ANALISIS *LOAD BALANCING* MENGGUNAKAN ALGORTIMA OPTIMASI KOLONI SEMUT DAN *LEAST CONNECTION* PADA JARINGAN *SOFTWARE DEFINED NETWORK*

(LOAD BALANCING ANALISYS USING ANT COLONY OPTIMIZATION AND LEAST CONNECTION ALGORITHM ON SOFTWARE DEFINED NETWORK)

Putra Prawhira Desa[1] Favian Dewanta[2] , Ridha Muldina negara[3]
[1,2,3]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom
prawhiradezza@student.telkomunivesity.ac.id[1] favian@telkomuniversity.ac.id[2] ,
ridhanegara@telkomuniversity.ac.id[3]

ABSTRAK

Berkembangnya teknologi dan kebutuhan terhadap informasi yang semakin tinggi mempengaruhi kepadatan yang terjadi pada sebuah *traffic* jaringan. Semakin banyak *user* yang melakukan permintaan terhadap sebuah *server* maka kepadatan pada sebuah *traffic* akan meningkat. Keadaan seperti ini dapat mengakibatkan kualitas sebuah jaringan menurun, hal ini dapat dilihat dari parameter QOS yang nilainya akan menurun.

Salah satu cara mengatasi masalah tersebut adalah dengan melakukan *load balancing. Load balancing* adalah proses pendistribusian beban pada suatu jaringan secara merata. Tugas akhir ini akan melakukan simulasi *load balancing* dengan algoritma optimasi koloni semut dan algoritma *least connection*, yang dilakukan dengan menggunakan opendaylight *controller* dan disimulasikan pada topologi Abilene dengan 16 *host* dan 11 *switch.*

Dari hasil pengujian dan analisis, penulis dapat menyimpulkan bahwa *load balancing* dengan menggunakan algoritma *least connection* lebih efektif dibandingkan *load balancing* dengan menggunakan algoritma optimasi koloni semut. Hal itu dibuktikan dengan nilai efektifitas pada algoritma *least connection* lebih besar 1,5 % untuk *throughput*, 2 % untuk *delay*, dan nilai *jitter* yang diperoleh lebih kecil 0.005 *millisecond* atau 1,1% dibandingkan dengan algoritma optimasi koloni semut. Hal tersebut terjadi karena algoritma optimasi koloni semut membutuhkan waktu lebih lama untuk pencarian jalur terbaik dibandingkan dengan algoritma *least connection*.

Kata kunci : SDN, *Load balancing*, Optimasi Koloni Semut, *Least Connection, Quality of Service*

ABSTRACT

The development of technology and the need for higher information affect the density that occurs in network traffic. The more users who make requests to a server, the density of traffic will increase. Conditions like this can cause the quality of a network to decline, this can be seen from the QOS parameter whose value will decrease.

One way to solve this problem is to do load balancing. Load balancing is the process of distributing loads on a network evenly. In this final project, a load balancing simulation will be carried out using the ant colony optimization algorithm and the least connection algorithm using the opendaylight controller and formulated on the Abilene topology with 16 hosts and 11 switches.

From the results of testing and analysis, it can be concluded that the performance of load balancing using the least connection algorithm is more effective than that of load balancing using the ant colony optimization algorithm. This is proven by the value of the effectiveness of the least connection algorithm greater than 1.5% for the throughput, 2% for the delay, and the value of the jitter is 0.005 millisecond or 1.1% smaller than the ant colony optimization algorithm. This is influenced by the fact that the ant colony optimization algorithm takes longer time to search for the best path with respect to the least connection algorithm..

Keyword : SDN, Load balancing, Ant Colony Optimization, Least Connection, Quality of Service

1. Pendahuluan

Perkembangan teknologi jaringan internet terjadi sangat pesat, hal ini dapat dilihat dari banyak hal baru yang tercipta dan terhubung ke jaringan internet. Perkembangan ini membuat kebutuhan manusia dimudahkan baik dalam pembangunan, pemeliharaan maupun pemantauan terhadap jaringan internet. Seiring berjalannya waktu dan pesatnya perkembangan jaringan internet, suatu konsep atau paradigma baru dalam teknologi jaringan muncul yaitu *Software Defined Network* (SDN). SDN adalah sebuah pendekatan baru untuk merancang, membangun dan mengelola jaringan komputer dengan memisahkan antara *control plane* dan *data plane*, konsep utama jaringan SDN adalah sentralisasi jaringan di mana pusat pengaturan jaringan berada di *control plane*[1].Kemampuan yang dimiliki oleh jaringan SDN dapat digunakan untuk automasi jaringan dan dapat memaksimalkan penggunaan perangkat jaringan seperti *load balancing* [2].

