Implementasi *Least Bandwidth Utilization Ratio* Sebagai Metode *Load Balancing* Pada Arsitektur *Software Defined Network*

Amir Husein[1], T.B. Utomo[2], Taufik Irfan[3]

[1]*Jurusan Teknik Elektro, Politeknik Negeri Bandung, Bandung 40012*
E-mail : amir.husein.tkom418@polban.ac.id
[2]*Jurusan Teknik Elektro, Politeknik Negeri Bandung, Bandung 40012*
E-mail : tb.utomo@polban.ac.id
[3]*Jurusan Teknik Elektro, Politeknik Negeri Bandung, Bandung 40012*
E-mail : taufik.irfan@polban.ac.id

ABSTRAK

Dewasa ini, pengguna internet kian meningkat seiring dengan perkembangan teknologi. Peningkatan ini mengakibatkan beban *server* bertambah dan perlunya sistem *load balancing* untuk menghindari *server down*. SDN atau *Software Defined Network* merupakan sebuah cara dalam melakukan jaringan dengan ide utama memisahkan antara *control* dan *data plane*. Protokol pada arsitektur SDN adalah OpenFlow, bertugas untuk menghubungkan *control plane* yang tersentralisasi dengan banyak *data plane*. Pada penelitian ini diimplementasikan sebuah *load balancer* pada SDN menggunakan metode *least bandwidth utilization ratio*. Pengujian dilakukan secara virtual menggunakan mininet dan Ryu pada 6 skenario. Pengukuran dilakukan pada 3 parameter *Quality of Service* (QoS), yaitu *throughput*, *response time*, dan *packet loss* yang kemudian dibandingkan dengan metode round robin. Nilai *throughput* pada skenario 1 hingga 6 cenderung lebih tinggi apabila dibandingkan dengan round robin, yang mana nilai *throughput* berbanding lurus dengan sedikit terjadinya *packet loss* pada seluruh skenario. *Packet loss* tidak terjadi sama sekali atau 0% dengan nilai rata-rata *throughput* tertinggi berada pada skenario 6 sebesar 2493.69 KB/s, sedangkan pada round robin terjadi *packet loss* di skenario 3 dengan rata-rata sebesar 1.32%. Nilai *response time* cukup bervariasi apabila dibandingkan dengan round robin pada seluruh skenario, dengan nilai rata-rata berada pada rentang 165.54 ms hingga 172.08 ms.

Kata Kunci
Bandwidth utilization, load balance, mininet, openflow, SDN, server

1. PENDAHULUAN

Dewasa ini, perkembangan teknologi khususnya pada bidang jaringan telekomunikasi berkembang dengan pesat yang diikuti oleh pengguna internet yang kian meningkat. Menurut data hasil survei Pengguna Internet Indonesia tahun 2019-2020 yang dilakukan oleh APJII, bahwa pengguna internet di Indonesia mencapai 196.7 juta pengguna, dengan kenaikan sebesar 8.9% pertahun. Hal ini tentunya akan berdampak pada pekerjaan dan beban *server* secara keseluruhan.

Penggunaan dan beban *server* yang berlebihan akan menyebabkan kondisi *server down*, yaitu kondisi dimana *server* tidak mampu mengolah *request* dari *client* untuk dikembalikan dalam bentuk *response*. Maka dari itu untuk mengatasi hal ini, perlu adanya sistem *load balancing* dengan tujuan untuk meningkatkan kinerja dan ketersediaan jaringan yang dikhususkan pada *server* [1]. Pada jaringan konvensional, sistem *load balancing* di desain pada beberapa perangkat keras dan tentunya membutuhkan biaya lebih untuk hal tersebut, akan tetapi pada arsitektur *Software Defined Network* atau SDN, sistem *load balancing* merupakan suatu kode program yang mudah untuk diimplementasikan. SDN merupakan suatu paradigma jaringan yang memisahkan antara *control* dan *data plane*, sehingga jaringan bersifat fleksibel dan mudah untuk beradaptasi terhadap perubahan yang terjadi. SDN memiliki sebuah *controller* dengan tugas untuk melakukan kontrol pada seluruh elemen topologi [2].

