Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 1, No. 12, Desember 2017, hlm. 1568-1577 http://j-ptiik.ub.ac.id

Analisis Perbandingan Performa Algoritma *Round Robin* dan *Least Connection* untuk *Load Balancing* pada Software Defined Network

Agung Nugroho[1], Widhi Yahya[2], Kasyful Amron[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]agungnugroho10294@gmail.com,[2]widhi.yahya@ub.ac.id, [3] kasyful @ub.ac.id

Abstrak

Perkembangan teknologi internet dimana para penggunanya yang menginginkan layanan yang selalu tersedia setiap saat semakin meningkat mengikuti perkembangan zaman yang biasanya belum tentu didukung dengan bertambahnya layanan *server* yang memadai. *Request* pengguna yang terlalu banyak menjadikan beban kerja pada *server* meningkat dengan cepat, mengakibatkan *server down* dalam waktu yang singkat serta dibutuhkan teknologi yang menangani penggunaan jaringan yang kompleks. Sebagai teknologi baru, *software defined network* menawarkan *scalability* dan *programmability* untuk penggunaan jaringan yang semakin kompleks seperti *load balancing*. Beberapa penelitian tentang *load balancing* pada *software defined network* (SDN) dengan berbagai algoritma yang diterapkan akan menghasilkan performa yang berbeda-beda, diantaranya yaitu algoritma *round robin* dan *least connection*. Pengujian dilakukan dengan 3 kategori *rate* yaitu *low*, *medium*, dan *high*. Parameter pengujian yang digunakan adalah *throughput*, *response time*, dan *CPU Usage* menggunakan *Httperf* dan *psutil*. Algoritma *round robin* lebih unggul dibandingkan algoritma *least connection* pada koneksi kecil. Sedangkan untuk koneksi besar algoritma *least connection* lebih unggul. Nilai rata-rata *response time* menujukan keunggulan algoritma *round robin* dibandingkan dengan algoritma *least connection*. *CPU Usage server* dengan algoritma *round robin* lebih stabil pada setiap kategori *rate*. Sedangkan algoritma *least connection* lebih ringan beban dari algoritma *round robin* meskipun nilainya mengalami kenaikan pada setiap kategori *rate*.

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

Abstract

The development of internet technology whom the users desire for the availaible services anytime tend to be increased following the current development which commonly not always supported by the increasing adequate server services. The highest users demand make the work load in server raised rapidly which cause down server in a short period along with the needs of technology which handle the utilization of complex connections. As a new technology, Software defined network offers scalability and programmability for the using of connections which become complex such as Balancing Web Server. There are some studies concerning with load balancing in Software defined network (SDN) with various algorithm which is applied will resulted difference performance from algorithm, , one of them is Round robin and Least connection. The experiment is done with 3 categories; low, medium, and high. The experiments parameter which is used are Throughput, Response time, and CPU Usage use Httperf and Psutil. Round robin Algorithm is more excellent than Least connection Algorithm in a tiny connection. On the other hand, for the bigger connection, Least connection Algorithm is excellent. The average value of time response shows the superiority of Round robin Algorithm rather than Least connection Algorithm. CPU Usage server with the Round robin Algorithm is more stable in each rate category. On the contrary, Least connection Algorithm is less of load than Round robin Algorithm in spite of the value is raise in each rate category.

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

1. PENDAHULUAN

Pekembangan teknologi saat ini berkembang sangat pesat khususnya pada pemanfaatan teknologi internet. Jaringan internet telah menjadi bagian infrasructur kritis dari bisnis, rumah, dan sekolah (McKeown, et al., 2008).Pengguna internet cenderung semakin meningkat mengikuti perkembangan zaman yang biasanya belum tentu didukung dengan bertambahnya layanan *server* yang cukup memadai. Menurut statistik Kementerian Komunikasi dan Informatika menyatakan, pengguna internet di Indonesia telah mencapai 82 juta orang. Dengan capaian tersebut, Indonesia berada pada peringkat ke-8 di dunia (Kemkominfo, 2014). Akibatnya akan menimbulkan masalah bagi para pengguna yang menginginkan layanan yang selalu tersedia setiap saat (*high availability*). *Request* pengguna yang terlalu banyak menjadikan beban kerja pada *server* meningkat dengan cepat yang mengakibatkan *server down* dalam waktu yang singkat serta dibutuhkan teknologi yang menangani penggunaan jaringan yang kompleks. Sehingga dibutuhkan sebuah teknologi yang mampu menangani masalah tersebut. Sebagai teknologi baru, *software defined network* menawarkan *scalability* dan *programmability* untuk penggunaan jaringan yang semakin kompleks seperti *load balancing web server*.

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 plene 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 *manage*men 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, 2012)

Beberapa mekanisme dalam pembagian beban pada *web server* dengan menggunakan teknik *load balancing*, yaitu suatu mekanisme pembagian beban *server* dengan beban trafik di distribusikan pada dua atau lebih jalur koneksi secara seimbang . Tujuan mekanisme tersebut agar memaksimalkan *throughput*, trafik dapat berjalan optimal,memperkecil waktu tanggap dan menghindari *overload* pada salah satu jalur koneksi (Sirajuddin, et al., 2012).