Load balancing adalah sebuah cara atau teknik untuk mendistribusikan beban di antara beberapa jaringan untuk mencapai pemanfaatan sumber daya yang maksimal. *Load balancing* dilakukan untuk menghindari *overload* dan memberikan kualitas layanan yang maksimal. *Load balancing* memungkinkan akses dalam jaringan dapat terdistribusi atau tidak terpusat sehingga kinerja dari jaringan komputer dapat bekerja secara stabil [3] .

Pada metode *load balancing* terdapat beberapa algoritma yaitu, *threshold, central manager, least loaded*, round-robin *dan least connection*. Pada tugas akhir algortima yang akan digunakan adalah

algoritma optimasi koloni semut dan algortima *least connection*. Kedua algoritma ini mempunyai karakteristik tersendiri dan keduanya berkerja secara dinamis untuk mencari jalur terbaik sehigga pendistribusian beban pada trafik terjadi secara merata[4]. Dengan melakukan penelitian untuk membandingkan algoritma optimasi koloni semut dan algoritma *least connection*, penulis akan mengetahui algoritma yang dapat bekerja secara optimal untuk optimasi pemerataan beban pada *traffic*

2. Dasar Teori

2.1 *Software Defined Network*

Software Defined Network (SDN) adalah paradigma baru yang bersifat inovatif untuk merancang, mengimplementasikan dan mengelola jaringan yang memisahkan antara *control plan*e dan *data plane* untuk mendapatkan pengalaman pengguna yang lebih baik. Konsep pada jaringan SDN ini menawarkan banyak manfaat dalam berbagai fleksibilitas dan pengendalian jaringan. Selain itu pada konsep SDN, *operation control* dipusatkan pada *controller* yang menentukan kebijakan pada jaringan. SDN memberikan fasilitas manajemen jaringan dan memungkinkan konfigurasi jaringan secara efesien dan terprogram untuk meningkatkan kinerja jaringan dan *monitoring*[5].

2.2 *Software Defined Network Controller*

*Controlle*r SDN adalah *software* yang digunakan untuk mengatur dan mengelola *flow control* untuk mengaktifkan *intelligent Networking* adalah inti dari jaringan SDN. *Controller* juga menggunkan *protocol openflow* untuk mengkonfigurasi perangkat jaringan[6]

2.3 Protokol *Open Flow*

OpenFlow adalah protokol yang relatif baru yang dirancang dan di implementasikan di Stanford University pada tahun 2008. Protokol baru ini bertujuan untuk mengontrol *data plane switch*, yang telah dipisahkan secara fisik dari *control plane* menggunakan perangkat lunak pengendali *(Controller*) pada sebuah *server*. *Control Plane* berkomunikasi dengan data plane melalui protocol OpenFlow. Bentuk *Software Defined Networking* (SDN) memungkinkan para peneliti, *administrator* dan operator untuk mengontrol jaringan mereka dengan perangkat lunak khusus dan menyediakan *Application Programming interface* (API) terhadap tabel *forwarding* dari *switch* dari *vendor* yang berbeda[7].

2.4 *Load Balancing*

Load balancing merupakan proses pendistribusian beban (*load*) terhadap sebuah pelayanan yang ada pada sekumpulan perangkat jaringan atau *server*. Ketika ada banyak

permintaan dari pengguna maka jaringan atau *server* akan terbebani karena harus melakukan pelayanan terhadap pengguna jaringan tersebut. Solusi yang cukup bermanfaat adalah dengan membagi-bagi beban yang datang ke beberapa *server*, jadi tidak berpusat ke salah satu perangkat jaringan saja. Teknologi itulah yang disebut Teknologi *Load balancing*. Dengan menggunakan Teknologi *Load balancing* dapat diperoleh keuntungan seperti menjamin reabilitas pelayanan, dan skalabilitas suatu jaringan [8].

