Analisis Optimasi Algoritma IP Hash pada P4 Menggunakan Algoritma Round Robin

IP Hash Algorithm Optimization Analysis on P4 Using the Round Robin Algorithm

1st Mohammad Rifki Baihaqi
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
baihaqirifki@student.telkomuniversity.ac.id

2nd Ridha Muldina Negara
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
ridhanegara@telkomuniversity.ac.id

3rd Rohmat Tulloh
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
rohmatth@telkomuniversity.ac.id

Abstrak —Load balancing adalah mekanisme untuk membagi beban komputasi ke beberapa server. Load balancing ini bertujuan untuk mengoptimalkan resource dan meningkatkan throughput agar server tidak mengalami overload. Software Defined Network (SDN) adalah teknologi jaringan komputer yang memisahkan fungsi Data Plane dan control plane. Protokol OpenFlow digunakan untuk mengontrol/mengelola arus lalu lintas, tetapi OpenFlow ini telah ditentukan dan tidak dapat diubah atau dimodifikasi. Untuk mengatasi masalah di atas, Programming Protocol-independent Packet Processors (P4) adalah bahasa pemrograman untuk pemrograman top-down yang dapat menentukan bagaimana pipeline pada switch bekerja dan bagaimana paket-paket ini dapat diproses. P4 ini dapat mengatasi kelemahan OpenFlow yang kurang fleksibel dalam mengontrol/mengatur arus trafik dan memungkinkan berjalannya proses load-balancing. Pada Tugas Akhir ini telah dilakukan simulasi dan analisis load balancing pada infrastruktur jaringan yang dapat diprogram berbasis bahasa P4. Algoritma round-robin dengan P4 memiliki nilai throughput rata-rata 127,61 KB/s. Algoritma hash IP dengan P4 memiliki nilai throughput rata-rata 127,50 KB/s. Algoritma round robin memiliki nilai response time rata-rata 3,13 ms; pada algoritma hash IP, nilai rata-ratanya adalah 12,16 ms. Nilai response time sistem berbasis P4 dengan algoritma round robin lebih baik daripada algoritma hash IP, dan request loss kedua sistem dapat mendistribusikan request dengan baik sehingga request loss sebesar 0%.

Kata Kunci—*load balancing, data plane, P4, round robin, IP hash*

Abstract— Load balancing is a mechanism for dividing the compute load into multiple servers. This load balancing aims to optimize resources and increase throughput so that the server does not experience overload. Software Defined Network (SDN) is a computer network technology that separates data plane and control plane functions. TheOpenFlow protocol is used to control/manage traffic flow, but this OpenFlow is predefined and cannot be changed or modified. To solve the above problem, Programming Protocol-independent Packet Processors (P4) is a programming language for top-down programming that can determine how pipelines on switches work and how these packets can be processed. This P4 can overcome the weakness of OpenFlow which is less flexible in controlling / regulating traffic flow and allows the load-balancing process to run. In this Final Project, simulation and load balancing analysis have been carried out on network infrastructure that can be programmed based on the P4 language. The round-robin algorithm with P4 has an average throughput value of 127.61 KB/s. The IP hash algorithm with P4 has an average throughput value of 127.50 KB/s. The round robin algorithm has an average response time value of 3.13 ms; in the IP hash algorithm, the average value is 12.16 ms. The response time value of the P4-based system with the round robin algorithm is better than the IP hash algorithm, and the request loss of both systems can distribute the request well so that the request loss is 0%.

Keywords—load balancing, data Plane, P4, round robin, IP hash

I. PENDAHULUAN

Pada saat teknologi jaringan komputer berkembang dengan sangat pesat, dan dampak dari perkembangan yang pesat ini membuat kebutuhan manusia dalam hal kebutuhan internet meningkat dengan drastis, ini membuat server seringkali mengalami *overload*[1]. Cara yang banyak dilakukan untuk mengatasi ini yaitu dengan menambahkan server baru atau menambahkan *harddisk* tambahan untuk *database* tetapi hal ini membutuhkan biaya yang besar, di sisi lain terdapat teknik yang dapat mengatasi masalah ini yaitu teknik *load balancing*. *Load balancing* adalah suatu mekanisme untuk membagi beban komputasi ke beberapa server, tujuan dari *load balancing* ini untuk mengoptimalkan sumber daya, meningkatkan *throughput,* agar server tidak mengalami *overload.*

Software-Defined Network (SDN) adalah konsep pendekatan jaringan komputer dimana sistem pengontrol (*control plane)* dari arus data dipisahkan dari perangkat kerasnya (*data plane*)[2]. Pada SDN terdapat protokol *openflow* ini digunakan untuk mengontrol/mengatur *traffic flows* tetapi *openflow* ini sudah ditentukan dan tidak dapat ubah atau dimodifikasi. Programming Protocol-independent Packet Processors (P4) adalah sebuah bahasa pemograman untuk *top-down programming* yang dapat

menentukan bagaimana *pipelines* pada switch bekerja dan bagaimana paket-paket ini dapat diproses[3]. P4 ini dapat mengatasi kelemahan *openflow* yang kurang fleksibel dalam mengontrol/mengatur *traffic-flow*. IP hash merupakan algoritma load balancing yang dalam mendristribusikan bebannya dengan cara mencocokan hash key dan ini membuat beban yang diterima oleh server tidak merata, untuk mengatasi kelemahan IP hash dilakukan optimasi dengan menggunakan algoritma Round Robin. untuk menerapkan load balancing dibutuhkan aplikasi tambahan seperti HAProxy, F5, Nginx dan lain-lain dengan P4 proses load balancing dapat langsung dilakukan pada data plane.

Pada penelitian sebelumnya yang berjudul "*Load Balancing Implementation Strategy for Various Services in Software Defined Network using ONOS Controller*"[4] pada penelitian ini sudah diterapkan pada jaringan sdn namun untuk protokol yang digunakan adalah openflow sehingga tidak bisa menerapkan *programmable* pada *pipeline* openflow akibatnya untuk melakukan proses load balancing memerlukan aplikasi tambahan yaitu F5. Pada penelitian yang berjudul "Analisis Dan Implementasi *Load Balancing* Pada Web Server Dengan Algoritme *Shortest Delay* Pada *Software Defined Network*"[5] namun pada penelitian ini masih menerapkan protokol openflow sehingga tidak bisa memodifikasi *pipeline* openflow dan proses load balancing terjadi pada controller ryu. Pada penelitian berjudul "Analisis Kinerja Load Balancing pada Server Web Menggunakan Algoritma Weighted Round Robin pada Proxmox VE"[6] namun pada penelitian ini untuk melakukan load balancing masih menggunakan aplikasi tambahan yaitu HAProxy dan masih di implementasikan pada jaringan konvensional sehingga pada switch tidak bisa diterapkan *programmable* di sisi *data plane*. Pada penelitian berjudul "Implementasi Load Balancing dengan Algoritma Weighted Round Robin menggunakan NGINX"[7] namun pada penelitian ini load balancing masih menggunakan aplikasi Nginx dan tidak diterapkan *programmable* pada switch. Pada penelitian berjudul "Comparative Analysis of Load Balancing Dynamic Ratio and Server Ratio Algorithms"[8] namun pada penelitian ini laod balancing masih menggunakan F5 BIG IP yang memerlukan *resource* yang besar dan tidak di implementasikan pada jaringan sdn sehingga switch yang dipakai tidak bisa untuk diprogram untuk melakukan load balancing. Pada penelitian berjudul "Implementasi High-Availability Web Server Menggunakan Load Balancing As a Service Pada Openstack Cloud"[9] namun pada penelitian ini untuk menjalankan proses load balancing menggunakan aplikasi Openstck. Pada penelitian berjudul "Load Balancing In Software Defined Networking (SDN)"[10] namum pada penelitian ini load balancing terdapat pada controller pox dan masih menggunakan protokol openflow sehingga tidak bisa melakukan *programmable* pada sisi *dataplane* switch. Pada penelitian yang berjudul "Implementasi Load Balancer Berdasarkan Server Status pada Arsitektur Software Defined Network (SDN)"[11] namun pada penelitian ini load balancing terjadi pada controller sdn dan masih menggunakan protokol openflow.

