Implementasi Load Balancing Pada Server Dengan Menggunakan Algoritme *Least Traffic* Pada Software-Defined Network

Nur Fauzi[1], Widhi Yahya[2], Adhitya Bhawiyuga[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]nfauzi42@gmail.com, [2]widhi.yahya@ub.ac.id, [3]bhawiyuga@ub.ac.id

Abstrak

Pada penggunaan teknologi jaringan komputer yang terus berkembang pesat. Salah satu solusi untuk permasalahan tersebut adalah *Software Defined Network* yang merupakan konsep memisahkan antara *control plane* dan *data plane*.*Control plane* SDN yang bersifat *programmable* memungkinkan untuk menerapkan berbagai aplikasi jaringan, salah satunya *load balancing*. *Load balancing* merupakan metode pendistribusian kinerja server dalam melayani setiap *request* yang ada dan membagi permintaan yang masuk untuk diberikan ke suatu server dengan memberikan layanan yang sama. Algoritme*load balancing* yang digunakan pada penelitian ini adalah Algoritme *least traffic* merupakan algoritme yang mengalokasikan jalur koneksi ke server dengan *traffic* terpendek. Algoritme ini akan diuji dan dianalisis kinerjanya. Pengujian dilakukan dengan 3 kategori yaitu *low*, *medium*, dan *high*. Parameter pengujian yang digunakan adalah pengujian fungsionalitas, *throughput*, *packet loss*, dan waktu korvegensi.Pada pengujian fungsionalitas mapping server algoritme *least traffic* menghandle *request* dari *client* lebih bagus dibandingkan algoritme *round-robin* karena algoritme *least traffic* tidak akan memilih jalur ke server yang terdapat beban *traffic* dan akan memilih jalur ke server lainnya.Pengujian *throughput* algoritme *least traffic* lebih unggul dibandingkan algoritme *round robin* pada kategori *high*.Sedangkan *packet loss roud robin* memiliki rata-rata 6,96% lebih tinggi dibandingkan *least traffic* dengan rata-rata 2,97%.Pada hasil pengujian waktu konvergensi, Waktu konvergensi yang diperoleh Algoritma *Round Robin* adalah 10,51 detik berbanding 14,80 detik waktu yang didapat algoritma *least traffic*.

Kata kunci: *software defined network, load balancing, least traffic, round robin*

Abstract

On the use of the computer network technology continues to evolve rapidly. One of the solutions to these problems are Software Defined Network which is the concept of separating the control plane and data plane. Control plane SDN that are programmable allowing to apply a variety of applications, including network load balancing. Load balancing is the method of distributing server performance in serving every request and served to divide the incoming requests to be given to a server by providing the same service. The load balancing algorithms used on this research is the least traffic Algorithms. Algorithm the algorithm is least traffic allocate line connection to the server with the shortest traffic. This algorithm will be tested and analyzed performance. Testing conducted with the 3 categories, namely low, medium, and high. The test parameters used aretesting functionality, throughput, packet loss, and time korvegensi. On testing the functionality of mapping server algorithms least traffic handling big request from the client better than round-robin algorithm because the algorithm least traffic would not choose the path to the server that there is a burden of traffic and will select the path to another server. Throughput testing algorithms least traffic is superior compared to the round robin algorithm on the category of low packet loss While roud robin has an average of 6.96% higher compared to the least traffic with an average of 2.97%. On the results of the test of convergence, convergence time is the time gained Round Robin Algorithm is 10.51 seconds compared to 14.80 seconds time obtained Algorithm Least Traffic.

Keywords: *software defined network, load balancing, least traffic, round robin*

1. PENDAHULUAN

Perkembangan Teknologi komputer pada saat ini telah mengalami peningkatan yang begitu pesat.Demikian juga dengan *internet* dimana setiap orang bisa saling terhubung dengan jaringan yang sangat luas.Peningkatan jumlah pengguna ini tentu saja berpengaruh terhadap kinerja server. Sebagai penyedia layanan server yang pada mulanya hanya bisa melayani koneksi beberapa klien saja, kini dituntut harus melayani banyak klien. *Software defined network* (SDN) adalah sebuah teknologi jaringan dengan paradigma pemisahan antara *control plane* dan *data plane* pada perangkat jaringan seperti *router* dan *switch*. *Control plane* berfungsi mengatur logika pada perangkat, sedangkan *data plane* berfungsi untuk meneruskan paket yang masuk ke suatu port menuju port tujuan dengan komunikasi pada *control plane*. Cara komunikasi antara perangkat dan *controller* menggunakan sebuah protocol yang disebut dengan *Openflow*. *Openflow* adalah standar komunikasi protokol yang mampu melakukan pemisahan antara *control plane* dan *data plane* dari sebuah perangkat jaringan, serta mampu menciptakan komunikasi yang sangat baik antara *control plane* dan *data plane*.