2.5 Mininet

Mininet adalah sebuah *emulator* yang membentuk jaringan yang terdiri dari *host, switch* dan *controller* yang divirtualisasi. *Hos*t pada mininet menggunakan *Operasting system* linux dan *switchnya* menggunakan OpenFlow. *Emulator* mininet bekerja dengan virtualisasi berbasis proses sehingga menghasilkan simulasi jaringan yang nyata[9]

2.6 Algoritma Optimasi Koloni Semut

Algoritma optimasi koloni semut adalah algortima yang dikemukakan oleh Marco Dorigo pada awal tahun 1990. Sesuai dengan namanya algoritma ini terinspirasi dari perilaku koloni semut pada saat mencari makanan. Dalam penerapannya algoritma ini memiliki dua tahapan. Tahap pertama saat semut maju, proses ini akan digunakan untuk mencari rute baru dan menemukan informasi tentang keadaan rute yang dilewati. Ketika semut sudah menemukan tempat tujuan, ditahap kedua semut mundur melalui jalur yang sama. Teknik yang dilakukan oleh semut ini bisa diadaptasi untuk memecahkan permasalahan sutau jaringan seperti menemukan jarak terdekat untuk mengirimkan paket [12].

2.7 Algoritma *Least Connections*

Algoritma *least connection* adalah algoritma load balancing yang mengarahkan pendistribusian beban trafik pada node yang memiliki jumlah koneksi aktif paling sedikit. Algoritma ini sangat baik untuk melakukan proses penditribusian beban ketika jumlah beban yang acak dan banyak [12]

2.8 *Quality of Service*

1. ***Throughput***

   Throughput adalah jumlah total kedatangan paket yang sukses yang diamati atau diproses dalam suatu interval waktu tertentu. [13].

2. ***Delay***

   Delay adalah waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan [14].

3. *Jitter*

Jitter adalah variasi delay perbedaan selang waktu kedatangan antar paket di terminal tujuan[14].

4. *Packet Loss*

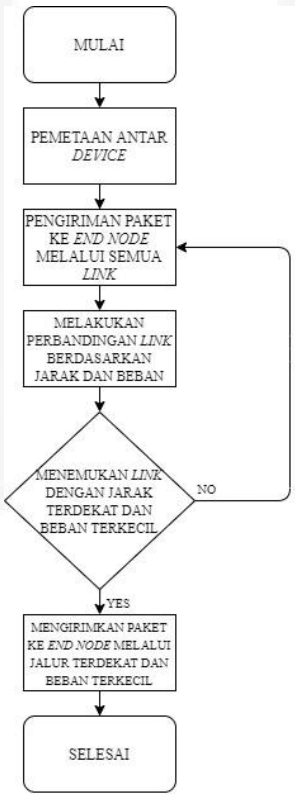
Packet loss adalah parameter yang menggambarkan suatu kondisi yang menentukan jumlah paket yang hilang pada saat proses transmisi paket. [14].

