

ANALISIS LOAD BALANCING ROUND ROBIN DAN FAULT DETECTION PADA SOFTWARE DEFINED NETWORK BERBASIS P4**Rizal Cerdas Kurniawan S¹, Gamel Mahesa Farosh², Yanto Setiawan^{3*}**rizal.kurniawan@binus.ac.id, gamel.farosh@binus.ac.id, yanto.setiawan@binus.ac.id

Computer Science Department, BINUS Online Learning, Bina Nusantara University, Jakarta, Indonesia 11480

Informasi ArtikelDiterima : 3 Feb 2023
Direview : 9 Apr 2023
Disetujui : 20 Apr 2023**Kata Kunci***Load balancing, Round robin, Software defined network, Programming protocol-independent packet processor, TIPHON.***Abstrak**

Tujuan penelitian ialah mengoptimalkan kinerja *server* pada jaringan Software Defined Network berbasis P4 language dengan menerapkan *load balancing round robin*. Metode penelitian dibagi menjadi identifikasi masalah, perancangan dan implementasi, pengujian dan analisa. *Load balancing* adalah mekanisme yang digunakan untuk membagi *workload* pada beberapa *server* yang dijalankan dengan tujuan untuk mengoptimalkan *resource* yang ada, meningkatkan nilai *throughput*, serta meningkatkan kinerja *server* agar tidak mendapatkan *workload* berlebih yang dapat menurunkan kualitas layanan. Hasil Penelitian adalah *throughput* pada implementasi 4 *server* memiliki nilai rata-rata 632,09Kb/s, 3 *server* memiliki rata-rata 632,14Kb/s, dan 2 *server* memiliki rata-rata 632,10Kb/s. Hasil pengukuran *request loss* pada implementasi 4 *server* memiliki nilai rata-rata 0%, 3 *server* memiliki rata-rata 0%, dan 2 *server* memiliki rata-rata 0%. Hasil pengukuran *response time* pada implementasi 4 *server* memiliki nilai rata-rata 3.9ms, 3 *server* memiliki rata-rata 0,73ms, dan 2 *server* memiliki rata-rata 0,87ms. Hasil pengukuran *fairness index* pada implementasi pada *server* yang diuji memiliki nilai rata-rata 1. Disimpulkan beban kerja pada masing-masing *server load balancing round robin* dengan *fault detection* dapat mengoptimalkan kinerja *server*.

Keywords*Load balancing, round robin, Software defined network, Programming protocol-independent packet processor, TIPHON.***Abstrak**

The purpose of this study is to optimize server performance on a network defined P4 language-based network software by applying round robin load balancing. Research methods are divided into problem identification, design and implementation, testing and analysis. Load balancing is a mechanism used to divide the workload on several servers that are run with the aim of optimizing existing resources, increasing throughput, and improving server performance so as not to get overloaded which can degrade the quality of Service. The result of this study is that the implementation throughput of 4 servers has an average value of 632.09 Kb/s, 3 servers have an average of 632.14 Kb / s, and 2 servers have an average of 632.10 Kb / s. The results of the measurement of request loss in the implementation of 4 servers have an average value of 0%, 3 servers have an average of 0%, and 2 servers have an average of 0%. The results of the measurement of response time on the implementation of 4 servers have an average value of 3.9 ms, 3 servers have an average of 0.73 ms, and 2 servers have an average of 0.87 ms. The measurement of the fairness index on the implementation on the server tested has an average value of 1. Summing up the workload on each server load balancing round robin can optimize server performance.

A. Pendahuluan

Pada saat ini teknologi jaringan enterprise di perusahaan sudah berkembang dengan pesat. Implementasi berbagai aplikasi perusahaan untuk mendukung bisnisnya di tengah pandemi dan banyaknya persaingan membuat perusahaan harus banyak menggunakan *server* untuk menyediakan layanan aplikasi yang digunakan oleh pelanggan (Agung Nugroho, Widhi Yahya, 2017). Dalam dunia industri banyak perusahaan menggunakan web sebagai bentuk promosi ataupun transaksi online seperti web e-commerce, web tersebut menangani jutaan hit pada web *server* mereka. Banyak hal yang menyebabkan kegagalan hit salah satunya adalah banyaknya permintaan ataupun transaksi ke web *server* yang tak mampu ditangani, hal ini sangat berbahaya dan merugikan perusahaan. Dampaknya adalah dapat membuat *server* mengalami *overload* (Murti et al., 2020). Untuk mengatasi ini banyak cara yang bisa dilakukan, cara yang paling banyak dilakukan adalah dengan menambah *server* baru atau dengan menambahkan kapasitas penyimpanan *server* tetapi hal ini membutuhkan biaya yang besar, di sisi lain terdapat teknik yang dapat mengatasi masalah ini yaitu menggunakan teknik *load balancing*. *Load balancing* adalah mekanisme yang digunakan untuk membagi *workload* pada beberapa *server* yang dijalankan dengan tujuan untuk mengoptimalkan *resource* yang ada, meningkatkan nilai *throughput*, serta meningkatkan kinerja *server* agar tidak mendapatkan *workload* berlebih yang dapat menurunkan kualitas layanan (Zhang et al., 2018). *Load balancing round robin* merupakan metode untuk menyalurkan beban secara berurutan dari *server* satu ke *server* lainnya, sehingga menciptakan sistem putaran atau rotasi (Nasser & Witono, 2016).

