Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 3, No. 9, September 2019, hlm. 8954-8963 http://j-ptiik.ub.ac.id

Analisis Perbandingan Penerapan *Stateless* Dan *Stateful* Pada Aplikasi *Server Load Balancing* Berbasis *Software-Defined Networking*

Fahmi Alfareza[1], Achmad Basuki[2], Kasyful Amron[2]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]fahmi_alfareza@student.ub.ac.id, [2]abazh@ub.ac.id, [3]kasyful@ub.ac.id

Abstrak

Software-defined networking atau biasa disebut SDN merupakan sebuah arsitektur jaringan komputer di mana aliran data dan kontrol komunikasi terpisah. Pemisahan tersebut membuat SDN dapat melakukan konfigurasi aplikasi jaringan lebih efisien dibandingkan dengan jaringan tradisional, salah satu contohnya adalah aplikasi *server load balancing*. *Server load balancing* adalah sebuah teknik untuk menyeimbangkan distribusi beban ke beberapa *server* yang menyediakan suatu layanan. Ada dua teknik yang dapat diimplementasikan sebagai *server load balancing* berbasis SDN, yaitu *stateless* dan *stateful*. Permasalahannya adalah, jika *state* pada SDN mempengaruhi kinerja pada aplikasi jaringan berbasis SDN. Lalu, teknik manakah yang mempunyai kinerja lebih baik untuk diimplementasikan sebagai *server load balancing* berbasis SDN yang dilihat pada sisi *client*, *SDN controller*, dan *server*. Pada sisi *client*, didapatkan bahwa kinerja *stateless* lebih baik daripada *stateful* dengan nilai *throughput* dari *stateless* 222,9% lebih besar pada *request rate* 250 *requests/second* daripada *stateful*. Pada sisi *SDN controller*, didapatkan bahwa kinerja *stateless* lebih baik daripada *stateful* dengan nilai *connection rate* dari *stateless* 222,9% lebih besar pada *request rate* 250 *requests/second* daripada *stateful*. Dan pada sisi *server*, didapatkan bahwa kinerja *stateful* lebih baik daripada *stateless* dengan distribusi *request* yang lebih merata. Pada *stateful*, *server 0 menerima* 6.667 *request*, *server 1 menerima* 6.667 *request*, dan *server 2 menerima* 6.666 *request*. Sedangkan pada *stateless*, *server 0 menerima* 10.000 *request*, *server 1 menerima* 5.000 *request*, dan *server 2 menerima* 5.000 *request*.

Kata kunci: *software-defined networking, stateless server load balancing, stateful server load balancing, openflow*

Abstract

Software-defined networking or commonly called SDN is a computer network architecture in which data flow and communication control are separate. This separation makes SDN able to configure network applications more efficiently than traditional networks, one example is the server load balancing application. Server load balancing is a technique for balancing load distribution to several servers that provide a service. There are two techniques that can be implemented as server load balancing, that is stateless and stateful. The problem is, if the state on SDN affects performance on SDN-based network applications. Then, which technique has better performance to be implemented as SDN-based server load balancing seen on the client side, SDN controller, and server. On the client side, it is found that the performance of stateless is better than stateful with the throughput value of stateless 222.9% greater at request rates of 250 requests/second than stateful. On the SDN controller side, it was found that the performance of the stateless is better than the stateful with the connection rate value of the stateless 222.9% greater at the request rate of 250 requests/second than the stateful. And on the server side, it was found that the performance of stateful was better than with a more even distribution of requests. In a stateful, server 0 receives 6,667 requests, server 1 receives 6,666 requests, and server 2 receives 6,666 requests. Whereas on stateless, server 0 receives 10,000 requests, server 1 receives 5,000 requests, and server 2 receives 5,000 requests.

Keywords: *software-defined networking, stateless server load balancing, stateful server load balancing, openflow*

1. PENDAHULUAN

Software-defined networking atau biasa disebut SDN merupakan arsitektur pada jaringan komputer yang melakukan pemisahan *control system* dan aliran data. Arsitektur SDN juga memisahkan *control plane* dan *data plane* pada jaringan komputer. Pada SDN, yang memberikan informasi tentang setiap *flow* yang masuk adalah *control plane*. Sedangkan, yang melakukan *forwarding packet* yang masuk dari *port* asal ke *port* tujuan yang didasari oleh informasi yang diberikan oleh *control plane* adalah *data plane* (Al-Najjar, 2016). Pada arsitektur SDN dapat diketahui dengan jelas kondisi dari suatu jaringan komputer dikarenakan SDN mempunyai konsep sentralisasi jaringan komputer. Pemisahan tersebut membuat SDN dapat melakukan konfigurasi aplikasi jaringan lebih efisien dibandingkan dengan jaringan tradisional, salah satu contohnya adalah aplikasi *server load balancing* (Zhong et al., 2017).

Server load balancing adalah sebuah teknik untuk menyeimbangkan distribusi beban ke beberapa server yang menyediakan suatu layanan (Gustin Anggraeni, Sukiswo & Ajub Ajulian Zahra, 2014). *Server load balancing* juga dapat didefinisikan sebagai, teknik yang digunakan untuk mendistribukan beban ke beberapa *server* dengan tujuan untuk memberikan distribusi beban yang merata pada setiap *server*. *Server load balancing* dapat mendistribusikan beban yang seimbang terhadap beberapa *server* dengan parameter yang dapat disesuaikan sesuai kebutuhan. Tujuan dari *server load balancing* sendiri adalah untuk memaksimalkan *throughput*, meminimalkan *response time* dan memberikan pembebanan yang seimbang pada *server* serta mengoptimalkan sumber daya yang ada pada jaringan (Setyawan, 2014).

SDN mempunyai dua mekanisme yang dapat dilakukan untuk mengimplementasikan aplikasi jaringan, yaitu dengan mekanisme *reaktif* dan *proaktif*. Pada penelitian yang dilakukan oleh Fernandez, dengan judul "*Comparing OpenFlow Controller Paradigms Scability: Reactive and Proactive*" membandingkan skalabilitas mekanisme reaktif dan proaktif pada SDN *OpenFlow controller*. Mekanisme rea*ctive* pada SDN mempunyai cara kerja di mana ketika sebuah *packet* datang pada SDN *switch*, maka *switch* akan melihat *flow table* pada SDN *switch* tersebut. Jika tidak ada *flow* yang sama pada *flow table*, maka *flow* tersebut akan diteruskan ke *SDN controller*. Kemudian, SDN *controller* akan memproses *flow* tersebut dan menentukan ke mana *flow* tersebut akan di-*forward*. Setelah itu, SDN *controller* akan menuliskan *flow* baru pada *flow table* SDN *switch* berdasarkan spesifikasi pada *flow* sebelumnya. Pada mekanisme rea*ctive* ini, juga dapat didefinisikan bahwa setiap *flow* akan melewati dan diproses pada *SDN controller*.