Oleh kerena itu pada Tugas Akhir ini telah dilakukan simulasi dan analisis load balancing pada *programmable network infrastructure* yang berbasis P4 *language* untuk sisi *data plane*. Untuk algoritma yang digunakan adalah IP hash dan round robin. Pada algoritma IP hash dilakukan Optimasi menggunakan algoritma round robin agar beban yang diterima oleh server sama atau didistribusikan secara merata. Penelitian ini akan menggunakan Mininet dan P4 BMv2 switch agar dapat melakukan simulasi infrastruktur jaringan.

II. KAJIAN TEORI

A. *Load Balancing*

Load Balancing merupakan teknik pendistribusian beban kerja secara seimbang atau merata antara dua atau lebih komputer. Ini bertujuan untuk mendapatkan sumber daya secara optimal dan menghindari terjadinya overload[6].

B. *Software Defined Network*

Software Defined Network (SDN) merupakan pendekatan jaringan yang dimungkinkan untuk diprogram sehingga mendukung pemisahan antara *control plane* dan *data plane*[7]. SDN dapat membuat perilaku jaringan dapat dikonrol secara *programmable* dan *centralized* menggunakan *sofware applications* melalui *open interface*[8].

C. *OpenFlow*

OpenFlow adalah standar pertama yang digunakan sebagai antarmuka komunikasi antara *controlplane* dengan *dataplane*, OpenFlow adalah protokol SDN yang paling banyak digunakan, Hal ini dikarenakan OpenFlow memungkinkan peneliti untuk mengembangkan bahkan merancang protokol yang baru[10].

D. *Programming Protocol-Independent Packet Processors (P4)*

Programming Protocol-Independent Packet Processors (P4) adalah *domain-specific programming language* yang dirancang untuk mendeskripsikan bagaimana perilaku pemrosesan paket pada *data plane*. P4 merupakan bahasa pemoograman yang sifatnya *statically-typed* , yang artinya program yang di tuliskan dengan P4 akan menggunakan sedikit memori dan lebih cepat. *Syntax* P4 berbasis dari bahasa pemograman C[13].

Perbedaan arsitektur dari tradisional switch dan P4-*defined switch* , *data plane* dari P4-*defined switch* tidak bersifat *fixed pipelines* dan *fixed table* tetapi dapat didefinisikan dalam program P4. Control plane P4-defined switch berkomunikasi dengan *data plane* menggunakan kanal yang sama dengan yang terdapat pada *fixed-fanction device*, hanya saja pada *pipelines, headers, parser, deparser*, dll yang ada pada data plane didefinisikan sendiri di dalam program P4. Compiler P4 membuat Application Programming Interface (API) untuk berkomunikasi dengan data plane[15]

E. P4Runtime

P4Runtime merupakan *Application Programming Interface* (API) berbasis gRPC dan *Protocol Buffers* (protobuf) yang menghubungkan antara P4 defined *switch* dengan *control plane*. Menggunakan desain arsitektur PSA tetapi dikemabngkan dengan arsitektur lain dan diinisialisasi oleh *google* dan *barefoot*[16].

F. Behavioral Model version 2 (BMv2)

BMv2 adalah P4 *sofware switch* versi 2. BMv2 ini menggunakan bahasa pemograman C++11. BMv2 akan menerima input dari file berformat *JavaScript Object Notation* (JSON) yang dihasilkan oleh P4 Compiler. BMv2 mengimplementasi perilaku pemrosesan paket yang telah didefinisikan didalam program P4. Pada saat Tugas Akhir ini ditulis BMv2 belum *production-grede switch* dan hanya digunakan untuk pengembangan dari P4. Kinerjanya dalam *throughput* dan *latensi*, secara singnifikan lebih rendah dari pada perangakat lunak tingkat produksi seperti *Open vSwitch*.

G. Mininet

Mininet merupakan sofware emulator yang digunakan untuk membuat jaringan virtual yang realistis, mempu menjalankan *real karnel*, *virtual host, link banwidth, switch*, kode aplikasi pada sautu mesin baik VM, *cloud*, atau *native* dalam hitungan detik menggunakan satu perintah[17]. Pada mininet memiliki keuntungan yaitu fleksibilitas untuk mengemulasikan SDN, tidak perlu akses testbed yang mahal, menggunakan *python* API untuk membuat topologi buatan yang terdiri dari *controller*, *switch*, *host*, dan *link*.

H. Virtualisasi Server

Virtualisasi Server merupakan konsep baru pada perkembangan teknologi. Hal ini dikarenakan, Virtualisasi Server memungkinkan penggunaan satu perangkat keras untuk menjalankan beberapa sistem operasi secara independent dengan layanan yang berbeda pada waktu bersamaan. Virtualisasi Server sebagai paradigma baru memberikan efisiensi yang lebih dibanding sistem server yang ada dengan meminimalkan jumlah perangkat fisik, tempat, dan biaya pemeliharaan sistem[18].

I. Web Server

Web server merupakan *software* yang menjadi inti utama dari *World Wide Web* (WWW). Data yang terdapat dalam web server memiliki format yang standar, disebut dengan format *Standar General Markup Language* (SGML). Web server bekerja dengan menunggu permintaan dari *client* yang menggunakan *browser* seperti *Google Chrome*, *Mozilla Firefox*, *Internet Explorer,* dan *browser* lainnya. Web server memproses permintaan dari *browser* kemudian memberikan hasil prosesnya berupa data yang diinginkan kembali ke *browser*[19].

J. Round Robin

Algoritma round robin merupakan algoritma yang paling sederhana dan banyak digunakan oleh perangkat load balancing. Algoritma ini membagi beban secara bergiliran dan berurutan dari satu server ke server lainnya. Konsep dasar dari algoritma round robin ini adalah dengan menggunakan time sharing[20]. Pada algoritma ini jika terdapat 2 server dan dari client mengerimkan beberapa *request* maka *request* tersebut akan terbagi secara merata ke setiap server, apabila terdapat 3 server maka algoritma ini membagi request dari client menjadi 3 sesuai dengan jumlah server.