Software Defined Network (SDN) adalah konsep pendekatan jaringan komputer dimana sistem pengontrol (*control plane*) dari arus data dipisahkan dari perangkat kerasnya (*data plane*) (Sanjoyo et al., 2020). Pada teknologi SDN terdapat protokol *openflow* dan *netconf*, protokol tersebut digunakan untuk mengontrol *traffic flows*, namun *openflow* dan *netconf* ini sudah ditentukan dan tidak dapat ubah atau dimodifikasi. *Programming Protocol-independent Packet Processors* (P4) adalah sebuah bahasa pemrograman untuk *top down* programming yang dapat menentukan bagaimana *pipelines* pada *switch* bekerja dan bagaimana paket-paket ini dapat diproses (Laraba et al., 2020).

Pada penelitian yang berjudul "*Load balancing Implementation Strategy for Various Services in Software defined network using ONOS Controller*" (R. Tulloh, M. Iqbal, M. Rifqi, 2021) ini menggunakan protokol *openflow* dengan algoritma shortest delay untuk *load balancing*. Pada penelitian yang berjudul "*Load balancing Implementation Strategy for Various Services in Software defined network using ONOS Controller*" (Sharma & Reddy, 2019) *load balancing* yang digunakan adalah Software-Defined Networking (SDN) based Cloud, menggunakan konsep SDN dengan protokolnya yaitu Cloud. Pada Penelitian "*LBFT: Load balancing and Fault Tolerance in distributed controllers*" (Mahjoubi et al., 2019) ini menerapkan *fault detection* algorithm pada *load balancing* di Jaringan SDN dengan *controllernya* yaitu *Floodlight*. Kemudian pada penelitian "*Load Balance in Data Center SDN Networks*" (Ali et al., 2018) dilakukan simulasi mengenai *Load Balance Data Center SDN Network* menggunakan algoritma *Dijkstra*. Pada penelitian "*Traffic Load balancing Using Software Defined Networking (SDN) Controller as Virtualized*

Network Function”(Ejaz et al., 2019) dilakukan validasi *load balancing* dalam topologi *Fat-Tree* menggunakan dua pengontrol vSDN di *emulator*.

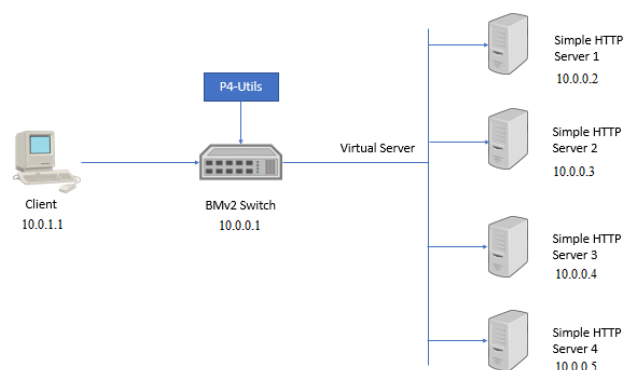
Tujuan dan manfaat pada penelitian ini adalah sebagai berikut:

1. Mengoptimalkan kinerja *server* pada jaringan *Software Defined Network* berbasis *P4 language* dengan menerapkan *load balancing round robin*.
2. Menerapkan *High Availability server* pada jaringan *Software Defined Network* berbasis *P4 language* dengan mengimplementasikan *fault detection system*.
3. Melakukan implementasi *load balancing round robin* dan *fault detection* di jaringan *Software Defined Network* berbasis *P4 language* pada *Virtual machine*.
4. Melakukan pengukuran *respons time*, *throughput*, *request loss*, dan *fairness index* dalam penerapan *load balancing round robin* pada jaringan SDN berbasis *P4 language*.

B. Metode Penelitian

Topologi Sistem

Pada penelitian ini, terdapat *topologi* yang akan digunakan untuk pengujian *load balancing* menggunakan *P4 language*. Pada topologi sistem ini terdiri dari *client* yang digunakan untuk mengakses *server*, 1 *controller*, *switch* BMv2, *Simple HTTP server* yang sudah *support* *P4*.



Gambar 1. Topologi Sistem

Pada implementasi ini *client* akan mengirimkan *request* menuju *server*, alamat tujuannya adalah *IP Virtual Server*, *request* yang sudah dikirimkan akan masuk ke *switch* BMv2 yang sudah diprogram *P4 load balancing* dengan algoritma *round robin*, kemudian algoritma tersebut akan membagi beban secara merata ke setiap *server*. *Server* akan memproses *request* tersebut dan mengirimkan kembali paket ke alamat *IP* milik *client*.

