server secara real-time, lalu mengarahkan permintaan baru ke server yang memiliki koneksi yang lebih ringan, sehingga mencegah terjadinya overload pada salah satu server.

3.9.2. Sistem *Load Balancing* Menggunakan Algoritma *Source IP Hash*

Sistem load balancing dioperasikan dengan algoritma *Source IP Hash* untuk mengelola distribusi trafik secara optimal. Algoritma ini bekerja dengan mengonversi alamat IP sumber setiap permintaan ke dalam nilai hash, yang kemudian menentukan server target yang konsisten untuk setiap klien. Dengan metode ini, setiap permintaan dari alamat IP yang sama akan selalu diarahkan ke server yang sama, menjaga kestabilan dalam sesi komunikasi dan memastikan pembagian beban yang merata. Pendekatan tersebut sangat membantu dalam mengoptimalkan pemanfaatan sumber daya jaringan dan mencegah terjadinya overload pada salah satu server, sehingga keseluruhan sistem dapat beroperasi dengan peningkatan efisiensi dan keandalan yang signifikan.

3.10. Membuat Topologi Jaringan Pada Mininet

Setelah menyelesaikan instalasi Mininet, berikut adalah command untuk membuat topologi jaringan pada mininet:

Menjalankan untuk membuat topologi jaringan virtual yang terdiri dari enam host yang terhubung melalui satu switch Open vSwitch, dengan command yang ditunjukkan pada Tabel 1 dibawah ini.

Tabel 1. Masuk ke Direktori Ext

1.	sudo mn --topo single,6 --mac controller=remote,ip=127.0.0.1 --switch ovsk

Setelah menjalankan command yang ditunjukkan pada Tabel 4.11 adalah mengkonfigurasi setiap server menggunakan protokol HTTP pada port 80 yang ditunjukkan Tabel 2 dibawah ini:

Tabel 2. Menjalankan server

1.	h1 python3 -m http.server 80
2.	h2 python3 -m http.server 80
3.	h3 python3 -m http.server 80

3.11. Pengujian Sistem Load Balancing

Dalam melakukan pengujian sistem load balancing, dilakukan pengujian dengan skenario server yang menerima hingga 3000 koneksi, dengan rate sebesar 300, 150, dan 75 koneksi per detik untuk mengukur kestabilan dan efisiensi penanganan trafik. Selanjutnya, pengujian dilanjutkan pada sistem load balancing menggunakan algoritma *Least Connection dan Source IP Hash* yang mengutamakan distribusi beban berdasarkan jumlah koneksi aktif guna mengoptimalkan performa pada server yang memiliki perbedaan beban kerja, dan kedua, pengujian menggunakan algoritma Source IP Hash yang mendistribusikan trafik berdasarkan nilai hash dari alamat IP sumber, sehingga memastikan bahwa distribusi koneksi tetap merata meskipun terjadi fluktuasi trafik yang signifikan.

3.11.1. Pengujian Sistem Load Balancing Menggunakan Algoritma Least Connection

Dalam pengujian sistem load balancing menggunakan algoritma Least Connection, dilakukan evaluasi pada server yang mampu menerima hingga 3000 koneksi dengan variasi laju koneksi masing-masing 300, 150, dan 75 koneksi per detik. Algoritma ini mengarahkan setiap koneksi masuk ke server dengan jumlah koneksi aktif terendah, sehingga distribusi beban dapat disesuaikan dengan kapasitas masing-masing server untuk menjaga kestabilan dan efisiensi sistem (Nitisara, 2024).

Untuk meningkatkan akurasi serta mendapatkan data yang representatif, setiap skenario pengujian dijalankan sebanyak 10 kali percobaan, sehingga memungkinkan analisis tren performa melalui perhitungan nilai rata-rata dan identifikasi titik-titik kritis yang memerlukan penyesuaian konfigurasi guna mengantisipasi lonjakan trafik. Hasil pengujian ini memberikan Gambaran mendalam mengenai responsivitas sistem dalam mendistribusikan trafik secara optimal di bawah kondisi beban tinggi. Berikut merupakan hasil dari pengujian Throughput (Mbps), Avg. Latency (ms), Jitter (ms), dan Packet Loss dilakukan dengan 10 kali dengan server yang menerima 3000 koneksi dengan rate yang berbeda, yakni 300, 150, dan 75 koneksi per detik.