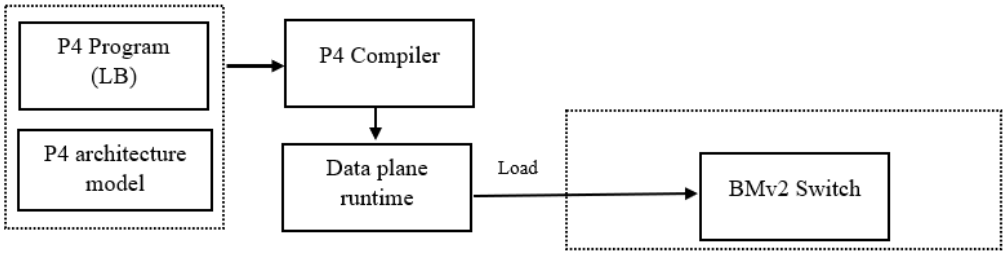
K. IP Hash

IP Hash load balancing menggunakan algoritma yang mengambil sumber dan tujuan alamat IP dari client dan server untuk menghasilkan key hash yang unik. Kunci ini digunakan untuk mengalokasikan client ke server tertentu. Karena kunci dapat diregenerasikan jika sesi rusak, metode load balancing ini dapat memastikan bahwa klien diarahkan ke server yang sama dengan yang digunakan sebelumnya. Ini berguna jika penting agar klien terhubung ke sesi yang masih aktif setelah pemutusan dan rekoneksi[21]. IP address yang sudah dirubah menjadi hash key ketika tersambung pada server. Hash key yang terbuat oleh proses ini akan menjadi penanda IP address pada salah satu server.

III. METODE

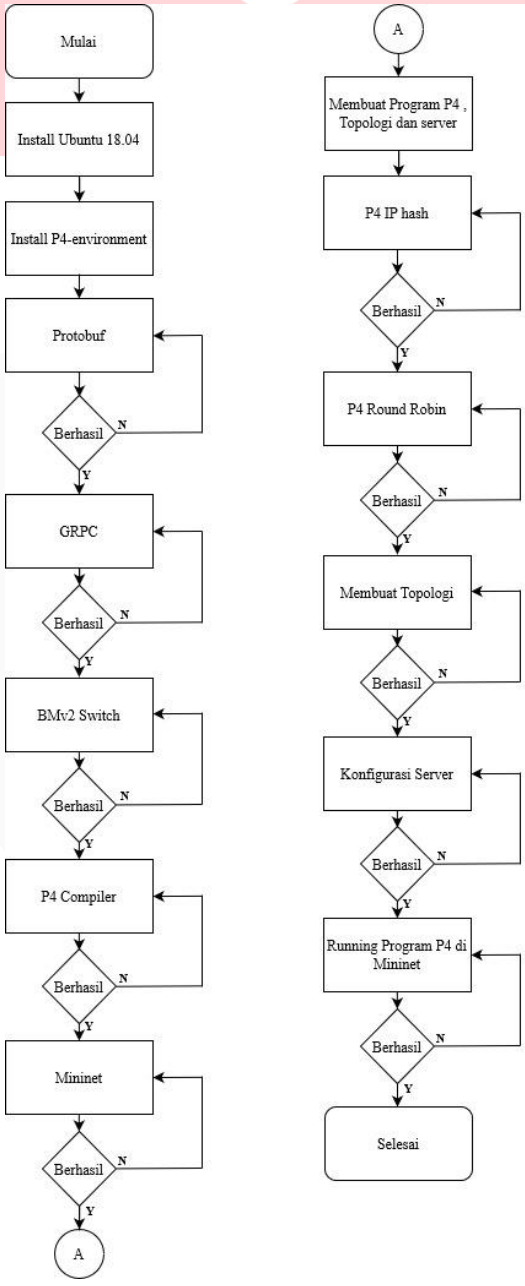
A. Perancangan Sistem

Sistem yang telah dirancang pada Tugas Akhir ini digunakan untuk menguji dan menganalisis *load balancing* pada infrastruktur jaringan terprogram berbasis P4 *language*. Gambar sistem Tugas Akhir ini dapat dilihat pada gambar 3.1, secara umum *infrastructure layer* menggunakan *data plane* BMv2 *switch* untuk dapat medukung P4 *language*. Program P4 *load balancing* dengan algoritma *round robin* ini akan dimasukan kedalam *switch* BMv2 dengan menggunakan P4 *Compiler* dimana pada algoritma *round robin* ini jika terdapat *request* maka algoritma ini akan membagi request tersebut ke beberapa server, jumlah *request* tersebut akan terbagi sesuai dengan jumlah server, misalkan terdapat 2 server dan jumlah *request* sebanyak 200 *request* ini akan terbagi menjadi server 1 menerima 100 *request* dan server 2 menerima100 *request* dikarena kan algoritma round robin akan membagi beban atau *request* secara merata ke setiap server. Pada load balancing dengan algoritma IP hash beban atau request akan mencocokkan *hash key* yang ada pada server dan client, jika terdapat 100 *request* dari client dan terdapat 2 server jika hash key cocok dengan server ke 1 maka request akan diteruskan ke server 1.



GAMBAR 3.1
GAMBARAN UMUM S*ISTEM*

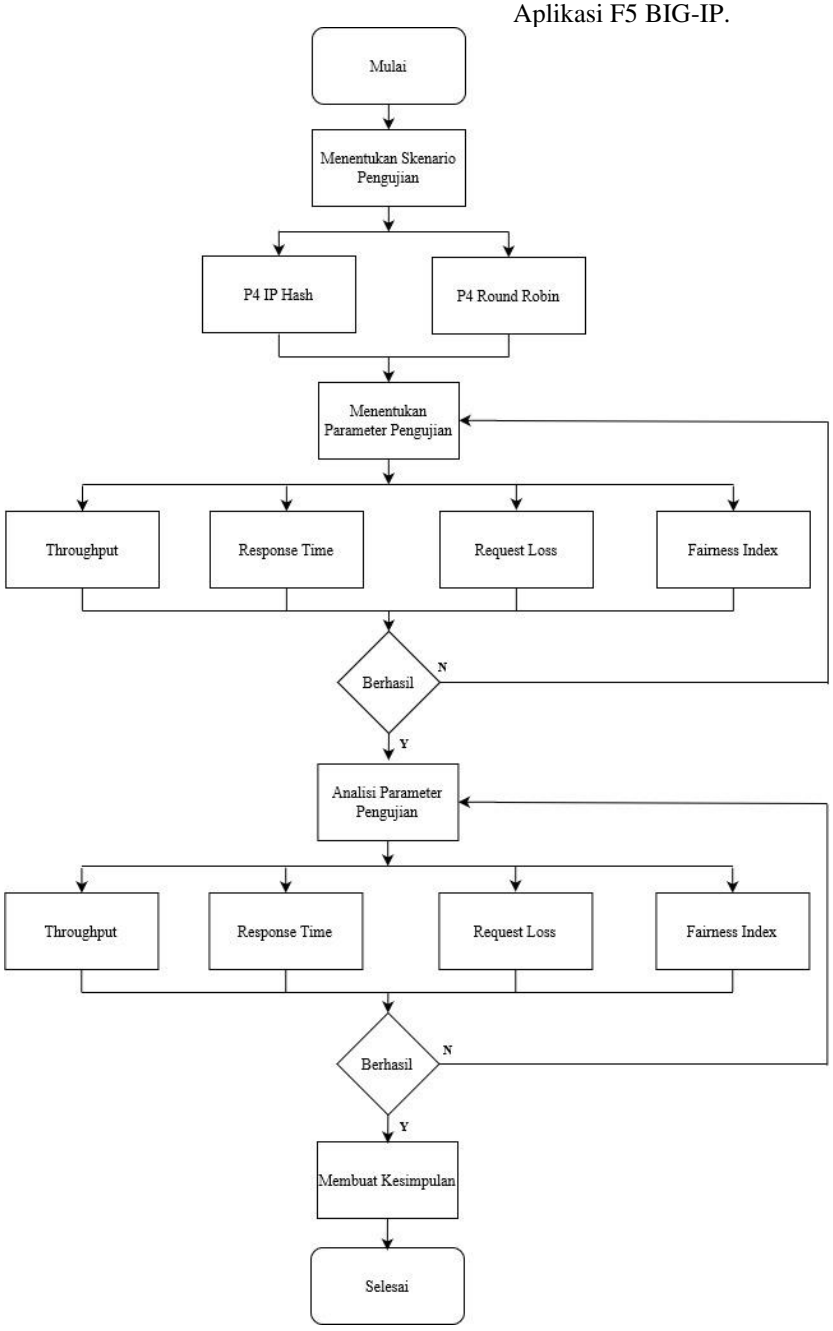
B. Diagram Blok Kebutuhan Implementasi Sistem



