ANALISIS LOAD BALANCING PADA JARINGAN SOFTWARE DEFINED NETWORK (SDN) MENGGUNAKAN ALGORITMA OPTIMASI KOLONI SEMUT

LOAD BALANCING ANALYSIS ON SOFTWARE DEFINED NETWORK (SDN) USING ANT-COLONY OPTIMIZATION ALGORITHM

Faizal Mulya[1], Dr. Tito Waluyo P., S.Si, S.T, M.PMat.[2], Roswan Latuconsina, S.T., M.T.,[3]

[1,2,3]Prodi S1 Sistem Komputer, Fakultas Teknik Elektro, Universitas Telkom
[1]faiizalle@gmail.com, [2]titowaluyo@gmail.com, [3]roswan78@gmail.com

Abstrak

Perkembangan teknologi pada jaringan internet semakin meningkat. Peningkatan ini pun berdampak pada server karena server sulit mendistribusikan muatan. Untuk mencukupi kebutuhan internet, teknik yang dapat digunakan adalah load balancing. Load balancing merupakan teknik untuk menggunakan dua atau lebih jalur koneksi internet dan menyeimbangkan beban di antara kedua jalur koneksi internet tersebut. Pada tugas akhir ini, masalah utama yang akan dibahas adalah simulasi pada algoritma optimasi koloni semut pada load balancing. Hasil pengujian meningkatkan throughput dan utilisasi CPU merata.

Kata Kunci: SDN *(Software Defined Network*), *Load-balancing*, Optimasi Koloni Semut

Abstract

The development of technology on internet networks is increasing. This increase also had an impact on the server because the server had difficulty distributing request. To meet internet needs, the technique that can be used is load balancing. Load balancing is a technique to use two or more internet connection lines and balance the request between the two internet connection lines. In this final project, the main problem to be discussed is a simulation on the ant colony optimization algorithm on load balancing. The test results of load balancing using ant-colony optimization algorithm is able to increase throughput and CPU utilization evenly.

Keywords: *SDN (Software Defined Network), Load-balancing, Ant-colony Optimization*

1. Pendahuluan

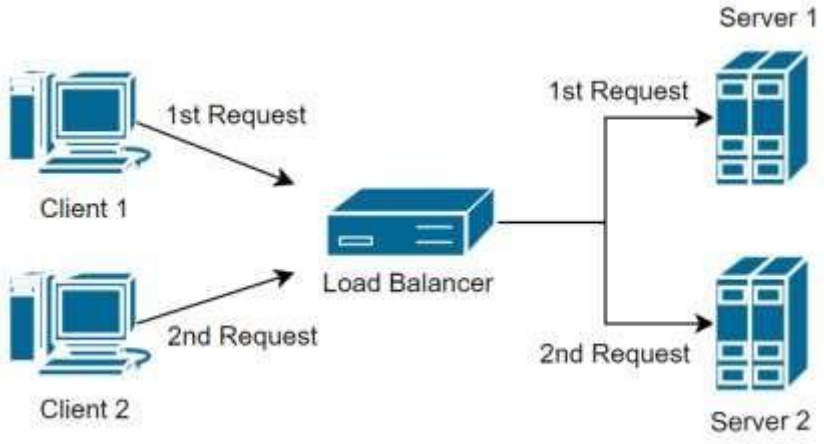
Perkembangan teknologi jaringan komputer belakangan ini berkembang sangat pesat, yang dimana perkembangannya membuat kebutuhan manusia dapat di mudahkan baik dalam pembangunan, monitoring atau memelihara jaringan komputer. Seiring berjalannya waktu dan pesatnya perkembangan teknologi jaringan yang memunculkan sebuah konsep baru dalam teknologi jaringan yaitu Software-Defined Network (SDN). SDN adalah istilah yang merujuk pada konsep baru dalam mendesain, mengelola dan pengimplementasian jaringan, terutama untuk mendukung kebutuhan dan inovasi pada jaringan ini yang semakin lama semakin luas. Kemampuan yang dimiliki oleh jaringan SDN juga dapat digunakan untuk mengubah prilaku jaringan serta dapat melakukan perubahan tersebut secara otomatis, dan dapat memaksimalkan pengunaan perangkat jaringan misalnya load balancing. Adanya konsep SDN ini dapat membantu menentukan peroutingan dengan menggunakan algoritma penentuan jalur.