Mekanisme *proactive* pada SDN mempunyai cara kerja di mana *flow entries* telah didefinisikan sebelumnya pada *flow table* SDN *switch*. Dengan demikian, *SDN switch* mengetahui *flow* akan di-*forwad* ke mana ketika ada *flow* yang datang. SDN *controller* tidak terlibat dalam penentuan *flow rule* pada *flow table* SDN *switch*. Pada mekanisme pro*active* ini, juga dapat didefinisikan bahwa setiap *flow* akan melewati dan diproses pada *SDN switch* karena *flow* sudah didefinisikan sebelumnya. Pada penelitian tersebut, implementasi dilakukan pada *emulator mininet* dengan *POX*, *Floodlight*, dan *NOX-C++* sebagai SDN *controller*. Lalu, peneliti melakukan pengujian dengan menggunakan 100 *switch* dan 200 *switch* dengan SDN *controller* yang berbeda-beda. Lalu, dari setiap pengujian itu dilakukanlah berapakah *Flows/sec* (*performance*) yang didapatkan dari setiap pengujian untuk membandingkan mekanisme rea*ctive* dan *proctive*. Dari semua pengujian, hasilnya mekanisme pro*active* mempunyai kinerja dan skalabilitas yang lebih baik dari pada *reactive*.

Penelitian awal telah dilakukan oleh beberapa peneliti untuk mengembangkan *server load balancing* pada arsitektur *SDN*. Peneliti sebelumnya, yaitu S. Kaur et al., dalam jurnalnya yang berjudul "*Round-Robin Based Load Balancing in Software Defined Network*" mengembangkan aplikasi *server load balancing* dengan algoritme *round-robin* dan *random* pada arsitektur *SDN*. *Server load balancing* kemudian diimplementasikan pada *emulator mininet* dengan *POX controller* dengan teknik *stateful server load balancing*. Teknik *stateful server load balancing* yang diimplementasikan pada penelitian tersebut adalah menggunakan mekanisme *reactive* pada SDN *controller*. Hasilnya, didapatkan kedua *server load balancing* dapat bekerja pada arsitektur *SDN* serta algoritme *round-robin* mempunyai kinerja yang lebih baik daripada algoritme *random* (Kaur et al. 2015). Tetapi, pada penelitian ini

tidak dilakukan implementasi *server load balancing* menggunakan teknik *stateless* pada SDN. Teknik *stateless server load balancing* yang dimaksud adalah teknik yang menggunakan mekanisme *proactive*.

Pada penelitian yang dilakukan oleh M. Aslan et al., dengan judul "*On the Impact of Network State Collection on the Performance of SDN Application"* membandingkan kinerja antara *passive* dan *active network state collection* pada aplikasi jaringan berbasis SDN. *Passive network state collection* menggunakan mekanisme *proactive* pada SDN *controller*. Sedangkan, *active network state collection* menggunakan mekanisme *reactive* pada SDN *controller*. Peneliti menggunakan 2 *switch*, dengan setiap *switch* terhubung dengan 3 host, 2 sebagai client dan 1 sebagai server. Peneliti menjalankan aplikasi jaringan berbasis SDN tersebut dengan *single controller* dan *distributed controller* yang terhubung dengan setiap *switch*. Lalu, peneliti membandingkan kinerja dari penerapan *passive* dan *active network state collection* dengan menggunakan *single controller* dan *distributed controller*. Hasilnya, *passive network state collection* mempunyai kinerja lebih baik daripada *active network state collection* pada penerapannya menggunakan *single controller* dan *distributed controller*.

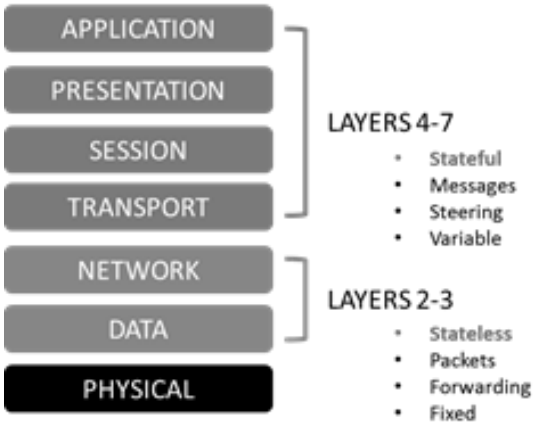
Ada dua teknik yang dapat diimplementasikan sebagai *server load balancing* berbasis SDN, yaitu menggunakan teknik *stateless* dan *stateful*. Pada *stateless server load balancing* berbasis SDN mempunyai cara kerja, di mana hanya *flow* baru pertama yang akan melewati *controller* untuk mencatat *flow* baru pada *flow table*. Sedangkan, *flow* yang sudah dikenali tidak akan melewati SDN *controller* karena sudah dikenali oleh *flow table* pada SDN *switch* (Lori MacVittie, 2014). Jika tidak ada *flow* yang sama datang sampai *flow* tersebut expired, maka jika ada *flow* yang sama tersebut akan dituliskan kembali ke *flow table* melalui SDN *controller*. *Stateless server load balancing* berbasis SDN menggunakan mekanisme *proactive* pada SDN *controller*. Sedangkan, pada *stateful server load balancing* berbasis SDN mempunyai cara kerja, di mana setiap *flow* akan melewati SDN *controller*, karena setiap *flow* akan dicatat setiap *state*-nya ke *flow table* pada SDN *switch* (Lori MacVittie, 2014). *Stateful server load balancing* berbasis SDN menggunakan mekanisme *reactive* pada SDN *controller*. Permasalahannya adalah, jika *state* pada SDN mempengaruhi kinerja pada aplikasi jaringan berbasis SDN yang dibuktikan dengan penelitian-penelitian sebelumnya. Lalu, teknik manakah yang mempunyai kinerja lebih baik untuk diimplementasikan sebagai *server load balancing* berbasis SDN yang dilihat pada sisi *client*, *SDN controller*, dan *server*.