Ada beberapa penelitian tentang *load balancing* pada *software defined network* (SDN) dengan berbagai algoritma yang diterapkan. Banyaknya algoritma yang dapat diterapkan untuk *load balancing* membuat performa yang dihasilkan dari lagoritma yang digunakan berbeda-beda. Dua diantara berbagai algoritma yang ada yaitu algoritma *round robin* dan *least connection*. Algoritma *round robin* bekerja dengan cara membagi beban secara bergiliran dan berurutan dari satu *server* ke *server* lainnya. Pada penggunaan algoritma *round robin*, saat *user* mengakses halaman *web server* pertama akan diarahkan ke *server* utama, permintaan *user* yang selanjutnya akan diberikan kepada *server* yang lainnya (Mustafa & Ibrahim, 2015). Sehingga algoritma *round robin* dapat digunakan sebagai penyeimbang beban *server* dalam memberikan respon terhadap setiap permintaan yang dilakukan oleh *user*. Algoritma *least connection* melakukan pembagian beban berdasarkan banyaknya koneksi yang sedang dilayani oleh sebuah *server*. Pada *server* yang memiliki kemampuan pemrosesan yang sama, algoritma penjadwalan *least connection* akan mendistribusikan beban permintaan dengan baik karena permintaan yang panjang tidak akan disalurkan ke sebuah *server* (Kurniawan, 2013). Sehingga dengan algorima *least connection server* dapat memberikan respon yang baik terhadap permintaan yang dilakukan oleh *user* dan juga dapat digunakan sebagai penyeimbang beban pada *server*.

Pada penelitian sebelumnya belum diterapkan pada *real network*. Tentu hasil analisis performa yang didapatkan akan memiliki perbedaan dibandingkan pada *real network*.

Berdasarkan kebutuhan diatas maka penelitian ini akan berfokus pada analisis perbandingan performa algoritma *round robin* dan algoritma *least connection* untuk *load*

balancing. Kedua algoritma tersebeut dipilih karena sudah dibahas pada penelitian sebelumnya yang hanya diimplementasikan pada simulator. Arsitektur jaringan yang akan digunakan adalah *software defined network* karena memberikan manajemen jaringan yang fleksibel, dinamis dan kemudahan dalam melakukan monitoring jaringan yang terpusat pada *controller*. Load balancer akan di letakan pada *physical switch* dimana harus diinstall OpenWRT yang memliki banyak fitur salah satunya untuk keperluan penelitian ini. Pada penelitian ini akan diterapkan pada *real network* agar mendapat hasil yang lebih nyata. Parameter pengujian pada penelitian ini adalah *throughput*, *response time*, dan *CPU Usage*.

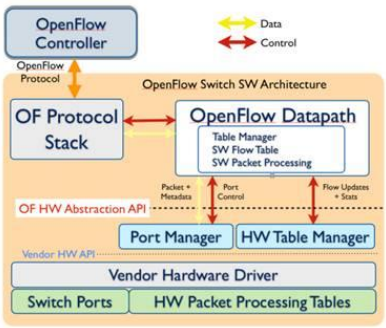
2. LANDASAN KEPUSTAKAAN

2.1 *Software defined network* (SDN)

Software defined network adalah sebuah paradigma baru di dunia *networking*, merupakan sebuah pendekatan baru untuk membangun, mendesain serta me-*manage* jaringan komputer. Sedangkan pengertian yang sebenarnya menurut *Open Networking Foundation* (organisasi pengembang *Sofware Defined Network*), bahwa *Sofware Defined Network* adalah suatu arsitektur jaringan dimana control network dipisahkan dari system forwardingnya, dan *controller* tersebut dapat kita program secara langsung.

2.4 *Openflow*

Openflow merupakan open standar komunikasi protokol yang mampu melakukan pemisahan antara control plene dan *data plane* dari sebuah perangkat jaringan, serta mampu menciptakan komunikasi yang sangat baik antara *control plane* dan *data plane* (*Openflow* Organization,2011).



Gambar 1. Arsitektur Open F*Low Switch*

Sumber: *Openflow* Organization

2.5 *Controller*

Controller merupakan "otak" pada jaringan *Software defined network* . Di dalam *Controller* ini terdapat pengendalian st*rate*gis yaitu mengelola kontrol aliran ke *switch* / router (melalui API) dan memiliki aturan logis di dalamnya sebagai penentu jalur terbaik yang dipilih. Selain itu, *controller* juga dapat melakukukan fungsional jaringan seperti *load balancing*,routing,dan banyak hal yang dapat di lakukan oleh *controller*.

2.6 *Load balancing*

Load balancing adalah suatu teknik yang digunakan untuk memisahkan antara dua atau banyak network link. Dengan mempunyai banyak link maka optimalisasi utilitas sumber daya, *throughput* atau *response time* akan semakin baik karena mempunyai lebih dari satu link yang bisa saling mem-backup pada saat network *down* dan menjadi cepat pada saat network normal. jika memerlukan *real*ibilitas tinggi yang memerlukan 100% koneksi uptime dan yang menginginkan koneksi upstream yang berbeda dan dibuat saling mem-backup (Lukitasari dan Oklilas,2010).

Load balancing adalah teknik untuk mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara seimbang, agar traffic dapat berjalan optimal, memaksimalkakn *throughput*, memperkecil waktu tanggap dan menghindari *overload* pada salah satu jalur koneksi (Sirajuddin,2012).