Kemudian dari pada itu, untuk membantu meringankan beban maka dibutuhkan load balancing yang dimana dapat dimaksimalkan oleh jaringan SDN. Load balancing juga merupakan salah satu mekanisme untuk membagi beban komputasi ke beberapa server. Load balancing bertujuan untuk peningkatan kinerja server diantaranya memaksimal-kan nilai throughput, meminimalkan respond time, dan menghindari pembebanan berlebihan di salah satu server, karena beberapa server juga dapat mengurangi kemampuan komputasi. Maka dari itu teknik untuk mendistribusikan beban traffic pada dua atau lebih jalur koneksi secara seimbang atau yang biasa disebut load balancing dibutuhkan agar traffic dapat berjalan dengan baik, memaksimalkan throughput, dan menghindari overload pada server.

2. Dasar Teori

2.1 Software Defined Network

Software Defined Network (SDN) merupakan sebuah arsitektur baru dalam bidang jaringan komputer, memiliki karakteristik yang dinamis, manageable, costeffective, dan adaptable, sehingga sangat ideal untuk kebutuhan aplikasi saat ini yang bersifat dinamis dan yang memiliki bandwidth tinggi. Arsitektur Software Defined Network (SDN) memisahkan antara network control dan fungsi forwarding, sehingga network control tersebut dapat diprogram secara langsung. Dalam arsitektur Software Defined Network (SDN) terdiri dari tiga layer, Applicaton plane, Controller plane, dan Data plane [7].



Gambar 2.1 Gambaran umum Load-balancing

Gambar 2.1 merupakan arsitektur komputer yang digabungkan dengan dengan Load Balancing. Kemudian di bagi menjadi beberapa devices, yaitu :
1. Client, merupakkan pengguna yang menggunakan service pada server.
2. Load Balancer, teknik membagi distribusi beban trafik.
3. Server, bertugas untuk menyediakan layanan terhadap user yaitu sebuah web server.

2.2 Load Balancing

Load balancing merupakan salah satu mekanisme untuk membagi beban komputasi ke beberapa server. Load balancing bertujuan untuk optimasi sumber daya, memaksimalkan troughput, meminimkan waktu respon, dan menghindari pembebanan berlebihan di satu sumber daya. Menggunakan beberapa sumber daya komputasi juga dapat mengurangi kemungkinan tidak berfungsinya suatu layanan karena setiap sumber daya dapat saling menggantikan (redundant) [5].

2.3 POX Controller

POX adalah sebuah platform pengembangan open source untuk aplikasi Software Defined Network (SDN) yang berdasarkan pada bahasa pemrograman Python dan merupakan controller OpenFlow[10]. POX memungkinkan proses perancangan dan pembangunan jaringan yang lebih cepat, serta menjadi lebih umum digunakan dari pada pendahulunya NOX POX membutuhkan Python 2.7 untuk eksekusinya. Dalam prakteknya, juga dapat dijalankan menggunakan Python 2.6. POX dapat dijalankan di sistem operasi Windows, Mac OS, dan Linux (meskipun telah digunakan pada sistem lain juga). Banyak pengembangan dilakukan di Mac OS, sehingga hampir selalu digunakan dalam Mac. POX dapat digunakan dengan "standar" Python interpreter (CPython), tetapi juga mendukung PyPy.

Daftar fitur POX sebagai berikut:

1. Tampilan menyerupai python untuk antarmuka OpenFlow pada grafik kinerja POX.

2. Dapat dijalankan di semua system. Secara khusus dioperasikan untuk sistem operasi Linux, Mac OS, dan Windows.

3. Mendukung tampilan GUI dan visualisasi yang sama seperti NOX. Memiliki kinerja yang lebih baik dibandingkan dengan aplikasi NOX.