Dengan membuat *control plane* secara terpusat, *Software defined network* (SDN) memberikan managemen jaringan yang pengaturanya lebih fleksibel, mudah diatur dalam segi keamanan, optimasi sumberdaya jaringan secara dinamis, bahkan pengaturan jaringan dapat dilakukan sendiri tanpa menunggu perkembangan dari vendor untuk pengoptimalan jaringan. (Foundation, 2017). *Controller* SDN yang bersifat programmable memungkinkan untuk menerapkan aplikasi seperti *load balancing*, *intrusion detection*, *multimedia multicast*, *routing* sampai berbagai macam *virtualisasi*(Azodolmolky, Software Defined Network with OpenFlow, 2013). Berkaitan dengan *load balancing*, jika beberapa client ingin mengakses layanan yang sama pada jaringan, masing-masing client akan di arahkan ke salah satu server dari beberapa server yang ada pada server yang terdistribusikan, sehingga client dapat dilayani dengan cepat dan meringankan beban server.

Terdapat beberapa algoritme load balancing yang paling banyak digunakan antara lain *round robin, least- connection*. *Round robin* merupakan algoritme *load balancing* yang dilakukan dengan memberi giliran masing-masing server secara sirkular, sehingga tidak membebankan traffic dari setiap server. *Least-connection* bekerja memilih server dengan koneksi *outgoing* (keluar) yang paling rendah.

Berdasarkan penjelasan diatas, algoritme tersebut tidak memperhatikan kepadatan lalu lintas yang ada pada jaringan. Semakin tinggi tingkat kepadatan lalu lintas jaringan maka akan membuat semakin tingginya resiko terjadinya kegagalan komunikasi data pada *link* tersebut menyebabkan paket yang dikirim mengalami keterlambatan datang ataupun paket tersebut hilang dikarenakan adanya kenaikan traffic mendekati throughput.

Oleh karena itu, penelitian ini bertujuan untuk mendistribusikan beban permintaan user dengan pemilihan jalur berdasarkan *traffic* terendah pada *Software Defined Network. Parameter* yang digunakan untuk mengukur kinerja algoritme diantaranya adalah mapping server, *throughput, packet loss*, waktu konvergensi. Diharapkan penelitian ini mampu memberikan alternatif penentuan algoritme load balancing pada *Software Defined Network.*

2. KAJIAN KEPUSTAKAAN

2.1 Software Defined Network

Software defined network merupakan sebuah paradigma baru di dunia networking, merupakan sebuah pendekatan baru untuk membangun, mendesain serta me-manage jaringan komputer. Pada dasarnya SDN adalah jaringan dipisahkan dari forwarding dan dapat diprogram secara langsung. Migrasi dari *control logic* yang digunakan pada networking devices misalkan ethernet *switches* yang bersifat tightly integrated menjadi jaringan yang *accessible* dan *logically centralized controller*, sehingga mampu menyediakan jaringan yang *fleksibel*, *programable*, *vendor-agnostic*, *cost efficient*, dan arsitektur jaringan yang lebih inovatif

2.2 Openflow

Openflow adalah sebuah *control interface* yang memungkinkan untuk memprogram switch pada data plan sehingga administrator

dapat mengontrol secara langsung lalu lintas paket pada forward plan atau data plan melalui *interface OpenFlow* ini. *OpenFlow* mendefinisikan infrastruktur *flow-based forwarding* dan *Application Programmatic interface* (API) standar yang memungkinkan *controller* untuk mengarahkan fungsi dari *switch* melalui saluran yang aman (secure chanel) (Sudiyatmoko, Hertiana, & Negara, Analisis Performansi Perutingan Link State Menggunakan Algoritma Djikstra Pada Platform Software Defined, 2016).

