JURNAL LITEK : Jurnal Listrik Telekomunikasi Elektronika, Vol. 19, No. 2, September 2022, pp. 57~62
pISSN: 1693-8097; eISSN: 2549-8762

ANALISIS UNJUK KERJA *LOAD BALANCING* F5 BIG-IP LTM MENGGUNAKAN ALGORITMA *LEAST CONNECTION* DAN *ROUND ROBIN* PADA *WEB SERVER*

Muhammad Ismail[1], Eka Wahyudi[2], Nanda Iryani[3]

[1,2,3)]Jurusan Teknik Telekomunikasi Institut Teknologi Telkom Purwokerto
Email: 15101023@ittelkom-pwt.ac.id[1], ekawahyudi@ittelkom-pwt.ac.id[2], nanda@ittelkom-pwt.ac.id[3]

Abstrak – Penelitian ini ditujukan untuk mengetahui kehandalan sebuah jaringan *web server* dengan *F5 BIG-IP LTM* sebagai *load balancer* yang membagi beban *traffic* yang masuk ke *web server*. Pada penelitian ini membandingkan penerapan algoritma *least connection* dan algoritma *round robin* pada *load balancing F5 BIG-IP LTM* dengan dua *web server*. Pengujian *load balancing* tersebut menggunakan *tool h2load benchmarking* dan *wireshark* serta melakukan pengujian dengan pengiriman jumlah beban *traffic* sebesar 2000, 4000, 6000, dan 8000 *request*. Hasil dari pengujian *Quality of Service* (QoS) diperoleh nilai performansi sangat baik pada ketiga parameter sesuai standarisasi TIPHON, dengan nilai *response time* sebesar 365,359 ms, 358,837 ms, 365,274 ms, dan 368,068 ms pada algoritma *least connection* sedangkan pada algoritma *round robin* diperoleh hasil 359,900 ms, 359,900 ms, 357,832 ms, dan 376,299 ms, nilai *throughput* sebesar 1876,761 Kbps, 1970,554 Kbps, 1965,507 Kbps, dan 1904,598 Kbps pada algoritma *least connection* sedangkan pada algoritma *round robin* diperoleh hasil 1829,785 Kbps, 1927,395 Kbps, 1884,955 Kbps, dan 1876,890 Kbps, dan nilai *delay* sebesar 1,35 ms, 1,38 ms, 1,40 ms, dan 1,43 ms pada algoritma *least connection* sedangkan pada algoritma *round robin* 1,36 ms, 1,40 ms, 1,42 ms, dan 1,46 ms. Dapat disimpulkan bahwa hasil pada algoritma *least connection* lebih unggul dibandingkan algoritma *round robin*.

Kata-kata kunci: Load Balancing, Web Server, F5 BIG-IP LTM, QoS, Response Time.

Abstract – This study aims to determine the reliability of a web server network with F5 BIG-IP LTM as a load balancer that divides the traffic load that enters the web server. This study compares the application of the least connection algorithm and the round robin algorithm on F5 BIG-IP LTM load balancing with two web servers. The load balancing test uses the h2load benchmarking tool and wireshark and performs testing by sending a total traffic load of 2000, 4000, 6000, and 8000 requests. The results of the Quality of Service (QoS) test obtained very good performance values for the three parameters according to the TIPHON standard, with response time values of 365.359 ms, 358.837 ms, 365.274 ms, and 368.068 ms on the least connection algorithm while the round robin algorithm obtained 359.900 ms, 359.900 ms, 357.832 ms, and 376.299 ms, the throughput value is 1876.761 Kbps, 1970.554 Kbps, 1965.507 Kbps, and 1904.598 Kbps on the least connection algorithm while the round robin algorithm results in 1829.785 Kbps, 1927.395 Kbps, 1884.955 Kbps, and 1876.890 Kbps, and the delay values are 1.35 ms, 1.38 ms, 1.40 ms, and 1.43 ms on the least connection algorithm while the round robin algorithm is 1.36 ms, 1.40 ms, 1.42 ms, and 1.46 ms. It can be concluded that the results of the least connection algorithm are superior to the round robin algorithm.

Keywords: Load Balancing, Web Server, F5 BIG-IP LTM, QoS, Response Time.