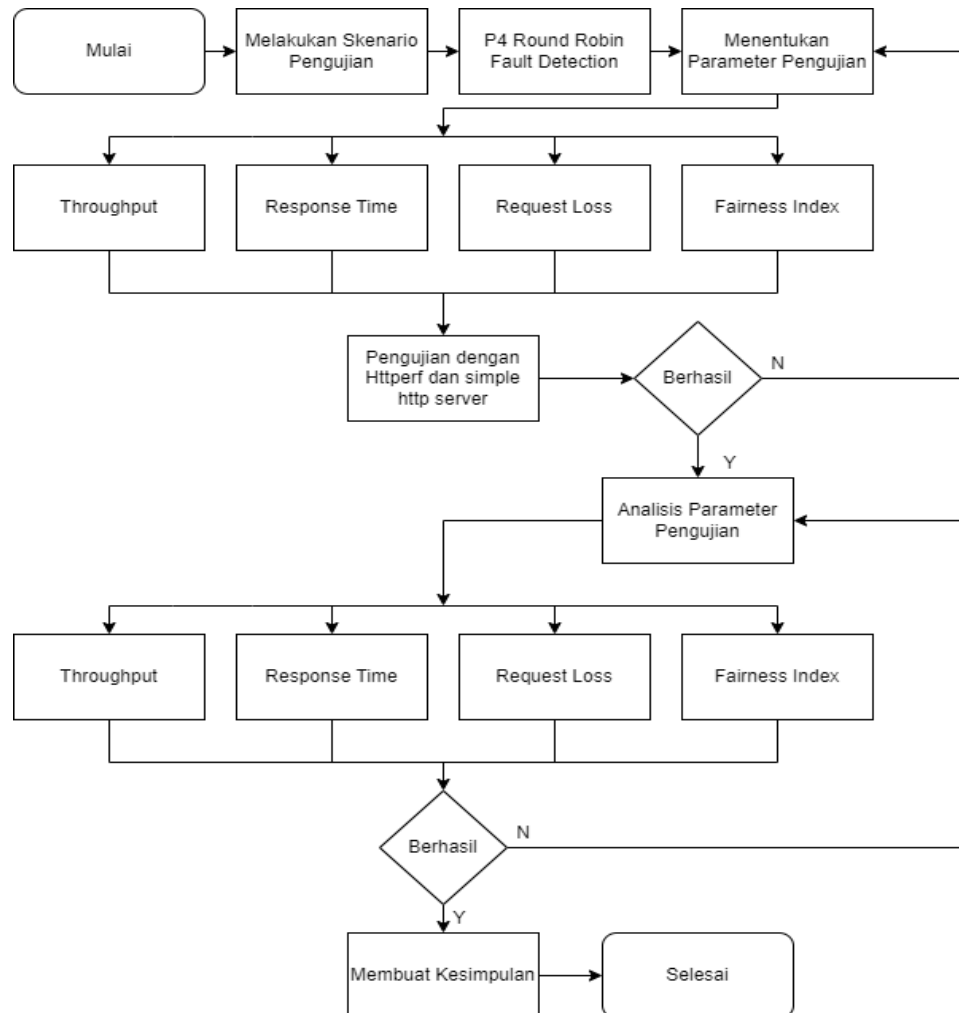
Skenario Pengujian

Pada sistem ini akan dilakukan pengujian terhadap *load balancing* yang diprogram dengan *P4* pada jaringan SDN.

Pada Gambar 2 langkah-langkah pengujian sistem dapat dijelaskan sebagai berikut:

1. Menentukan Skenario Pengujian Sistem.

2. Menentukan parameter pengujian dengan *httperf* dan *Simple HTTP server* dan mengambil data dari parameter yang akan diuji yaitu *Throughput*, *Response time*, *Request loss* dan *Fairness index*.
3. Melakukan analisis pada data yang sudah didapatkan pada pengujian
4. Mengambil kesimpulan dari analisis yang telah dilakukan



Gambar 2. Flowchart Pengujian Sistem

Tabel 1. Pengujian *Throughput*, *Request loss*, dan *Response Time*

No	Layanan <i>Simple HTTP Server</i> (Pengujian)
1	200 request
2	400 request
3	600 request
4	800 request
5	1000 request

Pada Tabel 1 dijelaskan mengenai pengujian yang dilakukan untuk mendapatkan *parameter throughput*, *response time*, *request loss*. Pada layanan

Simple HTTP Server dilakukan pengujian dengan memberikan beban kepada *server* sebanyak 200 *request*, 400 *request*, 600 *request*, 800 *request*, 1000 *request*. Skenario ini dalam pengambilan data dilakukan sebanyak 30 kali untuk mendapatkan hasil yang maksimal dan skenario ini didasarkan pada penelitian (R. Tulloh, M. Iqbal, M. Rifqi, 2021).

Tabel 2. Pengujian *Fairness index*

No	Layanan <i>Simple HTTP Server</i> (Pengujian)
1	10 <i>request</i>
2	20 <i>request</i>
3	30 <i>request</i>
4	40 <i>request</i>
5	50 <i>request</i>

Pada Tabel 2 dijelaskan mengenai pengujian yang dilakukan untuk mendapatkan parameter *fairness index*, baik dengan 4 buah *server*, 3 buah *server*, maupun dengan 2 buah *server* untuk membuktikan algoritma *round robin* dengan *fault detection* yang digunakan berfungsi normal dengan tetap membagi *request* yang seimbang pada setiap *server*. Skenario ini dilakukan dengan memberikan beban kepada *server* sebanyak 10 *request*, 20 *request*, 30 *request*, 40 *request*, dan 50 *request*.