2.3 Load Balancing

Load balancing adalah teknik untuk mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara seimbang, agar trafik dapat berjalan optimal, memaksimalkan *throughput* ,memperkecil waktu tanggap dan menghindari kelebihan beban pada salah satu jalur koneksi. Hingga saat ini, terdapat berbagai macam *algoritme* untuk *load balancing*, seperti *round robin*, *least-connection*, dan *least loaded*. *Round robin* merupakan algoritme *load balancing* yang dilakukan dengan memberi giliran masing-masing *switch/router* secara berurutan dan sirkular, sehingga tidak mementingkan beban *traffic* dari setiap *switch/router*. *Least-connection* bekerja dengan memilih *switch/router* dengan koneksi *outgoing* (keluar) yang paling rendah. Sedangkan *least-loaded* akan memilih *switch/router* dengan beban kerja yang paling rendah.

2.4 Algoritma Least traffic

Pada mekanisme pemilihan server yang digunakan untuk sistem *load balancing* menggunakan *software defined network* dilakukan pembagian *traffic* berdasarkan request yang dilakukan oleh *client*. Setiap *client* yang melakukkan *request* untuk koneksi pertama maka akan diarahkan traffic paket tersebut menuju ke server 1, selanjutnya diarahkan ke server berdasarkan jalur *traffic* yang rendah dan permintaan *request* ke sekian kali dari *client* akan terus diarahkan berdasarkan perulangan pemilihan server dengan *traffic* rendah hinggagseluruh request selesai. Pemilihan server berdasarkan variabel yang digunakan pada *traffic* dan *path* yang berbeda.

2.5 Controller Ryu

Ryu merupakan framework jaringan berbasis komponen untuk *software defined network*.Pengembangan aplikasi untuk Ryu dapat dilakukan dengan menggunakan bahasa *Python* atau dengan mengirimkan pesan JSON melalui API yang tersedia. Ryu mendukung berbagai protokol untuk memanajemen jaringan antara lain *OpenFlow*, *NetConf*, *Of-config* dll. Kebutuhan atas *Ryu* sebagai *OpenFlow controller* adalah karena *Ryu* mendukung *OpenFlow* versi 1.0 hingga 1.5, dimana pada *OpenFlow* versi 1.1 tersedia group actions yang dapat digunakan untuk *multipath routing*. *Ryu* di sini digunakan sebagai *Controller* SDN dan sebagai *framework* untuk mengembangkan sebuah sistem *multipath routing* di *OpenFlow* SDN

2.6 sFlow

sFlow-RT merupakangsebuah *tool*guntuk memonitor jaringangsecara *real-time* untuk SDN dengangmenggunakan teknologi analitik asinkron dari InMongda memungkinkan aplikasi pada jaringan SDN yang memperhatikan perform , seperti *load-balancing*, perlindungan terhadap DDoS, dan sebagainya. *sFlow-RT*akan menerima aliran telemetri secara kontinu dari agen-agen *sFlow* yang terpasang pada perangkat-perangkat jaringan, *host*, dan aplikasi dan mengubahnya menjadi metrik yang dapat di tindaklanjuti dan dapat diakses pada sebuah REST API. REST API tersebut memudahkan berbagai hal pada SDN antara lain melakukan konfigurasi terhadap pengukuran yang terkostumisasi, menerima metrik, menetapkan *threshold*, dan menerima notifikasi. Aplikasi terhadap *sFlow* dapat dibuat secara eksternal yang dapat ditulis dalam bahasa apapun yang mendukung pemanggilan HTTP/REST, ataupun secara internal dalam sFlow-RT dengan menggunakan bahasa JavaScript/ECMAScript(InMon, 2017)

2.7 Mininet

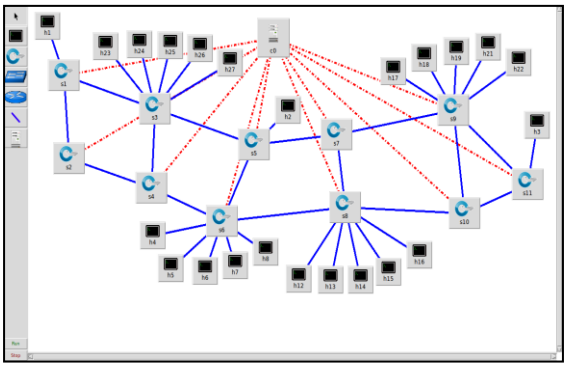
Mininet adalah merupakan sebuah *emulator* jaringan yang mensimulasikan koleksi dari *host end*, *switch* , *router* dan *link* pada single kernel linux.Dari masing -