3. Model dan Sitem Perancangan

3.1 Desain Sistem

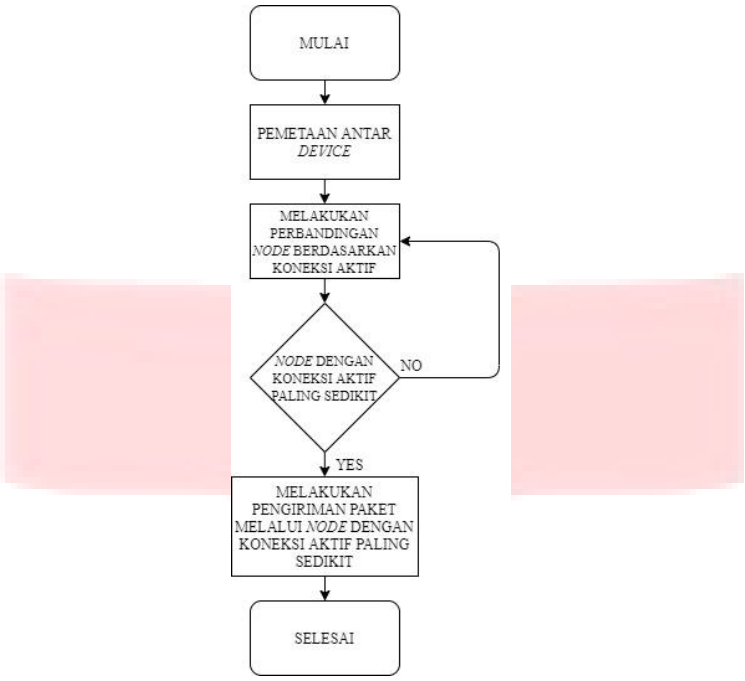
Untuk proses perancangan system yang akan dibangun untuk melakukan perbandingan performansi *Load balancing* dengan algoritma optimasi koloni semut dan algoritma *least connection* terbagin menjadi dua tahap, yaitu perancangan *hardware* dan perancangan *software*. Untuk percangan *hardware* yaitu menentukan dan melakukan penyesuaian spesifikasi PC yang digunakan untuk penelitian. Sedangkan untuk perangcangan *software* terdiri instalasi OS XUbuntu, Instalasi Mininet dan installasi *controller*. selanjutnya adalah melakukan simulasi *Load Balancing* dengan menggunkaan algoritma optimasi koloni semut dan algoritma *least connection.*

3.2 Diagram Blok Algoritma Optimasi Koloni Semut



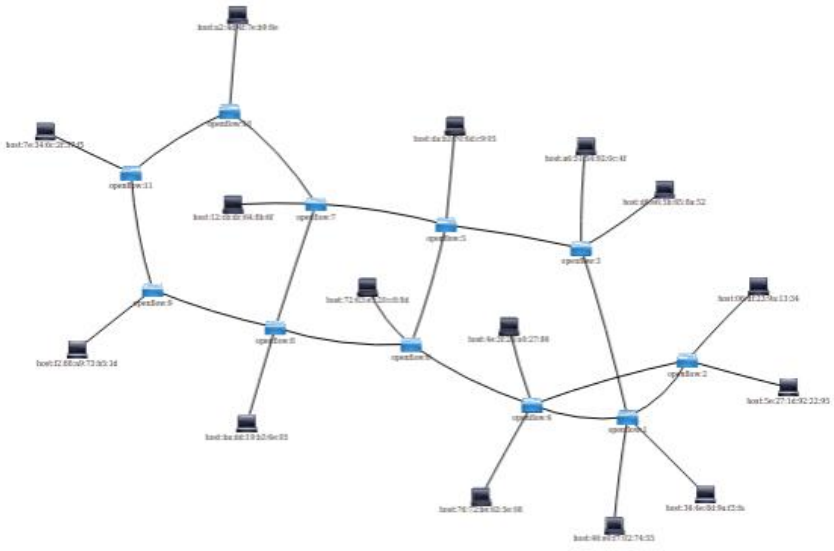
Gambar 3.1 Diagram blok algoritma optimasi koloni semut

3.3 Diagram Blok Algoritma *Least Connection*



Gambar 3.2 Diagram blok algoritma *least connection*

3.4 Topologi Jaringan



Gambar 3.3 Topologi jaringan

Gambar 3.3 adalah bentuk topologi yang digunakan dalam penelitian ini, dengan jumlah 11 switch dan 15 host.

