ANALISIS LOAD BALANCING PADA JARINGAN SOFTWARE DEFINED NETWORK (SDN) MENGGUNAKAN ALGORITMA JARINGAN SYARAF TIRUAN (JST)

LOAD BALANCING ANALYSIS ON SOFTWARE DEFINED NETWORK (SDN) USING ARTIFICIAL NEURAL NETWORK (ANN) ALGORITHM

Andika Malraherawan Pradana[1], Tito Waluyo Purboyo[2], Roswan Latuconsina[3]

[1,2,3]Prodi S1 Sistem Komputer, Fakultas Teknik Elektro, Universitas Telkom

[1]dhikapradana@student.telkomuniversity.ac.id, [2]titowaluyo@telkomuniveristy.co.id, [3]roswan@telkomuniversity.ac.id

Abstrak

Pada perkembangan teknologi jaringan, dimana perkembangannya membuat kita lebih dimudahkan baik dalam membangun, memonitoring atau memelihara suatu jaringan komputer. Perkembangan pemakaian internet yang meningkat juga menyebabkan permintaan akan mutu layanan harus ditingkatkan. Tidak cukup hanya bisa terhubung ke internet, performa konektivitas menjadi faktor penting dalam penggunaan internet sekarang ini. Load balancing merupakan salah satu mekanisme untuk membagi beban komputasi ke beberapa server. Load balancing bertujuan untuk mengoptimalkan sumber daya, memaksimal-kan troughput, meminimkan waktu respon, dan menghindari pembebanan berlebihan di satu sumber daya. Tugas Akhir ini membahas tentang analisis jaringan Software Definend Network (SDN) untuk peningkatkan dan pengoptimalan yang di terapkan dengan menggunakan teknik Load Balancing.

Keywords: Experimental, Software Defined Networking, Load Balancing, Jaringan Syaraf Tiruan

Abstract

In the development of network technology, where development makes it easier for us to monitor and build computer networks. The Increased use of the internet will also result in improved service quality. It's not enough just to be able to connect to the internet, the performance also become a concern. Load balancing is one of the behaviors to divide the load into several servers. The purpose of load balancing is to allocate resources, maximize throughput, minimize response time, and avoid charging resources. This Final Project discusses the analysis of the Network Definend Network (SDN) network for enhancement and optimization applied using the Load Balancing technique.

Keywords: Experimental, Software Defined Networking, Load Balancing, Artificial Neural Network

1. Pendahuluan

Perkembangan jaringan internet terus melesat cepat, hal ini dapat dilihat dari semakin banyaknya pengguna yang terhubung ke jaringan, terutama jaringan internet, di sisi lain karena semakin banyaknya pengguna yang menggunakan jaringan internet untuk memenuhi kebutuhan akan perkembangan teknologi maka seringkali suatu jaringan internet mengalami overload dan crash yang disebabkan oleh banyaknya request yang dilakukan oleh pengguna [2]. Cara umum mengatasinya yaitu dengan menambahkan Server atau menambahkan harddisk tambahan untuk database, namun cara ini membutuhkan biaya yang cukup besar dan hanya sebagian kecil pengguna yang dapat menyediakannya. Maka ada salah satu solusi untuk mengatasi masalah jaringan tersebut ialah dengan teknik load balancing.

Load balancing merupakan salah satu mekanisme untuk membagi beban komputasi ke beberapa server. Load balancing bertujuan untuk mengoptimalkan sumber daya, memaksimal-kan troughput, meminimalkan waktu respon, dan menghindari pembebanan berlebihan di satu sumber daya. Teknik ini juga sering di gunakan untuk mendistribusikan beban traffic pada dua atau lebih jalur koneksi secara seimbang, agar traffic dapat berjalan secara optimal [1].