masing elemen ini disebut "*host*" menggunakan virtualisasi ringan untuk membuat sistem tampilan tunggal sehingga terlihat jaringan yang lengkap,menjalankan *kernel* yang sama,sistem dan *user code*. Pada *mininet* ini dilakukan perancangan jaringan dengan topologi yang diinginkan. Secara sederhana *mininet* ini berfungsi untuk emulasi pada bagian data *path* untuk mengetes konfigurasi jaringan SDN. Sedangkan untuk melakukan testing pada *mininet* dapat dilakukan dengan *command* "sudo mn". Dengan *command* ini *mininet* akan mengemulasikan konfigurasi jaringan SDN yang terdiri dari 1 *controller*, 1 *switch* dan 2 *host*.

3. PERANCANGAN SISTEM

3.1 TOPOLOGI

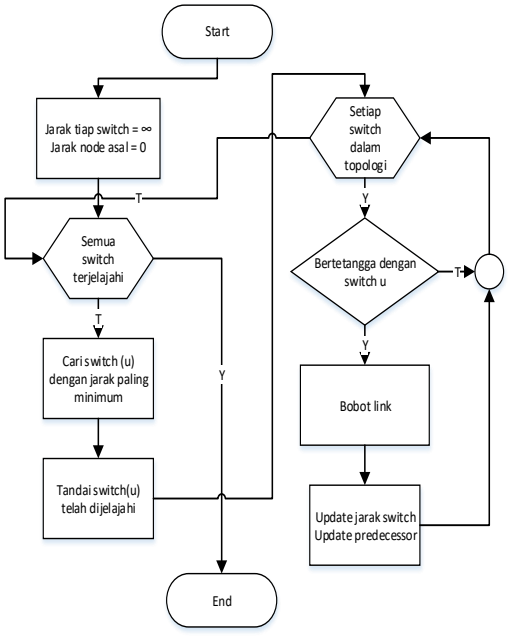
Perancang sistemakan dilakukan dengan simulasi pada aplikasi emulator jaringan bernama mininet yang terdapat di *Operating System* berbasis *linux*.dan *Controller* SDN bernama *Ryu*.Kemudian juga diperlukan perancangan topologi dan perancangan algoritme *load balancing* yaitu algoritme *least traffic* yang nantinya akan dibandingan dengan algoritme *round robin* untuk melihat algoritme yang memiliki performa lebih baik pada jaringan *Software Defined Network*.Topologi ini dibuat menggunakan 11 switch dan 23 host.Untuk menghubungkan antar host dan switch dilakukanh dari *create link* (*host* 1, *host* 2 dan *host* 3) sebagai server.(*host* 4, *host* 5, *host* 6, *host* 7, *host* 8, *host* 12, *host* 13, *host* 14, *host* 15, *host* 16, *host* 17, *host* 18, *host* 19, *host* 21, *host* 22, *host* 23, *host* 24, *host* 25, *host* 26, *host* 27) sebagai *client*.Untuk pengujian ini dilakukan setting pada *link bandwitch* sebesar 100 Mbit antar *switch*. Setelah semua topologi selesai dibuat *controller* dan *switch* akan melakukan *starting* sistem.



Gambar 1. Topologi Mininet

3.2 ROUTING

Pada perancangan *routing* berisi tentang bagaimana cara membangun sebuah fungsi *routing* yang dapat meneruskan paket dari pengirim menuju penerima dengan jalur yang telah ditentukan oleh algoritme *routing*. Algoritme *routing* yang digunakan adalah algoritme *Dijkstra*. Algoritme *Dijkstra* digunakan untuk mencari jalur terpendek berdasarkan bobot link yang telah ditentukan. Diagram alir algoritme *Dijkstra* seperti pada Gambar 2. bawah ini.



Gambar 2. Perancangan Routing

Pada gambar 2 flowchart diatas , langkah pertama adalah menginisialisasi jarak ke semua *switch* menjadi tak terhingga dan memberi nilai jarak *node* asal menjadi 0. Kemudian akan mencari *switch* dengan jarak paling minimum yang akan ditandai agar tidak dijelajahi pada iterasi selanjutnya. Setelah *switch* tersebut didapatkan, langkah

selanjutnya adalah melakukan *update* bobot dan *predecessor* pada *switch* yang bertetanggaan berdasarkan dengan bobot link yang telah ditentukan. Proses tersebut akan dilakukan sampai semua *switch* pada topologi berhasil dijelajahi.