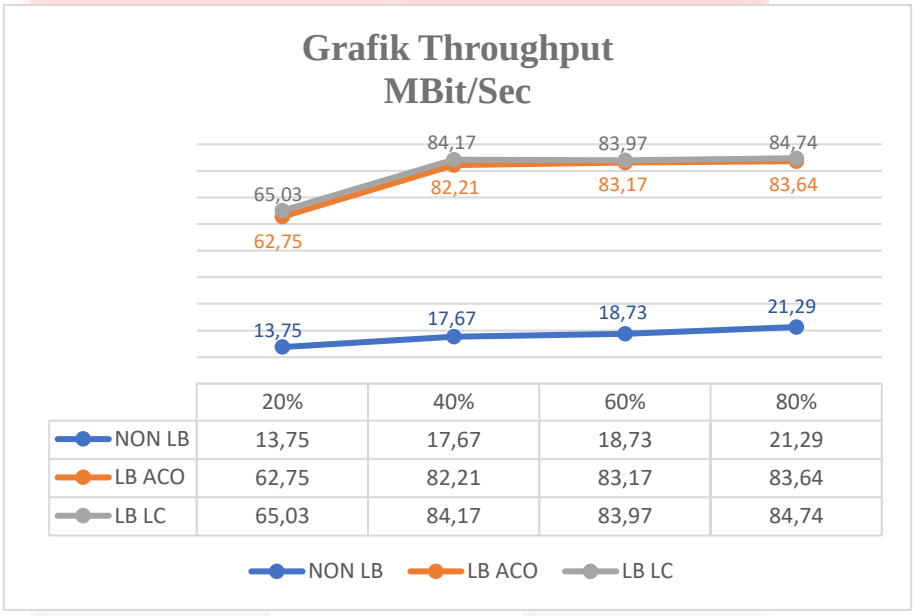
3.5 Skenario Pengujian

Adapun skenario yang digunakan untuk proses pengujian pada penelitiaan ini, yaitu :

1. Pengujian dilakukan dengan membangkitan traffic dari host 4 ke host 13 dan dari host 12 ke host 2 dengan tuuan untuk membuat traffic padat.
2. Pengujian dilakukan dengan cara mengukur *throughput*, *delay* dan *jitter* pada dari *host* 14 sebagai *client* ke *host* 1 sebagai *server*.
3. Pengambilan data menggunakan D-ITG

4. Hasil

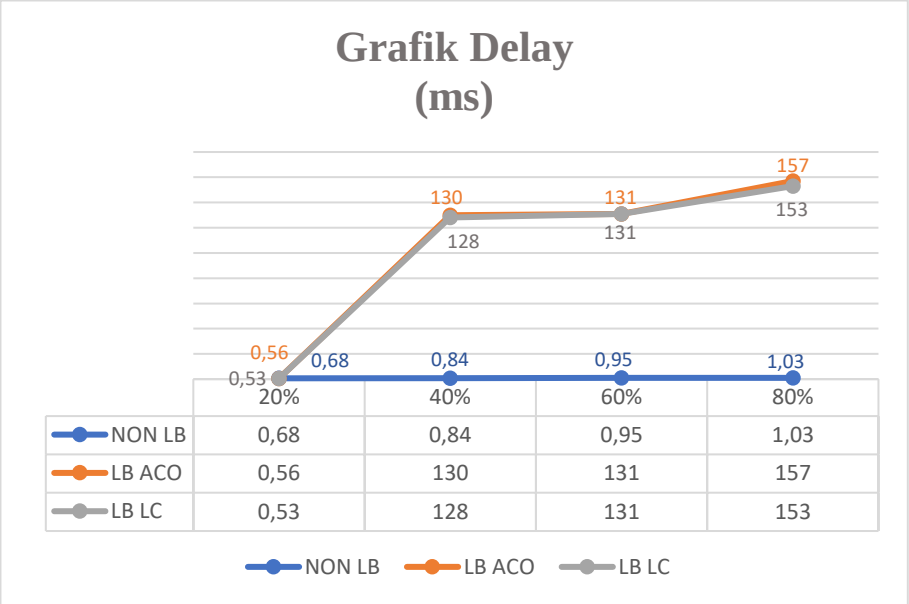
4.1 Pengujian *Troughput*



Gambar 4.1 Grafik perbadingan nilai *throughput*

Pada **Gambar 4.1** dilihat bahwa nilai *throughput* yang dihasilkan pada saat simulasi *load balancing* dengan algoritma optimasi koloni semut dan *least connection* nilainya lebih tinggi dibandingkan saat tidak melakukan *load balancing*. Pada saat simulasi ini nilai *throughput* dipengaruhi besarnya *background traffic* atau aliran data yang diaktifkan pada jaringan. Apabila nilai *throughput* dibandingkan secara keseluruhuan pada setiap algoritma, algortima *least connection* memiliki nilai efektifitas *load balancing* 1,2% lebih tinggi dibandingkan dengan algoritma optimasi koloni semut.