2.7 Algoritma *Round robin*

Algoritma *Round robin* merupakan algoritma yang paling sederhana dan paling banyak digunakan oleh perangkat *load balancing*. Algoritma *Round robin* bekerja dengan cara membagi beban secara bergiliran dan berurutan dari satu *server* ke *server* lainnya. Konsep dasar dari algoritma *Round robin* ini adalah dengan menggunakan time sharing, pada intinya algoritma ini memproses antrian secara bergiliran (Ellrod, 2010).

2.8 Algoritma *Least connection*

Algoritma *Least connection* melakukan pembagian beban berdasarkan banyaknya koneksi yang sedang dilayani oleh sebuah *server*. *Server* dengan koneksi yang paling sedikit akan diberikan beban berikutnya, begitu pula *server* dengan koneksi banyak akan dialihkan bebannya ke *server* lain yang bebannya lebih rendah (Ellrod, 2010)

2.9 Web Server

Web Server merupakan suatu perangkat baik perangkat keras mampun lunak yang menggunakan protokol *HTTP* atu *HTTPS* sebagai media dalam mengakses berkas-berkas di dalam suatu halaman *website* dengan menggunakan *web browser*. Penggunaan *web server* dapat sebagai penempatan suatu halaman *web*, sebagai penyimpanan atau *database* serta penggunaan aplikasi bisnis. (Kadir,2003)
Psutil

2.10 *Psutil*

Psutil merupakan salah satu library atau modul tambahan di dalam pemrograman *Python*. Library ini berfungsi untuk mengambil informasi sumber daya atau *resource* dari suatu perangkat komputer menggunakan *Python*. Nilai yang diambil bisa berupa Nilai *CPU*, *Memory*, dan sebagainya (Alexander Svendsen, 2014).

2.11 *Httperf*

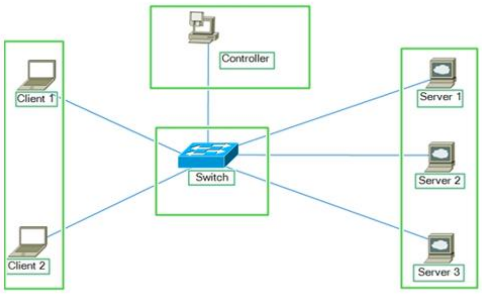
Pada saat melakukan pengujian terhadap *webserver*, banyak sekali *tool* yang dapat digunakan. Salah satu tools yang lumayan sering dipakai adalah *Httperf*. *Httperf*merupakan *tool* yangringan dan stabil ketika dilakukan pengujian performa sebuah *webserver* (David M, 1998).

3. METODOLOGI PENELITIAN

3.1. Studi Literatur

Studi literatur dilakukan untuk mempelajari mengenai dasar teori yang digunakan sebagai panduan penunjang penelitian ini. Dasar teori pendukung tersebut bisa diberoleh dari materi yang bersifat resmi seperti jurnal, e-book, forum, dan peneletian sebelumnya yang memiliki keterkaitan dengan penelitian ini.

3.2. Perancangan Sistem



Gambar 2. Diagram Perancangan Sistem

Pada gambar 2 merupakan diagran perancangan sistem. Dalam sistem ini akan dipecah menjadi beberapa subsistem yaitu Subsistem *Switch*, Subsistem *Controller*, Subsistem *Server* dan Subsistem *Client*

3.3. Analisi Kebutuhan

Analisis kebutuhan menentukan kebutuhan apa saja yang diperlukan pada penelitian ini. Sehingga penelitian ini dapat dilakukan untuk mendapatkan hasil penelitian serta dilakukan analisis. Berikut merupakan daftar kebutuhan yang diperlukan.

3.3.1 Kebutuhan Perangkat Keras

Perancangan sistem dalam peneletian ini membutuhkan perangkat keras yaitu

1. 3 buah *server* (virtual)
2. 1 buah PC sebagai *Client*.
3. 1 buah PC sebagai *controller*.
4. 1 buah *switch* (TP-Link TL-WR841ND).

3.3.2 Kebutuhan perangkat Lunak

Perangkat lunak memiliki peranan agar keseluruhan perangkat keras dapat terhubung dan mampu menjalankan peran sesuai dengan perancangan sistem.

1. Sistem operasi Ubuntu.
2. *Firmware* untuk perangkat *switch* yaitu OpenWRT versi 1.0.
3. VirtualBox digunakan sebagai pembangun *server* virtual.
4. *Python*
5. *POX*
6. Text editor
7. Browser

3.4. Implementasi Sistem

1. Konfigurasi pada perangkat *Switch*

Switch yang digunakan dalam penelitian adalah TP-Link TL-WR841ND. *Switch* ini telah dilakukan instalasi protokol *Openflow* agar sistem dapat berjalan pada perangkat Software Defined Network. Konfigurasi dilakukan sesuai aturan Pantou yang merupakan sumber resmi untuk panduan *Openflow* versi 1.0. Setelah itu lakukan konfigurasi standar *Openflow*.