Adapun rumusan masalah dalam Tugas Akhir ini yaitu, Seberapa baik analisis dari utilitas server dengan menggunakan mekanisme Load balancing dengan algoritma jaringan syaraf tiruan pada topologi jaringan berdasarkan scenario perancangan.Dengan merujuk pada rumusan masalah, maka tujuan dari tugas akhir ini adalah Mengkaji dan menganalisis kinerja Load balancing Jaringan syaraf tiruan berdasarkan parameter Response Time, bandwith, dan Utilitas CPU,Mensimulasikan jaringan Software Defined Network dalam mekanisme Load Balancing dengan algoritma Jaringan syaraf tiruan menggunakan mininet.

Metode ini dilaksanakan dengan melakukan studi kepustakaan melalui membaca bukubuku, skripsi, dan jurnal yang dapat mendukung penulisan Tugas Akhir yang relevan mengenai Software Defined Network (SDN), Load-balancing dan Jaringan syaraf tiruan. Perancangan dan Implementasi system.

Analisis ini dilakukan untuk mengkaji kebutuhan sistem, meliputi analisis kebutuhan perangkat lunak yang akan di bangun termasuk didalamnya yaitu perancangan flowchart, dan perancangan sistem. Pengujian sistem dan Analisis Pada tahap ini dilakukan pengujian Load-balancing dengan algoritma Jaringan syaraf tiruan dan topologi jaringan yang digunakan.

2. Dasar Teori /Material dan Metodologi/perancangan

2.1 Software Defined Network

Software Defined Network (SDN) merupakan sebuah arsitektur baru dalam bidang jaringan komputer, memiliki karakteristik yang dinamis, manageable, cost-effective, dan adaptable [2] sehingga sangat ideal untuk kebutuhan aplikasi saat ini yang bersifat dinamis dan yang memiliki bandwidth tinggi.[4] Arsitektur Software Defined Network (SDN) memisahkan antara network control dan fungsi forwarding, sehingga network control tersebut dapat diprogram secara langsung.[3]. *Controller Plane* merupakan yang utama dalam jaringan SDN, dapat dijalankan secara terpisah dari *Data Plane*. Sedangkan Data Plane merupakan *hardware* jaringan yang terprogram secara khusus dan dikendalikan penuh oleh *Control Plane*.[5] Pada SDN tersedia *open interface* yang memungkinkan sebuah entitas *software* atau aplikasi untuk mengendalikan konekvitas dari sumber daya jaringan, mengendalikan *traffic* jaringan, serta melakukan modifikasi di *traffic* jaringan tersebut.

2.2 Load Balancing

Load balancing merupakan salah satu mekanisme untuk membagi beban komputasi ke beberapa server.[8] Load balancing bertujuan untuk optimalisasi 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).[10]Load balance dapat membagi trafik jaringan secara adil dan meminimalisir terjadinya overload pada salah satu server,[9] akan tetapi sering terjadi diskoneksi untuk aplikasi realtime dikarenakan perpindahan gateway pada setiap jaringan yang menuju ke server. Load balance juga merupakan teknik untuk mendistribusikan beban trafik pada dua atau lebih jalur koneksi secara seimbang, agar trafik dapat berjalan optimal, memperkecil waktu tanggap dan menghindari overload pada salah satu jalur koneksi [7]. Dan beberapa keuntungan yang diperoleh dari teknik Load Balancing sebagai berikut:

1.Flexibility

Server tidak lagi menjadi inti sistem dan resource utama, tetapi menjadi bagian dari banyak server yang membentuk cluster. Hal ini membuat bahwa performa per-unit dari cluster tidak terlalu diperhitungkan, tetapi performa cluster secara keseluruhan. Sedangkan untuk meningkatkan performa dan cluster, server atau unit baru dapat ditarnbahkan tanpa mengganti unit yang lama.

2.Scalability

Sistem tidak memerlukan desain ulang karena seluruh arsitektur sistem dapat mengadaptasikan sistem tersebut ketika terjadi perubahan pada komponen sistem. Untuk semua trafik yang melewati Load Balancer, aturan keamanan dapat dimplementasikan dengan mudah. Dengan Private Network digunakan untuk Real servers, alamat Ipnya tidak akan diakses secara langsung dari luar sistem cluster

3.High-availability

Load balancer dapat mengetahui kondisi Real server dalam sistem secara otomatis, jika terdapat real server yang mati maka akan dihapus dari daftan Real server, dan jika Real server tersebut kernbali aktif maka akan dimasukkan ke dalam daftar Real server.[7] Load balancer juga dapat dikonfigurasi redundant dengan Load Balancer yang lain.