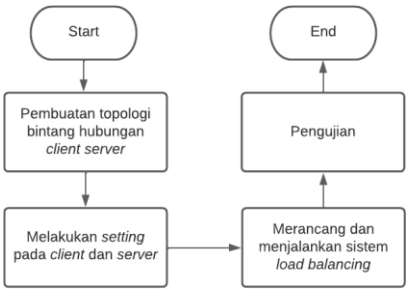
Saat ini, telah banyak sekali metode atau algoritma *load balancing* yang diimplementasikan guna meningkatkan

kinerja dan ketersediaan jaringan. Seperti halnya *round robin*, metode yang paling banyak digunakan saat ini, membagi beban pada *server* secara bergantian, akan tetapi metode ini tidak memperhatikan kondisi *link* maupun *server* sama sekali [1], [2]. Selain itu, terdapat pula metode *shortest delay* yang memilih *server* tujuan berdasarkan rata-rata *delay* paling rendah [3]. Kemudian terdapat pula metode *weighted least connection* yang memilih *server* tujuan berdasarkan koneksi TCP terhubung paling sedikit yang disertai pembobotan berdasarkan spesifikasi untuk tiap *server* [4]. Selain metode yang disebutkan tersebut, masih banyak metode lainnya yang digunakan, yang terbagi menjadi metode statis maupun dinamis seperti pemanfaatan *deep learning* dalam proses *load balancing*.

Penelitian mengenai sistem *load balancing* pada arsitektur SDN yang menggunakan metode perbandingan *bandwidth* pernah dilakukan sebelumnya, tetapi memiliki kelemahan dalam tahap pengujian yang tidak terlalu spesifik, karena hanya menguji nilai *throughput* dan *response time* saja dengan total 100 *request* [1]. Apabila dilihat dari penelitian sebelumnya, penelitian ini secara tidak langsung berusaha untuk mengembangkan hasil dari penelitian tersebut, dengan menekankan proses pengujian berupa tambahan skenario *server* yang mengembalikan data gambar serta gabungan *request* yang dikirim per detik. Penelitian ini akan menguji nilai dari tiga parameter QoS, yaitu *throughput, response time,* dan *packet loss*.

2. METODE PENELITIAN

Dalam melaksanakan penelitian ini, berikut di bawah ini adalah diagram alir dari metode penelitian yang dilakukan.

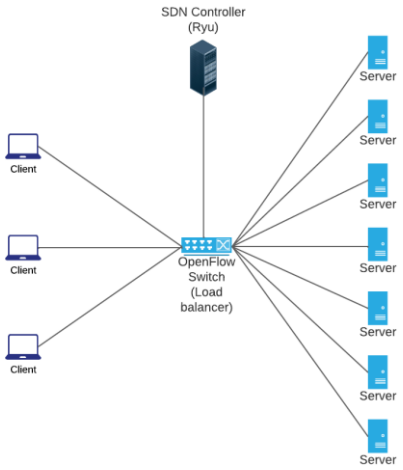


Gambar 1. Diagram alir perancangan sistem

Metode penelitian diawali dengan pembuatan topologi bintang hubungan *client server* yang dibuat di bawah *environment* mininet sebagai simulator topologi. Kemudian dilakukan konfigurasi pada setiap *client* maupun *server*, yang mana pada masing-masing *end-node* tersebut akan diatur dari mulai alamat IP, MAC, hingga nomor port yang akan saling terhubung menuju OpenFlow Switch yang akan bertindak sebagai *load balancer*. Setelah konfigurasi pada topologi selesai, maka perlu dibuat sebuah sistem *load balancing* menggunakan metode *least bandwidth utilization ratio* pada sisi kontroler sebagai *control plane* yang akan mengatur perpindahan data pada sisi *data plane* nantinya. Proses pengujian akan dilakukan menggunakan *tools httperf*, yaitu sebuah *tools* pengujian beban *server* yang mampu mengirimkan *request* dengan jumlah yang dapat diatur, dan metrik hasil pengujian yang dapat diamati dan diukur. Penjelasan lebih lanjut mengenai metode penelitian yang dilakukan adalah sebagai berikut.