I. PENDAHULUAN

Beban *traffic* yang besar pada sebuah *website* sangat berpengaruh pada kinerja *server* yang menjadi tidak maksimal serta menimbulkan resiko *overload* atau bahkan kegagalan fungsi pada suatu *link* (*down*) pada jaringan *web server*. Sebagai langkah antisipasi dalam hal tersebut, maka ditemukan solusi dengan digunakan lebih dari satu *server* dan dilengkapi dengan protokol *load balancing*. Penambahan *server* merupakan langkah untuk memberikan *availability* pada jaringan ketika terjadi peningkatan beban *traffic,* dan mekanisme *load balancing* berfungsi untuk melakukan pembagian beban *traffic* pada *server* untuk memaksimalkan *resource* yang ada serta meringankan kinerja tiap *server*. Penelitian ini, akan mengimplementasikan mekanisme *load balancing* dengan menggunakan algoritma *least connection* dan *round robin*. Perangkat lunak yang dipakai dalam penelitian ini adalah *F5 BIG-IP LTM*, *VMware*, dan *h2load benchmarking*. *F5 BIG-IP LTM* nantinya akan berfungsi sebagai *load balancer* sedangkan *VMware* berfungsi sebagai *virtual machine* yang akan digunakan untuk memvirtualisasikan beberapa perangkat seperti *web server* dan *F5 BIG-IP LTM*. *Traffic* yang telah dikirimkan mengunakan *tool h2load benchmarking* akan menghasilkan *log file* kemudian dilakukan pengambilan data untuk penelitian. Pengujian menggunakan *h2load benchmarking* nantinya akan diberikan beberapa skenario untuk menguji sistem *load balancing* yang telah dirancang.

A. *Load Balancing*

Load balancing merupakan salah satu mekanisme untuk membagi beban *traffic* ke beberapa *server*. *Load balancing* mendistribusikan beban *traffic* pada dua atau lebih jalur koneksi secara seimbang, agar *traffic* dapat berjalan optimal. Layanan *load balancing* memungkinkan pengaksesan sumber daya dalam jaringan didistribusikan ke beberapa *host* lainnya agar tidak terpusat sehingga unjuk kerja jaringan komputer secara keseluruhan bisa stabil. *Load balancing* pada umumnya digunakan apabila *server* memiliki jumlah pengguna yang melebihi kapasitas maksimal dari yang dapat *server* tangani. *Load balancing* mampu mengurangi beban kerja setiap *server* sehingga tidak ada *server* yang memiliki beban yang berlebihan [1].

1. *Least Connection*

Algoritma *least connection* adalah algoritma yang melakukan pembagian beban berdasarkan banyaknya koneksi yang sedang dilayani oleh sebuah *server*. *Server* dengan pelayanan koneksi yang paling sedikit akan diberikan beban yang berikutnya akan masuk. *Least connection* memiliki skenario di mana jika dua *server* dalam sebuah *cluster* memiliki spesifikasi yang sama, satu *server* masih bisa mendapatkan *overload* jauh lebih cepat daripada yang lain [2].

2. *Round Robin*

Algoritma *round robin* adalah salah satu algoritma yang umum digunakan dalam *load balancing*. Algoritma ini berjalan dengan cara membagi beban *traffic* secara bergiliran dan berurutan dari satu *server* ke *server* lain. Konsep dasar dari algoritma *round robin* adalah dengan menggunakan *time sharing*. Setiap proses mendapatkan waktu CPU yang disebut dengan waktu *quantum* untuk membatasi waktu prosesnya, biasanya 1-100 ms. Algoritma ini memproses antrian secara bergiliran [3].

B. *Web Server*

Web server adalah *server* yang melayani layanan HTTP dari *web browser* dan mengirimkan kode dinamis ke *server* aplikasi. *Server* ini akan memproses kode dinamis menjadi kode statis dalam suatu halaman statis yang selanjutnya dikirimkan ke *browser* oleh *web server*. *Web server* biasanya disebut juga sebagai HTTP *server*, karena menggunakan protokol HTTP [4].