2.3 Open Flow

OpenFlow adalah protokol paling utama pada SDN. Posisinya berada di antara Controller dan Forwarding (data plane). OpenFlow memungkinkan pengaturan routing dan pengiriman paket ketika melalui sebuah switch. Dalam sebuah jaringan, setiap switch hanya berfungsi meneruskan paket yang melalui suatu port tanpa mampu membedakan tipe protokol data yang dikirimkan. OpenFlow memungkinkan untuk mengakses dan memanipulasi Forwarding Plane secara langsung dari perangkatperangkat jaringan seperti switch dan router baik secara fisik maupun virtual. Nick McKeown et.al memberikan pembahasan yang rinci tentang latar belakang OpenFlow [4]. Ide dasarnya cukup sederhana, yaitu mengeksploitasi fakta bahwa switch dan router Ethernet yang paling modern mengandung flow table yang dijalankan pada tingkat linerate untuk mengimplementasikan firewall, NAT, QoS, dan juga untuk mengumpulkan data statistik. Flow table bisa berbeda implementasinya tergantung vendor dari perangkat keras tersebut. Akan tetapi terdapat beberapa common set of function (kumpulan fungsi atau primitive yang serupa) di antara berbagai switch dan router vendor-based tersebut. OpenFlow dalam hal ini menyediakan open protocol untuk memprogram flow-table pada berbagai switch dan router tersebut. Di lain pihak, vendor pun tidak dirugikan karena tidak perlu membuka mekanisme kerja internal dari produk switch atau router buatan masing-masing vendor. Untuk bisa menggunakan OpenFlow diperlukan Controller SDN yang mendukung jalannya protokol OpenFlow. Controller adalah salah satu platform pengembangan open source untuk aplikasi SDN yang mendukung protokol OpenFlow.

2.3.1 Route Flow

Route Flow terbentuk atas penggabungan proyek OpenFlow dengan routing engine Quagga. Sistem ini terdiri dari controller OpenFlow (RFProxy), RFClient dan Independent Server (RFServer) [10]. Tujuan utama dibuatnya RouteFlow adalah menerapkan virtualisasi IP routing secara terpusat, dengan memisahkan fungsi control-plane dan data-plane. Penelitian ini memanfaatkan RouteFlow sebagai sistem yang berjalan pada control-plane.

2.3.2 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.

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.