2.1 Perancangan Topologi Jaringan

Dalam penelitian ini, topologi jaringan yang akan digunakan ialah topologi bintang hubungan antara *client* dan *server* yang terdiri dari 3 *client*, 7 *server,* 1 OpenFlow Switch yang bertindak sebagai *load balancer* dan 1 buah kontroler SDN yang akan bertindak sebagai *control plane*. OpenFlow Switch yang bertindak sebagai *load balancer* akan memiliki virtual IP sebagai alamat IP yang hanya akan diketahui oleh *client*. *Delay* tambahan sebesar 40ms ditambahkan pada setiap link di topologi, yang mana nilai ini merupakan *delay* terburuk yang masih dapat diterima oleh pengguna berdasarkan standar F.746.1 ITU-T [5]. Nilai *bandwidth* maksimum diatur sebesar 100 Mbps pada setiap link. Gambar 2 menunjukkan topologi jaringan yang dirancang dan akan digunakan pada penelitian ini.

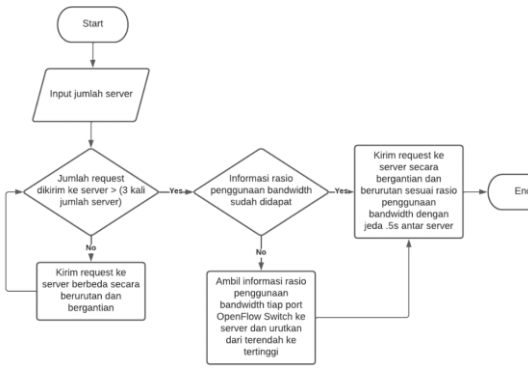


Gambar 2. Topologi jaringan

2.2 Perancangan Sistem *Load Balancing*

Metode *load balancing* yang akan digunakan dalam penelitian ini adalah *least bandwidth utilization ratio*. Metode ini akan bekerja dengan mengirimkan *request* dari *client* menuju *server* tujuan yang memiliki utilisasi *bandwidth* terendah pada *link* nya selama 0.5 detik, sehingga *server* yang memiliki utilisasi *bandwidth* terbesar pada *link* nya akan memperoleh prioritas terakhir dalam menerima *request* dari *client*.

Bandwidth utilization ratio (BWUtil) adalah sebuah kriteria evaluasi suatu *link* jaringan dalam menganalisis kemampuan suatu *link* dalam mengirimkan data. Nilai BWUtil sendiri merupakan rasio antara penggunaan *bandwidth* dengan nilai maksimum *bandwidth* pada suatu *link*. Nilai BWUtil yang besar dan mendekati 1 mengindikasikan bahwa rawan terjadinya *congestion* pada *link* tersebut [6].



Gambar 3. Diagram alir proses load balancing dengan metode *least bandwidth utilization ratio*

Diagram alir pada gambar 3 menunjukkan proses dari *load balancing* yang akan dilakukan. *Request* 1 sampai dengan *request* ke 21 akan dikirim dengan skema *round robin*. Hal ini dilakukan sebagai langkah awal dalam mendapatkan nilai BWUtil dari tiap *link* yang terhubung ke *server*. Setelah itu digunakan sebuah *tools* bwm-ng untuk memperoleh informasi mengenai penggunaan atau utilisasi *bandwidth* di tiap port OpenFlow Switch yang terhubung ke *server*. Nilai utilisasi *bandwidth* ini akan diurutkan mulai dari yang terkecil hingga terbesar, yang kemudian *request* ke 22 sampai seterusnya akan dikirim menuju *server* secara bergantian selama 0.5 detik mulai dari *server* yang memiliki nilai BWUtil terendah hingga terbesar. Adapun persamaan dari *bandwidth utilization ratio* adalah sebagai berikut:

$$BWUtil = \frac{Utilisasi\ bandwidth}{Bandwidth\ maksimum} \quad (1)$$

2.3 Skenario Pengujian

Skenario pengujian yang akan dilakukan terdiri dari 6 skenario, yaitu 3 skenario untuk *server* yang mengembalikan halaman HTML sederhana dengan total 1000 *request* untuk masing-masing skenario dan 3 skenario untuk *server* yang mengembalikan data gambar sebesar 4.4 MB dengan total 100 *request* untuk masing-masing skenario. Perbedaan dari tiap skenario akan difokuskan pada jumlah *request* per detik yang dikirim oleh *client*. Penjelasan lebih rinci untuk setiap skenario ditampilkan pada Tabel 1 berikut.