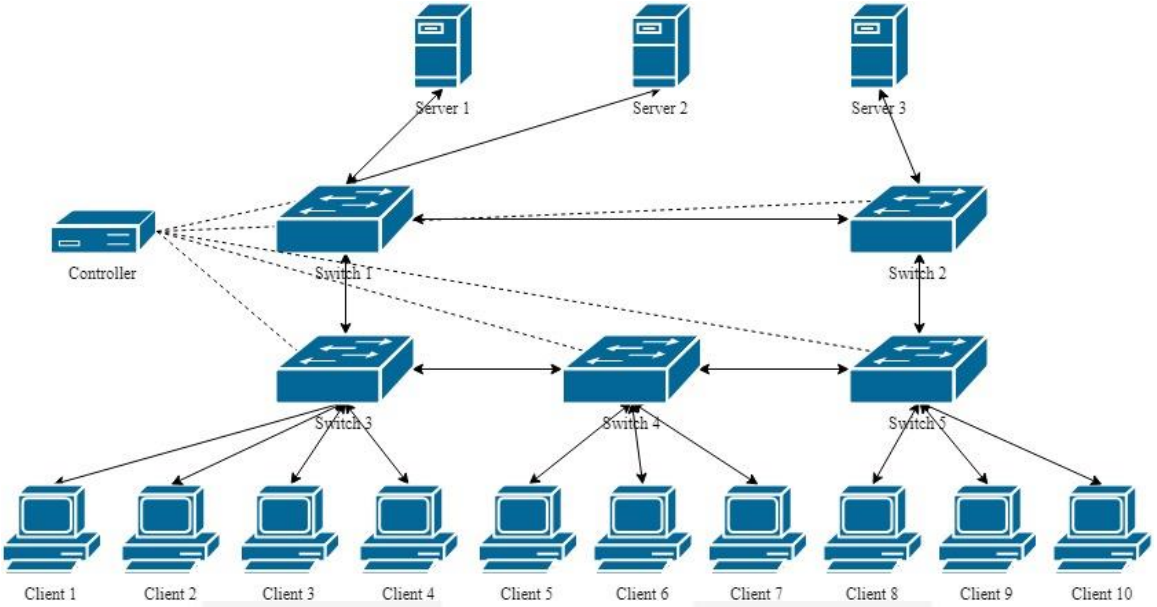
2.4 Mininet

Mininet merupakan sebuah emulator berbasis Command Line Interface (CLI) untuk membuat prototype jaringan yang berskala besar secara cepat pada sumber daya yang sangat terbatas.[1]. Mininet diciptakan dengan tujuan untuk mendukung riset dibidang SDN. Emulator Mininet sendiri memungkinkan untuk menjalankan sebuah kode secara interaktif tanpa harus memodifikasi kode tersebut.[10] Artinya, kode simulasi tersebut sama persis dengan kode pada real network environment. Mininet juga sebagai solusi yang dianggap paling unggul dalam berbagai hal seperti kemudahan penggunaan, akurasi dan skalabilitas. Mininet mampu menyediakan konfigurasi yang realitas dan mudah dengan harga yang murah. Dibandingkan dengan hardware testbed yang cenderung lebih mahal, sangat sulit untuk dikonfigurasi ulang, namun lebih akurat dibandingkan dengan menggunakan emulator Mininet.

3. Pembahasan

3.1 Gambaran Umum Sistem

Dalam Pada pengujian ini akan dirancang load balancing pada jaringan Software Definend Network (SDN) menggunakan algoritma optimasi koloni semut, dan penggunaan topologi tree pada jaringan SDN dan di jalankan dengan emulator mininet, konfigurasi menggunakan controller POX.



