ANALISIS PERFORMANSI LOAD BALANCING DENGAN ALGORITMA WEIGHTED ROUND ROBIN PADA SOFTWARE DEFINED NETWORK (SDN)

LOAD BALANCING PERFORMANCE ANALYSIS BASED ON WEIGTHED ROUND ROBIN ALGORITHM IN SOFTWARE DEFINED NETWORK (SDN)

Faris Putra Perdana[1], Budhi Irawan[2], Roswan Latuconsina [3]
[1,2,3]Prodi S1 Sistem Komputer, Fakultas Teknik, Universitas Telkom
[1]ris_rangers@yahoo.co.id, [2] budhiirawan@telkomuniversity.ac.id, [3] roswan@telkomuniversity.ac.id

Abstrak

Dewasa ini pengguna internet semakin meningkat seiring dengan kebutuhan masyarakat. Peningkatan ini mengakibatkan tingginya pengguna dan membuat beban kerja sebuah *server* kian bertambah. Di sisi lain, masyarakat menginginkan kecepatan akses yang maksimal. *Software Defined Network* (SDN) adalah sebuah pendekatan baru untuk merancang, membangun dan mengelola jaringan komputer dengan memisahkan *control plane* dan *data plane*. Konsep utama pada SDN adalah sentralisasi jaringan, di mana semua pengaturan berada pada *control plane*. Protokol yang paling menonjol pada SDN yaitu *OpenFlow*. *OpenFlow* adalah protokol/standar komunikasi antarmuka yang berada antara *control* dan *forwarding layer*.

Pada penelitian ini digunakan suatu teknik *Load Balancing* untuk meningkatan performansi *server*. Teknik *Load Balancing* yang di implementasikan adalah berbasis pada algoritma *Weighted Round Robin* dengan pengaplikasian teknik *Load Balancing* ini diharapkan *request client* dapat terdistribusi secara merata ke setiap *server*.

Untuk kebutuhan ini dilakukan sebuah simulasi jaringan virtual berbasis SDN, menggunakan tool/aplikasi Mininet. Simulasi dilaksanakan dengan memvariasikan jumlah *client*, jumlah *server* dan bobot *server*. Dengan pengujian dengan parameter *Cpu Utilization* mendapatkan hasil semakin besar jumlah bobot pada *server* maka semakin besar pula *Cpu Utilization*. Kemudian pada pengujian *Response Time* mendapatkan hasil semakin besar besar jumlah bobot pada *server* maka nilai *Response Time* semakin besar.

Kata kunci: *SDN, OpenFlow, Load Balancing, Mininet, Weighted Round Robin*.

Abstract

Today's internet users are increasing along with the needs of the community. This increase resulted in high users and increased workload of a server. On the other hand, people want maximum access speed. Software Defined Network (SDN) is a new approach for designing, building and managing computer networks by separating control plane and data plane. The main concept of SDN is network centralization, where all settings are on the control plane. The most prominent protocol on SDN is OpenFlow. OpenFlow is a protocol / interface communication standard that resides between control and forwarding layer.

In this research used a Load Balancing technique to improve server performance. Load Balancing technique implemented is based on Weighted Round Robin algorithm with application of Load Balancing technique is expected client request can be distributed evenly to each server.

For this need to do a virtual network simulation based on SDN, using tool / application Mininet. The simulation is done by varying the number of clients, server number and server weight. By testing with *Cpu Utilization* parameter getting result of bigger amount of weight on server then more and more *Cpu Utilization*. Then on the test of Response Time get bigger result of big amount of weight on server then Response Time value bigger.

Keyword: *SDN, OpenFlow, Load Balancing, Mininet, Weighted Round Robin*.

1. Pendahuluan

Dengan berkembangnya teknologi pada zaman ini menyebabkan seluruh bidang yang ada di dunia menjadi lebih maju akibat dari teknologi, hal ini dirasakan juga pada perkembangan jaringan, terutama jaringan internet. Hal ini dapat dilihat dari semakin banyaknya pengguna yang terhubung ke jaringan, terutama jaringan internet. Di sisi lain karena semakin banyaknya pengguna yang menggunakan jaringan internet untuk masing – masing kepentingan, maka seringkali suatu jaringan internet mengalami *overload* atau *crash* disebabkan oleh banyaknya *request* yang dilakukan oleh pengguna. Cara mengatasinya yaitu dengan menambahkan *Server* atau menambahkan harddisk tambahan untuk *database*. Tetapi cara ini membutuhkan biaya yang cukup besar dan hanya sebagian kecil pengguna yang dapat menyediakannya.