Tabel 1. Skenario pengujian

Skenario	Respon *server*	Total *request*	*Request*/s
1	Halaman HTML	1000	15
2			30
3			45
4	Gambar	100	6
5			12
6			18

2.4 Parameter Uji

Berikut merupakan parameter QoS yang akan diuji dan diukur untuk dianalisis:

1. *Throughput*

$$Throughput\ (bps) = \frac{Paket\ diterima}{Lama\ pengamatan} \quad (2)$$

Throughput adalah ukuran kecepatan transfer data yang diukur dalam satuan bps (*bits per second).*

2. *Response time*

$$Response\ time\ (ms) = \ Latency + Waktu\ proses \quad (3)$$

Response time merupakan total waktu tunggu antara dikirimnya bit pertama *request* sampai diterimanya bit pertama *response*.

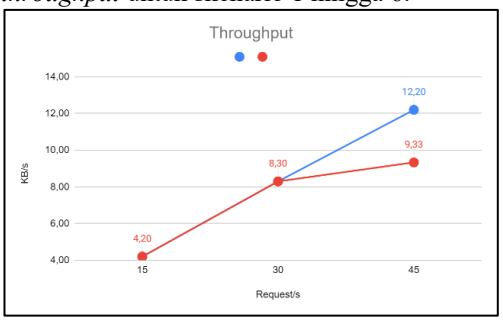
3. *Packet loss (PL)*

$$PL\ (\%) = \frac{(Paket\ diterima - dikirim)}{Paket\ diterima} \times 100 \quad (4)$$

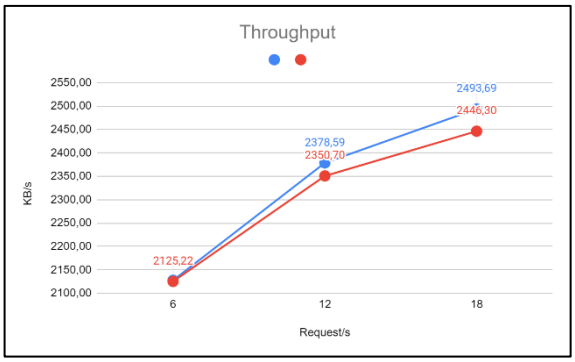
Packet loss adalah rasio hilangnya paket pada proses pengiriman dari total paket dikirim. Dalam penelitian ini, paket diilustrasikan sebagai *request*, karena keduanya memiliki nilai yang tetap dan dapat diukur *loss* nya.

3. HASIL DAN PEMBAHASAN

Pengujian dilakukan pada 6 skenario berbeda dan didapatkan nilai *throughput*, *response time*, dan *packet loss* untuk setiap skenario. Hasil dari keseluruhan pengujian disajikan dalam bentuk grafik. Hasil dari metode *least* BWUtil ditunjukkan dengan warna biru, sedangkan untuk *round robin* oleh warna merah. Berikut adalah hasil pengujian *throughput* untuk skenario 1 hingga 6.