GAMBAR 3.2
FLOWCHART PERANCANGAN SISTEM

Pada Gambar 3.3 langkah-langkah perancangan sistem dapat dijelaskan sebagi berikut:

1. Instalasi VM ubuntu 18.04 untuk penelitian tugas akhir dengan spesifikasi yang tertera pada Tabel 3.1.
2. Instalasi P4 Environment dan *depedency* yang dibutuhkan seperti Instalasi *Mininet* dengan tambahan BMv2 sehingga *switch support* sebagai *data plane programmibility* menggunakan P4 *language*.
3. Membuat *code* untuk P4 *switch* sehingga switch mendukung untuk menjalankan *packet processing*, *Load Balacing*, dan konfigurasi Sistem F5 dan Simple HTTP server yang digunakan sebagai server. Membuat *topology*.
4. Menjalankan Program P4 pada *mininet* dan Aplikasi F5 BIG-IP.



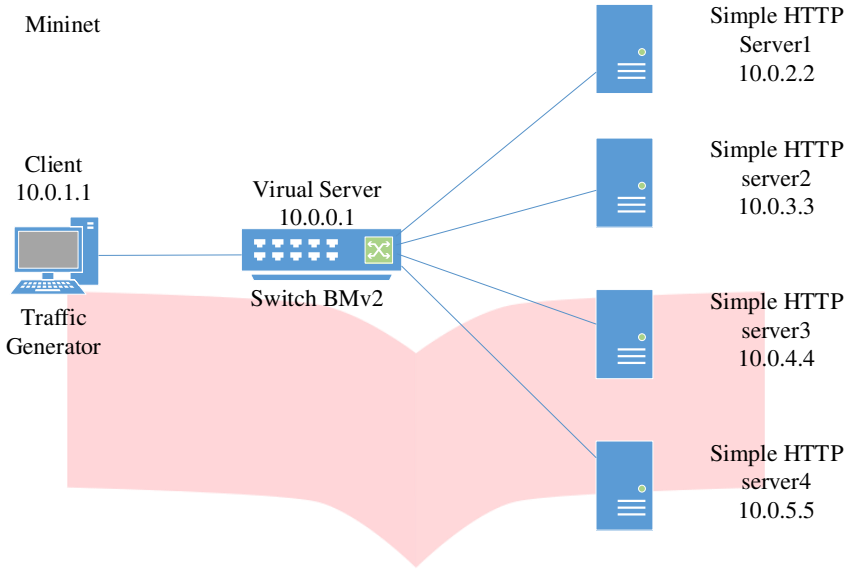
GAMBAR 3. 1
FLOWCHART PENGUJIAN SISTEM

Pada Gambar 3.4 langkah-langkah Pengujian sistem dapat dijelaskan sebagi berikut:

1. Menentukan Skenario Pengujian.
2. Menentukan parameter dan mengambil data dari parameter yang akan diuji yaitu *Throughput*, *Resoponse Time*, *Request Loss* dan *Fairness Index*.
3. Melakukan Analisis terhadap data yang sudah didapat.
4. Mengambil kesimpulan dari analisis yang telah dilakukan.

C. Implementasi Topologi Sistem

Pada penelitian Tugas Akhir ini, terdapat topologi yang akan digunakan untuk pengujian *Load Balancing* menggunakan P4 *Language*. Pada tolopogi terdiri dari client yang digunakan untuk mengakes server, switch BMv2, Simple HTTP server yang sudah support P4. Gambar topologi tertera pada gambar 3.5 berikut:

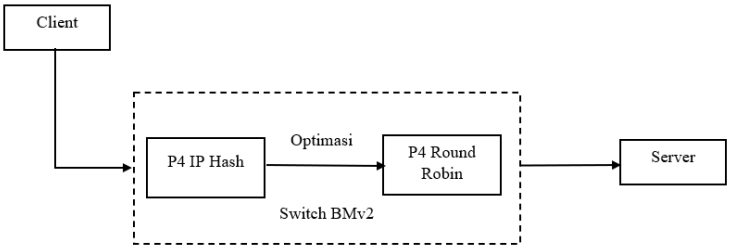


GAMBAR 3. 2
IMPLEMENTASI *TOPOLOGY* SISTEM P4 *ROUND* ROBIN DAN P4 IP HASH

Pada penelitian Tugas Akhir ini, terdapat topologi yang akan digunakan untuk pengujian *Load Balancing* menggunakan P4 *Language*. Pada tolopogi terdiri dari client yang digunakan untuk mengakes server, switch BMv2, Simple HTTP server yang sudah support P4. Gambar topologi tertera pada gambar 3.5. Pada gambar 3.5 merupakan implementasi dari P4 menggunakan algoritma round robin dan IP hash, pada sisi *clinet* akan mengirimkan *request* menuju server alamat tujuan nya adalah IP *Virtual Server* , *request* yang sudah dikirimkan akan masuk ke *switch* BMv2 yang sudah di program P4 *Load balancing* dengan algoritma Round Robin dan IP Hash , kemudian algoritma akan membagi beban ke setiap *server*. *Server* akan memproses *request* tersebut dan mengirimkan kembali paket ke alamat IP milik *client.*

D. Optimasi Algoritma IP Hash

Pada gambar 3.6 merupakan optimasi pada algoritma IP hash dimana algoritma ini akan membagi beban atau request dengan menggunakan hash key. Akibat penggunaan hash key ini request tidak didistribusikan secara merata ke setiap server, maka dilakukan optimasi agar request bisa didistribusikan secara merata menggunakan algoritma round robin. Algoritma round robin sangat cocok pada saat server memiliki spesifikasi yang sama.



GAMBAR 3. 3
OPTIMASI ALGORITMA IP HASH

Berikut merupakan proses dari algoritma IP Hash dan Round Robin bekerja optimasi yang dilakukan algoritma round robin terdapat pada baris 8-12 dimana *flowlet_select* akan meneruskan paket atau request ke server yang sudah ditentukan sebelumnya.

sebelumnya.