Tugas akhir ini dibuat untuk membantu mengatasi permasalahan diatas dengan menggunakan paradigma baru dalam jaringan komputer yang disebut *Software Defined Network* (SDN). Dengan cara ini pengguna *Software Defined Network* (SDN) dapat membangun sendiri sebuah aplikasi maupun gabungan dari beberapa aplikasi yang akan dijalankan pada Platform controller. Platform controller menyediakan Application Programming Interfaces (APIs) sehingga memudahkan dalam mengimplementasikan fitur dan layanan dalam jaringan komputer. Pada *Software Defined Network* (SDN), *controller* terpusat mengkonfigurasi tabel *forwarding* (flow-table) Switch yang bertanggung jawab untuk meneruskan aliran paket komunikasi.

Salah satu contoh fitur yang biasanya telah tersedia pada sebuah perangkat Switch ataupun Router adalah load-balancing dengan metode yang sudah diterapkan oleh vendor penyedianya. Load-balancing merupakan suatu teknik untuk membagi beban trafik yang diterima kepada beberapa Server dengan tujuan agar Server tidak mengalami overload dan tetap menyediakan layanan dengan baik. Teknik *load-balancing* dapat diterapkan dengan menggunakan beberapa metode antrian, antara lain: *Round-Robin*, *Weighted Round-Robin*, *Least Connection*, *Weighted Least Connection*. Sebagai salah satu contoh adalah penerapan metode *Weighted Round Robin* pada teknik *load-balancing* dimana metode ini mengarahkan koneksi jaringan pada *Server* dengan jumlah bobot yang paling besar dan diatur dengan menggunakan penjadwalan. *Server* dengan kapasitas yang lebih besar akan mendapatkan nilai bobot yang lebih tinggi. *Server* dengan nilai bobot yang lebih tinggi akan menerima beban yang lebih besar dari koneksi-koneksi aktif pada satu waktu. Dengan demikian, diharapkan penerapan *network Load Balancing* dengan algoritma *Weighted Round Robin* ini dapat membagi beban *request* terhadap suatu jaringan sehingga meminimalkan kegagalan terhadap request yang dilakukan oleh pengguna.

2. Dasar Teori

2.1 *Software Defined Networking (SDN)*

sebuah paradigma arsitektur baru dalam bidang jaringan komputer, yang memiliki karakteristik dinamis, *manageable*, *cost-effective*, dan *adaptable*, sehingga sangat ideal untuk kebutuhan aplikasi saat ini yang bersifat dinamis dan *high-bandwidth*. Arsitektur ini memisahkan *antara network control* dan fungsi *forwarding*, sehingga *network control* tersebut menjadi *directly programmable* (dapat diprogram secara langsung), sedangkan infrastruktur yang mendasarinya dapat diabstraksikan untuk layer aplikasi dan *network services*.

2.2 *OpenFlow*

OpenFlow adalah protokol paling utama pada SDN. Posisinya berada di antara *Controller* dan *Forwarding* (data plane). *OpenFlow* memungkinkan pengaturan routing dan pengiriman paket ketika melalui sebuah switch. Dalam sebuah jaringan, setiap switch hanya berfungsi meneruskan paket yang melalui suatu port tanpa mampu membedakan tipe protokol data yang dikirimkan. *OpenFlow* memungkinkan untuk mengakses dan memanipulasi *Forwarding Plane* secara langsung dari perangkat-perangkat jaringan seperti switch dan router baik secara fisik maupun virtual. Nick McKeown et.al memberikan pembahasan yang rinci tentang latar belakang *OpenFlow* . Ide dasarnya cukup sederhana, yaitu mengeksploitasi fakta bahwa switch dan router Ethernet yang paling modern mengandung flow table yang dijalankan pada tingkat line-rate untuk mengimplementasikan firewall, NAT, QoS, dan juga untuk mengumpulkan data statistik. Flow table bisa berbeda implementasinya tergantung vendor dari perangkat keras tersebut. Akan tetapi terdapat beberapa *common set of function* (kumpulan fungsi atau primitive yang serupa) di antara berbagai switch dan router vendor-based tersebut. *OpenFlow* dalam hal ini menyediakan open protocol untuk memprogram flow-table pada berbagai switch dan router tersebut. Di lain pihak, vendor pun tidak dirugikan karena tidak perlu membuka mekanisme kerja internal dari produk switch atau router buatan masing-masing vendor. Untuk bisa menggunakan *OpenFlow* diperlukan *Controller* SDN yang mendukung jalannya protokol *OpenFlow*. *Controller* adalah salah satu controller SDN yang mendukung protokol *OpenFlow*.

2.3 *Load Balancing*

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 *overload* pada salah satu jalur koneksi..