Implementasi Hardware

Pada penelitian ini komponen perangkat keras terdiri dari satu buah *laptop* yang digunakan untuk menjalankan VM yang sudah terinstall ubuntu 18.04.6. Berikut tabel 3 merupakan spesifikasi dari *laptop* yang digunakan pada penelitian ini.

Tabel 3. Spesifikasi Hardware

No	Spesifikasi	Detail
1	Processor	11th Gen Intel® Core™ i7-1165G7 @ 2.80GHz
2	Memory	8 GB
3	Operating System	Windows 10 pro 64 bit
4	Storage	1 TB HDD 120GB SATA SSD

Implementasi Software

Pada Tabel 4 merupakan rincian perangkat lunak yang digunakan untuk penelitian ini.

Tabel 4. Spesifikasi Software

No	Spesifikasi	Detail
1	Controller	P4 Utils versi 0.0
2	Emulator	Mininet versi 2.2.1
3	Packet Processor	P416
4	Software Switch	Stratum BMv2

Pada penelitian ini menggunakan P4 utils versi 0.0 sebagai SDN *Controller* dimana nantinya akan terhubung dengan mininet BMv2 yang *support* untuk P4 language sebagai data layer, dan aplikasi lain yang sudah tertera pada Tabel 4.

C. Hasil dan Pembahasan

Parameter Pengujian

Pada bab ini dijelaskan mengenai hasil pengujian dan analisis berdasarkan implementasi sistem dan skenario pengujian yang telah dilakukan. Berikut ada beberapa parameter pengujian pada penelitian ini :

a. *Througput*

Througput yaitu kecepatan transfer data yang diukur dalam satuan *bit per second*. *Througput* adalah jumlah total kedatangan paket yang sukses yang diamati pada tujuan selama interval waktu tertentu dibagi oleh durasi interval waktu tersebut. Pengukuran *througput* ini akan dilakukan pada sisi server (Murti et al., 2020). Nilai *Througput* sesuai dengan versi TIPHON pada tabel 5 berikut:

Tabel 5. Kategori *Througput*

No	Kategori <i>Througput</i>	<i>Througput</i>	<i>Indeks</i>
1	<i>Bad</i>	0 – 338 kbps	0
2	<i>Poor</i>	338 – 700 kbps	1
3	<i>Fair</i>	700 – 1200 kbps	2
4	<i>Good</i>	1200 – 2.1 Mbps	3
5	<i>Excellent</i>	>2.1 Mbps	4

(Sumber: TIPHON)

b. *Request loss*

Request loss yaitu jumlah *error request* yang dialami oleh *client* ketika mengirim *request* kepada *server* baik *request* yang tidak sampai *server* ataupun yang sudah sampai *server* tetapi tidak terlayani oleh *server* (Alssaheli et al., 2022). Nilai *packet loss* sesuai dengan TIPHON pada tabel 6 berikut:

Tabel 6. Kategori *Request loss*

No	Kategori <i>Request loss</i>	<i>Packet loss</i>	<i>Indeks</i>
1	<i>Poor</i>	>25%	1
2	<i>Medium</i>	12 – 24%	2
3	<i>Good</i>	3 – 14%	3
4	<i>Perfect</i>	0 – 2%	4

(Sumber: TIPHON)

c. *Response time*

Response time yaitu pengujian waktu respon yang mengacu pada waktu yang diperlukan suatu *server* untuk menanggapi permintaan dari *client*. Waktu yang dibutuhkan ketika *server* mengambil langkah untuk bereaksi

terhadap suatu input tertentu sampai proses selesai (Mohammadian et al., 2022). Nilai latency dengan versi TIPHON pada tabel 7 berikut:

Tabel 7. Kategori Latency

No	Kategori Latency	Latency	Indeks
1	<i>Poor</i>	> 450 s	1
2	<i>Medium</i>	300 – 450 s	2
3	<i>Good</i>	150 – 300 s	3
4	<i>Perfect</i>	<150 s	4

(Sumber: TIPHON)