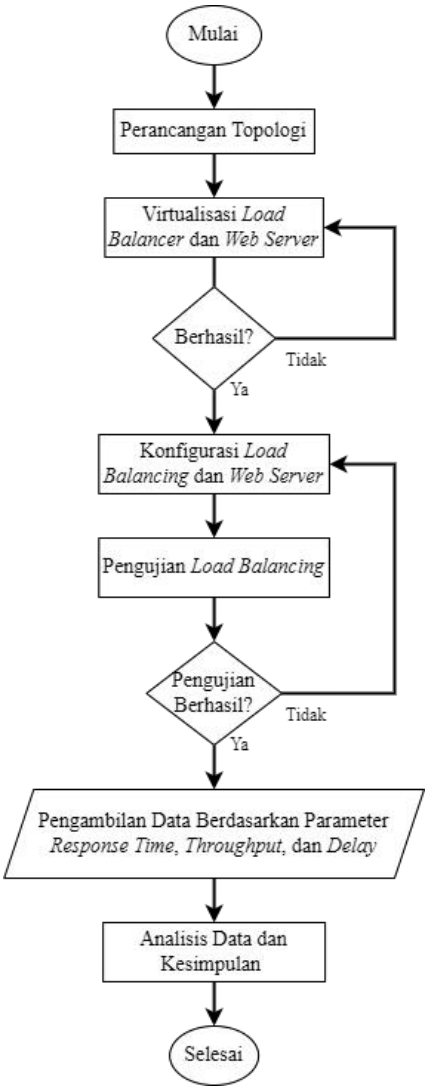
C. *F5 BIG-IP LTM*

F5 merupakan sebuah brand yang sedang mendominasi market *load balancer* dan juga *application delivery controller*. *F5* sendiri sudah berdiri sejak tahun 1996. Salah satu produk dari *F5* adalah *F5 Big-IP Local Traffic Manager* (LTM) yang pada dasarnya merupakan *load balancer*, namun memiliki keunggulan tersendiri dengan sejumlah besar fitur tambahan yang diantaranya dirancang untuk lebih memudahkan *operator* dalam mengontrol *traffic* jaringan, memilih tujuan *traffic* dengan benar berdasarkan peforma *server*, fitur keamanan, dan ketersediaan. Keunggulan dari *F5 Big-IP LTM* adalah *Full Proxy*, *Blazing fas SSL*, *TCP Optimization*, *Performance Optimization*, *Programmability*, *Scale*, dan *Speed* [5].

II. METODOLOGI

Penelitian ini merupakan penelitian kuantitatif yang dilakukan dengan pendekatan simulasi di mana didalamnya terdapat tahapan-tahapan sesuai diagram alir seperti pada Gambar 1. Dalam hal ini, akan dilakukan simulasi perancangan jaringan *web server* dengan protokol *load balancing F5 BIG-IP LTM* dengan konsentrasi penelitian tentang performansi jaringan pada dua beban *web server* menggunakan algoritma *least connection* dan algoritma *round robin*.

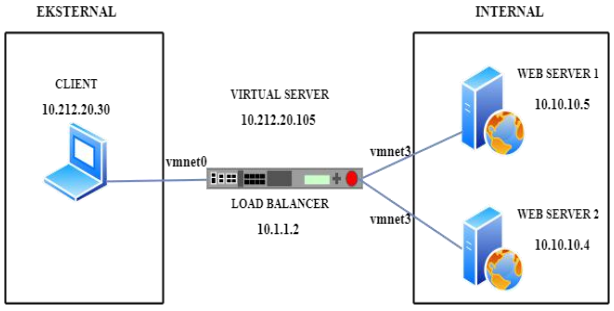
Pengujian dilakukan menggunakan tool *h2load* dengan 4 skenario uji yaitu 2000 *request*, 4000 *request*, 6000 *request*, dan 8000 *request* yang dilanjutkan pengambilan data menggunakan *tool wireshark* untuk selanjutnya dilakukan pengukuran *throughput*, *delay*, *response time*.



Gbr.1 *Flowchart* alur penelitian

A. Perancangan Topologi Jaringan

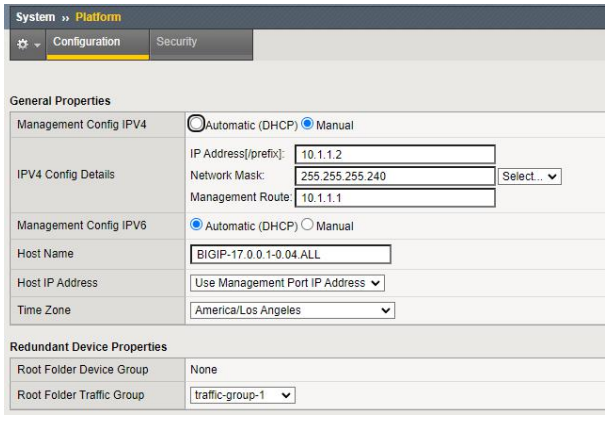
Gambar 2 menunjukkan skenario sebuah jaringan *web server* di mana topologi yang digunakan dalam penelitian ini merupakan topologi *hybrid* dengan menggunakan 1 perangkat fisik yang diposisikan sebagai *client* dan 3 perangkat *virtual* yang diposisikan sebagai *load balancer* dan *server*. PC 1 diposisikan menjadi *client* yang memiliki fungsi sebagai pengirim *traffic* yang nantinya akan mengirim paket *request* sesuai skenario pengujian menggunakan *tool h2load benchmarking*. Sedangkan pada PC 2 dilakukan instalasi *virtual machine* untuk memvirtualisasikan sebuah *load balancer F5 BIG-IP LTM* sebagai sistem *load balancing* yang memiliki fungsi sebagai pengatur beban antrian *traffic* yang masuk kedalam *server* berdasarkan algoritma *least connection* dan *round robin*, serta dua buah *web server* sebagai *user interface* layanan.