Tabel 3. Server Menerima Koneksi 3000 dengan Rate 300 Koneksi/Detik

No.	Throughput (Mbps)	Avg. Latency (ms)	Jitter (ms)	Packet Loss (%)
1.	283.14	22.63	2.6	0.14
2.	286.02	22.28	2.51	0
3.	285.96	21.65	2.87	0
4.	286.64	21.15	2.59	0.16
5.	284.01	21.95	3.21	0
6.	284.66	22.28	2.81	0

7.	283.67	22.58	3.11	0
8.	283.25	21.73	2.84	0
9.	283.64	21.08	2.85	0.07
10.	285.2	21.28	2.81	0
Rata - rata	**284.61**	**21.861**	**2.845**	**0**

Pada Tabel 3 Menunjukkan Pengujian load balancing menerapkan algoritma *least connection* yang diuji sebanyak 10 kali dengan client mengirimkan koneksi 3000 dengan rate 300 koneksi/detik ke server untuk mendapatkan data yang lebih akurat dan representatif. Hasil analisis menunjukkan bahwa rata-rata Throughput mencapai 284.619 Mbps, sementara Avg. Latency berada di angka 21. 861 ms, dengan Jitter rata-rata 2.845 ms, serta Packet Loss yang relatif rendah di 0%. Data ini memberikan Gambaran tentang bagaimana sistem mampu menangani lalu lintas jaringan dalam kondisi beban tinggi secara optimal, yang dapat menjadi acuan meningkatkan efisiensi sistem.

Tabel 4. Server Menerima Koneksi 3000 dengan Rate 150 Koneksi/Detik

No.	Throughput (Mbps)	Avg. Latency (ms)	Jitter (ms)	Packet Loss (%)
1.	143.54	14.91	2.01	0.1
2.	141.96	15.34	1.52	0
3.	143.53	15.32	2.26	0.1
4.	142.69	15.51	2.21	0.08
5.	140.53	14.86	2.19	0
6.	142.38	14.52	1.95	0
7.	141.83	15.56	2.43	0
8.	141.74	14.19	1.6	0.04
9.	140.78	14.25	1.76	0.07
10.	144.37	15.11	2.22	0
Rata - rata	**142.46**	**14.95**	**2.01**	**0**

Pada Tabel 4 Menunjukkan Pengujian load balancing menerapkan algoritma *least connection* yang diuji sebanyak 10 kali dengan client mengirimkan koneksi 3000 dengan rate 150 koneksi/detik ke server untuk meningkatkan akurasi. Parameter yang diukur mencakup Throughput (Mbps), Avg. Latency (ms), Jitter (ms), dan Packet Loss (%), yang memberikan Gambaran tentang efisiensi sistem dalam mengelola lalu lintas jaringan. Hasil analisis menunjukkan rata-rata Throughput sebesar 142.46 Mbps, Avg. Latency sebesar 14.95 ms, Jitter sebesar 2.01 ms, serta Packet Loss yang sangat rendah, yaitu 0%, mengindikasikan bahwa sistem bekerja dengan stabil dan optimal dalam kondisi beban tinggi.

Tabel 5. Server Menerima Koneksi 3000 dengan Rate 75 Koneksi/Detik

No.	Throughput (Mbps)	Avg. Latency (ms)	Jitter (ms)	Packet Loss (%)
1.	71.3	12.67	0.9	0
2.	69.48	11.13	1.39	0
3.	72.54	12.76	0.72	0.2
4.	70.88	11.85	1.47	0
5.	72.35	11.53	1.3	0
6.	69.36	12.62	1.41	0.07
7.	70.01	11.76	1.13	0.02
8.	72.08	11.29	1.17	0.15
9.	72.93	12.59	1.22	0.06
10.	70.04	11.01	0.81	0
Rata - rata	**71.09**	**12.05**	**1.15**	**0.053**