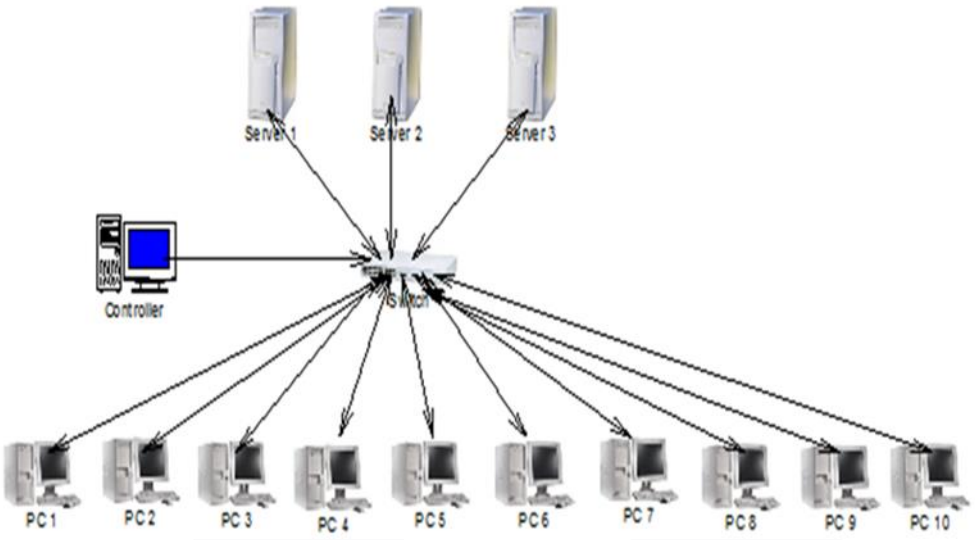
2.4 *Weighted Round Robin*

Pengembangan dari algoritma *Round Robin*, sedikit perbedaannya adalah algoritma ini bisa membagi beban lebih tinggi ke *server* atau cluster yang memiliki resource yang lebih besar atau dapat melakukan perhitungan perbedaan kemampuan proccessing dari masing - masing *server* anggota cluster. Administrator memasukan secara manual parameter beban yang akan ditangani oleh masing masing *server* anggota cluster, kemudian *scheduling sequence* secara otomatis dilakukan berdasarkan beban *server*. *Request* kemudian diarahkan ke *server* yang berbeda sesuai.

3. Pembahasan

3.1 Gambaran Umum Sistem

Mempelajari tentang teori-teori dan konsep mengenai *Weighted Round Robin* dan Teknik *Load Balancing* dan memodifikasinya dari artikel,jurnal dan refrensi dari internet. Dan mensimulasikan pada Mininet kemudian melihat hasilnya pada *controller* dan aplikasi *Wireshark*.



Gambar 3.1 Gambaran Topologi Jaringan

3.2 *Client*

Pada gambar 3.1, *Client*, merupakan pengguna/*client* yang menggunakan layanan yang terdapat pada masing-masing *server*.

3.3 *OpenFlow Switch*

OpenFlow Switch, perangkat ini bertujuan untuk meneruskan paket dari *controller*.

3.4 *Controller*

Controller, perangkat ini bertugas untuk mengatur OpenFlow Switch yang di berikan jaringan.

3.5 *Switch*

Perangkat ini menghubungkan antara *Client* dengan *OpenFlow Switch*.

3.6 *Server*

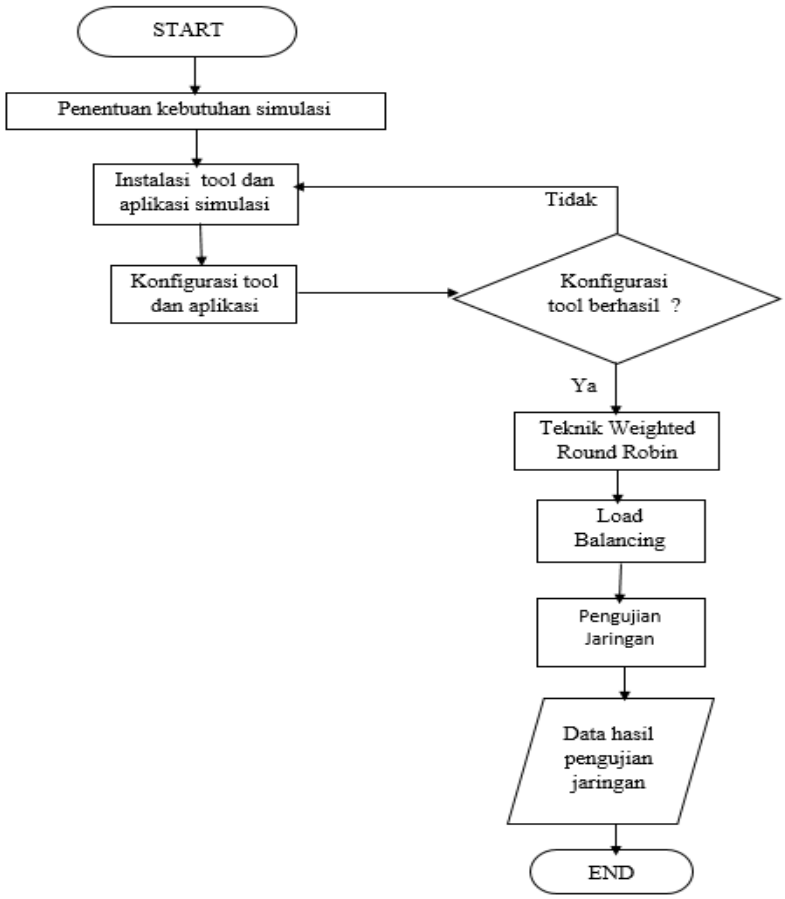
Server, bertugas untuk menyediakan layanan terhadap user yaitu sebuah web *server*. terdapat 3 buah *server* yang merupakan replica.