Gbr. 2 Topologi Jaringan

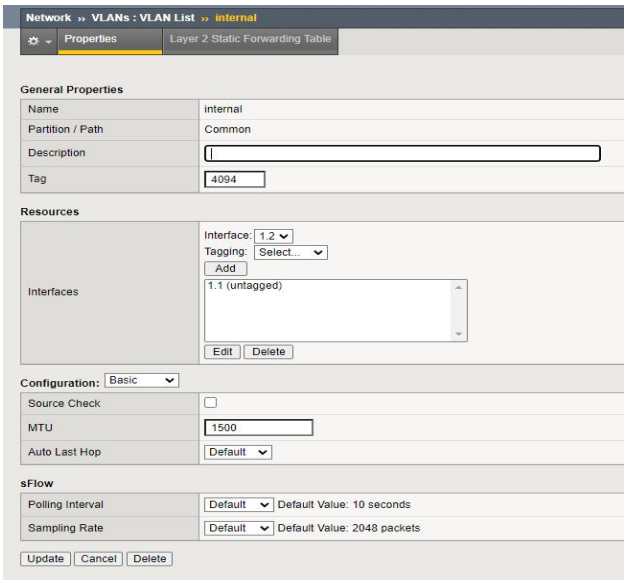
B. Konfigurasi *Load Balancer F5 BIG-IP*

Seperti yang terlihat pada Gambar 3, dilakukan konfigurasi *load balancer* dengan IP 10.1.1.2 sebagai IP dari *load balancer* dengan *subnet* 255.255.255.240 serta *hostname* BIGIP-17.0.0.1-0.04.ALL sebagai identitas *device* yang akan ditampilkan pada *interface* GUI dan *Host IP address* dengan *Use Management Port IP Address* yang menandakan bahwa *port* IP pada konfigurasi ini merupakan IP *management*.



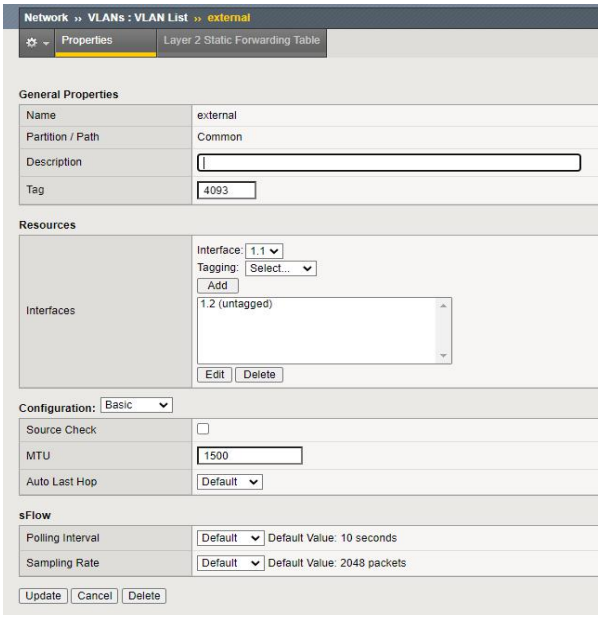
Gbr. 3 Konfigurasi IP *Management F5 BIG-IP LTM*

Konfigurasi dilanjutkan seperti yang terlihat pada Gambar 4. Pada menu VLANs dilakukan konfigurasi *internal network* dengan IP *address* 10.10.10.2 di mana nantinya akan menjadi *default gateway web server*. *Port lockdown* dipilih *allow default* dan VLAN *interfacenya* adalah 1.2 dengan kondisi *untagged*.