3.3 MONITORING LINK

Berdasarkan pada perancangan topologi dan observasi terhadap sistem yang akan dibangun, algoritme *monitoring* jalur diimplementasikan menggunakan library *ryu controller*. Algoritme ini bertujuan untuk mengukur *traffic* antar 2 buah *switch*. *Traffic* antar *switch* yang temukan akan ditotal untuk mengetahui *cost* dari sebuah jalur. Dengan menggunakan *sFlow-RT*, bisa mengambil *traffic rate* dari tiap *link* secara *real-time* dan mengirimkan data tersebut ke *controller*.

3.3 ALGORITME LEAST TRAFFIC

Pada perancangan algoritme *least traffic* berisi tentang bagaimana cara membangun sebuah fungsi *load balancing* yang dapat mendistribusikan beban *trafik* pada dua atau lebih jalur koneksi secara seimbang. Algoritme *routing* yang digunakan adalah algoritme *least traffic* digunakan untuk mencari jalur terpendek berdasarkan bobot *traffic* yang rendah yang telah ditentukan. Diagram alir algoritme *least traffic* seperti pada Gambar 3. bawah ini

Gambar 3. Flowchart Algoritme Least Traffic

Pada gambar 3 flowchart diatas. Pencarian algoritme *least traffic* dimulai denganmelakukan perulangan yang digunakan untuk mencari jalur menuju tiap server.Kemudian dari semua jarak tersebut akan dipilih server dengan jarak paling kecil.

4. IMPLEMENTASI

Pada bagian implementasi memuat langkah-langkah yang dilakukan untuk memasang perangkat lunak pendukung untuk melakukan pengembangan sistem yaitu *Mininet*, *Ryu Controller* dan *sFlow-RT* pengembangan pada program *controller*.

5. PENGUJIAN

Tujuan pengujian ini adalah untuk mengetahui perbandingan performa dari algoritma *least traffic* dan *round robin* untuk *load balancin* pada *sofware defined network*. Pengujian Fungsionalitas digunakan untuk melihat server dalam menghandle *request* dari beberapa client. Pengujian *throughput* dan *packet loss* menggunakan *tool iperf* dengan satuan Mbps, waktu konvergensi menggunakan *tool ping*. Dalam penelitian ini digunakan untuk mencari jumlah maksimal *request* menggunakan iperf dimulai dengan request pada jumlah host dibawah 50 hingga ditemukan maksimal request yang dikategorikan menjadi beberapa kategori yaitu low, medium, dan high. Nilai high tentu saja 20 host. Untuk nilai medium adalah setengah dari nilai high yaitu 10 host. Dan untuk nilai low adalah setengah dari nilai medium yaitu 5 host

5.1 PENGUJIAN FUNGSIONALITAS

Untuk pengujian mengukur *fungsionalitas*, Pengujian ini dilakuka untuk membandingkan *throughput* antara algoritme *least traffic* dan algoritme *round-robin* Pada Pengujian ini, h1, h2,h3 sebagai *server* dan h4 sampai h16, h17 sampai h27 sebagai *client*. Pengujian *fungsionalitas* ini dilakukan dengan beban traffic TCP selama 1000 detik di salah satu host yaitu host 29 dan host 30. Pengujian ini digunakan untuk melihat server dalam menghandle request dari beberapa client pada tabel 1. berikut ini.

Tabel 1. Pengujian Fungsionalitas

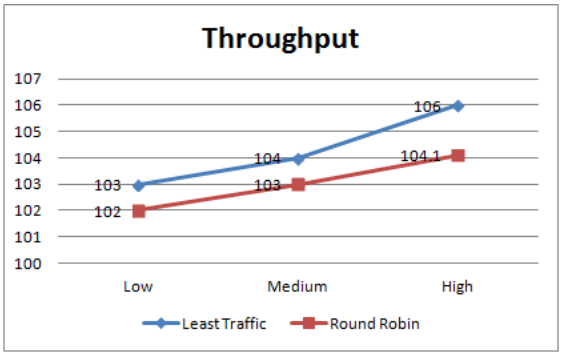
Algoritma

Server yang di uji	Least Traffic	Round Robin
Server 1	0	7
Server 2	12	7
Server 3	8	6