Gambar 3.1 Topologi Pengujian

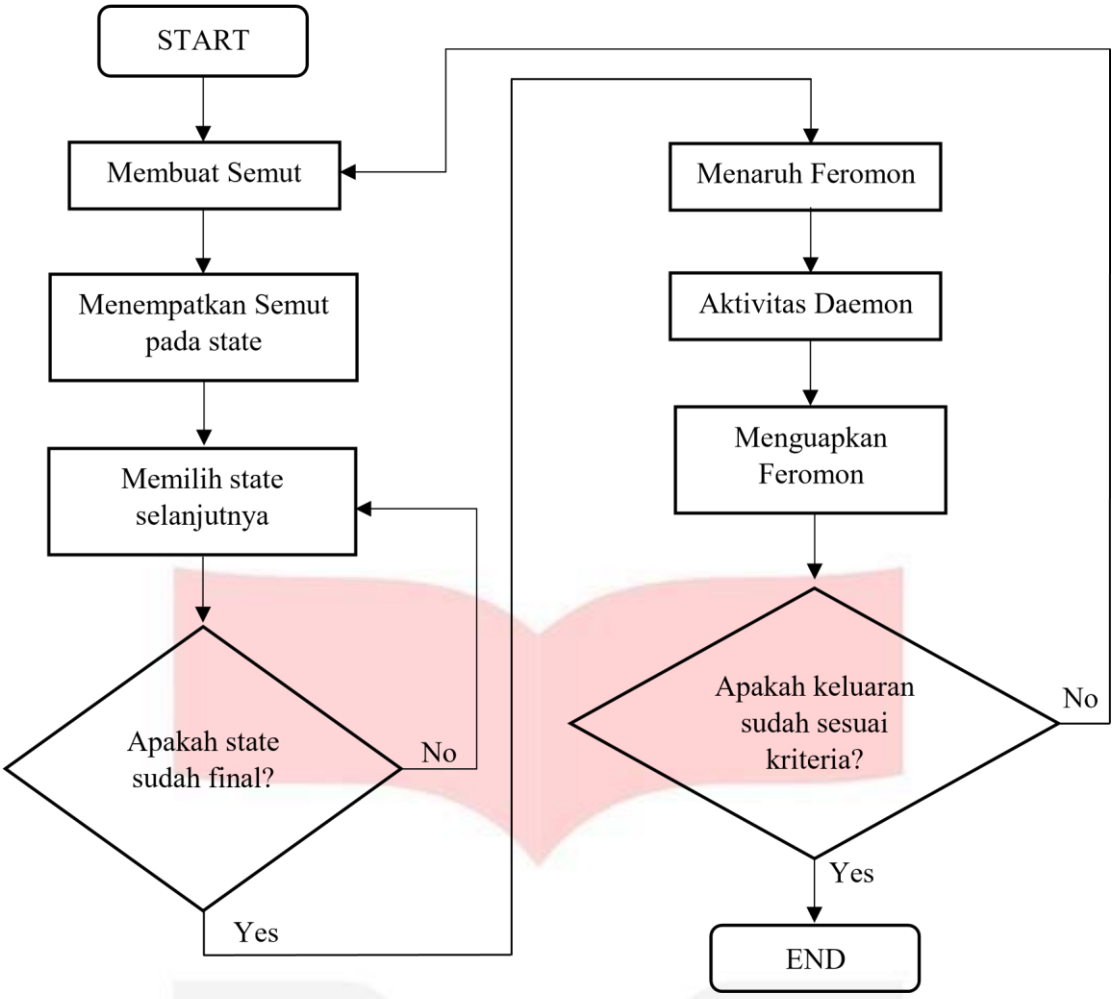
Gambar 3.1 merupakan gambaran umum dari load balance pada jaringan SDN yang akan di terapkan menggunakan algoritma optimasi koloni semut.

Berikut penjelasan gambar 3.1:

1. Client, mengirim request
2. Switch, perangkat ini bertujuan untuk meneruskan request dari controller.
3. POX Controller, bertugas utuk mengatur switch yang sedang mengirim request.
4. Server, bertugas sebagai penyedia layanan dari request client.

3.2 Perancangan Sistem

Pada perancangan analisis load balancing pada jaringan SDN yang di simulasikan pada tugas akhir adalah perancangan untuk meningkatkan performansi pada saat request masuk ke server yaitu dengan teknik load balancing, dan mengoptimalkan respon time yang ada, serta mngambil data dari utilitas CPU. Ruang lingkup analisis tugas akhir ini ialah pada emulator mininet dan perhitungan pada algoritma optimasi koloni semut jaringan SDN. Lingkup kerja dari tugas besar ini meliputi pengerjaan data analisis jaringan SDN terlebih dahulu, kemudian di lakukan tenik load balncing dan meneraprakan analisis pada algoritma optimasi koloni semut.



Gambar 3.2 Perancangan Sistem Optimasi koloni Semut

Pada Gambar 3.2 mempunyai penjelasan dan tahapan sebagai berikut:

1. Start adalah saat permulaan sistem perancangan beban yang akan di eksekusi.
2. Membuat Semut adalah memasukan data beban.
3. Menempatkan semut pada state adalah menempatkan beban pada state pertama.
4. Memilih state selanjutnya adalah mengirimkan beban pada state selanjutnya.
5. Jika state sudah final maka semut akan menaruh feromon jika tidak, kembali memilih state selanjutnya.
6. Menaruh feromon ditjukan untuk mencari jalur terpendek.
7. Aktivasi daemon merupakan proses pengiriman beban ke server dengan jalur terpendek dari server yang utilitasnya kecil.
8. Menguapkan feromon adalah penyimpanan traffic perjalanan semut.
9. Jika sudah sesuai kriteria maka data akan disimpan server, dan jika belum sesuai kriteria maka akan dikembalikan.
10. End adalah proses di akhir setelah semua eksekusi dan analisis selesai.