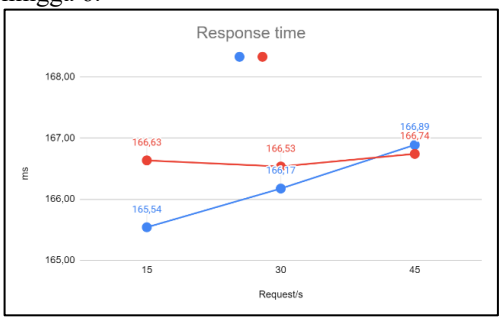


Gambar 4. Hasil pengujian *throughput* untuk *server* dengan respon HTML

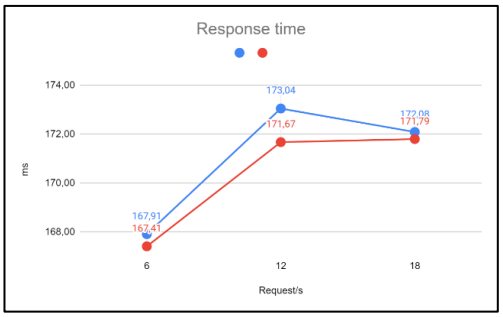


Gambar 5. Hasil pengujian *throughput* untuk *server* dengan respon gambar

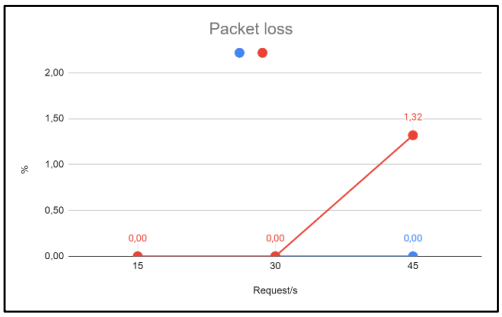
Pada gambar 4 dan 5, dapat dilihat bahwa nilai *throughput* yang dihasilkan oleh metode *least* BWUtil memiliki nilai yang lebih tinggi apabila dibandingkan dengan metode *round robin* pada 4 skenario yaitu skenario 3, 4, 5, dan 6. Pada skenario 1 dan 2 dengan 15 dan 30 *request* per detik, nilai *throughput* yang dihasilkan dari kedua metode adalah sama, yaitu sebesar 4.2 dan 8.3 KB/s, kesamaan nilai ini disebabkan karena besar data HTML yang dikembalikan cenderung kecil, tetapi pada skenario 3, nilai *throughput* pada metode *round robin* dipengaruhi oleh *packet loss* yang terjadi. Dari grafik, dapat dilihat pula bahwa peningkatan jumlah *request* per detik berbanding lurus dengan peningkatan nilai *throughput* yang dihasilkan. Metode *least* BWUtil terbukti mampu membagi beban *request* menuju *server* lebih baik apabila dibandingkan dengan *round robin* dari sisi pengujian *throughput*. Berikut adalah hasil pengujian *response time* untuk skenario 1 hingga 6.



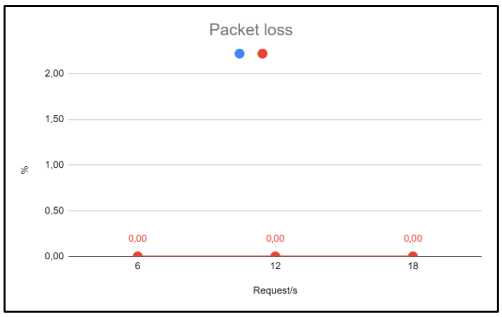
Gambar 6. Hasil pengujian *response time* untuk *server* dengan respon HTML



Gambar 7. Hasil pengujian *response time* untuk *server* dengan respon gambar



Gambar 8. Hasil pengujian *packet loss* untuk *server* dengan respon HTML



Gambar 9. Hasil pengujian *packet loss* untuk *server* dengan respon gambar

Nilai *response time* yang ditunjukkan oleh gambar 6 dan 7 untuk kedua metode cenderung memiliki nilai yang fluktuatif. Pada skenario 1 dan 2, nilai *response time* dari metode *least BWUtil* lebih rendah bila dibandingkan dengan *round robin*, akan tetapi pada skenario 3 hingga 6, *response time* dari *least* BWUtil lebih besar bila dibandingkan dengan metode *round robin*. Hal ini disebabkan karena *response time* sendiri ialah penjumlahan dari waktu latensi jaringan dengan waktu pemrosesan, baik pemrosesan pada sisi *server* maupun kontroler SDN. Hal ini tentunya berpengaruh, karena pada skenario 4 hingga 6, data yang dikirim sebesar 4.4 MB, sehingga menyebabkan waktu pemrosesan yang lebih pada sisi kontroler SDN maupun *server*. Selain itu, pada metode *least* BWUtil, terdapat pula proses perhitungan nilai utilisasi *bandwidth* menggunakan *tools* pihak ketiga bwm-ng yang tentunya membutuhkan waktu apabila dibandingkan dengan metode *round robin* yang tidak melaksanakan proses perhitungan apapun. Pada kondisi *link* yang mengalami *delay* tambahan sebesar 40 ms sesuai standar F.746.1 ITU-T mengenai *delay* terburuk yang masih dapat diterima oleh pengguna, nilai *response time* yang didapatkan dari proses pengujian masih berada pada predikat 'Baik' sesuai dengan standar TIPHON mengenai level *delay* pada jaringan [7]. Kemudian dibawah ini akan ditampilkan grafik hasil pengujian *packet loss* untuk skenario 1 hingga 6.