Pengujian yang telah dilakukan pada tabel 1 bahwa algoritme *least traffic* dalam menghandle *request* dari *client* lebih bagus dibandingkan algoritme *round-robin* karena algoritme *least traffic* tidak akan memilih jalur ke server yang terdapat beban *traffic* dan akan memilih jalur ke server lainnya, algoritme ini mengalokasikan koneksi ke server dengan *traffic* terendah. Sedangkan algoritme *round-robin* akan tetap melewati jalur yang telah diberikan beban *traffic* karna algoritme tersebut tidak memperhatikan *disturbance conditions* dan tetap akan memilih jalur ke server dengan cara berurutan.

5.2 THROUGHPUT

Pada pengujian ini telah ditentukan 3 host yang menjadi server. *Host* yang bertindak sebagai server dijalankan terlebih dahulu lalu host client dilakukan reques secaabersamaan. Nilai *throughput* yang diambil adalah nilai rata-rata yang tercetak pada terminal host client. Pada pengujian ini menggunakan interval waktu 2 dan waktu kirim 10. Grafik *throughput* pada perbandingan 2 algoritme dapat dilihat pada gambar 4 berikut ini.

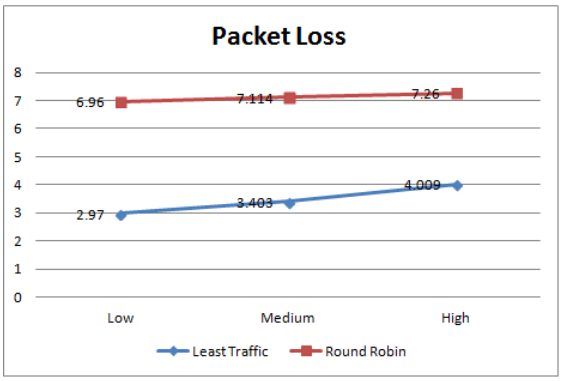


Gambar 4. Pengujian Throughput

Pada gambar 4 merupakan masing-masing algoritme least traffic dan round robin.Memperoleh nilai throughput yang cukup maksimal ketika pada kategori *high*. Keduanya pun tetap mengalami penurunan saat kategori *medium* dan *low*. Dari keseluruhan pengujian, perbedaan pada *throughput* kedua algoritme memiliki perbedaan yang cukup jauh dimana algoritme *least traffic* lebih unggul.

5.3 PACKET LOSS

Pada pengujian ini hampir sama dengan throughput yaitu telah ditentukan 3 host yang menjadi server. Host yang bertindak sebagai server dijalankan terlebih dahulu lalu host client dilakukan request secara bersamaan. Nilai packet loss yang diambil adalah nilai rata-rata yang tercetak pada terminal host client. Pada pengujian ini menggunakan interval waktu 2 dan waktu kirim 10. Grafik *packet loss* pada perbandingan 2 algoritme dapat dilihat pada gambar 5 berikut ini

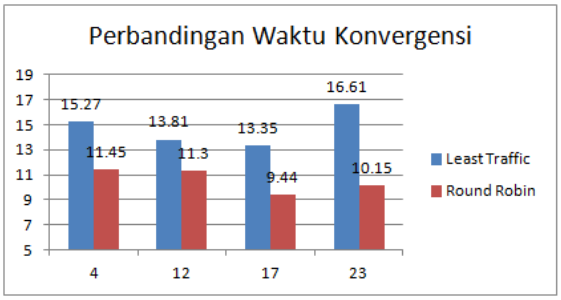


Gambar 5. Pengujian Packet Loss

Pada gambar 5 perbandingan packet loss, algoritme *least traffic* sedikit lebih unggul karena *packet loss* yang terjadi di bawah 3% pada kategori *low*, dimana nilai tersebut merupakan kategori sangat bagus. Sedangkan algoritme *round robin* saat kategori *low*, packet loss yang terjadi sebesar 6,96%. Namun ketika kategori *medium* dan *high*, kedua algoritme sama-sama mengalami *packet loss* di atas 3% yang merupakan kategori bagus.