Dalam penelitian ini, algoritma Weighted Round Robin merupakan metode untuk membagi beban bedasarkan bobot *server* yang paling besar dahulu. membuat skenario untuk pengujian *Software Defined Network*. Di akhir proses, kita akan mendapatkan hasil dari pengujian parameter QOS dan *Cpu Utilization*.
Langkah - langkah pada pengujian *Software Defined Network* sebagai berikut :

1. Persiapkan terlebih dahulu kebutuhan simulasi pada *Software Defined Network*.
2. Instalasi *Tool* dan Aplikasi yang di butuhkan untuk melakukan sebuah simulasi pada perancagan *Software Defined Network*.
3. Kemudian konfigurasi tool dan aplikasi yang telah selesai di instalasi.
4. Jika konfigurasi tool berhasil maka lanjut ke tahap berikutnya, jika tidak kembali ke instalasi Tool dan Aplikasi yang di butuhkan untuk melakukan sebuah simulasi pada perancagan *Software Defined Network*.

5. Setelah berhasil melalui konfigurasi Tool maka perancangan jaringan tersebut memasuki bagian algoritma *Weighted Round Robin* denga memilih bobot *server* terbesar.
6. Kemudia perancanga jaringan yang tadi telah melewati tahap algoritma *Weighted Round Robin* maka aka memasuki taha pembagian paket datajaringan secara seimbang oleh teknik *Load Balancing*.
7. Setelah itu lakukan pengujian jaringan dengan parameter yang telah di tetapkan.
8. Kemudian terakhir catat hasil data pengujian dan berikan hasil analisis.

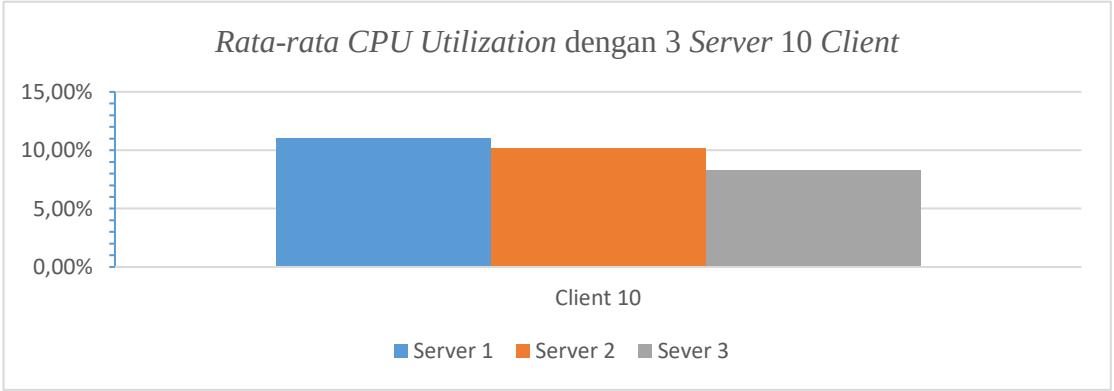
Flowchart alur perancangan seperti gambar 3. 3.