4.2 Pengujian Delay

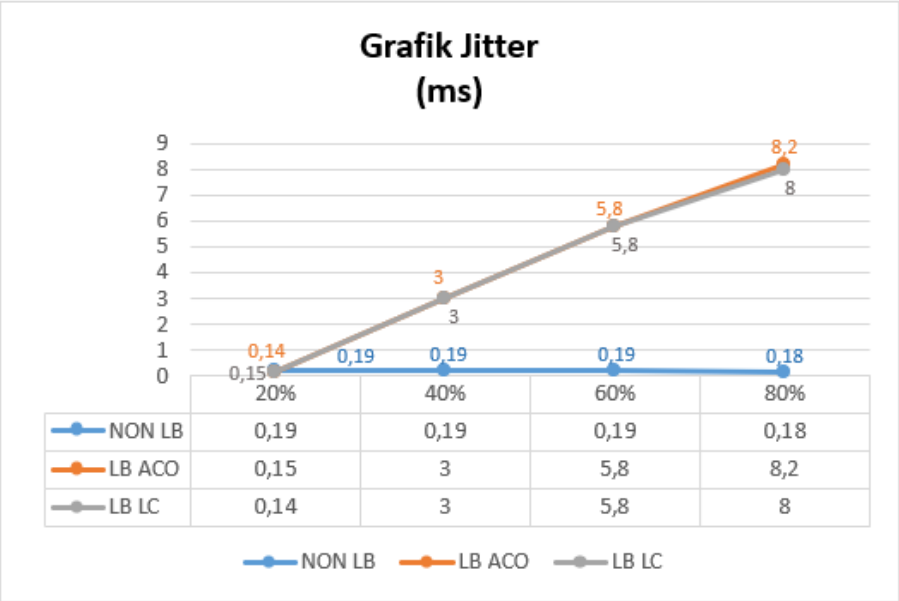


	20%	40%	60%	80%
NON LB	0,68	0,84	0,95	1,03
LB ACO	0,56	130	131	157
LB LC	0,53	128	131	153

Gambar 4.2 Grafik perbadingan nilai *delay*

Pada **Gambar 4.2** dapat dilihat untuk nilai *delay* pada saat simulasi terdapat perbedaan pada setiap skenario. Pada simulasi ini nilai *delay* yang didapat untuk algoritma optimasi koloni semut dan algoritma *least connection* lebih tinggi di bandingkan pada saat simulasi tanpa melakukan *load balancing.* Hal itu dipengaruhi oleh besaran aliran data yang diaktifkan di *traffic* dan juga dipengaruhi oleh waktu untuk melakukan pencarian jalur terbaik dari setiap algoritma.

4.2 Penujian Jitter



	20%	40%	60%	80%
NON LB	0,19	0,19	0,19	0,18
LB ACO	0,15	3	5,8	8,2
LB LC	0,14	3	5,8	8

Gambar 4.3 Grafik perbadingan nilai Jitter

Pada **Gambar 4.3** dapat dilihat nilai *jitter* yang diperoleh pada saat dilakukan simulasi. Pada saat simulasi tanpa melakukan *load balancing* nilai *jitter* yang diperoleh cenderung tidak mengalami perbuahan yang siginifikan atau cenderung stabil. Beda halnya pada saat dilakukan *load balancing* menggunakan algoritma optimasi koloni semut dan *least connection*. Nilai *jitter* terus naik berbanding lurus dengan *background* yang diaktifkan. Hal itu dipengaruhi oleh cara kerja dari algoritma yang digunakan. Apabila nilai *jitter* keseluruhan dari setiap algortima dirata-ratakan, nilai *jitter* pada lagoritma optimasi koloni semut lebih besar 0.05 *second* dibandingkan dengan nilai *jitter* yang diperoleh untuk algoritma *least connection*.

5. Kesimpulan

Bedasarkan nilai parameter yang didapat *load balancing* dengan algoritma *least connection* lebih effesien untuk digunakan dibandingkan dengan algoritma optimasi koloni semut. Hal itu dikarenakan waktu yang dibutuhkan algoritma optimasi koloni semut lebih lama dibandingkan algorima *least connection* untuk melakukan iterasi pada saat mencari jalur terbaik.