Dari permasalahan di atas, untuk mengatasi masalah antara *stateless* dan *stateful* yang akan diimplementasikan sebagai *server load balancing* berbasis SDN, maka peneliti menganalisis perbandingan penerapan *stateless* dan *stateful server load balancing* berbasis SDN. Diharapkan, dari penelitian ini didapatkan kesimpulan di mana analisis perbandingan dalam penelitian ini, dapat dimanfaatkan sebagai acuan dalam melakukan pengembangan penelitian kedepannya tentang teknik *stateless* atau *stateful* yang akan digunakan dalam penerapan *server load balancing* berbasis SDN.

2. *STATELESS VS STATEFUL SERVER LOAD BALANCING* BERBASIS SDN

Pada SDN, ada dua teknik yang dapat diimplementasikan sebagai *server load balancing*, yaitu teknik *stateless* dan *stateful*. *Stateless* berbasis SDN tidak banyak melibatkan *SDN controller* untuk penentuan jalur pada *flow table*. Dikarenakan, *stateless* berbasis *SDN* menggunakan mekanisme *proactive* pada SDN *controller*. *Stateless* mempunyai batasan, yaitu hanya dapat mengidentifikasi *packet* yang datang pada layer 2-3 OSI layer untuk menentukan jalur yang dilewati oleh *packet* tersebut. Dikarenakan SDN *switch* hanya dapat menggali informasi *packet* hanya pada layer 2-3 OSI layer. Informasi *state* yang ada pada *packet* yang datang tidak disimpan, tetapi bukan berarti mempunyai *state*. *Packet* diteruskan keluar *port* yang sesuai berdasarkan apa yang ada di FIB (*Forwarding Information Base*) atau pada SDN disebut sebagai *flow table*.

Pada *stateful* berbasis SDN selalu meibatkan SDN *controller* pada penentuan jalur pada *flow table*. Dikarenakan, *stateful berbasis SDN* menggunakan mekanisme *proactive* pada SDN *controller*. *Stateful* berbasis SDN, bekerja mulai dari layer 4-7 OSI layer untuk menentukan jalur pada *flow table*. Setiap informasi yang datang, disimpan sebagai *state* pada *flow table*. Semakin tinggi *layer* dari *packet* yang datang, maka akan semakin banyak melibatkan SDN *controller* dalam mengidentifikasi *packet* yang datang. Gambar 2.1 menunjukkan perbedaan level partisipasi OSI layer pada *stateless vs stateful*.



Gambar 1. *Stateful vs Stateless* pada SDN

Pada konteks *server load balancing* berbasis SDN, *stateless server load balancing* berbasis SDN mempunyai cara kerja, di mana hanya *flow* baru pertama yang akan melewati *controller* untuk mencatat *flow* baru pada *flow table*. Sedangkan, *flow* yang sudah dikenali tidak akan melewati SDN *controller* karena sudah dikenali oleh *flow table* pada SDN *switch* (Lori MacVittie, 2014). Jika tidak ada *flow* yang sama datang sampai *flow* tersebut expired, maka jika ada *flow* yang sama tersebut akan dituliskan kembali ke *flow table* melalui SDN *controller*. *Stateless server load balancing* pada SDN menggunakan mekanisme *proactive* pada SDN *controller*.

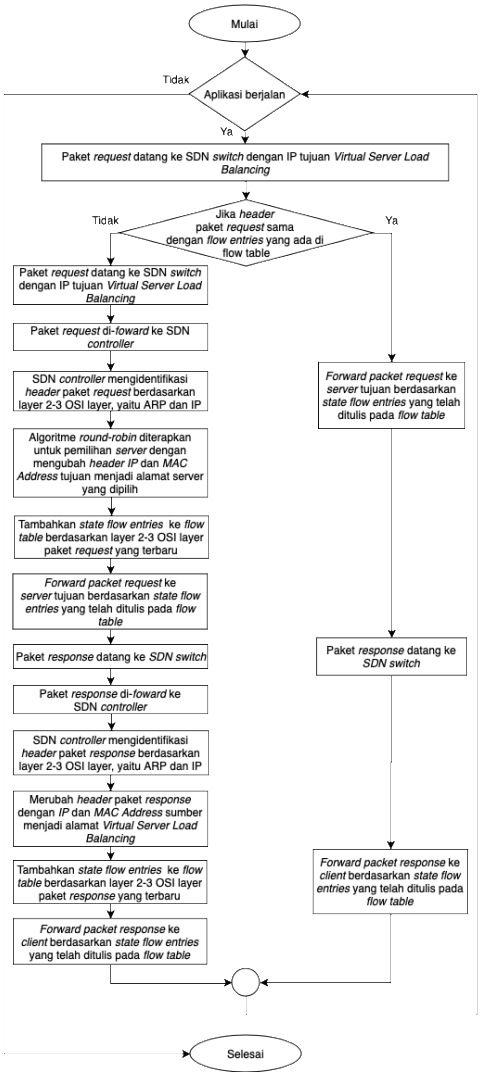
Pada *stateful server load balancing* pada SDN mempunyai cara kerja, di mana setiap *flow* akan melewati SDN *controller*, karena setiap *flow* akan dicatat setiap *state*-nya ke *flow table* pada SDN *switch* (Lori MacVittie, 2014). *Stateful server load balancing* pada SDN menggunakan mekanisme reactive pada SDN *controller*. Permasalahannya adalah, jika *state* pada SDN mempengaruhi kinerja pada aplikasi jaringan berbasis SDN yang dibuktikan dengan penelitian-penelitian sebelumnya. Lalu, teknik manakah yang mempunyai kinerja lebih baik untuk diimplementasikan sebagai *server load balancing* berbasis SDN yang dilihat pada sisi *client*, *SDN controller*, dan *server*.

3. METODE PENELITIAN

Pada penelitian ini ada dua *server load balancing* yang akan diuji, yaitu dengan menggunakan teknik *stateless* dan *stateful* dengan menggunakan algoritme *round-robin*. Berikut adalah perancangan dari *stateless* dan *stateful server load balancing*.