Gambar 3.3 Flowchart alur perancangan

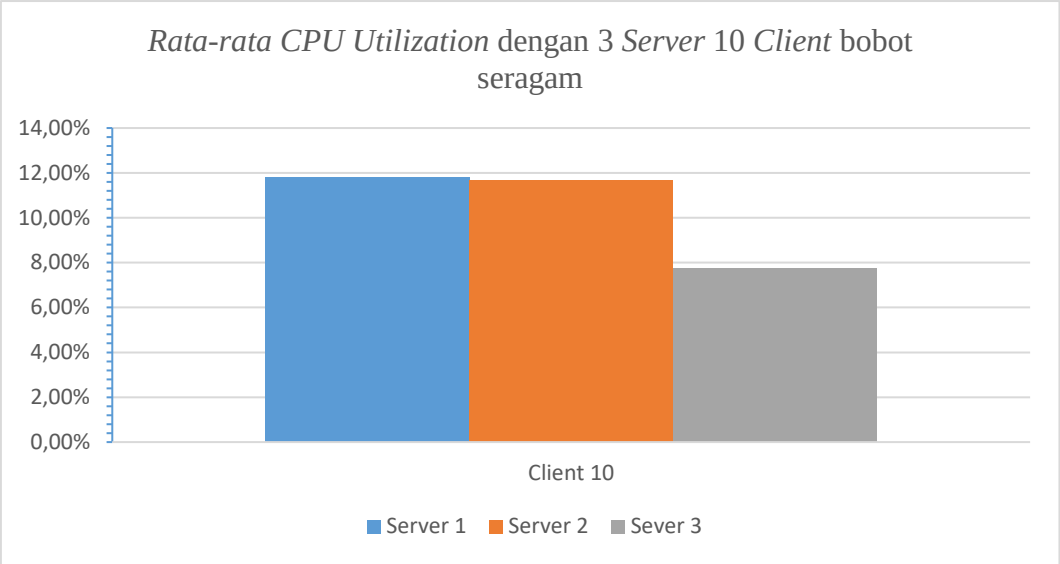
4. Analisis

Pada penelitian tugas akhir ini, yaitu, pengujian performansi akan mendapatkan beberapa hasil dengan menggunakan topologi satu switch dengan masing – masing *Client* mempunyai Bandiwth 100 MBps mempunya hasil analisis sebaga berikut :



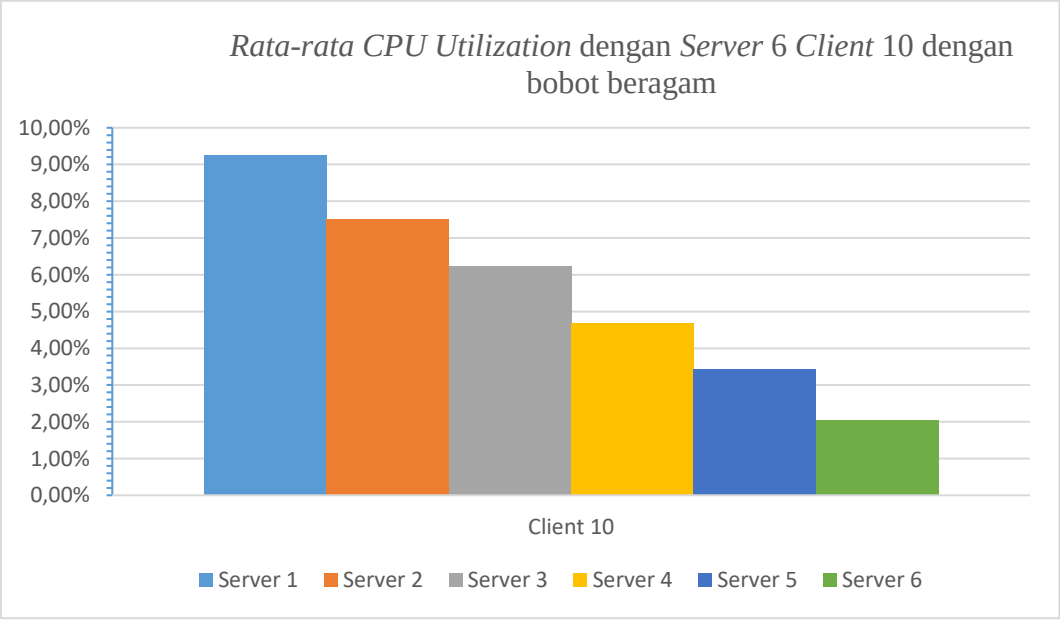
Gambar 4.1 Pengujian *CPU Utilization Server* (3) untuk *Client* (10) dengan bobot beragam

Analisis : dari gambar grafik 4.1 menunjukkan bahwa pengujian bedasarkan Cpu Utiliztion dari 3 *server* untuk 10 *Client* dengan utilitas Cpu pada *server* masuk dari bobot *server* yang telah di tentukkan sebelumnya yaitu bobot *server* 1 = 4 , bobot *server* 2 = 3 dan bobot *server* 3 = 2 menunjukkan bahwa kinerja *cpu utilization* sesuai yang di harapkan