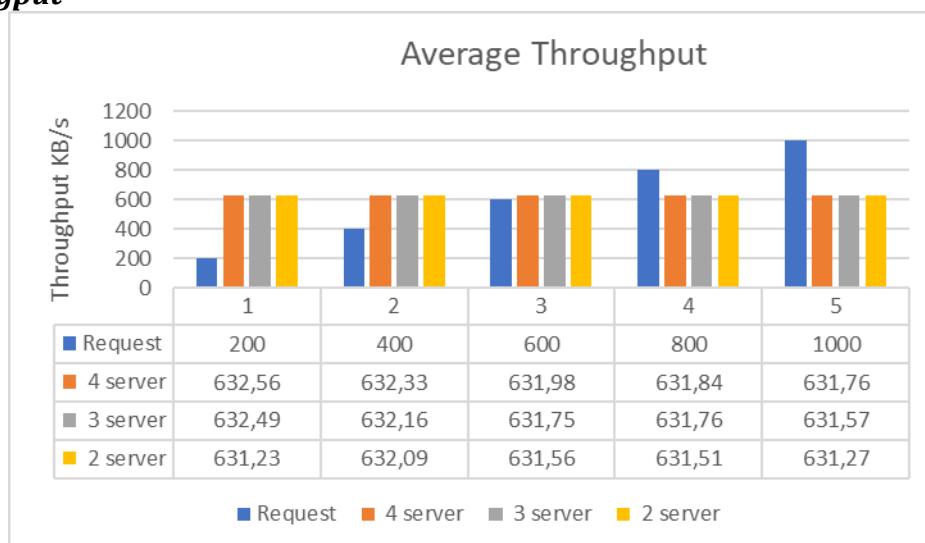
d. *Fairness index*

Fairness index yaitu parameter untuk mengukur beban *server* yang diterima dari permintaan *client*. Nilai sempurna dari *fairness index* adalah 1 dimana *server – server* membagi kerja secara seimbang.

Hasil Pengukuran

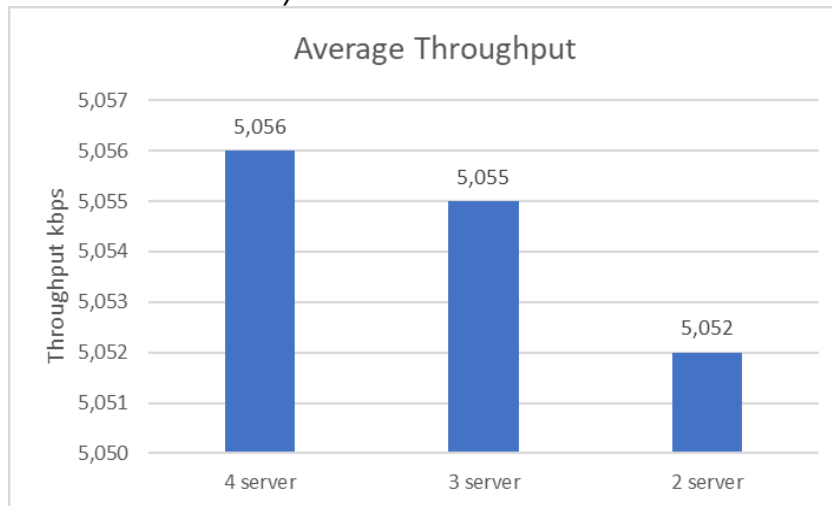
Pengukuran *load balancing round robin* dan *fault detection* di jaringan *software defined network* berbasis P4 ini disimulasikan melalui *mininet*. *Load balancing* yang dijalankan diaplikasikan pada *BMv2 switch* dengan pemrograman berbasis P4. Pengukuran ini dilakukan menggunakan *traffic generator httpperf* untuk *client*, *httpperf* digunakan untuk membuat banyak *request* secara bersamaan sesuai dengan jumlah *request* yang diharapkan. Untuk membuat banyak *request* pada *traffic generator httpperf* yaitu dengan perintah "*httpperf --server --num-conn --num-call --rate*" dimana *--server* adalah *ip bm2 switch* sebagai *load balancer*, *--num-conn* adalah jumlah koneksi yang dikirimkan, *--num-call* adalah jumlah *request* yang dikirimkan setiap sesinya sebelum sesi berakhir, dan *--rate* adalah jumlah waktu yang dibutuhkan saat mengirim *request*. Dan pada sisi *server* menggunakan *Simple HTTP server*. Output data yang dihasilkan adalah *throughput*, *request loss*, *response time*, dan *fairness index*. Berikut adalah hasil pengukuran yang telah dilakukan pada implementasi *load balancing round robin* dan *fault detection* pada jaringan *software defined network* berbasis P4.

Throughput



Gambar 3. Hasil Pengukuran *Throughput*

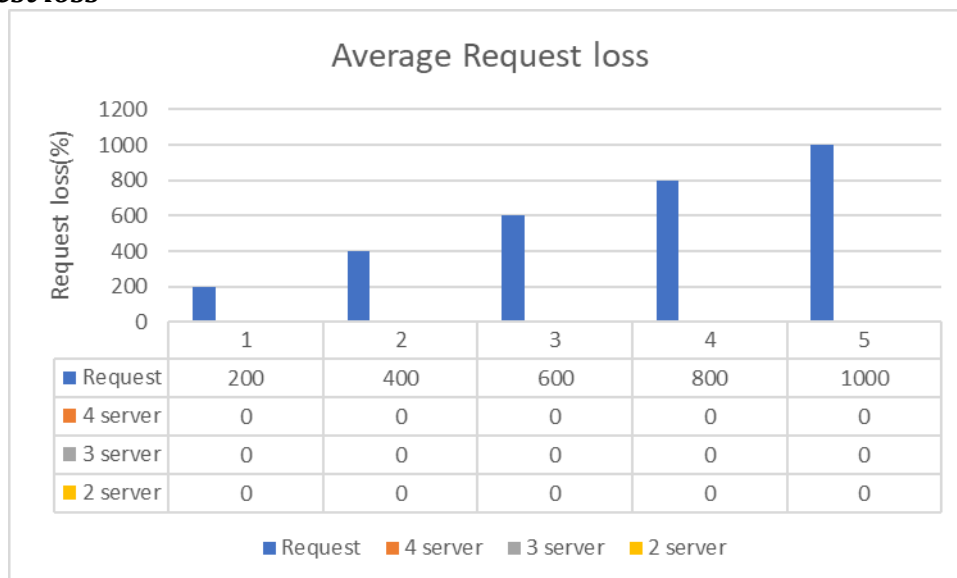
Pada Gambar 3 menunjukkan hasil nilai rata-rata *throughput* menggunakan algoritma *round robin* yang terdiri dari 4 server, 3 server, dan 2 server yang sudah dilakukan pengujian sebanyak 30 kali. Server 4 mendapat nilai rata-rata yaitu 632,09 Kb/s. Server 3 mendapat nilai rata-rata yaitu 631,95 Kb/s. Server 2 mendapat nilai rata-rata yaitu 631,93 Kb/s. karena satuannya masih Kb/s maka akan dikonversikan dahulu ke Kb/s.