3.1. *Stateless Server Load Balancing*

Pada *stateless server load balancing* penanganannya, *flow* baru yang tidak dikenali akan melewati *controller* terlebih dahulu lalu akan dituliskan ke *flow table* dengan *expired time* tertentu. Lalu, setiap *flow* yang dikenali akan mengikuti aturan pada *flow table*. Setelah itu, setiap *flow* baru yang tidak dikenali akan melakukan pergantian akses ke *server* secara bergantian menggunakan algoritme *round-robin*. Gambar 2 menjelaskan tentang *flowchart* dari *stateless server load balancing*.



Gambar 2. *Flowchart stateless server load balancing*

Pada Gambar 2, menggambarkan *flowchart* dari *stateless server load balancing* dengan menggunakan algoritme *round-robin*. Yang pertama adalah, apakah aplikasi berjalan yang digunakan untuk memastikan bahwa aplikasi *stateless server load balancing* menggunakan algoritme *round-robin* terus berjalan sampai

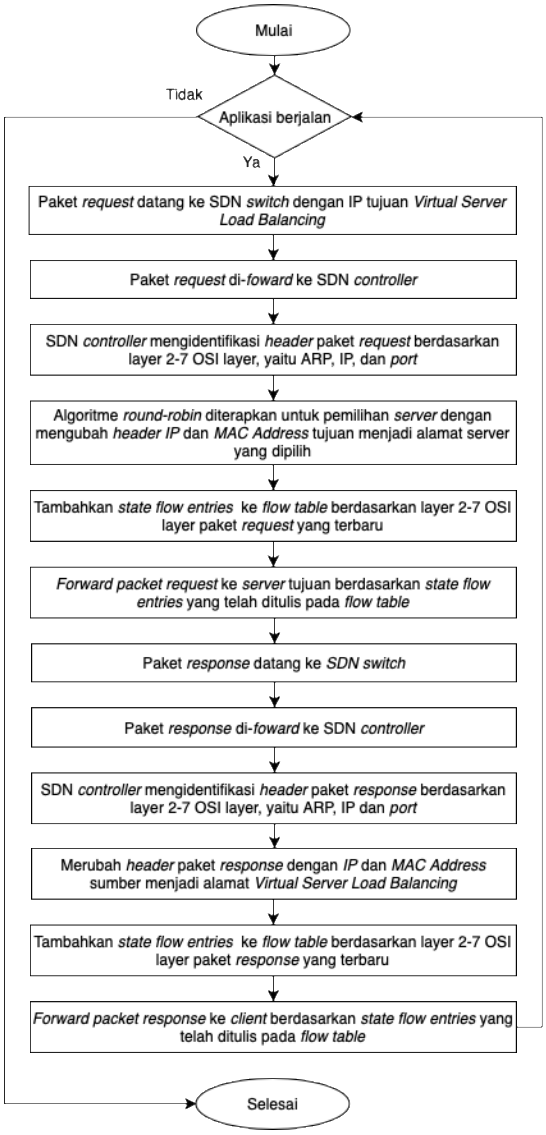
aplikasi tersebut dihentikan. Pada *stateless server load balancing*, paket *request* dari *client* pertama kali datang ke SDN *switch* dengan IP tujuan, IP dari *virtual server load balancing* yang telah didefinisikan oleh aplikasi. Jika pada paket *header* tersebut tidak sama dengan *flow entries* yang ada pada *flow table*, maka paket akan dikirimkan ke SDN *controller*. Lalu, SDN *controller* akan mengidentifikasi *header packet* tersebut berdasarkan layer 2-3 OSI layer, yaitu ARP dan IP. Kemudian, algoritme *round-robin* akan diterapkan untuk pemilihan server dengan cara mengubah *header* IP dan *MAC Address* tujuan yang sebelumnya, yaitu alamat dari *virtual server load balancing* menjadi alamat *server* yang dipilih. Lalu, SDN *controller* akan melakukan komunikasi dengan SDN *switch* untuk menambahkan *state flow entries* ke *flow table* berdasarkan layer 2-3 OSI layer paket tersebut yang telah dirubah *header*-nya. Setelah itu, SDN *controller* akan memerintahkan SDN *switch* untuk mem-*forward* paket *request* tersebut ke *server* tujuan yang telah dipilih.

Paket *response* akan datang ke SDN *switch*. Lalu, paket tersebut akan dikirimkan ke SDN *controller*. SDN *controller* mengidentifikasi *header* paket *response* berdasarkan layer 2-3 OSI layer, yaitu ARP dan IP. Kemudian, SDN *controller* akan merubah header paket *response* dengan IP dan *MAC Address* sumber dengan alamat *virtual server load balancing*. Lalu, SDN *controller* akan melakukan komunikasi dengan SDN *switch* untuk menambahkan *state flow entries* ke *flow table* berdasarkan layer 2-3 OSI layer paket tersebut yang telah dirubah *header*-nya. Setelah itu, SDN *controller* akan memerintahkan SDN *switch* untuk mem-*forward* paket *response* tersebut ke *client* yang melakukan *request* tersebut. Kondisi ini berlaku jika paket tersebut adalah paket baru dengan *header IP client* yang belum mempunyai *state flow entries* pada *flow table*.

Jika ada paket *request* dari *client* dengan spesifikasi *header IP client* sudah mempunyai *state flow entries* pada *flow table* telah ditulis, maka paket akan di-*forward* ke *server* tujuan yang telah dipilih sebelumnya pada *state* tersebut. Kemudian, paket *response* akan datang ke *SDN switch*. Setelah itu, paket *response* tersebut akan di-*forward* ke *client* yang melakukan *request* tersebut.

3.2. *Stateful Server Load Balancing*

Pada *stateful server load balancing* penanganannya, setiap *flow* akan melewati SDN *controller* karena setiap *flow* akan dicatatat setiap *state*-nya ke *flow table* pada SDN *switch*. Jadi, setiap *flow* akan diproses melalui SDN *controller*. Lalu, setiap *flow* yang masuk akan melakukan pergantian akses ke *server* secara bergantian menggunakan algoritme *round-robin*. Gambar 3 menjelaskan tentang *flowchart* dari *stateful server load balancing*.