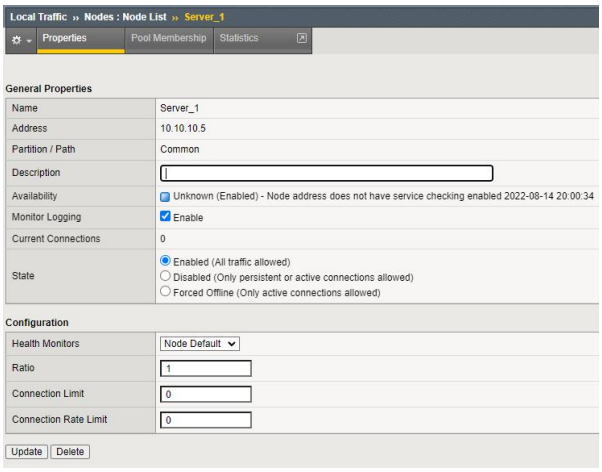
Gbr. 4 Konfigurasi Internal *Network*

Konfigurasi *external network F5 Big-IP LTM* seperti yang ditunjukkan pada Gambar 5, dilakukan konfigurasi *external* dengan IP *address* 10.212.20.2 yang akan menjadi *gateway client*. Untuk *port lockdown* diatur pada *mode none*. VLAN *interface* yang digunakan adalah 1.1 dengan kondisi *untagged*.



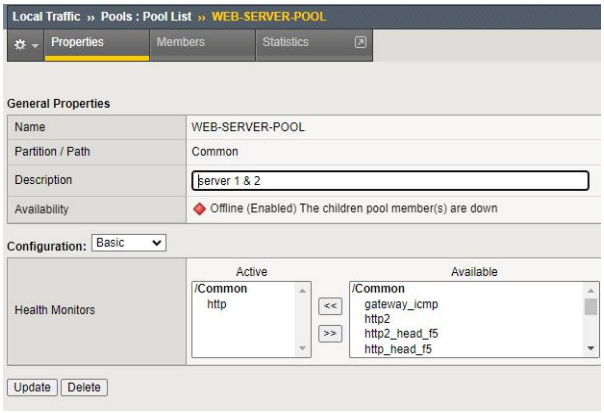
Gbr. 5 Konfigurasi Eksternal *Network*

Konfigurasi *node* sendiri merupakan tahapan dalam menambahkan *node server* yang akan digunakan untuk melayani *request client*. Seperti yang terlihat pada Gambar 6, dilakukan konfigurasi *node* pada *load balancer* dengan menjadikan *Server*_1 sebagai nama dari *node*, kemudian *node address* yang merupakan IP *address* dari *web server* 1 yaitu 10.10.10.5 dan 10.10.10.4 untuk *web server* 2.

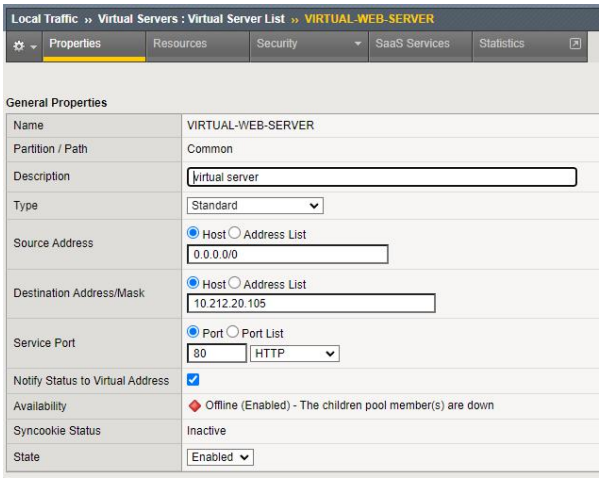


Gbr. 6 Konfigurasi *Node*

Gambar 7, dilakukan konfigurasi *server pool* pada *load balancing* dengan memilih WEB-*SERVER*-POOL sebagai nama dari *server pool* dan memilih *health monitor* yang akan digunakan, yaitu *http* di mana protokol tersebut digunakan pada kedua *server* yang menjadi *node* di dalam *web server pool*.



Gbr. 7 Konfigurasi *Server Pool*



Gbr. 8 Konfigurasi *Virtual Server*

Gambar 8 merupakan tahapan konfigurasi *virtual server* pada *load balancing* dengan *address* tujuannya adalah 10.212.20.150 yang berfungsi sebagai IP untuk kedua *web server* dengan *port* 80 yang merupakan port *service* HTTP. IP *address* 10.212.20.105 merupakan alamat IP yang akan dituju oleh *client* untuk selanjutnya dilakukan pengujian dengan mengirimkan beban *traffic* sesuai skenario yang telah ditentukan.