Pada Tabel 5. Menunjukkan Pengujian load balancing menerapkan algoritma least connection yang diuji sebanyak 10 kali dengan client mengirimkan koneksi 3000 dengan rate 75 koneksi/detik ke server untuk mendapatkan data yang lebih akurat. Parameter yang diukur mencakup Throughput (Mbps), Avg. Latency (ms), Jitter (ms), dan Packet Loss (%), memberikan Gambaran tentang bagaimana sistem menangani lalu lintas jaringan dalam kondisi beban tinggi. Hasil analisis menunjukkan rata-rata Throughput sebesar 71.09 Mbps, Avg. Latency sebesar 12.05 ms, Jitter sebesar 1.15 ms, serta Packet Loss yang sangat kecil, yaitu 0.053%, menunjukkan bahwa sistem beroperasi dengan stabil dan efisien dalam menangani koneksi secara optimal.

3.11.2. Pengujian Sistem Load Balancing Menggunakan Algoritma Source Ip Hash

Dalam pengujian sistem load balancing menggunakan algoritma Source IP Hash, evaluasi kinerja dilakukan dengan mengukur empat parameter utama, yaitu Throughput (Mbps), Avg. Latency (ms), Jitter (ms), dan Packet Loss (Darso, 2024). Parameter-parameter ini tidak hanya diukur dalam satu kali percobaan, melainkan dengan dilakukan 10 kali percobaan secara independen. Pendekatan tersebut memungkinkan analisis statistik yang lebih mendalam, dengan perhitungan nilai rata-rata, deviasi standar, dan identifikasi tren performa pada berbagai kondisi trafik

Algoritma Source IP Hash yang digunakan mengarahkan setiap koneksi dari alamat IP yang sama ke server yang sama secara konsisten sehingga distribusi beban tetap stabil di setiap percobaan. Hasil pengukuran diharapkan memberikan Gambaran yang komprehensif mengenai kestabilan dan konsistensi kinerja sistem load balancing ini, sehingga memungkinkan para pengelola jaringan untuk menentukan langkah penyempurnaan yang tepat. Berikut merupakan hasil dari pengujian Throughput (Mbps), Avg. Latency (ms), Jitter (ms), dan Packet Loss dilakukan dengan 10 kali dengan server yang menerima 3000 koneksi dengan rate yang berbeda, yakni 300, 150, dan 75 koneksi per detik.

Tabel 6. Server Menerima Koneksi 3000 dengan Rate 300 Koneksi/Detik

No.	Throughput (Mbps)	Avg. Latency (ms)	Jitter (ms)	Packet Loss (%)
1.	273	22.17	3.02	1.14
2.	272.27	21.42	2.85	0.93
3.	274.18	21.47	2.52	1.1
4.	274.95	21.87	3.13	0.83
5.	271.58	21.18	2.93	0.86
6.	271.27	21.06	2.76	0.9
7.	273.65	21.51	2.71	0.95
8.	271.62	22.35	3.06	0.84
9.	274.42	21.61	3.05	1.1
10.	272.54	21.45	2.91	1.07
Rata - rata	**272.94**	**21.60**	**2.89**	**0.97**

Pada Tabel 6 Menunjukkan Pengujian load balancing menerapkan algoritma *source ip hash* yang diuji sebanyak 10 kali dengan server menerima koneksi 3000 dengan rate 300 koneksi/detik untuk mendapatkan data yang lebih akurat. Parameter yang diukur mencakup Throughput (Mbps), Avg. Latency (ms), Jitter (ms), dan Packet Loss (%), memberikan Gambaran tentang efisiensi sistem dalam mengelola lalu lintas jaringan dalam kondisi beban tinggi. Hasil analisis menunjukkan rata-rata Throughput sebesar 273.01 Mbps, Avg. Latency sebesar 22.07 ms, Jitter sebesar 3.08 ms, serta Packet Loss sebesar 1.06%, yang mengindikasikan bahwa sistem bekerja dengan stabil dan efisien dalam menangani koneksi secara optimal.