Gambar 4.2 Pengujian *CPU Utilization Server* (6) untuk *Client* (10) dengan bobot seragam

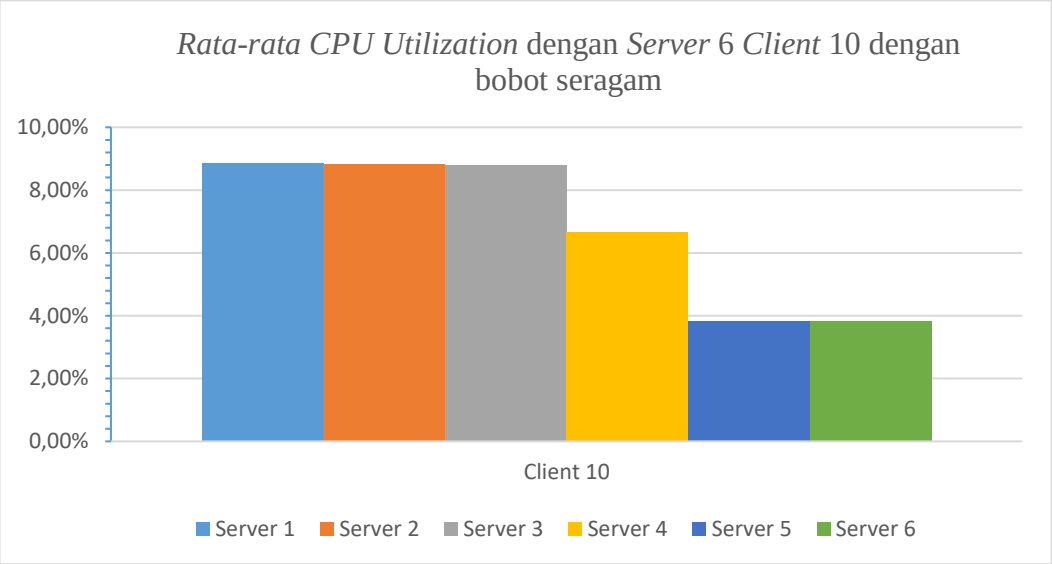
Analisis : dari gambar grafik 4.2 menunjukkan bahwa pengujian bedasarkan Cpu Utiliztion dari 3 *server* untuk 10 *Client* dengan besar utilitas penggunaan Cpu pada *server* masuk dari bobot *server* yang telah di tentukkan sebelumnya yaitu bobot *server* 1 = 3 , bobot *server* 2 = 3 , bobot *server* 3 = 2 menunjukkan bahwa kinerja *Cpu Utilization* sesuai yang di harapkan.



Gambar 4.3 Pengujian *CPU Utilization Server* (10) untuk *Client* (10) dengan bobot beragam

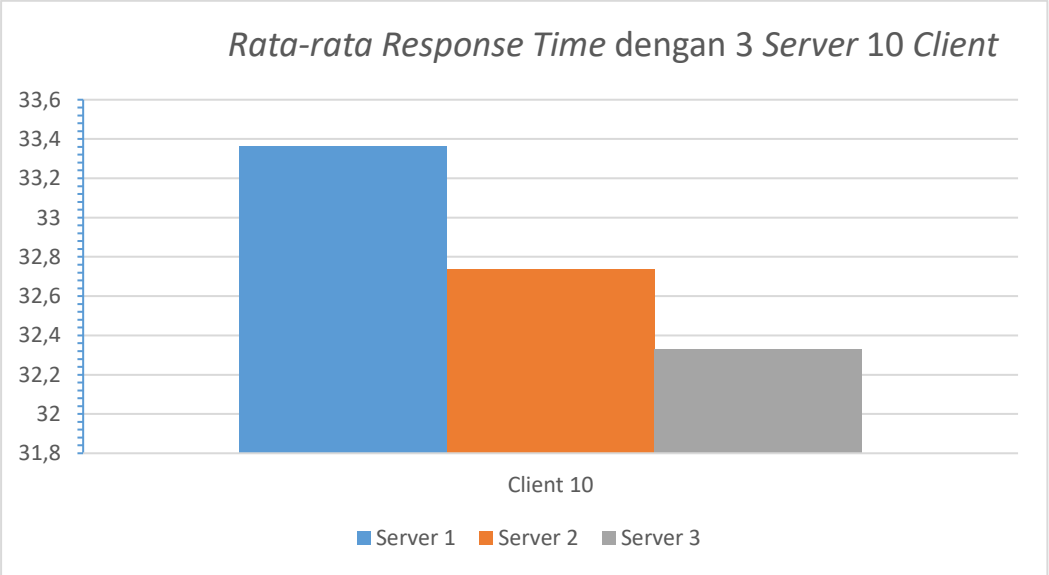
Analisis : dari gambar grafik 4.3 menunjukkan bahwa pengujian bedasarkan Cpu Utiliztion dari 10 *server* untuk 10 *Client* dengan besar utilitas penggunaan Cpu pada *server* masuk dari bobot *server* yang telah di tentukkan sebelumnya yaitu bobot *server* 1 = 10 , bobot *server* 2 = 9 , bobot *server* 3 = 8 , bobot *server* 4 = 7 , bobot *server* 5 = 6 , bobot *server* 6 = 5 , bobot *server* 7 = 4 , bobot *server* 8 = 3 , bobot *server* 9 = 2 dan bobot *server* 10 = 1 menunjukkan bahwa kinerja *Cpu Utilization* pada *Server* 4 , *server*