Gambar 3. TP-Link TL-WR841ND

2. Konfigurasi pada perangkat *Controller*

Konfigurasi yang dilakukan pada *controller* adalah melakukan pengesetan IP sesuai dengan konfigurasi pada perangkat *switch* sebelumnya yaitu pada IP 192.168.2.10. *Controller* sudah harus memiliki library *Python* dan *POX*. Sehingga *controller* mampu menjalankan *load balancing* untuk penelitian ini.

3. Konfigurasi pada perangkat *Server*

Server pada penelitian ini menggunakan *server* virtual yaitu menggunakan Virtual Box sebagai pengganti *server real*. Dilakukan konfigurasi sebagai berikut.

- Jumlah core pada tiap *server* diset 2 core pada virtualbox untuk masing – masing *server* virtual.
- Melakukan set alamat IP dengan 3 *server* dengan cara yang sama dengan alamat IP :
  - 192.168.2.5 , alamat IP *Server* 1
  - 192.168.2.6 , alamat IP *Server* 2
  - 192.168.2.7 , alamat IP *Server* 3
- Installasi apache *web server*

4. Konfigurasi pada perangkat *Client* yaitu dengan melakukan pengesetan alamat IP dalam pada *Client* yaitu : 192.168.2.20. Inti dari ip *Client* tidak berbenturan dengan ip yang lain agar tidak confict.

5. Implementasi Algoritma *Load balancing* dalam bentuk kode program pada *controller POX*.

3.5. Pengujian dan Analasis

Tujuan pengujian ini adalah untuk mengetahui perbandingan performa dari algoritma *round robin* dan *least connection* untuk *load balancing* pada sofware defined network . Pengujian juga dilakukan untuk mengetahui hasil dari nilai rata-rata tiap paramater yang diberikan. *Request* digunakan untuk memberikan beban permintaan *Client* kepada *server* dalam satu waktu sehingga dapat diketahui kemampuan *server* dalam melakukan respon terhadap *Client*

Pengujian menggunakan *Httperf* menggunakan parameter *throughput* dengan satuan KB/s, *response time* dengan satuan ms/req, *CPU Usage* dengan satuan prosentase (%). Nilai pengujian *throughput* dan *response time* diambil dari percobaan yang dilakukan sebanyak 5 kali. Sedangkan untuk *CPU Usage* hanya dilakukan sekali pengujian dikarenakan sudah mencakup nilai variasi *CPU Usage*.. Pengujian dilakukan pada masing-masing algoritma dengan kondisi yang sama.

Skenario pengujian peratama kali dilakukan dengan mencari besar *rate* maksimal yang mampu ditangani sistem hingga sistem mengalami satu*rate*d point. Pada proses mencari maksimal *rate* dilakukan *request* menggunakan *Httperf* dimulai dengan *rate* dengan nilai kecil dibawah 100 hingga ditemukan maksimal *rate* dengan waktu pengujian 1 detik. Nilai *rate* maksimal yang didapatkan yaitu 160 req/s. Selanjutnya nilai *rate* dikategorikan menjadi 3 kategori yaitu *low*, *medium*, dan *high*. Nilai *high* tentu saja 160 req/s. Untuk nilai *medium* adalah setengah dari nilai *high* yaitu 80 req/s. Dan untuk nilai *low* adalah setengah dari nilai *medium* yaitu 40 req/s.

3.6. Kesimpulan

Pengambilan kesimpulan dilakukan setelah semua tahapann perancangan, implemetasi, dan pengujian sistem telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap penelitian yang telah dilakukan. Tahap terakhir penulisan adalah saran dan yang dimaksudkan untuk memberikan pertimbangan atas pengembangan penelitan yang lebih lanjut.

4. HASIL

4.1 Hasil Pengujian *Throughput*

Proses pengujian *throughput* dapat dilihat nilai perbandingan *throughput* antara algoritma *round robin* dan algoritma *least connection*. Satuan yang digunakan untuk *throughput* yaitu KB/s. Berikut hasil yang diperoleh :

Tabel 1. Hasil pengujian *throughput*

Algoritma	*Round robin*			*Least connection*		
Kategori	*Low*	Med	*High*	*Low*	Med	*High*
Rata - rata	455,72	879,08	1.746,22	453,52	885,94	1.751,36

Berdasarkan tabel diatas didapatkan hasil pengukuran *throughput* untuk tiap alogritma pada nilai Net I/O yang ditampilkan oleh *Httperf*. *Throughput* yang diamati dilakukan pada banyak koneksi *low*,*medium*, dan *high* diukur dalam satuan *KB/s*. Nilai tersebut merupakan jumlah total *transfer* data yang sukses yang dapat dilakukan. Dari tabel diatas dapat dijelaskan sebagai berikut :

- Didapatkan nilai rata – rata *throughput* algoritma *round robin* dengan kategori *low* adalah 455,72 *KB/s*. Dengan kategori *medium* adalah 879,08 *KB/s*. Dengan kategori *high* adalah 1746,22 *KB/s*.
- Didapatkan nilai rata – rata *throughput* algoritma *least connection* dengan kategori *low* adalah 453,52 *KB/s*. Dengan kategori *medium* adalah 885,94 *KB/s*. Dengan kategori *high* adalah 1751,36 *KB/s*.