Pada gambar 8 dan 9 dapat dilihat bahwa nilai *packet loss* pada 5 skenario yaitu skenario 1,2,4 hingga 6 bernilai 0 atau tidak terdapat *packet loss* sama sekali. *Packet loss* hanya terjadi pada skenario 3, tepatnya pada metode *round robin* sebesar 1.32%. Hal ini membuktikan bahwa metode *round robin* belum begitu baik dalam menangani *request* dalam jumlah yang banyak per detiknya, yaitu 45 *request* per detik. Berbeda halnya dengan metode *least* BWUtil yang tidak mengalami *packet loss* sama sekali pada seluruh skenario pengujian. Hal ini dikarenakan metode *least* BWUtil memilih *server* berdasarkan rasio penggunaan *bandwidth* terendah, sehingga *request* yang dikirim minim terjadi *congestion* pada link yang terhubung ke *server*. Sedangkan metode *round robin* tidak memperhatikan apapun, baik itu *server* maupun *link* pada jaringan, sehingga rawan terjadinya kegagalan *server* dalam mengembalikan respon pada *client*.

4. KESIMPULAN

Dari penelitian yang telah dilakukan, dapat disimpulkan bahwa *least bandwidth utilization ratio* sebagai metode *load balancing* telah cukup baik dalam membagi beban menuju *server* pada kondisi *delay* terburuk 40 ms apabila dibandingkan dengan metode *round robin* pada pengujian

throughput dan *packet loss*. Nilai *throughput* yang dihasilkan lebih tinggi dari metode pembanding dengan nilai tertinggi sebesar 2493.69 KB/s. Selain itu, metode *least bandwidth utilization ratio* ini telah cukup baik dalam menangani *request* per detik dalam jumlah yang besar, khususnya pada skenario 3 tanpa adanya *packet loss*, dimana pada metode *round robin* masih terdapat *loss* yang terjadi sebesar 1.32%. Nilai *response time* secara keseluruhan tidak menunjukkan hasil yang begitu baik apabila dibandingkan dengan metode *round robin*, dikarenakan terdapat waktu pemrosesan pada sisi kontroler SDN dan *server* serta proses perhitungan nilai utilisasi *bandwidth* pada kontroler SDN yang mempengaruhi *response time* itu sendiri.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih atas bantuan dana finansial dari Politeknik Negeri Bandung melalui skema Bantuan Dana Tugas Akhir Mahasiswa tahun anggaran 2022.

DAFTAR PUSTAKA

[1] A. Arahunashi, G. Vaidya, Neethu, and K. V. Reddy, "Implementation of Server Load Balancing Techniques Using Software-Defined Networking," 2018.

[2] "Server Load Balancing in Software Defined Networking," *National Journal of Parallel and Soft Computing*, vol. 1, no. 1, pp. 261–265, Mar. 2019.

[3] L. Ronny, C. Negara, W. Yahya, and R. Primananda, "Analisis Dan Implementasi Load Balancing Pada Web Server Dengan Algoritme Shortest Delay Pada Software Defined Network," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 2, no. 9, pp. 2791–2797, Sep. 2018, [Online]. Available: http://j-ptiik.ub.ac.id

[4] H. Karim, R. Primananda, and W. Yahya, "Implementasi Load Balancing Web Server Dengan Algoritme Weighted Least Connection Pada Software Defined Network," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 3, no. 1, pp. 109–118, Jan. 2019, [Online]. Available: http://j-ptiik.ub.ac.id

[5] F 746, "ITU-T Rec. F.746.1 (10/2014) Requirements for low-latency interactive multimedia streaming," 2014. [Online]. Available: http://handle.itu.int/11.1002/1000/11

[6] C. X. Cui and Y. bin Xu, "Research on load balance method in SDN," *International Journal of Grid and Distributed Computing*, vol. 9, no. 1, pp. 25–36, 2016, doi: 10.14257/ijgdc.2016.9.1.03.

[7] "Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON); General aspects of Quality of Service (QoS)," Jun. 1999. [Online]. Available: http://www.etsi.org