Algoritma 1. IP hash dan Round Robin

Ingress Pipeline

1. Jika sebuah paket ditujukan ke IP virtual dan bukan TCP paket SYN.
2. Gunakan lima tupel untuk mendapatkan nilai hash sebagai indeks di *flowlet_select* agar server dapat melayani.

3. Selesai
4. Jika sebuah paket ditujukan ke client.
5. Ubah alamat MAC tujuan menjadi MAC client.
6. Kirim paket ke port yang sesuai.
7. Selesai
8. Jika ini adalah paket TCP SYN yang ditunjukan ke IP virtual.
9. Gunakan lima tupel untuk mendapatkan nilai hash i (i is 1~n).
10. Gunakan lima tupel untuk mendapatkan nilai hash sebagai indeks untuk *flowlet_select*.
11. simpan i di *flowlet_select* dengan yang sesuai indeks.
12. Selesai
13. jika Si down.
14. Temukan server berfungsi dengan baik berikutnya dengan indeks j.
15. Simpan j di *flowlet_select*.
16. Jika semua server sedang down.
17. Drop paket tersebut.
18. Selesai.
19. Jika sebuah paket ditujukan ke IP virtual.
20. ubah alamat MAC tujuan menjadi MAC Server yang dipilih.
21. ubah alamat IP tujuan ke yang dipilih IP server.
22. Kirim paket ini ke port output yang sesuai.
23. Selesai.

Engress Pipeline

24. jika sebuah paket ditujukan ke client.
25. ubah alamat IP sumber menjadi VIP.
26. Selesai.

E. Skenario Pengujian

Pada Tugas Akhir ini telah dilakukan pengujian terhadap *Load balancing* yang di program dengan P4 pada jaringan SDN Pengujian di lakukan untuk mendapatkan parameter *throughput*, *respon time*, dan *request loss*, dan *Fairness Index*, pada skenario 1 dilakukan pengujian pada *load balancing* dengan menggunakan program P4 dengan menggunakan algoritma IP hash, program P4 dengan menggunakan algoritma Round Robin, *Request* yang di berikan sebanyak 5000, 10000, 15000, 20000, dan 25000 *request*. Untuk pengujian *Fairness Index* request yang diberikan sebanyak 25, 50, 75, dan 100 request[22][23]. Skenario ini dalam pengambilan data skenario ini didasarkan pada penelitian[24].

TABEL 3.4
PENGUJIAN SISTEM LOAD BALANCING

Skenario	Jenis Layanan	Pengujian 1	Pengujian 2	Pengujian 3	Pengujian 4	Pengujian 5
P4 IP Hash	Simple HTTP Server	5000 *request*	10000 *request*	15000 *request*	20000 *request*	25000 *request*
P4 Round Robin	Simple HTTP Server	5000 *request*	10000 *request*	15000 *request*	20000 *request*	25000 *request*

IV. HASIL DAN PEMBAHASAN

Dalam tugas akhir ini, parameter yang dianalisis yaitu:

a. *Throughput* adalah rate (kecepatan) transfer data efektif yang di ukur dalam satuan *bit per second* (bps) dimana pengukuran ini dilakukan pada sisi server.
b. *Response time* berfungsi untuk mengetahui seberapa cepat server dalam menanggapi request dari client dan pengukuran ini dilakukan pada sisi server.
c. *Request loss* merupakan jumlah kegagalan request yang dikirimkan oleh client, request yang dikirimkan tidak sampai pada server dan request yang sampai tetapi tidak dilayani oleh server termasuk dalam *request loss*.
d. *Fairness Index* bertujuan untuk mengetahui apakah setiap server yang dijalankan menerima request atau beban yang sudah dikirimkan oleh client.

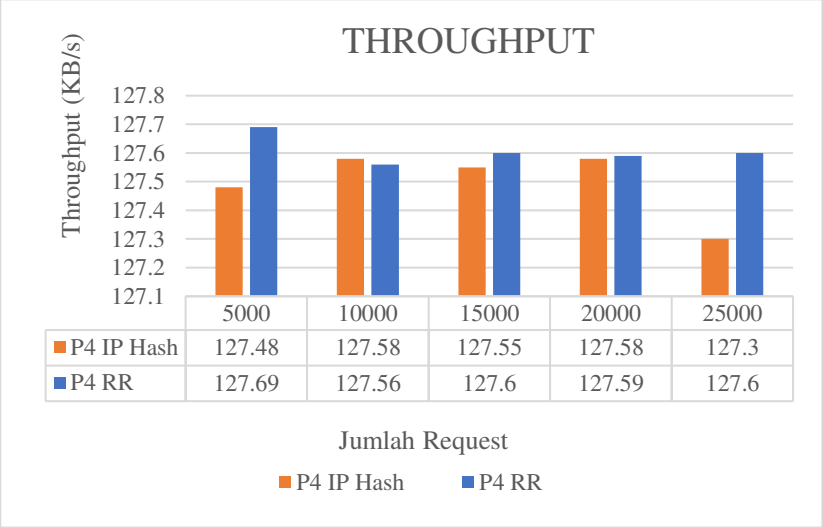
A. Hasil Pengukuran

Pada pengukuran *load balancing* dengan menggunakan Program P4 yang di masukan kedalam switch BMv2 dan disimulasikan pada mininet, algoritma yang digunakan pada sistem ini adalah algoritma *Round Robin* dan IP hash . Pengukuran ini menggunakan *simple http server* sebagai *server*, dan pada sisi *client*

menggunakan *traffic generator* yaitu httperf. Data yang diambil yaitu *throughput*, *response time*, *request loss,* dan *Fairness Index.* Untuk membuat banyak request menggunkan traffic generator yaitu httperf, command yang dipakai adalah httperf -- server --num-conn --num-call –rate dimana –server merupakan virtual ip load balancing, --num-conn jumlah koneksi atau request yang akan dikirimkan, --num-call merupakan jumlah total request yang dikirimkan pada setiap sesi sebelum ditutup, --rete merupakan total waktu yang dibutuhkan untuk mengirim request. Berikut hasil pengukuran sistem *load balancing* menggunakan program P4.