C. Skenario Pengujian dan Pengambilan Data

Pengujian ini menggunakan 4 skenario. Skenario pengujian yang diberikan dapat dilihat pada Tabel 1. Skenario pada masing-masing pengujian dilakukan sebanyak 10 kali, dengan variasi *traffic* 2000, 4000, 6000, dan 8000 jumlah koneksi dengan 100 *request* per detik dengan tujuan mendapatkan hasil rata-rata dari parameter *QoS* dan mendapatkan hasil waktu tercepat pada *load balancer* dalam menangani *request*. Aplikasi yang digunakan untuk skenario pengujian ini menggunakan *h2load benchmarking* untuk mendapatkan nilai *response time* dan kemudian data yang dihasilkan akan di-*capture* oleh aplikasi *wireshark* pada komputer *client* untuk mengetahui nilai *throughput* dan *delay*.

TABEL I
Skenario Pengujian Data

No	Jumlah Koneksi	*Request* per detik	Jumlah Skenario
1	2000	100	10
2	4000	100	10
3	6000	100	10
4	8000	100	10

III. HASIL DAN PEMBAHASAN

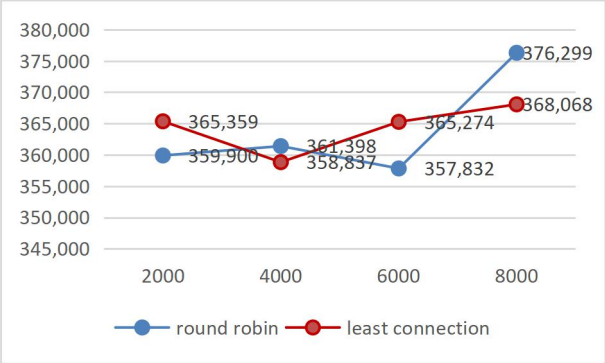
A. Hasil Pengujan *Response Time*

Pengujian *response time* bertujuan untuk mengukur keterlambatan respon *server* pada saat *client* mengirimkan *request* data yang dihitung dalam satuan *millisecond*. Semakin kecil nilai *response time* maka *load balancing* dapat melayani *request* atau koneksi dari *client* semakin cepat.

Pengujian dilakukan dengan mengirimkan *traffic* menggunakan *h2load benchmark* sebanyak 10 kali pada masing-masing skenario pengujian yang telah ditentukan, dan selanjutnya diambil nilai rata-rata pada tiap periode skenario. Hasil pengujian parameter *response time* menghasilkan nilai seperti yang terdapat pada Tabel 2 dan Gambar 9.

TABEL II
Nilai Parameter *Response Time*

No.	*Respon Time* (ms)				Keterangan
	2000 *request*	4000 *request*	6000 *request*	8000 *request*	
1.	365,3	358,8	365,2	368,0	*Least Connection*
2.	359,9	359,9	357,8	376,2	*Round Robin*



Gbr. 9 Diagram Hasil Uji *Response Time*

Gambar 9 menunjukkan diagram hasil pengujian *response time*. Pengukuran pertama pada jumlah koneksi 2000 *request*, rata-rata *response time* yang diberikan oleh *load balancing web server* dengan algoritma *least connection* adalah sebesar 365,359 ms sedangkan pada algoritma *round robin* adalah sebesar 359,900 ms. Pengujian pada saat jumlah koneksi 4000 *request*, rata-rata *response time* yang diberikan oleh *load balancing web server* dengan algoritma *least connection* adalah sebesar 358,837 ms dan pada algoritma *round robin* didapatkan hasil sebesar 361,398 ms. Pengujian selanjutnya dilakukan skenario pengujian 6000 *request*, di mana rata-rata *response time* yang diberikan oleh *load balancing web server* dengan algoritma *least connection* adalah sebesar 365,274 ms dan pada algoritma *round robin* adalah sebesar 357,832 ms. Skenario selanjutnya diberikan beban *traffic* sebesar 8000 *request* dan diperoleh hasil rata-rata *response time* yang diberikan oleh *load balancing web server* dengan algoritma *least connection* adalah sebesar 368,068 ms sedangkan pada algoritma *round robin* adalah sebesar 376,299 ms.

Nilai *response time* relatif bertambah sebanding dengan bertambahnya jumlah koneksi. Semakin tinggi jumlah koneksi maka semakin tinggi juga nilai *response time* yang diberikan oleh *load balancing web server*. Terdapat penurunan grafik pada pengujian 4000 *request*. Mengacu pada tabel 2, hasil pengujian yang diperoleh pada masing-masing algoritma menunjukkan nilai yang baik dengan memperoleh hasil kurang dari 1 *second*. Hasil pengujian menunjukkan bahwa algoritma *least connection* cenderung lebih stabil dibandingkan algoritma *round robin* dimana terdapat kenaikan grafik secara signifikan pada pengukuran 8000 *request*.