Tabel 7. Server Menerima Koneksi 3000 dengan Rate 150 Koneksi/Detik

No.	Throughput (Mbps)	Avg. Latency (ms)	Jitter (ms)	Packet Loss (%)
1.	135.41	15.01	1.72	0.39
2.	135.28	14.8	1.56	0.53
3.	135.25	15.69	1.7	0.68
4.	137.63	14.95	1.91	0.6
5.	135.82	15.88	1.65	0.36
6.	137	15.01	2.14	0.46
7.	137.81	15.74	1.72	0.6
8.	137.5	14.86	2.12	0.67
9.	136.31	14.81	2.25	0.34
10.	136.39	15.31	1.76	0.66
Rata - rata	**136.44**	**15.20**	**1.85**	**0.52**

Pada Tabel 7 Menunjukkan Pengujian load balancing menerapkan algoritma *source ip hash* yang diuji sebanyak 10 kali dengan server menerima koneksi 3000 dengan rate 150 koneksi/detik untuk mendapatkan data yang lebih akurat. Parameter yang diukur mencakup Throughput (Mbps), Avg. Latency (ms), Jitter (ms), dan Packet Loss (%), memberikan Gambaran tentang efisiensi sistem dalam mengelola lalu lintas jaringan dalam kondisi beban tinggi. Hasil analisis menunjukkan rata-rata Throughput sebesar 137.45 Mbps, Avg. Latency sebesar 14.67 ms, Jitter sebesar 2.02 ms, serta Packet Loss sebesar 0.48%, yang mengindikasikan bahwa sistem bekerja dengan stabil dan efisien dalam menangani koneksi secara optimal menggunakan algoritma yang diterapkan.

Tabel 8. Server Menerima Koneksi 3000 dengan Rate 75 Koneksi/Detik

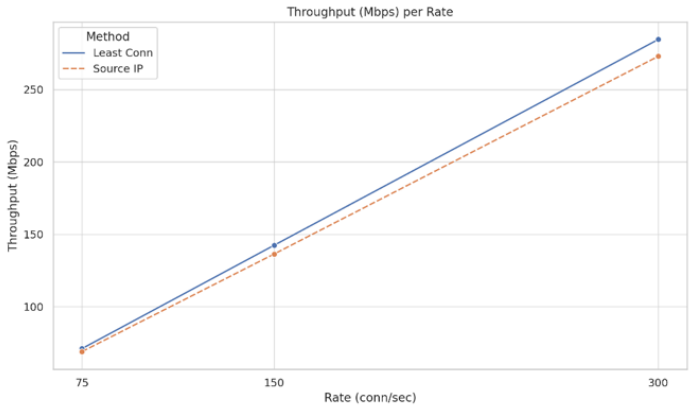
No.	Throughput (Mbps)	Avg. Latency (ms)	Jitter (ms)	Packet Loss (%)
1.	69.6	12.38	0.96	0.39
2.	70.13	11.93	1.47	0.37
3.	70.23	12.89	0.99	0.07
4.	67	11.17	0.7	0.33
5.	69.78	12.53	1.24	0.32
6.	69.15	12.61	0.62	0.11
7.	68.34	11.02	0.66	0.15
8.	70.15	11.35	1.03	0.39
9.	66.94	11.61	1.19	0.12
10.	68.91	12.37	0.81	0.35
Rata - rata	**69.02**	**11.98**	**0.97**	**0.26**

Pada Tabel 8 Menunjukkan Pengujian load balancing menerapkan algoritma *source ip hash* yang diuji sebanyak 10 kali dengan server menerima koneksi 3000 dengan rate 150 koneksi/detik untuk mendapatkan data yang lebih akurat. Parameter yang diukur mencakup Throughput (Mbps), Avg. Latency (ms), Jitter (ms), dan Packet Loss (%), memberikan Gambaran tentang bagaimana sistem menangani lalu lintas jaringan dalam kondisi beban tinggi. Hasil analisis menunjukkan rata-rata Throughput sebesar 69.02 Mbps, Avg. Latency sebesar 11.98 ms, Jitter sebesar 0.97 ms, serta Packet Loss sebesar 0.26%, yang menunjukkan bahwa sistem beroperasi dengan stabil dan efisien dalam menangani koneksi secara optimal menggunakan algoritma yang diterapkan.