4.2 Hasil Pengujian *Response time*

Proses pengujian *response time* dapat dilihat nilai perbandingan *response time* antara algoritma *round robin* dan algoritma *least connection*. *Satuan yang digunakan untuk response time yaitu ms/req. Berikut hasil yang diperoleh :*

Tabel 2. Hasil pengujian *response time*

Algoritma	*Round robin*			*Least connection*		
Kategori	*Low*	Med	*High*	*Low*	Med	*High*
Rata - rata	2,24	2,42	9,12	2,30	2,42	9,72

Berdasarkan tabel diatas didapatkan hasil pengukuran *response time* untuk tiap alogritma pada nilai *request rate* yang memiliki satuan ms/req yang ditampilkan oleh *Httperf*. *Response time* yang diamati dilakukan pada banyak koneksi *low*,midle, dan *high* diukur dalam satuan *ms/req*. Nilai tersebut merupakan lamanya transfer per *request* data yang sukses yang dapat dilakukan. Dari tabel diatas dapat dijelaskan sebagai berikut :

- Didapatkan nilai rata – rata *response time* algoritma *round robin* dengan kategori *low* adalah 2,24 *ms/req*. Dengan kategori *medium* adalah 2,42 *ms/req*. Dengan kategori *high* adalah 9,12 *ms/req*.
- Didapatkan nilai rata – rata *response time* algoritma *least connection* dengan kategori *low* adalah 2,30 *ms/req*. Dengan kategori *medium* adalah 2,42 *ms/req*. Dengan kategori *high* adalah 9,72 *ms/req*.

4.3 Hasil Pengujian *CPU Usage*

Proses pengujian *CPU Usage* dapat dilihat nilai perbandingan *CPU Usage* antara algoritma *round robin* dan *least connection*..*Satuan yang digunakan untuk CPU Usage* adalah prosentase (%). Data didapatkan dari hasil record yang dikumpulkan dari masing-masing *server* menggunakan *tools* pencetak *CPU Usage* yang dijalankan pada tiap *server*. Berikut hasil yang diperoleh :

Tabel 3. Hasil pengujian *CPU Usage server* 1

Algoritma	*Round robin*			*Least connection*		
Kategori	*Low*	Med	*High*	*Low*	Med	*High*
Rata-rata	84,74	88,80	89,96	66,90	68,42	76,39

Dari tabel diatas dapat dijelaskan sebagai berikut:

- Didapatkan nilai rata – rata *CPU Usage* algoritma *round robin* dengan kategori *low* adalah 84,74 %. Dengan kategori *medium* adalah 88,80 %. Dengan kategori *high* adalah 89,96 %.
- Didapatkan nilai rata – rata *CPU Usage* algoritma *least connection* dengan kategori *low* adalah 66,90 %. Dengan kategori *medium* adalah 68,42 %. Dengan kategori *high*adalah 76,39 %.

Tabel 4. Hasil pengujian *CPU Usage server* 2

Algoritma	*Round robin*			*Least connection*		
Kategori	*Low*	Med	*High*	*Low*	Med	*High*
Rata-rata	80,97	82,12	83,50	70,15	73,35	78,31

Dari tabel diatas dapat dijelaskan sebagai berikut:

- Didapatkan nilai rata – rata *CPU Usage* algoritma *round robin* dengan kategori *low*

adalah 80,97 %. Dengan kategori *medium* adalah 82,12 %. Dengan kategori *high* adalaha 83,50 %.

- Didapatkan nilai rata – rata *CPU Usage* algoritma *least connection* dengan ketegori *low* adalah 70,15 %. Dengan kategori *medium* adalah 73,35 %. Dengan kategori *high* adalah 78,31 %.

Tabel 5. Hasil pengujian *CPU Usage server* 3

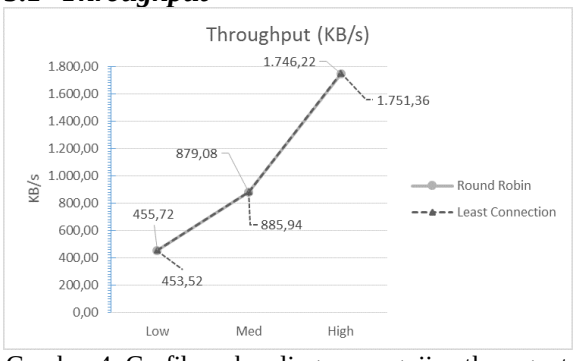
Algoritma	*Round robin*			*Least connection*		
Kategori	*Low*	Med	*High*	*Low*	Med	*High*
Rata-rata	89,73	91,28	91,34	88,06	90,27	90,35

Dari tabel diatas dapat dijelaskan sebagai berikut:

- Didapatkan nilai rata – rata *CPU Usage* algoritma *round robin* dengan kategori *low* adalah 89,73 %. Dengan kategori *medium* adalah 91,28 %. Dengan kategori *high* adalah 91,34 %.
- Didapatkan nilai rata – rata *CPU Usage* algoritma *least connection* dengan kategori *low* adalah 88,06 %. Dengan kategori *medium* adalah 90,27 %. Dengan kategori *high* adalah 90,35 %.