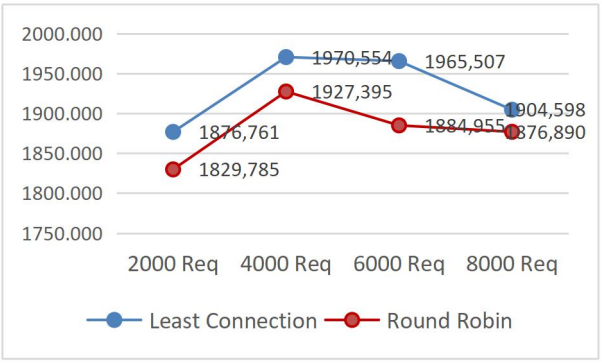
B. Hasil Pengujian *Throughput*

Hasil pengujian *throughput* pada Tabel 3 dan Gambar 10 menunjukkan nilai *throughput* pada penggunaan kedua algoritma tersebut terlihat tidak terlalu signifikan dan cenderung sama serta mengalami penurunan. Hasil pengujian menggunakan algoritma *least connection* memiliki rata-rata *throughput* yang lebih baik dibandingkan dengan algoritma *round robin*.

Nilai *throughput* tertinggi pada 4000 *request* dan cenderung mengalami penurunan seiring peningkatan *request*. Hasil yang ditampilkan pada grafik tidak jauh berbeda antara kedua algoritma. Algoritma *least connection* memiliki rata-rata *throughput* lebih tinggi dibandingkan dengan algoritma *round robin* pada tiap skenario, seperti yang terlihat pada grafik saat dikirimkan *request* sebesar 6000 *request*.

TABEL III
Nilai Parameter *Throughput*

No.	*Throughput* (Kbps)				Keterangan
	2000 *request*	4000 *request*	6000 *request*	8000 *request*	
1.	1876,7	1970,5	1965,5	1904,5	*Least Connection*
2.	1829,7	1927,3	1884,9	1876,8	*Round Robin*



Gbr. 10 Diagram Hasil Uji *Throughput*

Mengacu pada tabel IV terkait kualitas parameter *throughput* sesuai standarisasi TIPHON TR 101 309 [6] menunjukkan hasil pengujian yang diperoleh pada masing-masing algoritma termasuk dalam kategori yang baik dikarenakan keduanya mendapat hasil yang berada diantara 1200 Kbps – 2,1 Mbps.

TABEL IV
Klasifikasi Standar *Throughput*

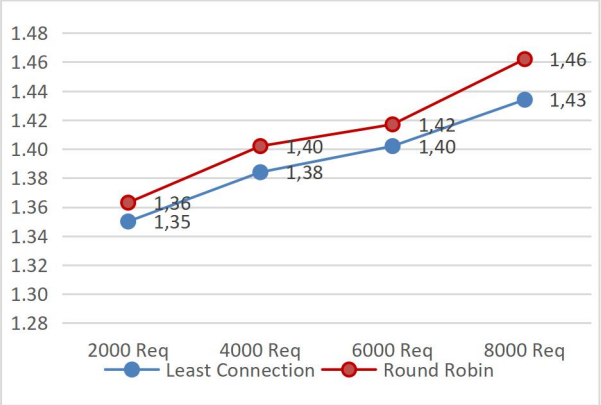
Kategori	**Nilai Throughput**
Sangat Baik	>2,1 Mbps
Baik	1200 Kbps – 2,1 Mbps
Cukup	700 – 1200 Kbps
Kurang Baik	338 – 700 Kbps
Buruk	0 – 338 Kbps

C. Hasil Pengujian *Delay*

Hasil pengujian *delay* pada Tabel 5 dan Gambar 11 menunjukkan nilai *delay* pada penggunaan kedua algoritma tersebut terlihat tidak terlalu signifikan dan cenderung sama serta mengalami peningkatan seiring bertambahnya jumlah *request*. Hasil pengujian menggunakan algoritma *least connection* memiliki rata-rata *delay* yang lebih kecil dibandingkan dengan algoritma *round robin*.