Gambar 4. Hasil konversi pengukuran *throughput*

Pada Gambar 4 menunjukkan hasil konversi dari satuan Kb/s ke Kb/s. Dapat dilihat pada pengujian ini jumlah server dan *throughput* berbanding lurus bahwa semakin banyak request maka *throughput* yang dihasilkan juga semakin kecil.

Request loss



Gambar 5. Hasil Pengukuran *Request loss*

Pada Gambar 5 menunjukkan nilai hasil rata-rata *request loss* menggunakan algoritma *round robin* yaitu terdiri dari 4 server, 3 server, dan 2 server yang sudah dilakukan pengujian sebanyak 30 kali. Untuk pengujian 30 kali ini terdapat nilai

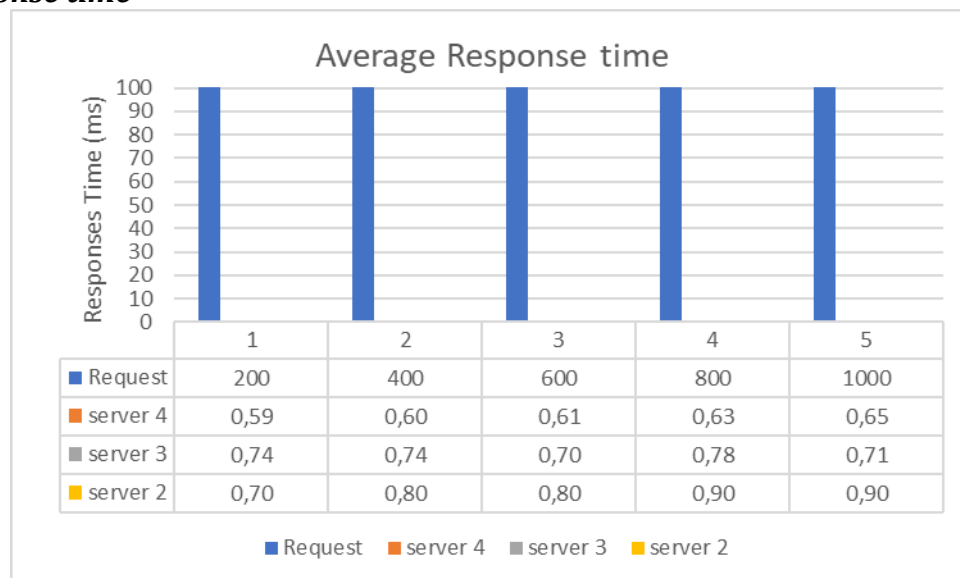
loss yaitu pada 4 *server* mendapat nilai rata-rata 0 % *loss*. Pada 3 *server* mendapat nilai rata-rata 0 % *loss*. Pada 2 *server* mendapat nilai rata-rata 0 % *loss*.



Gambar 6. Hasil Pengujian 30 kali *request loss*

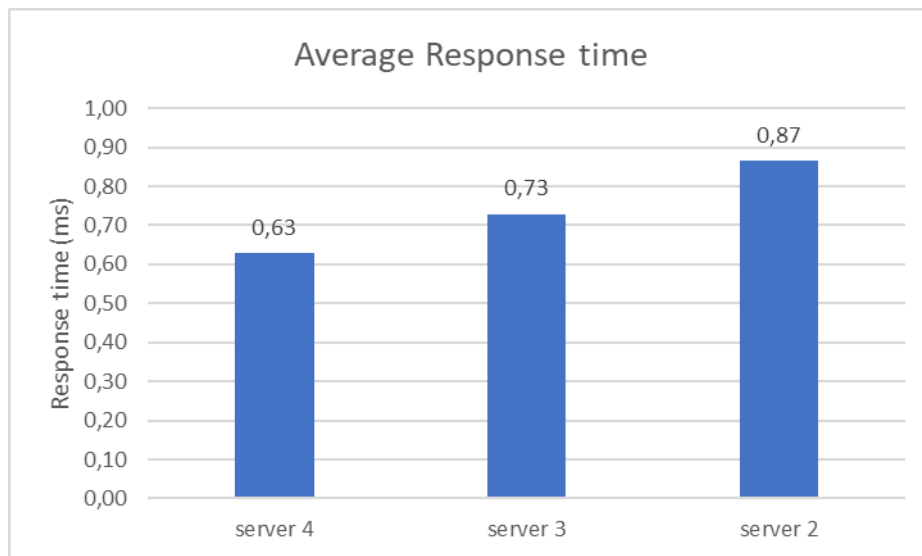
Pada Gambar 6 dapat dilihat hasil pengujian 30 kali ini bahwa semakin banyak *request* dan semakin sedikit *server* maka tidak terdapat *request loss*.