4. Implementasi dan Pengujian Sistem

Skenario pengujian merupakan perencanaan yang telah disusun secara rinci. Skenario pengujian dilakukan setelah perencanaan sudah dianggap siap. Sedangkan parameter pengujian adalah keluaran hasil dari skenario pengujian load balancing menggunakan algoritma optimasi koloni semut pada jaringan SDN untuk mencari nilai throughput dan cpu utilization.

Pengujian ini bertujuan untuk melihat hasil throughput dan utilitas cpu dari request yang dikirimkan client menuju server.

1. Throughput

Kecepatan actual pada sebuah jalur koneksi saat melakukan request dari client ke server. Formula untuk menghitung nilai throughput ialah:

$$\text{Throughput} = \frac{\text{Jumlah data yang dikirim1}}{Waktu\ pengiriman\ data} \qquad (1)$$

2. Utilitas CPU

Memonitoring utilitas dilakukan agar dapat melihat info penggunaan aktifitas proses, system dan juga aplikasi yang sedang berjalan saat client melakukan request pada server. Formula yang digunakan yaitu :

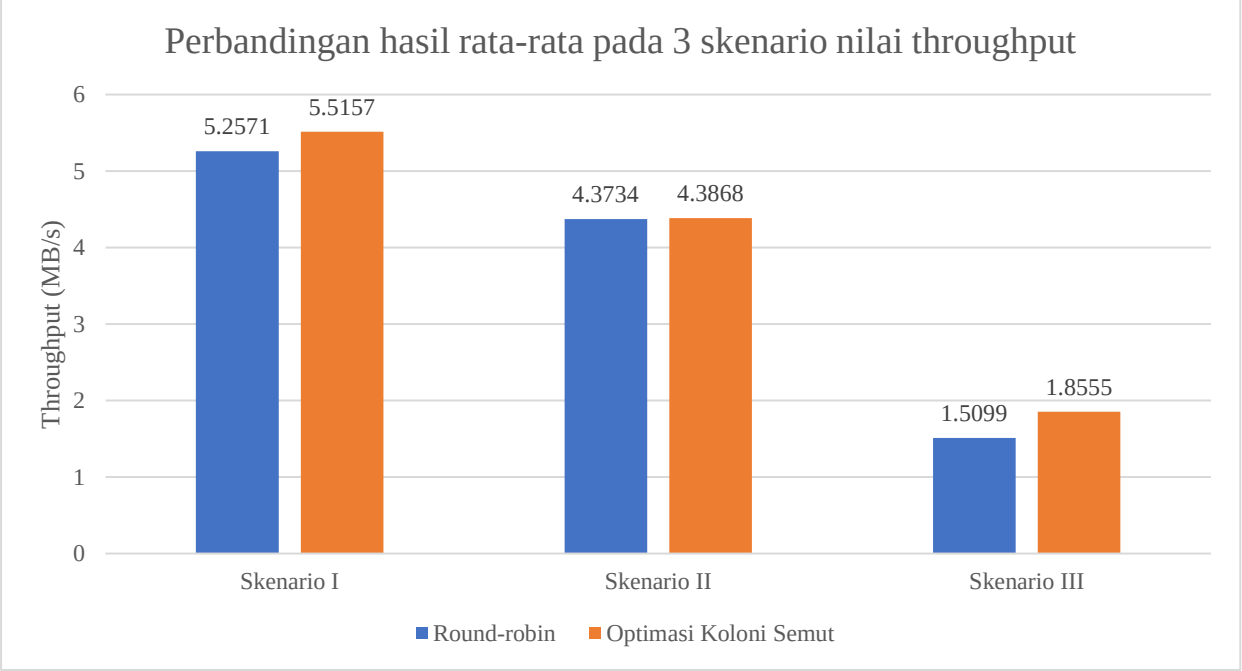
$$CPU\ Utilization = 100\% - \text{CPU idle}(\%) \qquad (2)$$

4.1 Pengujian Throughput

Pengujian pertama menggunakan skenario 3 server dan 10 client, 5 server dan 10 client, serta 10 server dana 50 client dengan memberikan bobot bandwidth pada jalur trafik sebesar 80 MBps. Jumlah request sebanyak 100 request dimana 1 request berisi data sebesar 10485760bit atau 10MB. Pertama-tama, untuk mendapatkan nilai data hasil throughput dibutuh kan nilai waktu penyelesaian (s). Setelah mendapatkan hasil waktu penyelesaian, jumlah data (MB) pada setiap request dibagi dengan waktu penyelesaian request maka nilai hasil throughput akan keluar. Pengujian pengambilan waktu penyelesaian pada algoritma optimasi koloni semut dengan 100 request yang dikirimkan dari client ke server. Hasil informasi data yang diberikan mengenai waktu penyelesaian dapat dilihat pada terminal yang mengacu dari hasil keluaran pada mininet.