Gambar 3. *Flowchart stateful server load balancing*

Pada Gambar 3, menggambarkan *flowchart* dari *stateful server load balancing* dengan menggunakan algoritme *round-robin*. Yang pertama adalah, apakah aplikasi berjalan yang digunakan untuk memastikan bahwa aplikasi *stateful server load balancing* menggunakan algoritme *round-robin* terus berjalan sampai

aplikasi tersebut dihentikan. Pada *stateful server load balancing*, paket *request* dari *client* pertama kali datang ke SDN *switch* dengan IP tujuan, IP dari *virtual server load balancing* yang telah didefinisikan oleh aplikasi. Kemudian, paket akan dikirimkan ke SDN *controller*. Lalu, SDN *controller* akan mengidentifikasi *header packet* tersebut berdasarkan layer 2-7 OSI layer, yaitu ARP, IP, dan *port*. Kemudian, algoritme *round-robin* akan diterapkan untuk pemilihan server dengan cara mengubah *header* IP dan *MAC Address* tujuan yang sebelumnya, yaitu alamat dari *virtual server load balancing* menjadi alamat *server* yang dipilih. Lalu, SDN *controller* akan melakukan komunikasi dengan SDN *switch* untuk menambahkan *state flow entries* ke *flow table* berdasarkan layer 2-7 OSI layer paket tersebut yang telah dirubah *header*-nya. Setelah itu, SDN *controller* akan memerintahkan SDN *switch* untuk mem-*forward* paket *request* tersebut ke *server* tujuan yang telah dipilih.

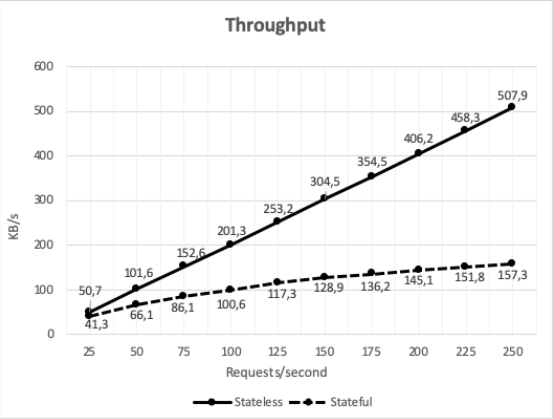
Paket *response* akan datang ke SDN *switch*. Lalu, paket tersebut akan dikirimkan ke SDN *controller*. SDN *controller* mengidentifikasi *header* paket *response* berdasarkan layer 2-7 OSI layer, yaitu ARP, IP, dan *port*. Kemudian, SDN *controller* akan merubah header paket *response* dengan IP dan *MAC Address* sumber dengan alamat *virtual server load balancing*. Lalu, SDN *controller* akan melakukan komunikasi dengan SDN *switch* untuk menambahkan *state flow entries* ke *flow table* berdasarkan layer 2-7 OSI layer paket tersebut yang telah dirubah *header*-nya. Setelah itu, SDN *controller* akan memerintahkan SDN *switch* untuk mem-*forward* paket *response* tersebut ke *client* yang melakukan *request* tersebut.

4. HASIL DAN PEMBAHASAN

Pada bagian ini, dilakukan dua pembahasan, yaitu analisis perbandingan kinerja dari penerapan *stateless* dan *stateful server load balancing* yang dilihat dari sisi *client*, *SDN controller*, dan *server*.

4.1. *Client*

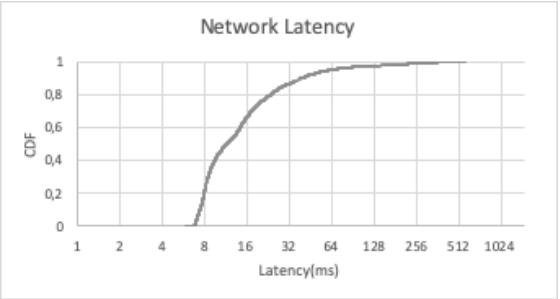
Pada analisis perbandingan kinerja dengan parameter *throughput*, dilakukan pengiriman *request* dengan *rate* 10, 20, 30, 40, 50, 60, 70, 80, 90 dan 100 *requests per second* dengan jumlah setiap *client* mengirimkan 100 *request* pada setiap pengujian. Gambar 4 menunjukkan perbandingan *throughput* dari *stateless* dan *stateful server load balancing*.



Gambar 4. Perbandingan *throughput*

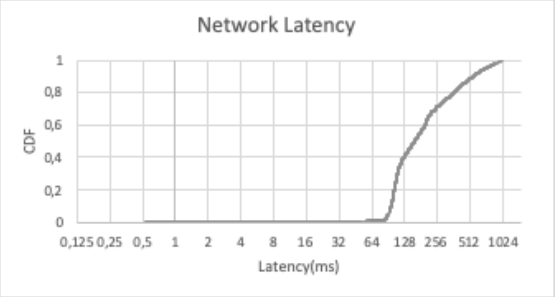
Pada Gambar 4 didapatkan bahwa *stateless server load balancing* mempunyai *throughput* yang lebih baik daripada *stateful server load balancing*. Pada *stateless* dan *stateful server load balancing*, *throughput* terus meningkat semakin banyaknya *request* yang dilakukan oleh *client*. Perbandingan pada 25 *requests/second* menunjukkan bahwa perbandingan antara *stateless* dan *stateful* tidak terlalu jauh, dengan *stateless server load balancing* mempunyai *throughput* 22,8% lebih tinggi daripada *stateful server load balancing*. Sedangkan, pada 250 *requests/second* menunjukkan bahwa perbandingan antara *stateless* dan *stateful* mulai jauh, dengan *stateless server load balancing* mempunyai *throughput* 222,9% lebih tinggi daripada *stateful server load balancing*. Pada analisis ini, dapat diperoleh bahwa *stateless server load balancing* mempunyai kinerja yang lebih baik daripada *stateful server load balancing*. Itu membuktikan bahwa *stateless server load balancing* mempunyai kinerja yang lebih baik daripada *stateful server load balancing*.

Hasil dari kinerja tersebut juga dipengaruhi oleh *network latency* yang dihasilkan oleh *stateless server load balancing* yang ditunjukkan pada Gambar 5.