Response time



Gambar 7. Hasil Pengukuran *Response time*

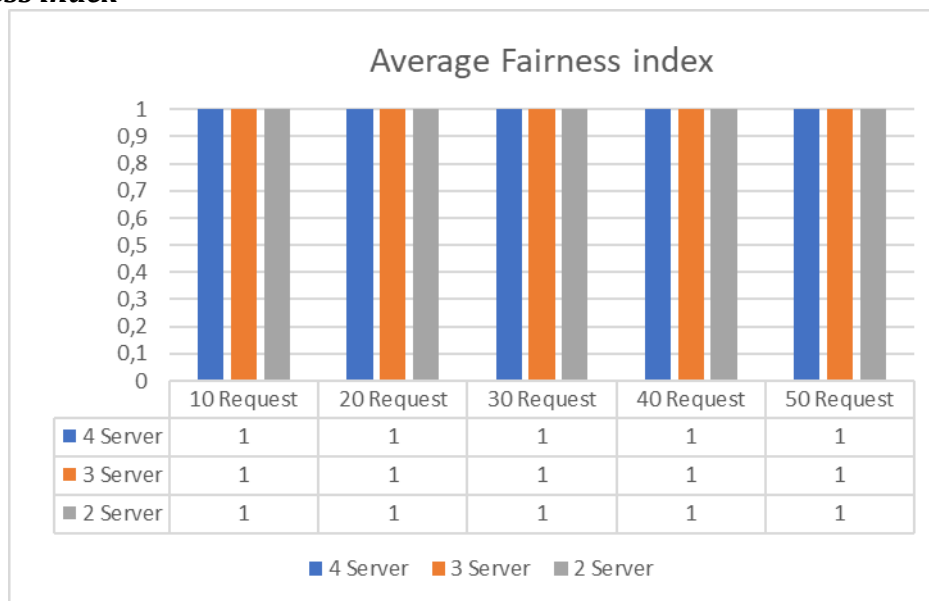
Pada Gambar 7 menunjukkan hasil nilai rata-rata dari pengujian sebanyak 30 kali pada beberapa *server response time* menggunakan P4 algoritma *round robin*. Pada 4 *server* hasil nilai rata-rata yaitu 0,63 ms. Pada 3 *server* hasil nilai rata-rata 0,73 ms. Pada 2 *server* yaitu 0,87 ms.



Gambar 8. Hasil Pengujian 30 kali *response time*

Pada Gambar 8 dapat dilihat pengujian sebanyak 30 kali bahwa semakin banyak *request* dan semakin sedikit *server* maka *response time* yang dihasilkan semakin besar.

Fairness index



Gambar 9. Hasil Pengukuran *Fairness index*

Pada gambar 9 merupakan hasil pengukuran *Fairness index* dari 10 *request* sampai 50 *request* ke beberapa *server* yang telah diuji selama sebanyak 30 kali untuk menunjukkan stabilitas, kinerja, dan cara kerja *server* ketika sistem menjalankan algoritma *round robin* dan *fault detection*.

D. Simpulan

Berdasarkan hasil penelitian selama pengerjaan tugas akhir ini dapat Tarik kesimpulan bahwa:

1. Hasil dari pengujian sebanyak 30 kali dalam *throughput* pada beberapa server algoritma *round robin* mendapatkan nilai rata-rata server 4 yaitu 5,056Kb/s tergolong "*Excelent*" sesuai standar TIPHON. Server 3 mendapat nilai rata-rata yaitu 5,055Kb/s tergolong "*Excelent*" sesuai standar TIPHON. Server 2 mendapat nilai rata-rata yaitu 5,052Kb/s tergolong "*Excelent*" sesuai standar TIPHON. Jadi rata – rata pengukuran *throughput* selama 30 kali hasilnya tergolong sangat baik.
2. Hasil dari pengujian selama sebanyak 30 kali dalam *request loss* pada beberapa algoritma *round robin* menggunakan P4 yaitu pada 4 server mendapat nilai rata-rata 0 % *loss* tergolong "*Perfect*" sesuai standar TIPHON. Pada 3 server mendapat nilai rata-rata 0 % *loss* tergolong "*Perfect*" sesuai standar TIPHON. Pada 2 server mendapat nilai rata-rata 0% *loss* tergolong "*Perfect*" sesuai standar TIPHON. Jadi rata – rata pengukuran *request loss* selama 30 kali hasilnya tergolong sangat baik.
3. Hasil dari pengujian sebanyak 30 kali dalam *response time* pada beberapa server algoritma *round robin* dengan P4 yaitu pada 4 server hasil nilai rata-rata yaitu 0,63 ms tergolong "*Perfect*" sesuai standar TIPHON. Pada 3 server hasil nilai rata-rata 0,73 ms tergolong "*Perfect*" sesuai standar TIPHON. Pada 2 server yaitu 0,87 ms tergolong "*Perfect*" sesuai standar TIPHON. Jadi rata – rata pengukuran *response time* selama 30 kali hasilnya tergolong sangat baik.
4. Hasil dari pengujian sebanyak 30 kali dalam *fairness index*, beban yang diterima server menunjukan nilai 1 yang artinya sama rata. Sehingga algoritma *round robin* baik digunakan untuk server-server yang memiliki spesifikasi yang sama. *fault detection* membantu mengoptimalkan kinerja server sehingga tidak terjadi *request loss* ketika ada salah satu server bermasalah.

E. Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada Jurusan Teknik Informatika, Binus Online Learning, dan Universitas Bina Nusantara yang telah memberikan bimbingan dan arahan bagi penulis untuk melakukan penelitian ini.

F. Referensi

- [1] Agung Nugroho, Widhi Yahya, K. A. (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.
- [2] Ali, T. E., Morad, A. H., & Abdala, M. A. (2018). Load balance in data center SDN networks. International Journal of Electrical and Computer Engineering, 8(5), 3084–3091. <https://doi.org/10.11591/ijece.v8i5.pp3084-3091>
- [3] Alssaheli, O. M. A., Abidin, Z. Z., Zakaria, N. A., & Abas, Z. A. (2022). Software Defined Network based Load balancing for Network Performance Evaluation. International Journal of Advanced Computer Science and Applications, 13(4), 117–124. <https://doi.org/10.14569/IJACSA.2022.0130414>
- [4] Ejaz, S., Iqbal, Z., Azmat Shah, P., Bukhari, B. H., Ali, A., & Aadil, F. (2019). Traffic Load balancing Using Software Defined Networking (SDN) Controller

- as Virtualized Network Function. *IEEE Access*, 7, 46646–46658. <https://doi.org/10.1109/ACCESS.2019.2909356>
- [5] Laraba, A., François, J., Chrisment, I., Chowdhury, S. R., & Boutaba, R. (2020). Defeating Protocol Abuse with P4: Application to Explicit Congestion Notification. *IFIP Networking 2020 Conference and Workshops, Networking 2020*, 431–439.
- [6] Mahjoubi, A., Zeynalpour, O., Eslami, B., & Yazdani, N. (2019). LBFT: Load balancing and fault tolerance in distributed controllers. *2019 International Symposium on Networks, Computers and Communications, ISNCC 2019*, 1–6. <https://doi.org/10.1109/ISNCC.2019.8909087>
- [7] Mohammadian, V., Navimipour, N. J., Hosseinzadeh, M., & Darwesh, A. (2022). Fault-Tolerant Load balancing in Cloud Computing: A Systematic Literature Review. *IEEE Access*, 10, 12714–12731. <https://doi.org/10.1109/ACCESS.2021.3139730>
- [8] Murti, K. W., Riza, T. A., & Mulyana, A. (2020). Comparative Analysis of Load balancing Dynamic Ratio and Server Ratio Algorithms. *Proceeding - 1st FORTEI-International Conference on Electrical Engineering, FORTEI-ICEE 2020*, 162–167. <https://doi.org/10.1109/FORTEI-ICEE50915.2020.9249815>
- [9] Nasser, H., & Witono, T. (2016). Analisis Algoritma Round robin, Least Connection, Dan Ratio Pada Load balancing Menggunakan Opnet Modeler. *Jurnal Informatika*, 12(1). <https://doi.org/10.21460/inf.2016.121.455>
- [10] R. Tulloh, M. Iqbal, M. Rifqi, A. B. (2021). Load balancing Implementation Strategy for Various Services in Software Defined Network using ONOS Controller. ... *Journal Of Computing* <https://137.117.138.59/handle/123456789/4509>
- [11] Sanjoyo, D. D., Elektro, F. T., & Telkom, U. (2020). Analisis Performansi Sistem Traffic Engineering Pada Controller Onos Dengan Metode Intent Monitor and Re- Route (Imr) Menggunakan Off-Platform Application (Opa) Berbasis Software Defined Network Performance Analysis of Traffic Engineering on Onos Cont. 7(1), 487–495.
- [12] Sharma, R., & Reddy, H. (2019). Effect of load balancer on software-defined networking (SDN) based Cloud. *2019 IEEE 16th India Council International Conference, INDICON 2019 - Symposium Proceedings, March*. <https://doi.org/10.1109/INDICON47234.2019.9030327>
- [13] Zhang, J., Yu, F. R., Wang, S., Huang, T., Liu, Z., & Liu, Y. (2018). Load balancing in data center networks: A survey. *IEEE Communications Surveys and Tutorials*, 20(3), 2324–2325. <https://doi.org/10.1109/COMST.2018.2816042>