5. PEMBAHASAN

5.1 *Throughput*



Gambar 4. Grafik perbandingan pengujian througput

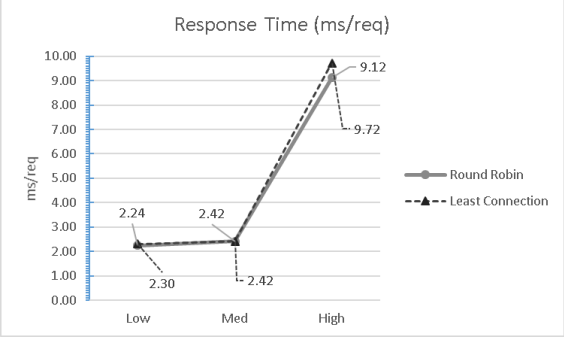
Grafik diatas merupakan grafik hasil pengujian *throughput* dari algoritma *round robin* dan *least connection*. Berdasarkan grafik diatas pengaruh perubahan kategori *rate* memberikan konsekuensi perubahan terhadap nilai *throughput*. Pada *rate* 40 req/s nilai pengujian *throughput* dengan algoritma *round robin* adalah 455,72 *KB/s* sedangkan pengujian dengan algoritma *least connection* adalah 453,52 *KB/s*. Pada *rate* 80 req/s nilai pengujian *throughput* dengan algoritma *round robin* adalah 879,08 *KB/s* sedangkan dengan algoritma *least connection* adalah 885,94 *KB/s*. Terakhir pada *rate* 160 req/s nilai pengujian *throughput* dengan algoritma *round robin* adalah 1746,22 *KB/s* sedangkan dengan algoritma *least connection* adalah 1751,36 *KB/s.*

Nilai *throughput* naik seiring dengan perubahan kategori *rate*, semakin tinggi kategori *rate* maka semakin tinggi nilai *througput* yang didapatkan semakin besar. Nilai througput algoritma *round robin* lebih besar dibandingkan algoritma *least connection* pada kategori *low*. Pada kategori *medium* dan *high* algoritma *round robin* memiliki nilai *throughput* lebih kecil dibandingkan algoritma *least connection*. Nilai rata – rata *throughput* kedua algortima mengalami kenaikan nilai dari kategori *low* hingga *high*.

Pada penelitian ini algoritma *round robin* lebih unggul hanya pada kategori *low*. Hal ini dikarenakan komputasi *round robin* lebih sederhana tanpa membandingkan jumlah koneksi tiap *server*. Sehingga untuk menangani koneksi yang kecil masih lebih unggul.

Sedangkan algoritma *least connection* lebih unggul pada kategori *medium* dan *high* karena komputasi algoritma *least connection* yang lebih kompleks dengan membandingkan jumlah koneksi tiap *server* lebih efektif.

5.2 *Response time*



Gambar 5. Grafik perbandingan pengujian *response time*

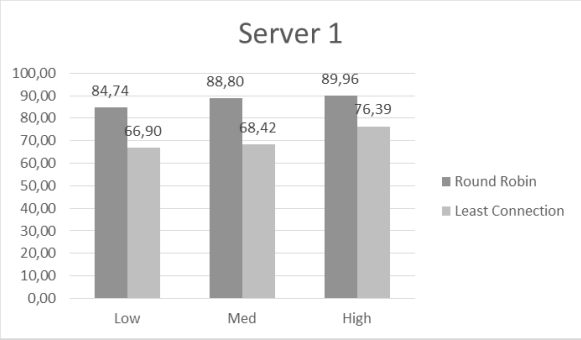
Grafik diatas merupakan grafik hasil pengujian *response time* dari algoritma *round robin* dan *least connection*. Berdasarkan grafik diatas pengaruh perubahan kategori *rate* memberikan konsekuensi perubahan terhadap nilai *response time*. Pada *rate* 40 req/s nilai pengujian *response time* dengan algoritma *round robin* adalah 2,24 *ms/req* sedangkan pengujian dengan algoritma *least connection* adalah *ms/req*. Pada *rate* 80 req/s nilai pengujian *response time* dengan algoritma *round robin* dan *least connection* sama, yaitu 2,42 *ms/req*. Terakhir pada *rate* 160 req/s nilai pengujian

response time dengan algoritma *round robin* adalah 9,12 *ms/req* sedangkan dengan algoritma *least connection* adalah 9,72 *ms/req*.

Nilai *response time* bertambah seiring dengan perubahan kategori *rate*, semakin tinggi kategori *rate* maka semakin tinggi nilai *response time* yang didapatkan. Nilai rata – rata *response time* algoritma *round robin* lebih kecil dibandingkan algoritma *least connection* pada kategori *low* dan *high*. Sedangkan pada kategori *medium* memiliki nilai rata – rata *response time* yang sama. Nilai rata – rata *response time* kedua algortima mengalami kenaikan nilai dari kategori *low* hingga *high*.

Pada penelitian ini algoritma *round robin* lebih unggul dibandingkan algoritma least dikarenakan memiliki nilai rata – rata respose time yang lebih kecil. Hal ini dikarenakan komputasi yang dilakukan *round robin* hanya melakukan perhitungan tanpa harus melakukan perbandingan yang membuat proses menjadi lebih lama.