1. *Throughput*

Pengukuran *throughput* berguna untuk mengetahui kemampuan server dalam menyediakan layanan kepada *client*. Pada parameter ini semakin tinggi nilai *throughput* maka semakin baik juga kinerja dari *load balancing* dalam mendistribusikan *request*. Berikut hasil pengukuran dari *throughput.*

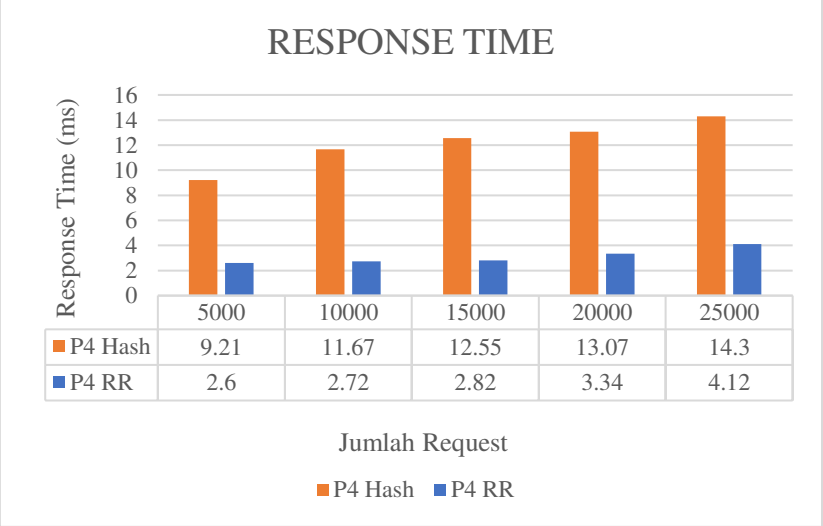


GAMBAR 4. 1
HASIL PENGUKURAN *THROUGHPUT*

Pada Gambar 4.1 menunjukan nilai *Throughput* menggunakan algoritma round robin dan IP hash. Nilai rata-rata dari algoritma IP hash dan Round Robin memiliki nilai yang tidak jauh berbeda, pada P4 *load balancing* algoritma round robin mendapat nilai rata-rata sebesar 127.61 KB/s, pada P4 *load balancing* algoritma IP hash mendapat nilai rata-rata sebesar 127.50 KB/s.

2. *Response Time*

Pengukuran *response time* berfungsi untuk mengukur seberapa cepat suatu server dapat menerima *request* dari clinet. Semakin kecil nilai *response time* maka semakin cepat server untuk melayani *request* dari *client*. Berikut hasil pengukuran dari *response time*.

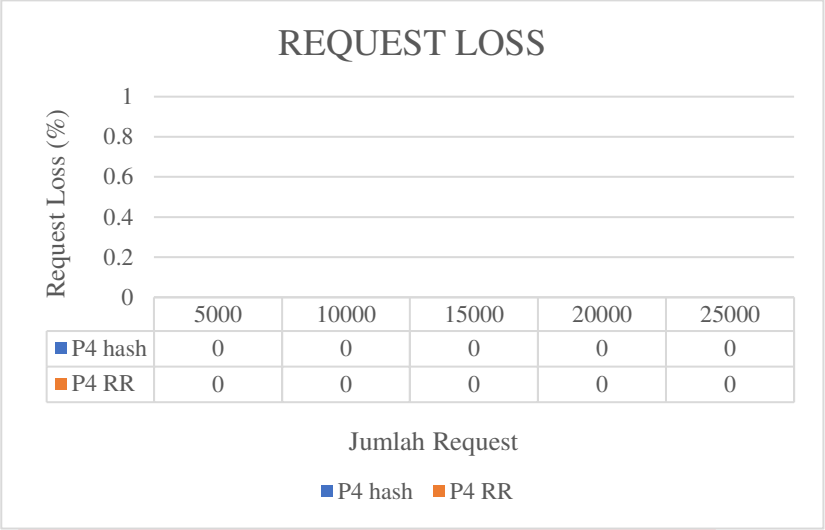


GAMBAR 4. 2
HASIL PENGUKURAN RESPONSE TIME

Pada Gambar 4.2 menunjukan hasil dari pengujian *response time*, nilai terendah terdapat pada *load balancing* menggunakan program P4 dengan algoritma round robin dimana server dapat merespon request lebih cepat dibandingkan dengan P4 yang menggunakan algoritma IP hash karena pada algoritma round robin tidak ada proses pencocokan hash key. pada P4 *load balancing* algoritma round robin mendapat nilai rata-rata sebesar 3.13 ms, pada P4 *load balancing* algoritma IP hash mendapat nilai rata-rata sebesar 12.16 ms.

3. *Request Loss*

Pengukuran *request loss* digunakan untuk menunjukan jumlah kegagalan permintaan yang tidak dapat di layani oleh server. Berikut hasil pengukuran dari *request loss*.

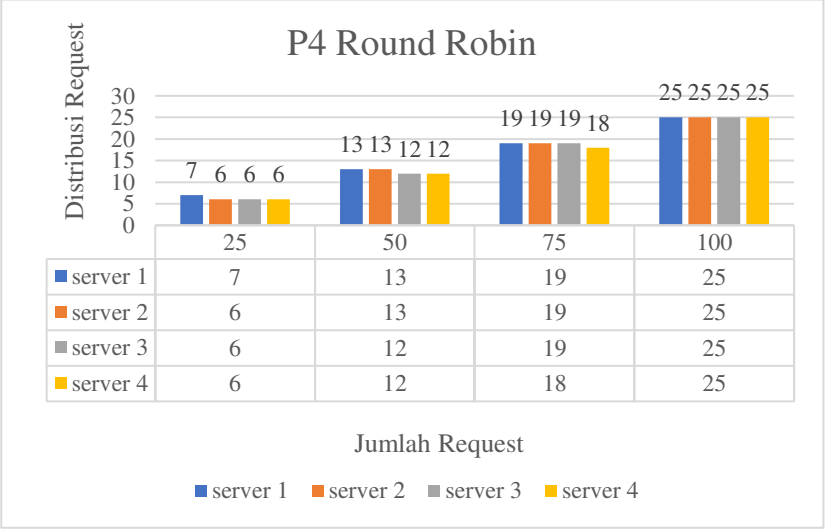


GAMBAR 4. 3
HASIL PENGUKURAN REQUEST LOSS

Pada Gambar 4.3 menunjukan hasil dari pengukuran *request loss*. Pada *load balancing* menggunakan program P4 dengan algoritma round robin, IP hash tidak terdapat *loss* ini dikarenakan *request* didistribusikan ke beberapa server.

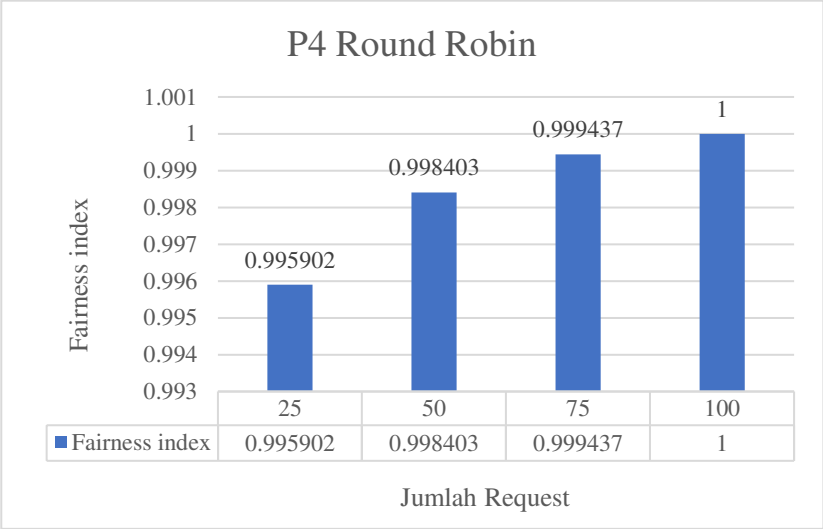
4. *Fairness Index* P4 Round Robin

Pada gambar 4.4 merupakan data distribusi request dengan menggunakan P4 Round Robin. Pengambilan data dilakukan dengan mengirimkan request sebanyak 25, 50, 75 dan 100 request. Algoritma Round Robin dapat bekerja sebagaima mestinya yaitu dengan membagi request secara merata ke setiap server. Algoritma ini cocok digunakan untuk server yang memiliki spesifikasi yang sama.