TABEL V
Nilai Parameter *Delay*

No.	*Delay* (ms)				Keterangan
	2000 *request*	4000 *request*	6000 *request*	8000 *request*	
1.	1,35	1,38	1,40	1,43	*Least Connection*
2.	1,36	1,40	1,42	1,46	*Round Robin*



Gbr. 11 Diagram Hasil Uji *Delay*

Hasil dari pengujian *delay* yang ditampilkan melalui grafik pada Gambar 11 menunjukkan perubahan grafik antara kedua algoritma cenderung sama dan mengalami peningkatan seiring bertambahnya beban *traffic* yang dikirim. Didapatkan hasil bahwa algoritma *least connection* lebih unggul dalam menangani *delay* pada skenario yang sama dibandingkan dengan algoritma *round robin*. Mengacu pada Tabel VI terkait kualitas parameter *delay* pada standarisasi TIPHON TR 101 309 [6], hasil pengujian yang diperoleh pada masing-masing algoritma menunjukkan nilai yang sangat baik yaitu <150 ms.

TABEL VI
Klasifikasi Standar *Delay*

Kategori	**Nilai *Delay* (ms)**
Sangat Baik	≤ 150
Baik	150 – 300
Sedang	300 – 450
Kurang	> 400

IV. KESIMPULAN

Berdasarkan hasil dari pembahasan mengenai analisis unjuk kerja *load balancing web server* dapat ditarik kesimpulan:

1. Algoritma *least connection* dan *round robin* dapat diterapkan pada *load balancer F5 BIG-IP LTM*.
2. Hasil pengujian parameter *response time* pada empat skenario percobaan menunjukkan bahwa algoritma *least connection* memiliki hasil rata-rata *response time* yang lebih stabil dibandingkan algoritma *round robin*, terlihat pada saat pengujian 6000 *request* dan 8000 *request* yaitu 365,274 ms dan 368,068 ms pada *least connection* dibandingkan dengan 357,832 ms dan 376,299 ms pada *round robin*.
3. Dalam segi kecepatan waktu respon, algoritma *round robin* mendapatkan hasil yang lebih cepat pada pengujian 2000 *request* yaitu 359,900 ms dan 6000 *request* yaitu 357,832 ms. Algoritma *least connection* mendapatkan hasil yang lebih cepat pada pengujian 4000 *request* yaitu 358,837 ms dan 8000 *request* yaitu 368,068 ms.
4. Hasil pengujian parameter *throughput* pada empat skenario percobaan menunjukkan bahwa algoritma *least connection* memiliki hasil rata-rata *throughput* yang lebih tinggi dibandingkan algoritma *round robin*. Perbedaan paling signifikan terjadi pada pengujian 6000 *request* di mana algoritma *least connection* mendapatkan hasil 1965,5 Kbps sedangkan pada algoritma *round robin* didaptkan hasil 1884,9 Kbps.
5. Hasil pengujian parameter *delay* menunjukkan bahwa algoritma *least connection* mendapatkan hasil rata-rata *delay* yang lebih kecil dibandingkan dengan algoritma *round robin* di mana *delay* terbesar pada algoritma *least connection* didapatkan hasil 1,43 ms sedangkan *round robin* didapatkan hasil 1,46 ms.

REFERENSI

[1] Supramana, Prismana, I. G. L. P. E. (2016). Implementasi Load Balancing Pada Web Server Dengan Menggunakan Apache. *Jurusan Teknik Informatika, Fakultas Teknik, Universitas Negeri Surabaya*, *5*(2), 117-125.

[2] Nugroho, H. A., Ikhwan, S., & Khair, F. (2018). Analisis Performansi Webserver Menggunakan Loadbalancingserver Dengan Algoritma Leastconnection. In *Conference on Electrical Engineering, Telematics, Industrial technology, and Creative Media (CENTIVE)* (pp. 332-337).

[3] Amiruddin, A., Akbar, S. R., & Arwani, I. (2018). Implementasi Security Pada Load Balancing Layanan Web Multidomain Dengan SSL. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *2*(2), 622-631.

[4] Rahmana, D., Primananda, R., & Yahya, W. (2017). Analisis Load Balancing Pada Web Server Menggunakan Algoritme Weighted Least Connection. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, *2*(3), 915-920.

[5] Jonsson, P., & Iveson, S. (2019). F5 Networks Application Delivery Fundamentals Study Guide. New York: Philip Jonsson and Steven Iveson.

[6] ETSI. (1999). Telecommunications and Internet Protocol Harmonization over Networks (TIPHON); General aspects of Quality of Service (QoS). ETSI TR 101 329 V2.1.1 (1999-06), vol. 1, pp. 1-37.