5.3 *CPU Usage*



Gambar 6. Grafik perbandingan pengujian *CPU Usage Server* 1

Gambar 7. Grafik perbandingan pengujian *CPU Usage Server* 2

Gambar 8. Grafik perbandingan pengujian *CPU Usage Server* 3

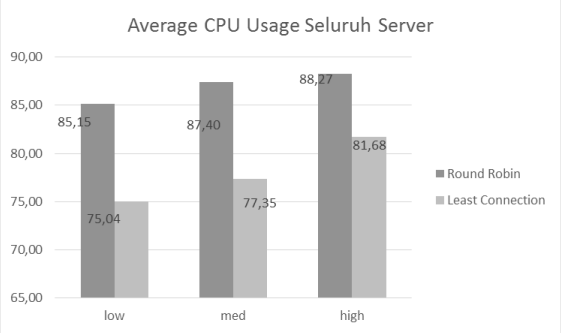
Grafik diatas merupakan grafik hasil pengujian *CPU Usage* tiap *server*. Pada grafik diatas menunjukan dari masing – masing *server* saat pengujian menggunakan algoritma *round robin* memiliki *CPU Usage* yang lebih tinggi dibandingkan dengan algoritma *least connection* pada setiap kategori *rate*. Dari kategori *request low* hingga *high*, *CPU Usage* selalu mengalami peningkatan nilai prosentase. Nilai *CPU Usage* bertambah seiring dengan perubahan kategori *rate*, semakin tinggi kategori *rate* maka semakin tinggi nilai *CPU Usage* yang didapatkan.

Pada penelitian ini dibutuhkan perhitungan standar deviasi dan average dari keseluruhan *server*. Standar deviasi adalah ukuran yang digunakan untuk mengukur jumlah variasi atau sebaran sejumlah nilai data. Semakin rendah standar deviasi, maka semakin mendekati rata-rata, sedangkan jika nilai standar deviasi semakin tinggi maka semakin lebar rentang variasi datanya. Sedangkan average adalah bilangan yang dapat dipakai sebagai wakil dari rentetan nilai rata-rata itu wujudnya hanya satu bilangan saja. Namun dengan satu bilangan itu akan dapat tercermin gambaran secara umum mengenai kumpulan atau deretan bahan keterangan yang berupa angka atau bilangan itu suatu bilangan yang mewakili sekumpulan data.

Tabel 6. Tabel Standar Deviasi dan Average *CPU Usage* Seluruh *Server*

Round robin					
Rate	*server* 1	*server* 2	*server* 3	STDDEV	AVERAGE
40	84,74	80,97	89,73	4,39	85,15
80	88,80	82,12	91,28	4,73	87,40
160	89,96	83,50	91,34	4,19	88,27
Least connection					
Rate	*server* 1	*server* 2	*server* 3	STDDEV	AVERAGE
40	66,90	70,15	88,06	11,39	75,04
80	68,42	73,35	90,27	11,46	77,35
160	76,39	78,31	90,35	7,57	81,68

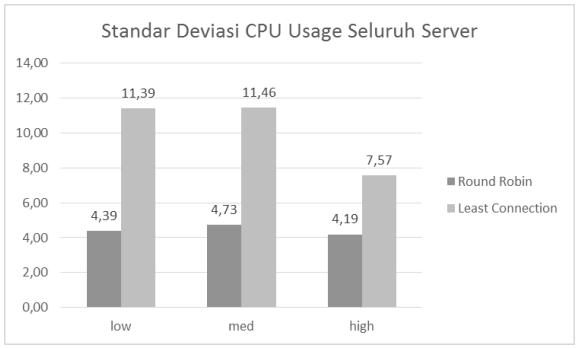
Nilai tabel diatas diambil dari grafik pengujian *CPU Usage* tiap *server*. Dilakukan perhitungan standar deviasi dan average *CPU Usage* seluruh *server* sesuai dengan kategori *rate*. Barikut adalah grafik average *CPU Usage* seluruh *server* dan grafik standar deviasi.



Gambar 9. Grafik perbandingan average *CPU Usage* seluruh *server*

Berdasarkan grafik diatas pengaruh perubahan kategori *rate* dengan jumlah koneksi yang semakin banyak akan memberikan konsekuensi perubahan terhadap nilai *CPU Usage.* Pada kategori *low*, yaitu dengan jumlah koneksi 1200 nilai pengujian *CPU Usage* dengan algoritma *round robin* adalah 81,15% sedangkan pengujian dengan algoritma *least connection* adalah 75,04%. Pada kategori *medium*, yaitu dengan jumlah koneksi 2400 nilai pengujian *CPU Usage* dengan algoritma *round robin* adalah 87,40% sedangkan pengujian dengan algoritma *least connection* adalah 77,35%. Terakhir pada *r* Pada kategori *high*, yaitu dengan jumlah koneksi 4800 nilai pengujian *CPU Usage* dengan algoritma *round robin* adalah 88,27% sedangkan dengan algoritma *least connection* 81,68%.

Nilai average seluruh *server* menunjukan *CPU Usage* saat pengujian dengan algoritma *round robin* memiliki nilai yang lebih tinggi dibandingkan dengan algoritma *least connection*. Pengujian *CPU Usage* kedua algoritma memiliki prosentase diatas 50%. Algoritma *round robin* melakukan komputasi yang berjalan tanpa memperhatikan perbandingan jumlah koneksi tiap *server* dan memiliki proses cepat. Sehingga *server* akan lebih cepat menerima *request* . Sedangkan algoritma *least connection* melakukan komputasi dengan melakukan perbandingan koneksi tiap *server*. Sehingga *server* memiliki jeda ketika menerima *request* .