Daftar Pustaka

[1] I. Ummah, "Perancangan Simulasi Jaringan Virtual Berbasis Software-Define Networking," *Indones. J. Comput.*, vol. 1, no. 1, pp. 95–106, 2016, doi: 10.21108/indojc.2016.1.1.20.

[2] F. Mulya1, M. Pm. Dr. Tito Waluyo P., S.Si, S.T, and M. T. Roswan Latuconsina, S.T., "Analisis Load Balancing Pada Jaringan Software Defined Network (Sdn) Menggunakan

Algoritma Optimasi Koloni Semut," vol. 6, no. 1, pp. 1393–1400, 2019.

[3] I. Ahmad, S. N. Karunarathna, M. Ylianttila, and A. Gurtov, "Load Balancing in Software Defined Mobile Networks," *Cent. Wirel. Commun. (CWC), Univ. Oulu, Oulu, FinlandDepartment Comput. Sci. Aalto Univ. Espoo, Finl.*, 2015.

[4] H. Nasser and T. Witono, "Analisis Algoritma Round Robin, Least Connection, Dan Ratio Pada Load Balancing Menggunakan Opnet Modeler," *J. Inform.*, vol. 12, no. 1, 2016, doi: 10.21460/inf.2016.121.455.

[5] A. K. Arahunashi, G. G. Vaidya, S. Neethu, and K. V. Reddy, "Implementation of Server Load Balancing Techniques Using Software-Defined Networking," *Proc. 2018 3rd Int. Conf. Comput. Syst. Inf. Technol. Sustain. Solut. CSITSS 2018*, pp. 87–90, 2018, doi: 10.1109/CSITSS.2018.8768754.

[6] M. H. Hidayat and N. R. Rosyid, "Analisis Kinerja dan Karakteristik Arsitektur Software-Defined Network Berbasis OpenDaylight Controller," *Citee*, no. 2085–6350, pp. 194–200, 2017.

[7] R. Kartadie, E. Utami, and E. Pramono, "Prototipe Infrastruktur Software-Defined Network Dengan Protokol OpenFlow Menggunakan UBUNTU Sebagai Kontroler," *Dasi*, vol. 15, no. 1, 2014, [Online]. Available: http://ojs.amikom.ac.id/index.php/dasi/article/view/179.

[8] M. Arman, N. Wijaya, and H. Irsyad, "Analisis Kinerja Web Server Menggunakan Algoritma Round Robin dan Least Connection," *J. Sisfokom (Sistem Inf. dan Komputer)*, vol. 6, no. 1, p. 55, 2017, doi: 10.32736/sisfokom.v6i1.143.

[9] K. S. T. University, *Video Pengenalan Emulator Mininet*. Indonesia, Indonesia, 2017.

[10] S. Sathyanarayana and M. Moh, "Joint route-server load balancing in software defined networks using ant colony optimization," *2016 Int. Conf. High Perform. Comput. Simulation, HPCS 2016*, pp. 156–163, 2016, doi: 10.1109/HPCSim.2016.7568330.

[11] J. Li, L. Yang, J. Wang, and S. Yang, "Research on SDN load balancing based on ant colony optimization algorithm," *Proc. 2018 IEEE 4th Inf. Technol. Mechatronics Eng. Conf. ITOEC 2018*, pp. 979–982, 2018, doi: 10.1109/ITOEC.2018.8740366.

[12] "'The Least Connection Method,'" 2017. https://docs.citrix.com/en-us/netscaler/11-1/load-balancing/load-balancing-customizing-algorithms/leastconnection-method.html (accessed Sep. 10, 2020).

[13] Y. Pratama, U. Ependi, and H. Suroyo, "Optimization of Wireless Network Performance Using the Hierarchical Token Bucket (Case Study : Muhammadiyah University of Palembang) Optimasi Kinerja Jaringan Nirkabel Menggunakan Hierarchical (Studi Kasus : Universitas Muhammadiyah Palembang) Journal," vol. 1, no. 1, pp. 49–59, 2019.

[14] A. Darojat and I. Nurhaida, "Analisa Qos Administrative Distance Static Route Pada Failover Vpn Ipsec," *J. Ilmu Tek. dan Komput.*, vol. 3, no. 1, p. 11, 2019, doi: 10.22441/jitkom.2020.v3.i1.002.