5 , *server* 6 , *server* 7 , *server* 8 hampir menujukkan kinerja daya *Cpu Utilization* yang sama karena pembagian bobot *server* yang masuk ke dalam daya *Cpu Utilization* itu pada saat di pembagian *Server* 10 dan *Client* 10 terbagi secara merata dan karena dari jumlah *Server* dan Jumlah *Client* terbagi secara merata.



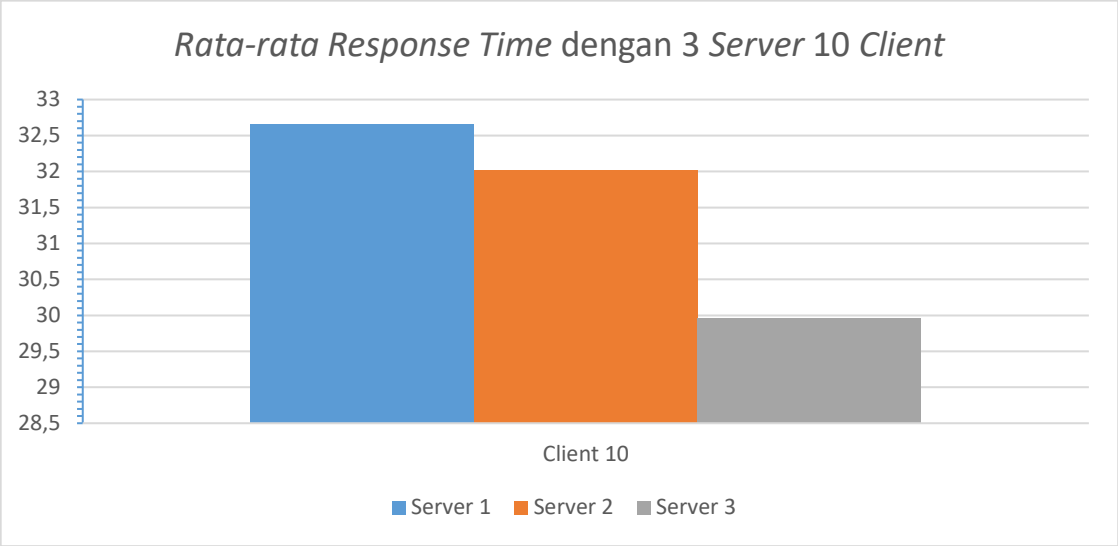
Gambar 4.4 Pengujian *CPU Utilization Server* (3) untuk *Client* (50) dengan bobot seragam

Analisis : dari gambar grafik 4.4 menunjukkan bahwa pengujian bedasarkan Cpu Utiliztion dari 6 *server* untuk 10 *Client* dengan besar utilitas penggunaan Cpu pada *server* masuk dari bobot *server* yang telah di tentukkan sebelumnya yaitu bobot *server* 1 = 4 , bobot *server* 2 = 4 , bobot *server* 3 = 3 , bobot *server* 4 = 2 dan bobot *server* 5 = 2menunjukkan bahwa kinerja *cpu utilization* sesuai yang di harapkan.



Gambar 4.5 Pengujian *Response Time Server* (3) untuk *Client* (10)

Analisis : dari gambar grafik 4.5 menunjukkan bahwa pengujian bedasarkan *Response Time* dari 3 server untuk 10 *Client* dengan masing – masing link pada *Client* mempunyai Bandwith 100 MBps dan memberikkan bobot *server* 1 = 3, *server* 2 = 3, *server* 3 = 2, bahwa waktu Response time dengan bobot *server* lebih besar memberitahukan *Response Time* lebih besar di bandingkan dengan bobot *Server* yang lebih kecil.



Gambar 4.6 Pengujian *Response Time Server* 30) untuk *Client* (10).