Gambar 10. Grafik perbandingan standar deviasi *CPU Usage* seluruh *server*

Pada grafik diatas menunjukan standar deviasi dari seluruh *server*. Pengujian dengan algoritma *round robin* memiliki nilai yang lebih kecil dan stabil dibandingkan dengan algoritma *least connection* yang memiliki nilai besar dan cenderung tidak stabil. Semakin kecil nilai standar deviasi dan stabil maka pembagian beban *server* lebih seimbang.

CPU Usage dengan algoritma *round robin* lebih stabil pada setiap kategori *request* . Sedangkan algoritma *least connection* lebih ringan beban dari algoritma *round robin* meskipun nilainya mengalami kenaikan pada kategori *request* . Algoritma *round robin* memiliki *CPU Usage* tinggi akan tetapi lebih seimbang dikarenakan memiliki standar deviasi yang stabil dan rendah.

6. KESIMPULAN

1. Penerapan algoritma *round robin* dan *least connection* untuk *load balancing* pada *software defined network* dapat berjalan dengan baik dan membantu meringankan beban *server* dalam memberikan perlayan terhadap permintaan yang dilakukan oleh *user*. Karena kedua algoritma masih mampu melakukan pembagian beban dengan baik dengan nilai rata-rata *CPU Usage* diatas 50%.
2. Performa algoritma *round robin* pada pengujian *throughput* lebih unggul dibandingkan algoritma *least connection* pada koneksi yang kecil. Sedangkan algoritma *least connection* lebih unggul pada koneksi yang besar. Pada pengujian *response time* menujukan keunggulan algoritma *round robin* dibandingkan dengan algoritma *least connection*. *CPU Usage server* dengan algoritma *round robin* lebih stabil pada setiap kategori *request* . Sedangkan algoritma *least connection* lebih ringan beban dari algoritma *round robin* meskipun nilainya mengalami kenaikan pada kategori *rate*.

7. DAFTAR PUSTAKA

Ellrod, C., 2010. *Load balancing – Least connection*. [Online]
Available at: https://www.citrix.com/blogs/2010/09/02/load-balancing-least-connections/
[Accessed 18 Oktober 2016].

Ellrod, C., 2010. *Load balancing – Round robin*. [Online]
Available at: https://www.citrix.com/blogs/2010/09/03/load-balancing-round-robin/
[Accessed 18 Oktober 2016].

Feamster, N., Rexford, H. B. J., Shaikh, A. & Der Merwe, J. V., 2004. The Case for Separating Routing from Routers. IEEE.

Foundation, O. . N., 2012. opennetworking. [Online]
Available at: https://www.opennetworking.org/sdn-resources/sdn-definition [Accessed 19 Oketober 2016].

Kadir, A., 2003. Pengenalan Sistem Informasi. Yogyakarta: Andi.

Karantha, D., 2016. Analisis *Openflow* loadbalancing *Web server* Dengan Algoritma *Least connection* pada Software Defined Network. s.l.:s.n.

Kemkominfo, 2014. Kementrian Komunikasi dan Informatika Republik Indonesia. [Online]
Available at: https://kominfo.go.id/index.php/content/detail/3980/Kemkominfo%3A+Pengguna+Internet+di+Indonesia+Capai+82+Juta/0/berita_satker [Accessed 17 Oktober 2016].

Kurniawan, Y., 2013. Analisis Kinerja Algoritma Load Balancer dan Implementasi pada Layanan *Web*. s.l.:s.n.

Lara, A., Kolasani, A. & Ramamurthy, B., 2014. Network Innovation using *Openflow*: A Survey. CSE Journal Articles.

McKeown, N., Anderson, T. & Balakrishnan, H., 2008. *Openflow*: Enabling Innovation in Campus Networks.

McKeown, N., Anderson, T. & Balakrishnan, H., 2008. *Openflow*: Enabling Innovation in Campus Networks.

Midgley , J. T. J., 2001. The Linux HTTP Benchmarking HOWTO.

Mosberger, D. & Jin, T., 1998. *Httperf* A Tool for Measuring.

Mustafa, M. E. & Ibrahim, A. M., 2015. *Load balancing* Algorithms Round-Robin, Least-Conncetion And Least Load Efficency. International Journal of Computer and Information Technology, 04(02).

Naela, W., 2016. Implementasi *Load balancing* Di *Web Server* Menggunakan Algoritma *Round robin* pada Software Defined Networking. s.l.:s.n.

Organization, O., 2011. *Openflow* Organization. [Online]
Available at: http://archive.*Openflow*.org/wp/learnmore/
[Accessed 18 Oktober 2016].

Rodola, G., 2009. *psutil* documentation. [Online]
Available at: https://*Python*hosted.org/*psutil*/
[Accessed 20 Oktober 2016].

Sirajuddin, S., Affandi, A. & Setijadi, E., 2012. Rancang Bangun *Server* Learning *Manage*ment System Menggunakan Load Balancer dan Reverse Proxy. 01(01).