3.12. Hasil Perbandingan Load Balancing Dengan Menerapkan Algoritma Least Connection dan IP Hash

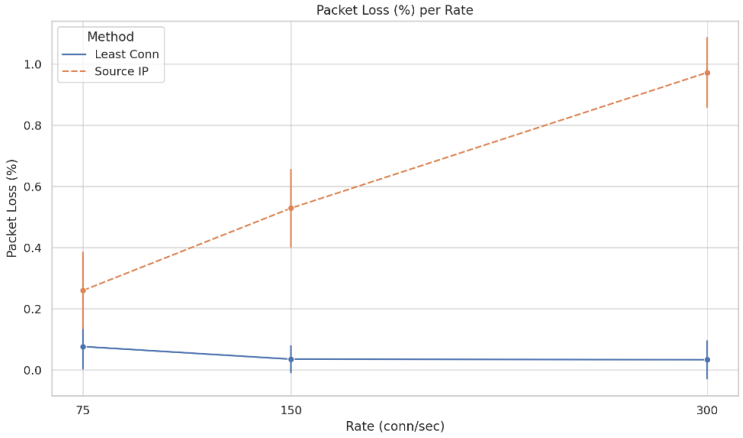
Pada penelitian ini hasil perbandingan performa load balancing dengan menerapkan algoritma Least Connection dan IP Hash Evaluasi dilakukan berdasarkan beberapa metrik

utama, termasuk Throughput, Avg. Latency, Jitter dan Packet Loss. Hasil pengukuran dari kedua algoritma ini ditampilkan dalam Gambar 6 berbentuk grafik yang memberikan Gambaran mengenai perbedaan kinerja masing-masing metode.



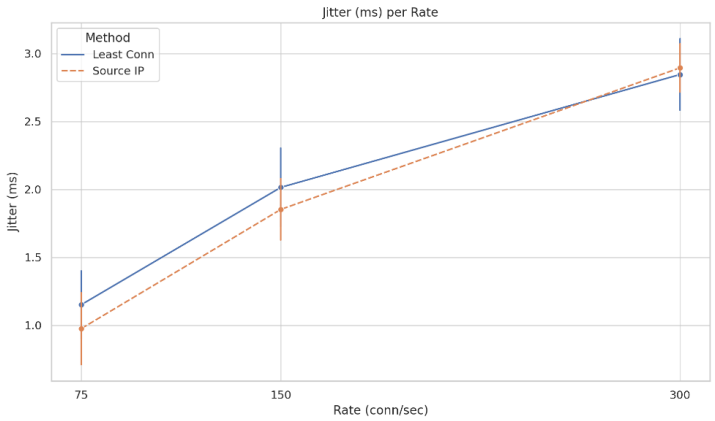
Gambar 6. Hasil Perbandingan Throughput (Mbps) per Rate

Berdasarkan hasil grafik pada Gambar 6, terlihat bahwa performa throughput antara algoritma Least Connection dan IP Hash menunjukkan perbedaan signifikan. Grafik tersebut mengilustrasikan bahwa saat koneksi sedang meningkat, algoritma Least Connection cenderung menghasilkan throughput yang lebih tinggi, yang mengindikasikan kemampuannya dalam mengalokasikan dan menangani beban yang terus bertambah secara lebih efisien. Sementara itu, meskipun IP Hash memberikan kestabilan pada kondisi beban rendah karena metode penetapan berdasarkan alamat IP, peningkatan throughput pada beban tinggi tidak terlihat optimal pada Least Connection. Selanjutnya adalah hasil grafik hasil perbandingan dari sisi packet loss yang ditunjukkan Gambar 5.



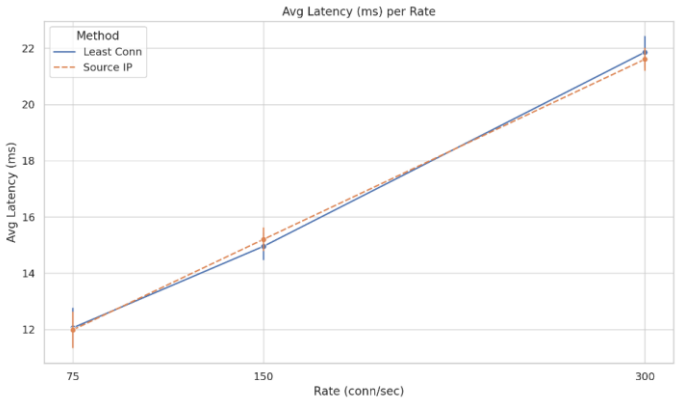
Gambar 7. Hasil Perbandingan Packet Loss (%) per Rate