Gambar 5. *Network latency* pada *stateless server load balancing*

Pada Gambar 5 dapat dilihat hasil nilai *network latency* pada *stateless server load balancing*. Grafik tersebut menunjukkan bahwa sekitar 94% dari jumlah paket yang dikirimkan memiliki nilai *network latency* kurang dari 64 *milisecond*. Dan rata-rata dari *network latency* dari *stateless server load balancing* adalah 23,35654 *milisecond*. Sedangkan, *network latency* pada *stateless server load balancing* ditunjukkan pada Gambar 6.



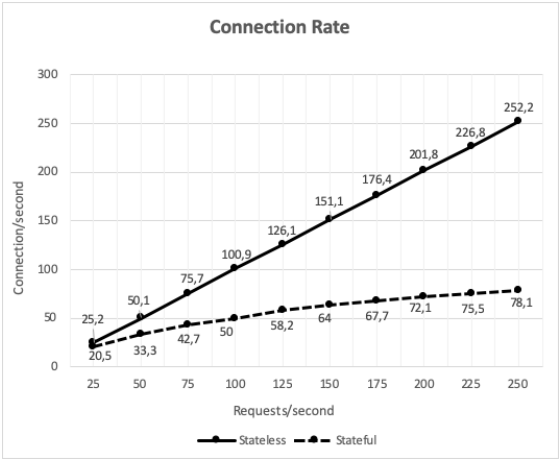
Gambar 6. *Network latency* pada *stateful server load balancing*

Pada Gambar 6 dapat dilihat hasil nilai *network latency* pada *stateful server load balancing*. Grafik tersebut menunjukkan bahwa sekitar 99% dari jumlah paket yang dikirimkan memiliki nilai *network latency* lebih dari 64 *milisecond*. Dan rata-rata dari *network latency* dari *stateful server load balancing* adalah 242,5678 *milisecond*. Ini membuktikan bahwa *network latency* pada *stateful server load balancing* lebih tinggi dibandingkan dengan *stateless server load balancing*. Pada analisis ini membuktikan bahwa *network latency* berpengaruh terhadap *throughput*. Semakin besar *network latency* maka akan semakin kecil *throughput*.

Dari hasil dan pembahasan ini, membuktikan bahwa *stateful server load balancing* berbasis SDN cocok digunakan pada jaringan yang tidak memprioritaskan kebutuhan kinerja pada sisi *client*. Sedangkan, *stateless server load balancing* berbasis SDN cocok digunakan pada jaringan yang yang memprioritaskan kebutuhan kinerja pada sisi *client*.

4.2. *SDN Controller*

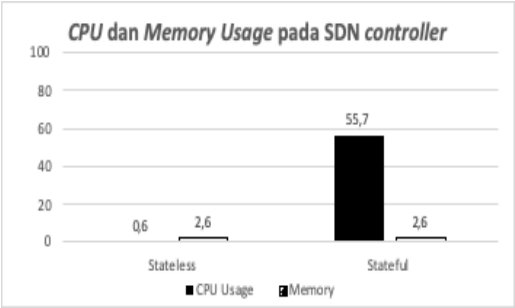
Gambar 7 menunjukkan perbandingan *connection rate* dari *stateless* dan *stateful server load balancing*.



Gambar 7. Perbandingan *connection rate*

Pada Gambar 7 didapatkan bahwa *stateless server load balancing* mempunyai *connection rate* yang lebih baik daripada *stateful server load balancing*. Pada *stateless* dan *stateful server load balancing*, *connection rate* terus meningkat semakin banyaknya *request* yang dilakukan oleh *client*. Perbandingan pada 25 *requests/second* menunjukkan bahwa perbandingan antara *stateless* dan *stateful* tidak terlalu jauh, dengan *stateless server load balancing* mempunyai *connection rate* 22,9% lebih tinggi daripada *stateful server load balancing*. Sedangkan, pada 250 *requests/second* menunjukkan bahwa perbandingan antara *stateless* dan *stateful* mulai jauh, dengan *stateless server load balancing* mempunyai *connection rate* 222,9% lebih tinggi daripada *stateful server load balancing*. Pada analisis ini, dapat diperoleh bahwa *stateless server load balancing* mempunyai kinerja yang lebih baik daripada *stateful server load balancing*. Itu membuktikan bahwa *stateless server load balancing* mempunyai kinerja yang lebih baik daripada *stateful server load balancing*.

Hasil dari kinerja tersebut juga dipengaruhi oleh *CPU* dan *memory usage* pada SDN *controller* yang ditunjukkan pada Gambar 8.

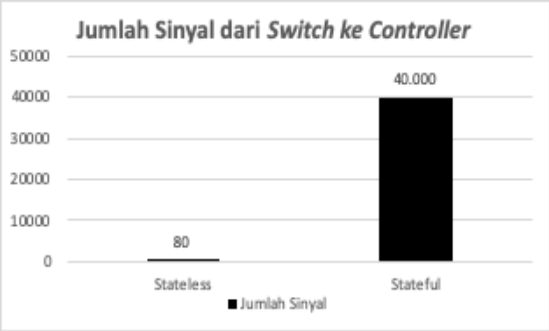


Gambar 8. Perbandingan *CPU* dan *memory usage* pada *SDN controller*

Pada Gambar 8 membuktikan bahwa *CPU* dan *memory usage* dari *stateless server load balancing* mempunyai nilai yang lebih rendah daripada *stateful server load balancing*. Perbandingan menunjukkan bahwa perbedaan sangat tinggi pada *CPU usage* antara *stateless* dan *stateful server load balancing* dengan *stateless server load balancing* mempunyai nilai 0,6% penggunaan *CPU* pada SDN *controller*. Sedangkan, *stateful server load balancing* mempunyai nilai 55,7% penggunaan *CPU* pada SDN *controller*.

Nilai pada *memory usage* pada *stateless* dan *stateful server load balancing* mempunyai nilai yang sama. Perbandingan menunjukkan bahwa tidak ada perbedaan pada *memory usage* antara *stateless* dan *stateful server load balancing* dengan *stateless* dan *stateful server load balancing* mempunyai nilai yang sama, yaitu 2,6% penggunaan *memory* pada SDN *controller*. Analisis ini juga membuktikan bahwa ada pengaruh *CPU usage* terhadap *connection* yang dihasilkan. Semakin terbebani SDN *controller*, maka akan semakin kecil *connection rate* yang dihasilkan.