Analisis : dari gambar grafik 4.2.3 menunjukkan bahwa pengujian bedasarkan *Response Time* dari 3 *server* untuk 10 *Client* dengan masing – masing link pada *Client* mempunyai Bandwith 100 MBps dan memberikkan bobot *server* 1 = 3, *server* 2 = 3, *server* 3 = 1, Adalah waktu *Response time* dengan bobot *server* yang sama menunjukkan *Response Time* lebih besar dan hampir seimbang pada *server* 1 dan *server* 2 di bandingkan dengan bobot *Server* 3 yang tidak sama jumlah bobotnya menghasilkan nilai *Response Time* yang lebih kecil dan lebih baik.

5. Kesimpulan dan saran

5.1 Kesimpulan

Bahwa Algoritma *Weighted Round Robin* dapat di terapkan dan di simulasikan pada *Software Defined Network* dengan cara paket data masuk kepada bobot *server* dengan menggunakan *Controller* POX untuk pengujian parameter dan performansi, kemudian mendapatkan hasil kesimpulan sebagai berikut :

1. Pada skenario moderat dengan 50 *client* dan 6 *server* didapatkan rata-rata nilai *Cpu Utilization* Pada skenario moderat dengan 50 *client* dan 6 *server* dengan bobot beragam didapatkan rata-rata nilai *Cpu Utilization* 11,46% pada *server* dengan bobot 6, dan rata-rata *Cpu Utilization* 4,90% pada *server* dengan bobot 1. Dari hasil simulasi tersebut dapat ditarik suatu kesimpulan bahwa semakin besar jumlah bobot *server* maka semakin besar pula *Cpu Utilization*.
2. *Response Time* pada *server* dengan bobot seragam hanya mengalami sedikit perbedaan antar satu *server* dengan *server* yang lain. Sebaliknya *Response Time* pada *server* dengan bobot beragam mengalami perbedaan beragam antar satu *server* dengan *server* yang lain.
3. Pengaruh *Response Time* terhadap bobot *server* adalah makin besar bobot *server* maka makin besar pula nilai *Response Time* yang didapatkan. Jika bobot *server* kecil maka hasil dari nilai *Response Time* yang didapatkan makin kecil.
4. Algoritma *Weighted Round Robin* bekerja dengan cukup baik di dalam pengalokasian beban (*request client*) yang hampir merata pada setiap *server*.

5.2 Saran

Saran yang di harapkan guna membantu kedepannya analisis yang di berikan lebih baik dan akurat yaitu :

1. Mebandingkan antara algoritma *Weighted Round Robin* dengan algoritma lain terhadap parameter *Cpu Utilization* untuk memperlihatkan apakah algoritma yang lain berpengaruh terhadapa parameter *Cpu Utilization* dan memberikan analisis yang lebih baik.
2. Di usahaakan menerapkan topologi tree *pada Software Defined Network* untuk melihat perfomansi pada QOS.
3. Apabila melakukan menggunakkan Virtualisasi diharapkan mempunya komputer atau laptop dengan *processor* yang besar

DAFTAR PUSTAKA

[1] Archaya, J., & Patel, C. (t.thn.). A riview Load ABalancing Techniques in Cloud Computing.

[2] Gajjar, D., Kotak, H., Perigo, D. L., & Dkk. (2016). Round Robin Load Balancer Using Software Defined Networking (SDN). *Capstone Team Research Project*.

[3] Kaur, S., Kumar, K., Japinder, S., & Ghumman, N. (2015). Round-robin based load balancing in Software Defined Networking.

[4] N, S., & S, R. (2015). Load Balancing. *Dynamic Load Balancing using SOftware Defined Networks*, 1-4.

[5] Perigo, D., & Schnitzer, J. (2016). Round Robin Load Balancer using Software Defined Networking (SDN). *Capstone Team Research Project*.

[6] Qiang, Z., Zhen, K., Hijun, Z., & Leung, C. (2016). Delay Optimal Virtualized Radi Resource Scheduling in Software Defined Vehicular Networks via Stochastic Learning. *IEEE*.

[7] Rubinstein, A. (2013). Judgment and Decision Making. *Response time and decision making: An experimental study*, 8.

[8] Sevcik, P., & Bartllet, J. (2001). Understanding Web Performance. 6.

[9] Ummah , I., & Abdillah, D. (2016). Perancangan simulasi Jaringan Virtual Berbasis Software Defined Networking.

[10] Zhong, H., Lin, Q., Cui, J., & Dkk. (2015). An Efficient SDN Load Balancing Scheme Based on Variance Analysis for Massive Mobile Users. *Hindawi Publishing Corporation Mobile Information Systems*, 9.