Berdasarkan grafik Gambar 7, hasil Perbandingan Packet Loss (%) per rate. Terlihat bahwa perbandingan kinerja antara algoritma Least Connection dan IP Hash. ditinjau dari metrik packet loss pada tingkat koneksi yang berbeda yaitu 75, 150, dan 300 koneksi per detik. Grafik tersebut menunjukkan bahwa garis berwarna biru, yang mewakili *least connection*, mempertahankan nilai packet loss yang rendah dan stabil meskipun koneksi sedang meningkat, sedangkan garis putus-putus berwarna oranye, yang mewakili IP Hash, mengalami peningkatan packet loss seiring bertambahnya laju koneksi.



Gambar 8. Hasil Perbandingan Jitter (ms) per Rate

Berdasarkan grafik pada Gambar 4.5 hasil perbandingan jitter (ms) per rate, terlihat adanya perbedaan performa antara metode Least Connection dan Source IP ketika diukur pada metrik jitter. Sumbu-x grafik menunjukkan laju koneksi (dalam conn/sec) dengan pengukuran pada nilai 75, 150, dan 300, sedangkan sumbu-y menunjukkan nilai jitter dalam milidetik, berkisar antara 1,0 hingga 3,0 ms. Grafik ini memperlihatkan bahwa seiring dengan peningkatan laju koneksi, nilai jitter naik untuk kedua metode, namun dengan tren yang berbeda. Garis biru solid, yang mewakili Least Connection, menunjukkan kenaikan jitter yang lebih tajam, terutama pada laju koneksi tertinggi, dibandingkan dengan garis merah putus-putus yang mewakili Source IP. Hal ini mengindikasikan bahwa dalam skenario beban tinggi, pendekatan berdasarkan Source IP cenderung memberikan variasi waktu pengiriman paket yang relatif lebih stabil.



Gambar 9. Hasil Perbandingan Avg. Latency (ms) per Rate

Berdasarkan Gambar 9 Hasil Perbandingan Avg. Latency (ms) per Rate, grafik ini menunjukkan perbandingan nilai avg latency antara algoritma load balancing *Least Connection* dan *IP Hash*. pada variasi koneksi 75, 150, dan 300 koneksi per detik. Sumbu-x grafik mengGambarkan laju koneksi, sedangkan sumbu-y menampilkan nilai rata-rata latency dalam milisecond. Dari grafik tersebut terlihat bahwa peningkatan koneksi mengakibatkan kenaikan nilai latency pada kedua metode. Namun, penambahan latency untuk metode *IP Hash* cenderung lebih tajam dibandingkan dengan *Least Connection* saat beban mencapai kondisi tinggi, yang mengindikasikan bahwa strategi distribusi beban dinamis dari Least Connection lebih mampu menjaga performa jaringan dalam situasi trafik padat.

4. Kesimpulan

Berdasarkan hasil penelitian dan analisis grafik dari berbagai metrik performa, seperti rata-rata latency, jitter, packet loss, dan throughput, dapat disimpulkan beberapa hal penting terkait performa algoritma load balancing yang diuji. Pertama, algoritma Least Connection menunjukkan performa yang lebih baik dibandingkan dengan algoritma IP Hash dan Source IP dalam metrik latency, packet loss, dan throughput, terutama saat jumlah koneksi meningkat. Hal ini mengindikasikan bahwa strategi dinamis dari algoritma Least Connection lebih efektif dalam mendistribusikan beban secara lebih merata, sehingga mengurangi terjadinya penurunan performa pada server. Sebaliknya, algoritma IP Hash menunjukkan peningkatan yang lebih tajam pada latency dan packet loss ketika jumlah koneksi meningkat, yang mengindikasikan kurangnya fleksibilitas metode ini dalam menangani beban yang tinggi dan perubahan trafik yang dinamis. Sementara itu, dalam pengukuran jitter, algoritma Source IP menunjukkan performa yang lebih stabil dibandingkan dengan Least Connection, terutama pada kondisi beban tinggi. Hal ini menunjukkan bahwa dalam skenario-skenario tertentu yang sangat sensitif terhadap waktu pengiriman paket, algoritma Source IP bisa menjadi pilihan yang lebih baik, meskipun memiliki kelemahan dalam hal distribusi beban yang tidak merata.