GAMBAR 4. 4
DISTRIBUSI P4 ALGORITMA ROUND ROBIN

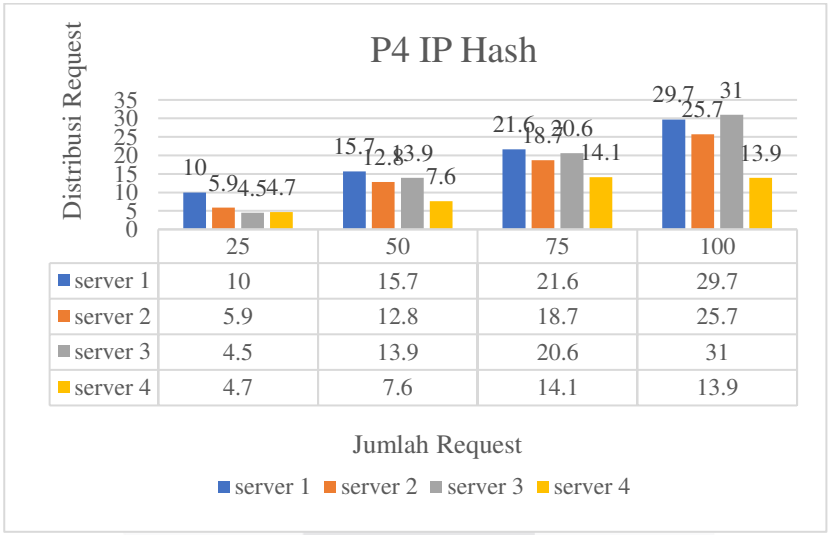
Pada gambar 4.5 merupakan hasil dari *Fairness Index* dari algoritma Round Robin. Setiap server memerima beban atau *request* secara merata mulai dari pemberian request sebanyak 25 sampai 100. Pada pemberian 100 request memiliki nilai *fairness index* yaitu 1 yang artinya setiap server menerima beban yang sama.



GAMBAR 4. 5
FAIRNESS INDEX P4 ALGORITMA ROUND ROBIN

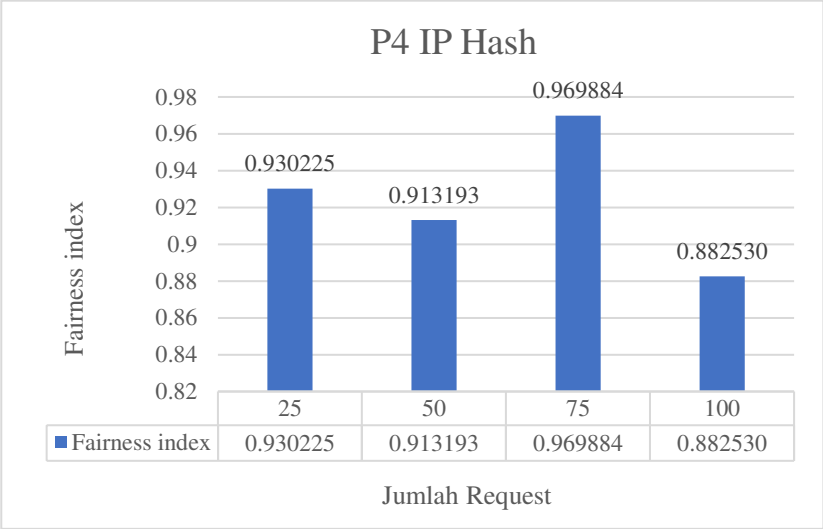
5. *Fairness Index* P4 IP Hash

Pada gambar 4.6 merupakan distribusi request dengan menggunakan P4 IP hash. Algortima ini akan mengambil sumber dan tujuan alamat IP dari client dan server untuk menghasilkan key hash yang unik. Kunci ini digunakan untuk mengalokasikan client ke server tertentu. Sehingga request yang diterima oleh server tidak merata untuk setiap servernya



GAMBAR 4. *6*
DISTRIBUSI P4 ALGORITMA IP HASH

Pada Gambar 4.7 merupakan hasil dari *Fairness Index* dari algoritma IP hash. Pada pemberian beban atau request setiap server menerima beban yang berbeda-beda ini dikarenakan IP hash mendistribusikan beban sesuai pencocokan *hash key* dari client dan server, satu server bisa menerima beban atau request lebih besar dari server lainnya. Seperti pada pemberian 100 *request* server 4 memiliki nilai *fairness index* terendah. Algoritma tidak cocok digunakan untuk server yang memiliki spresifikasi yang sama karena beban tidak di distribusikan secara merata.



GAMBAR 4. 7
FAIRNESS INDEX P4 ALGORITMA IP HASH

V. KESIMPULAN

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

A. Kesimpulan

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

1. Pengaruh dari algoritma round robin dan IP hash menggunakan P4 adalah proses load balancing langsung terjadi pada *data plane* switch dan dengan P4 program dapat membut program load balancing sesuai dengan kebutuhan agar kinerja load balancing bisa lebih efektif.
2. Hasil dari *fairness index* algoritma round robin memiliki hasil yang lebih baik dibandingkan dengan algoritma IP hash karena pada algoritma round robin beban yang diterima server sama dan ini menandakan optimasi yang dilakukan algoritma round robin bekerja dengan baik. Algoritma round robin sangat cocok untuk digunakan pada server yang memliki spesifikasi yang sama.
3. Hasil pengujian *throughput* pada algoritma round robin mendapatkan nilai rata-rata 127.61 KB/s dan IP hash dengan P4 memiliki nilai rata-rata 127.50 KB/s. Hasil *throughput* antara algoritma round robin dan IP hash tidak jauh berbeda.
4. Hasil pengujian *response time* algoritma round robin dengan P4 lebih cepat dibandingkan dengan P4 algoritma IP hash dan ini merupakan hasil optimasi yang dilakukan oleh algoritma round robin. Nilai rata-rata algoritma round robin yaitu 3.31 ms sedangkan pada IP hash rata-rata nya 12.16 ms.
5. Hasil pengujian *request loss* algoritma round robin dan IP hash dengan P4 memiliki nilai yang sama. Dimana niali *request loss* yaitu 0% yang artinya dapat mendistribusikan *request* dengan baik.

B. Saran

Saran yang dapat diberikan dalam penelitian proyek akhir ini. Sebagai berikut.

1. Penggunaan Controller SDN seperti ONOS, Pox, Ryu, dan lain-lain.
2. Penggunaan algoritma load balancing selain Round Robin dan IP hash.
3. Penggunaan *hardware* yang terinstall ubuntu agar *resource* yang digunakan lebih efesien, tidak menggunakan virtualisasi *software*.

REFERENSI

[1] T. Emad Ali, A. Hussein Morad, and M. A. Abdala, “Load Balance in Data Center SDN Networks,” *Int. J. Electr. Comput. Eng.*, vol. 8, no. 5, p. 3084, 2018, doi: 10.11591/ijece.v8i5.pp3084-3091.