5.3 WAKTU KONVERGENSI

Pengujian ini terdiri dari 11 *switch* dan 23 *host*, seperti (h1,h2,h3) sebagai server dan lainya sebagai client. Pada pengujian waktu konvergensi dilakukan disisi client dengan cara pada setiap *host* dilakukan pengujian sebanyak 1 kali dengan percobaan di salah satu host saja, seperti : (s3,h23), (s6,h4), (s8,h12), (s9,h17). Grafik waktu konvergensi pada perbandingan 2 algoritme dapat dilihat pada gambar 6 berikut ini.



Gambar 6. Pengujian Waktu Konvergensi

Berdasarkan Gambar 6 waktu konvergensi perbedaan waktu konvergensi.Algoritme *round robin* lebih unggul pada skenario *ping* dari *host* 4, *host* 12, *host* 17 dan *host* 23 karena *round robin* tidak memerlukan pebandingan jalur *traffic* hanya memerlukan perulangan index server.Semakin cepat waktu konvergensi, maka semakin baik sistem dalam menjalankan *load balancing*.

6. KESIMPULAN

Berdasarkan rumusan masalah, pengujian beserta hasil analisis terhadap data uji didapat beberapa kesimpulan, berikut adalah kesimpulan yang dapat diambil

1. Berdasarkan konsep perancangan, implementasi dan pengujian terhadap sistem load balancing menggunakan algoritme least traffic pada software defined network maka hasil yang didapatkan pembagian traffic pada jaringan dapat membantu meringankan beban server utama dalam memberikan pelayanan terhadap permintaan user.Pada pengujian fungsionalitasjmasing- masing Algoritme Least Traffic dalam menghandle request dari client lebih bagus dibandingkan Algoritme Round Robin karena Algoritme Least Traffic tidak akan memilih jalur ke server yang terdapat beban traffic dan akan memilih jalur ke server lainnya, Algoritme ini mengalokasikan koneksi ke server dengan traffic terendah. Sedangkan algorit me round robin akan tetap melewati jalur yang telah diberikan beban traffic.

2. Kinerja masing masing algoritme load balancing mengalami kenaikkan throughp ut yang signifikan ketika pada kategori high. Keduanya pun sama-sama mengalami penurunan saat kategori medium dan low , namun tidak begitu signifikan. Akan tetapi dari keseluruhan pengujian, algoritme least traffic memberikan hasil yang lebih baik daripada algoritmeround robin.Dalam pengujian packet loss, algoritme least traffic mendapatkan hasil dibawah 3% yang merupakan kategori sangat bagus dari semua percobaan. Sedangkan algoritme round robin mendapatkan lebih dari 3% yang merupakan kategori bagus.Untuk waktu konvergensi, pada pengujian algoritme least traffic diperoleh nilai rata-rata 14.80 detik, sedangkan algoritme round robin diperoleh nilai 10,51 detik. Dari hasil pengujian waktu konvergensi, algoritme round robin didapatkan hasil yang lebih unggul dibandingkan least traffic.

7. DAFTAR PUSTAKA

Azodolmolky, S. (2013). *Software Defined Network with OpenFlow*. Birmingham: Packt Publishing Ltd.

InMon. (2017). *sFlow-RT*. Retrieved from http://www.inmon.com/products/sFlow-RT.php

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., . . . Turner, J. (2008). *OpenFlow: Enabling Innovation in Campus Networks*. OpenFlow.

Mustafa, M. E., & Ibrahim, A. M. (2015). Load Balancing Algorithms Round-Robin (RR), Least-Connection, Least Loaded Efficiency. *International Journal of Computer and Information Technology*.

Open Networking Foundation. (2017, juli 19). Retrieved from Software-Defined Networking (SDN) Definition: https://www.opennetworking.org/sdn-definition/

Pepelnjak, I. (2013). OpenFlow and SDN. *Hype, Useful Tools or Panacea?,IP Space*.

Road, E. B., & Alto, P. (2012). Software Defined Networking. *The New Norm for Networks*, pp. 1-12.

Senthil Ganesha N, & Ranjani S. (2015). Dynamic Load Balancing using Software Defined Networks.

Sudiyatmoko, A. R.AAAAAAA, Hertiana, S. N., & Negara, R. M. (2AAA016). Analisis Performansi Perutingan Link StateMenggunakan Algoritma Djikstra Pada Platform Software Defined.

Syahidillah, W. MAAASSSS. (2012). *MULTIPATH ROUTING DENGAN LOAD-BALANCINGAAAA PADA OPENFLOW AAAAAASOFTWARE-DEFINED NETWORK.* Universitas Brawijaya.