Daftar Pustaka

[1] Abidin, N. Z. (2021). *Analisis Performansi Controller Pox Dan Ryu Pada Jaringan Software Defined Network Dengan Protokol Spanning Tree* (Bachelor's Thesis, Fakultas Sains Dan Teknologi Uin Syarif Hidayatullah Jakarta).

[2] Adlini, M. N., Dinda, A. H., Yulinda, S., Chotimah, O., & Merliyana, S. J. (2022). Metode Penelitian Kualitatif Studi Pustaka. *Jurnal Edumaspul*, *6*(1), 974-980.

[3] Darso, D., Mubarok, M. R., & Ramadhan, M. D. (2024). Analisa Kinerja Jaringan Nirkabel Universitas Amikom Purwokerto Berdasarkan Konsep Quality Of Service (Qos). *Storage: Jurnal Ilmiah Teknik Dan Ilmu Komputer*, *3*(4), 235-241.

[4] Erkamim, M., Prihatin, T., Saraswati, S. D., & Tonggiroh, M. (2024). Optimalisasi Throughput Pada Penerapan Load Balancing Dalam Jaringan Cloud Menggunakan Round Robin Dan Least Connection. *Journal Of System And Computer Engineering*, *5*(1), 13-23.

[5] Hakim, D. K., Yulianto, D. Y., & Fauzan, A. (2019). Pengujian Algoritma Load Balancing Pada Web Server Menggunakan Nginx. *Jrst (Jurnal Riset Sains Dan Teknologi)*, *3*(2), 85-92.

[6] Iryani, N., Ramadhani, A. D., & Sari, M. K. (2021). Analisis Performansi Routing Ospf Menggunakan Ryu Controller Dan Pox Controller Pada Software Defined Networking. *Incomtech: Jurnal Telekomunikasi Dan Komputer*, *11*(1), 73-84.

[7] Karim, A., Primananda, D., & Yahya, Y. (2019). Desain Dan Pengelolaan Jaringan Sdn Berbasis Pox Controller.

[8] Luthfi, H., Tulloh, R., & Iqbal, M. (2023). Perancangan Dan Implementasi Load Balancing Menggunakan Algoritma Least Connection Dan Ip Hash Pada Kubernetes. *Eproceedings Of Applied Science*, *9*(5).

[9] Narassati, D. A., Islam, H. F., & Putra, I. S. (2023). *Analisis Hubungan Digitalisasi Sistem Pembayaran Terhadap Pertumbuhan Ekonomi Indonesia*.

[10] Nitisara, N. M. (2024). *Implementasi Load Balancing Pada Sistem Software Defined Network (Sdn) Dengan Membandingkan Algoritma Fastest Dan Least Connection* (Doctoral Dissertation, Institut Teknologi Sepuluh Nopember).

[11] Nugroho, A., Yahya, W., & Amron, K. (2017). Analisis Perbandingan Performa Algoritma Round Robin Dan Least Connection Untuk Load Balancing Pada Software Defined Network. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *1*(12), 1568-1577.

[12] Pradana, A. M., Purboyo, T. W., & Latuconsina, R. (2019). Analisis Load Balancing Pada Jaringan Software Defined Network (Sdn) Menggunakan Algoritma Jaringan Syaraf Tiruan (Jst). *Eproceedings Of Engineering*, *6*(1).

[13] Rahmika, A. R. (2023). *Mekanisme Distribusi Beban Pada Load Balancing Web Server Dengan Algoritma Least Connection Dan Multi-Agent System Pada Lingkungan Virtual* (Doctoral Dissertation, Universitas Hasanuddin).

[14] Riskiono, S. D., & Pasha, D. (2020). Analisis Metode Load Balancing Dalam Meningkatkan Kinerja Website E-Learning. *Jurnal Teknoinfo*, *14*(1), 22-26.

[15] Sumiati, A., Trisnawan, P. H., & Fauzi, M. A. (2020). Implementasi Load Balancing Web Server Dengan Algoritma Source Ip Hash Pada Software Defined Network (Sdn). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *4*(3), 919-928.