Tabel 4.1 Nilai Hasil Throughput

Algoritma Pengujian	Round-robin	Optimasi Koloni Semut
Pengujian skenario I	5.2571	5.5157
Pengujian skenario II	4.3734	4.3868
Pengujian skenario III	1.5099	1.8555



Gambar 4.1 Nilai hasil rata-rata pada 3 skenario nilai throughput

Pada pengujian ini dapat disimpulkan bahwa nilai hasil throughput yang menggunakan algoritma optimasi koloni semut memiliki nilai throughput lebih besar dibandingkan nilai throughput yang dihasilkan oleh algoritma round-robin.
Nilai hasil throughput akan mengecil jika jumlah client semakin banyak, begitu pun sebaliknya, nilai throughput akan semakin besar jika client semakin sedikit. Begitu juga jika jumlah server semakin banyak maka nilai throughput akan semakin kecil, dan semakin sedikit jumlah server semakin besar nilai throughput yang didapatkan.

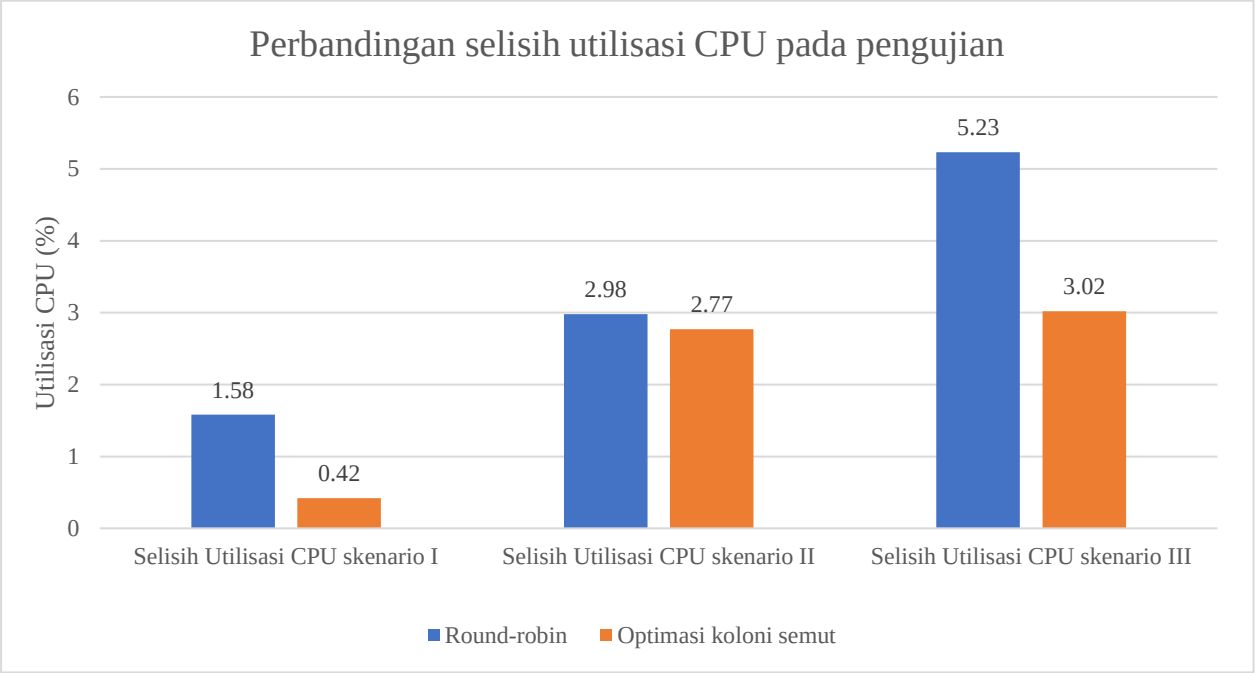
4.2 Pengujian CPU Utilization

Pengujian pertama menggunakan skenario 3 skenario dengan memberikan bobot bandwidth pada jalur trafik sebesar 80 Mbps. Jumlah request sebanyak 100 request dimana 1 request berisi data sebesar 10485760bit atau 10MB. Pada pengujian ini, didapat informasi data utilitas CPU sebagai berikut:

Tabel 4.2 Nilai Hasil Utilitas CPU

Algoritma Pengujian	Round-robin	Optimasi Koloni Semut
Selisih Utilisasi CPU skenario I	1.58	0.42
Selisih Utilisasi CPU skenario II	2.98	2.77
Selisih Utilisasi CPU skenario III	5.23	3.02

Dari tabel pengujian diatas dapat dilihat grafik perbandingannya dibawah ini :



Gambar 4.2 Perbandingan selisih utilisasi CPU pada pengujian

Berdasarkan grafik pada gambar 4.21, hasil dari kedua algoritma menyatakan bahwa utilisasi CPU yang dihasilkan algoritma optimasi koloni semut lebih merata dibandingkan utilisasi yang dihasilkan algoritma round-robin karena algoritma round-robin memiliki nilai selisih utilisasi CPU lebih besar pada tiap server nya.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan dari hasil Tugas Akhir ini, dapat ditarik beberapa kesimpulan yaitu :

1. Dari semua hasil pengujian nilai *throughput* pada algoritma optimasi koloni semut lebih besar dari algoritma round-robin.
2. Semakin banyak *client* semakin kecil hasil nilai *throughput* dihasilkan.
3. Semakin banyak *server*, semakin kecil nilai *throughput*.
4. Algoritma optimasi koloni semut menghasilkan nilai utilisasi CPU lebih merata dibanding algoritma round-robin.
5. Semakin rendah selisih utilisasi CPU semakin baik.
6. Semakin banyak *server* semakin besar selisih utilisasi CPU.
7. Algoritma Optimasi koloni Semut berjalan dengan baik dipengujian *load balancing* pada *software defined network*.

5.1 Saran

Berdasarkan dari Penelitian Tugas akhir ini maka, penulis menyarankan untuk penelitian selanjutnya adalah sebagai berikut :

1. Penelitian lebih lanjut dapat ditambah jumlah *server*, *client*, dan *request*-nya.
2. Penelitian lebih lanjut dapat menambah nilai hasil QoS.

3. Penelitian lebih lanjut dapat memakai sistem operasi terbaru dan spesifikasi PC yang tinggi.
4. Penelitian lebih lanjut dapat menggunakan monitor tambahan untuk mempermudah mengambil informasi.

Daftar Pustaka :

[1] Sachin Thakur, dan Saurabh Tripathi (2014): *Ant-based load balancing in telecommunications networks.*

[2] Kristiawan Nugroho (2015): *PENGGUNAAN ALGORITMA SEMU UNTUK PENENTUAN OPTIMISASI JALUR TIM MARKETING.*

[3] Pan Zhu, Jiangxing Zhang (2017): *Load balancing Algorithm for Web Server Based on Weighted Minimal Connections.*

[4] Dr. Mustafa El Gili Mustafa (2016): *LOAD BALANCING ALGORITHMS ROUND-ROBIN (RR), LEAST-CONNECTION, AND LEAST LOADED EFFICIENCY.*

[5] Tkachova O, Uzor Chinaobi Dan Abdulghafoor Raed Yahya (2016): *A Load balancing Algorithm for SDN.*

[6] Faris Keti, Shavan Askar (2016): *Emulation of Software defined networks Using Mininet in Different Simulation.*

[7] Izzatul Ummah, Desianto Abdillah (2016): Perancangan Simulasi Jaringan Virtual Berbasis Software-Define Networking.

[8] Isaac Keslassy Cheng-Shang Chang, Nick McKeown, Duan-Shin Lee (2017): *Optimal Load-Balancing.*

[9] Edwin Tenda, Imas Sukaesih Sitanggang, Baba Barus (2014): Optimasi Metaheuristik Koloni Semut untuk Solusi Masalah Jalur Terpendek pada Jaringan Jalan Riil.

[10] Wucai Lin, Lichen Zhang (2016): *The Load balancing Research of SDN based on Ant Colony Algorithm with Job Classification.*

[11] CISCO (2012) Edisi Redistributing Routing Protocols, diperoleh melalui situs internet: https://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/8606-redist.pdf. Diunduh pada 09 januari 2019.