Dan pada analisis kinerja ini, penulis juga melakukan pengamatan terhadap jumlah sinyal *switch* ke *controller*. Pada hasilnya bahwa jumlah sinyal dari *switch* ke *controller* mempengaruhi kinerja dari *stateless* dan *stateful server load balancing* yang ditunjukkan pada Gambar 9.



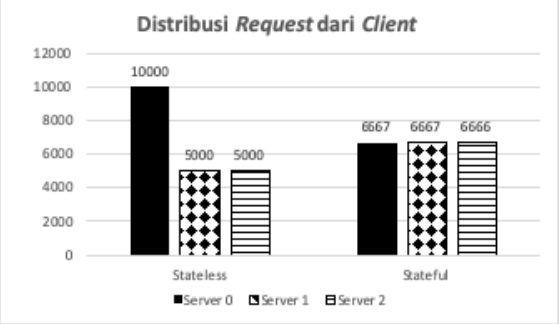
Gambar 9. Perbandingan jumlah sinyal *switch* ke *controller*

Pada Gambar 9 memberikan bukti bahwa jumlah sinyal *switch* ke *controller* dari *stateless* dan *stateful server load balancing* mempunyai perbedaan yang sangat besar dengan *stateful server load balancing* mempunyai nilai 40.000, sedangkan *stateless server load balancing* hanya mempunyai nilai 80. Pada Gambar 9 juga membuktikan bahwa jumlah sinyal *data plane* ke *control plane* akan mempengaruhi beban (*load*) pada *SDN controller*. Semakin banyak jumlah sinyal yang dihasilkan maka akan semakin membebani SDN *controller* yang dibuktikan dari bagian ini.

Dari hasil dan pembahasan ini, membuktikan bahwa *stateful server load balancing* mempunyai keterbatasan pada skalabilitas dikarenakan setiap *packet* yang masuk harus diproses oleh SDN *controller* yang menyebabkan SDN *controller* dapat mengalami *overload* dan berpengaruh terhadap kinerja dari *stateful server load balancing*. *Stateful server load balancing* berbasis SDN cocok digunakan pada jaringan yang mempunyai skala yang kecil dan tidak memprioritaskan kebutuhan kinerja pada sisi *SDN controller*. Sedangkan, *stateless server load balancing* berbasis SDN cocok digunakan pada jaringan yang mempunyai skala yang besar dan memprioritaskan kebutuhan kinerja pada *SDN controller*.

4.3. Server

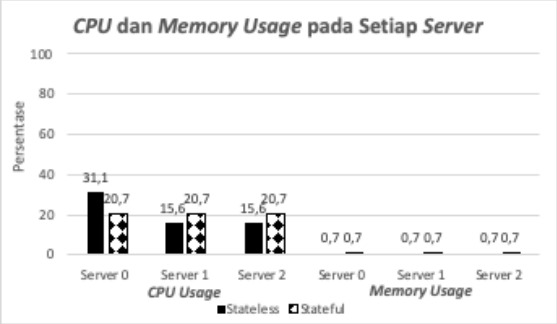
Gambar 10 menunjukkan perbandingan distribusi *request client* dari *stateless* dan *stateful server load balancing* berdasarkan data pada Lampiran B.6 Distribusi *Request Client* terhadap *Server*.



Gambar 10. Perbandingan distribusi *request*

Pada Gambar 10, ada perbedaan distribusi *request* dari *client* ke *server* pada *stateless* dan *stateful server load balancing*. Pada *stateless server load balancing* mempunyai perbedaan di mana distribusi *request* ke *server* 0 sebanyak 10.000 *request*, distribusi *request* ke *server* 1 sebanyak 5.000 *request*, dan distribusi *request* ke *server* 2 sebanyak 5.000 *request*. Sedangkan, pada *stateful server load balancing*, setiap pendistribusian *request* dilakukan secara merata antara *server 0*, *server 1*, dan *server 2*. Dari 20.000 *request* yang dilakukan, distribusi ke *server* 0 sebanyak 6.667 *request*, distribusi ke *server* 1 sebanyak 6.667 *request*, dan distribusi *request* ke *server* 2 sebanyak 6.666.

Pada bagian ini, dapat dianalisis bahwa *stateful server load balancing* lebih baik dibandingkan *stateful server load balancing* pada pendistribusian beban setiap *server* dengan algoritme *round-robin*. Dikarenakan, setiap *server* menerima beban yang sama rata. Sedangkan, pada *stateless server load balancing*, dapat membebani salah satu *server* jika *client* yang sudah tercatat pada *flow table SDN switch* melakukan *request* yang lebih banyak daripada *client* yang lain. Pendistribusian *request* tersebut juga mempengaruhi *CPU* dan *memory usage* pada *server* yang dibuktikan pada Gambar 11.



Gambar 11. Perbandingan *CPU* dan *memory usage* pada *server*

Pada Gambar 11, membuktikan bahwa *CPU usage* server 0 pada *stateless server load balancing* mempunyai nilai yang lebih besar daripada *stateful server load balancing* dengan nilai 31,1% dibandingkan dengan 20,7%. Sedangkan, pada *server 1* dan *server 2* pada *stateful server load balancing* mempunyai nilai yang lebih besar daripada *stateless server load balancing* dengan *stateful server load balancing* mempunyai nilai 15,6% dan *stateless server load balancing* mempunyai nilai 20,7%. Data tersebut membuktikan bahwa distribusi *request* mempengaruhi *CPU usage* pada *server*. Semakin banyak *request* yang dilakukan oleh *client* terhadap *server* maka semakin besar pula *CPU usage* pada *server* tersebut. Itu dibuktikan dengan server 0 yang menangani *request* lebih banyak daripada *server* yang lain pada *stateless server load balancing* yang mempengaruhi CPU *usage* dari *server 0* lebih besar daripada *server* yang lain.

Sedangkan, nilai pada *memory usage* pada *stateless* dan *stateful server load balancing* mempunyai nilai yang sama. Data ini juga membuktikan bahwa proses yang dilakukan untuk menangani *request* pada setiap *server* tidak terlalu melibatkan *memory* tetapi lebih melibatkan *CPU*.