[2] P. Monika, “Analisis performansi Jaringan Software Defined Network Menggunakan Metode Intent Monitor and Reroute (IMR) Pada Kontroler ONOS,” *J. Elektro dan Komun. Terap.*, 2019.

[3] M. Budiu and C. Dodd, “The P416 programming language,” *Oper. Syst. Rev.*, vol. 51, no. 1, pp. 5–14, 2017, doi: 10.1145/3139645.3139648.

[4] D. Systems, “Load Balancing Implementation Strategy for Various Services in Software Defined Network using ONOS Controller.”

[5] L. Ronny, C. Negara, W. Yahya, and R. Primananda, “Analisis Dan Implementasi Load Balancing Pada Web Server Dengan Algoritme Shortest Delay Pada Software Defined Network,” *J. Pengemb. Teknol. Inf. dan Ilmu Komput. Univ. Brawijaya*, vol. 2, no. 9, pp. 2791–2797, 2018.

[6] B. Arifwidodo, V. Metayasha, and S. Ikhwan, “Analisis Kinerja Load Balancing pada Server Web Menggunakan Algoritma Weighted Round Robin pada Proxmox VE,” *J. Telekomun. dan Komput.*, vol. 11, no. 3, p. 210, 2021, doi: 10.22441/incomtech.v11i3.11775.

[7] A. F. Pangestu and N. Fajar, “Implementasi Load Balancing dengan Algoritma Weighted Round

Robin menggunakan NGINX," no. December, 2020.

[8] K. W. Murti, T. A. Riza, and A. Mulyana, "Comparative Analysis of Load Balancing Dynamic Ratio and Server Ratio Algorithms," *Proceeding - 1st FORTEI-International Conf. Electr. Eng. FORTEI-ICEE 2020*, pp. 162–167, 2020, doi: 10.1109/FORTEI-ICEE50915.2020.9249815.

[9] I. Hidayah, R. Munadi, and I. D. Irawati, "Implementasi High-Availability Web Server Menggunakan Load Balancing As a Service Pada Openstack Cloud," *e-Proceeding Eng.*, vol. 6, no. 3, pp. 10278–10285, 2019.

[10] A. Ikram, S. Arif, N. Ayub, and W. Arif, "Load Balancing In Software Defined Networking (SDN)," *MAGNT Res. Rep.*, vol. 5, no. 1, pp. 298–305, 2018, doi: 1444-8939.2018/5-1/MRR.33.

[11] L. Fani, I. Ardy, A. Bhawiyuga, and W. Yahya, "Implementasi Load Balancer Berdasarkan Server Status pada Arsitektur Software Defined Network (SDN)," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 2, no. 5, pp. 2135–2143, 2018.

[12] A. Mustofa and D. Ramayanti, "Implementasi Load Balancing dan Failover to Device Mikrotik Router Menggunakan Metode NTH (Studi Kasus: PT.GO-JEK Indonesia)," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 7, no. 1, p. 139, 2020, doi: 10.25126/jtiik.2020701638.

[13] E. Haleplidis, S. Denazis, J. H. Salim, O. Koufopavlou, D. Meyer, and K. Pentikousis, "SDN Layers and Architecture Terminology," vol. RFC7426, pp. 1–35, 2015, [Online]. Available: http://tools.ietf.org/html/draft-haleplidis-sdnrg-layer-terminology-04.

[14] A. Maleki, M. Hossain, J. Georges, E. Rondeau, and T. Divoux, "An SDN Perspective to Mitigate the Energy Consumption of Core Networks – GÉANT2," *Int. Seeds Conf. 2017, Leeds*, no. September, 2017.

[15] R. M. Negara and R. Tulloh, "Analisis Simulasi Penerapan Algoritma OSPF Menggunakan RouteFlow pada Jaringan Software Defined Network (SDN)," *J. Infotel*, vol. 9, no. 1, p. 75, 2017, doi: 10.20895/infotel.v9i1.172.

[16] H. A. Friwansya, I. D. Irawati, Y. S. Hariyani, F. I. Terapan, and U. Telkom, "Implementasi Protokol Routing Ebgp Pada Software Defined," *E-Proceeding Apllied Sci.*, vol. 4, no. 3, pp. 2453–2462, 2018.

[17] O. N. Foundation, "OpenFlow Switch Specification Version 1.3.0 (Wire Protocol 0x04)," *Current*, vol. 0, pp. 1–36, 2012.

[18] W. Braun and M. Menth, "Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices," *Futur. Internet*, vol. 6, no. 2, pp. 302–336, 2014, doi: 10.3390/fi6020302.

[19] M. Budiu, and C. Dodd, "The P4-16 Programming Language," ACM SIGOPS Operating Systems Review (OSR), Vol. 51, no 1., 2017..

[20] Stordis, "P4 Programming Language", [online]. Tersedia di : https://stordis.com/p4-programming-language/..

[21] The P4 Language Consortium, "P4 16 Language Specification version 1.2.2," pp. 1–163, 2020, [Online]. Available: http://p4.org.

[22] A. P. I. W. Group, "P4Runtime Specification," pp. 1–54, 2018.

[23] Mininet, "Mininet Official Documentation", [online]. Tersedia di : http://mininet.org/, Diakses pada: 25/11/2021..

[24] M. Rosalia, R. Munadi, and R. Mayasari, "Implementasi HigRosalia, M., Munadi, R., & Mayasari, R. (2016). Implementasi High Availability Server Menggunakan Metode Load Balancing dan Failover pada Virtual Web Server Cluster. E-Proceeding of Engineering, 3(3), 4496–4503.h Availability Server Menggu," *e-Proceeding Eng.*, vol. 3, no. 3, pp. 4496–4503, 2016.

[25] E. Nurmiati, "Analisis Dan Perancangan Web Server Pada Handphone," *Stud. Inform. J. Sist. Inf.*, vol. 5, no. 2, pp. 1–17, 2012, [Online]. Available: http://download.portalgaruda.org/article.php?article=2481&val=329&title=ANALISIS DAN PERANCANGAN WEB SERVER PADA HANDPHONE.

[26] H. Nasser and T. Witono, "Analisis Algoritma Round Robin, Least Connection, Dan Ratio Pada Load Balancing Menggunakan Opnet Modeler," *J. Inform.*, vol. 12, no. 1, 2016, doi: 10.21460/inf.2016.121.455.

[27] I. P. A. Suwandika, M. Abdurohman, and M. A. Nugroho, "Analisis Performansi Load Balancing menggunakan Algoritma Least Connection dan IP Hash melalui Jaringan SDN pada Web Server."

[28] C. H. Ke and S. J. Hsu, "Load balancing using P4 in software-defined networks," *J. Internet Technol.*, vol. 21, no. 6, pp. 1671–1679, 2021, doi: 10.3966/160792642020112106009.

[29] I. A. Prakoso, S. N. Hertiana, and F. Dewanta, "Analysis of Fuzzy Logic Algorithm for Load Balancing in SDN," *2021 4th Int. Semin. Res. Inf. Technol. Intell. Syst. ISRITI 2021*, pp. 401–406, 2021, doi: 10.1109/ISRITI54043.2021.9702876.