Dan dari hasil dan pengujian ini, *stateless server load balancing* cocok digunakan pada jaringan yang mempunyai *server* yang heterogen atau mempunyai spesifikasi yang berbeda-beda dengan algoritme yang sesuai dengan kebutuhan. Sedangkan, *stateful server load balancing* cocok untuk digunakan pada jaringan yang mempunyai *server* yang homogen atau mempunyai spesifikasi yang sama dengan algoritme *round-robin*.

5. KESIMPULAN

Kesimpulan yang dapat diambil dari penelitian "Analisis Perbandingan Penerapan *Stateless* dan *Stateful* pada Aplikasi *Server Load Balancing* Berbasis *Software-Defined Networking*" mencakup dua faktor utama yaitu implementasi pengujian dan analisis. Implementasi dilakukan dengan membuat dua *server load balancing* dengan teknik *stateless* dan *stateful* yang berjalan pada *emulator Mininet* dengan aplikasi *SDN controller Ryu* dan *SDN switch Open vSwitch*. Kemudian, kesimpulan yang didapatkan dari penelitian ini adalah, sebagai berikut.

Pertama, *stateless server load balancing* mempunyai kinerja yang lebih baik pada sisi *client* daripada *stateful server load balancing*, dilihat dari hasil pengujian *throughput*. Perbandingan pada 25 *requests/second* menunjukkan bahwa perbandingan antara *stateless* dan *stateful* tidak terlalu jauh, dengan *stateless server load balancing* mempunyai *throughput* 22,8% lebih tinggi daripada *stateful*. Sedangkan, pada 250 *requests/second* menunjukkan bahwa perbandingan antara *stateless* dan *stateful* mulai jauh, dengan *stateless* mempunyai *throughput* 222,9% lebih tinggi daripada *stateful*. Hasil tersebut juga dipengaruhi oleh nilai *netwok latency* pada *stateless* dan *stateful server load balancing*. Semakin tinggi nilai *network latency*, maka akan semakin kecil *throughput* yang dihasilkan yang dibuktikan dengan *stateless server load balancing* mempunyai rata-rata nilai 23,35654 *milisecond* dan *stateful server load balancing* mempunyai rata-rata nilai 242,5678 *milisecond*.

Kedua, *stateless server load balancing* mempunyai kinerja yang lebih baik pada sisi *SDN controller* daripada *stateful server load balancing*, dilihat dari hasil pengujian *connection rate*. Perbandingan pada 25 *requests/second* menunjukkan bahwa perbandingan antara *stateless* dan *stateful* tidak

terlalu jauh, dengan *stateless server load balancing* mempunyai *connection rate* 22,9% lebih tinggi daripada *stateful*. Sedangkan, pada 250 *requests/second* menunjukkan bahwa perbandingan antara *stateless* dan *stateful* mulai jauh, dengan *stateless* mempunyai *connection rate* 222,9% lebih tinggi daripada *stateful*. Hasil tersebut dipengaruhi oleh *CPU* usage pada SDN *controller*. Semakin tinggi nilai *CPU usage* pada *SDN controller*, maka akan semakin kecil *connection rate* yang dihasilkan oleh *SDN controller* dengan s*tateless server load balancing* mempunyai *CPU usage* 0,6% dan *stateful server load balancing* mempunyai *CPU usage* 55,7%. Hasil dari CPU *usage* juga dipengaruhi oleh banyaknya jumlah sinyal dari *switch* ke *controller*. Semakin banyak jumlah sinyal yang dikirimkan, maka akan semakin besar *CPU usage* pada *SDN controller* dengan *stateless server load balancing* mempunyai jumlah sinyal 80 dan *stateful server load balancing* mempunyai jumlah sinyal 40.000.

Ketiga, *stateful server load balancing* mempunyai kinerja yang lebih baik pada sisi *server* daripada *stateless server load balancing*, dilihat dari hasil pengujian distribusi *request*. Dikarenakan, setiap *server* menerima beban yang sama rata dengan *server 0 menerima* 6.667 *request*, *server 1 menerima* 6.667 *request*, dan *server 2 menerima* 6.666 *request*. Sedangkan, pada *stateless server load balancing*, dapat membebani salah satu *server* jika *client* yang sudah tercatat pada *flow table* pada *SDN switch* melakukan *request* yang lebih banyak daripada *client* yang lain. Hal itu dibuktikan dengan *server 0 menerima* 10.000 *request*, *server 1 menerima* 5.000 *request*, dan *server 2 menerima* 5.000 *request*. Itu juga dibuktikan dengan *CPU usage server 0* mempunyai nilai 31,1%, *server 1* mempunyai nilai 15,6%, dan *server 2* mempunyai nilai 15,6% pada *stateful server load balancing*. Sedangkan pada *stateful server load balancing*, *server 0* mempunyai nilai 20,7%, *server 1* mempunyai nilai 20,7%, dan *server 2* mempunyai nilai 20,7%.

6. DAFTAR PUSTAKA

Al-Najjar, A. L. (2016). *Pushing SDN to the end-host, network load balancing using OpenFlow*.

Aslan, M., & Matrawy, A. (2016). *On the Impact of Network State Collection on the Performance of SDN Applications*. IEEE Communications Letters, 20(1), 5–8. doi:10.1109/lcomm.2015.2496955

Fernandez, M.P., 2013. Comparing OpenFlow controller paradigms scalability: Reactive and proactive. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, pp.1009–1016.

Gustin Anggraeni, Sukiswo & Ajub Ajulian Zahra. (2014). Analisis Kinerja Jaringan *Wireless LAN* Berdasarkan Mekanisme *Load balancing* dengan Algoritme *Round robin* Menggunakan Simulator OPNET 14.5.

Ibe, O. C., Choi, H., & Trivedi, K. S. (1993). Performance evaluation of client-server systems. IEEE Transactions on Parallel and Distributed Systems, 4(11), 1217–1229. doi:10.1109/71.250101

J. Esch, "Prolog to, "software-defined networking: a comprehensive survey"," in *Proceedings of the IEEE*, vol. 103, no. 1, pp. 10-13, Jan. 2015. doi: 10.1109/JPROC.2014.2374752

Kaur, S. et al., 2015. Round-Robin Based Load Balancing in Software Defined Network. *Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference*, pp.2–5.

MacVittie, Lori. 2014. *SDN Prerequisite: Stateful versus Stateless.* [Internet]. Available at: https://devcentral.f5.com/s/articles/sdn-prerequisite-stateful-versus-stateless.