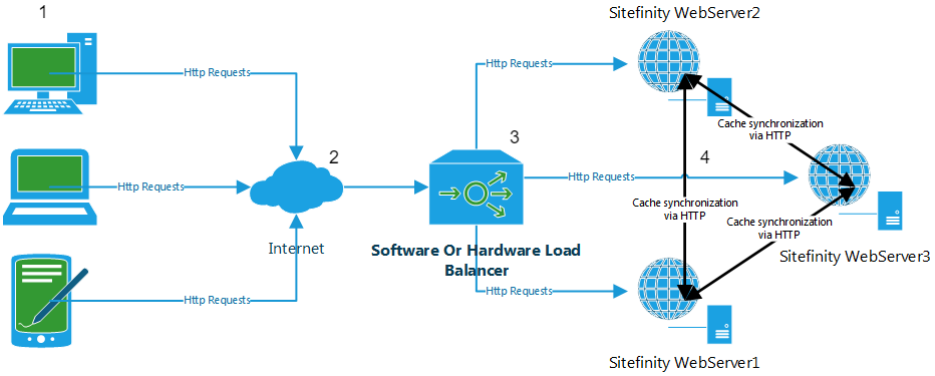
2.5 Metode Jaringan Syaraf Tiruan

Jaringan saraf tiruan merupakan sistem pemrosesan informasi yang memiliki kemampuan pembelajaran terhadap data dan informasi yang diterima, kemampuan untuk memodelkan fungsi linear, komputasi paralel, dan mempunyai sifat mentolerir ketidakpastian (fault tolerance).[3] Penerapan jaringan saraf tiruan sangatlah luas, diantaranya adalah dalam hal peramalan (forecasting), analisis data (data analysis), dan pengenalan pola.[4]
Jaringan syaraf tiruan (JST) juga salah satu perkembangan paling menarik dalam kecerdasan buatan dalam beberapa tahun terakhir, dan yang telah digunakan untuk memodelkan serta memprediksi sistem dinamis secara optimal.[3] JST telah menunjukkan kinerja yang sangat baik dalam hal mempelajari hubungan input-output untuk sistem nonlinier dan kompleks. Hubungan ini dapat ditemukan dengan cepat dan efisien dengan mengurangi kesalahan antara output jaringan dan keluaran sebenarnya. Outputnya bisa diprediksi hanya dalam beberapa detik. Model berbasis JST telah digunakan untuk memecahkan sejumlah masalah teknik di berbagai bidang, termasuk kontrol adaptif, pengenalan pola, robotika, pengolahan citra, diagnosis medis, deteksi kesalahan, pemantauan proses, energi terbarukan dan berkelanjutan, aplikasi laser dan identifikasi sistem nonlinier.

3. Pembahasan

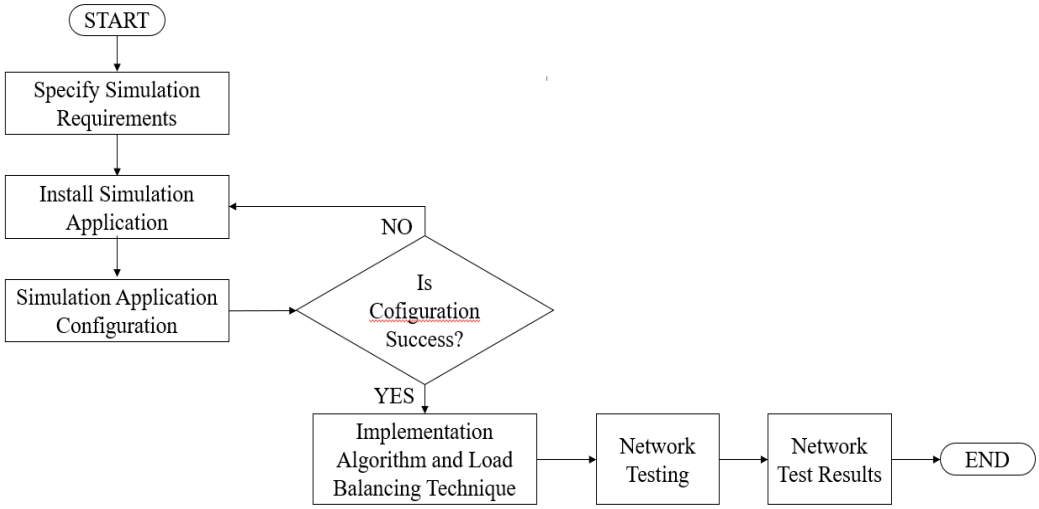
3.1 Gambaran Umum

Pada peneliatian ini akan dirancang sebuah sistem load balancing dengan jaringan Software Definend Network (SDN) menggunakan metode JST, dan penggunaan topologi single pada studi kasus jaringan SDN ini di jalankan dengan emulator mininet, di konfigurasi dengan RouteFlow serta controller POX. Dengan multiple gateway yang bertugas sebagai management jaringan sehingga koneksi jaringan SDN sendiri bisa berjalan secara optimal untuk masing-masing link. Maka dari itu metode JST (Jaringan Syaraf Tiruan) yang akan memecahkan analisis dalam proses system ini.



Gambar 1. Flowchart Skema yang Diusulkan

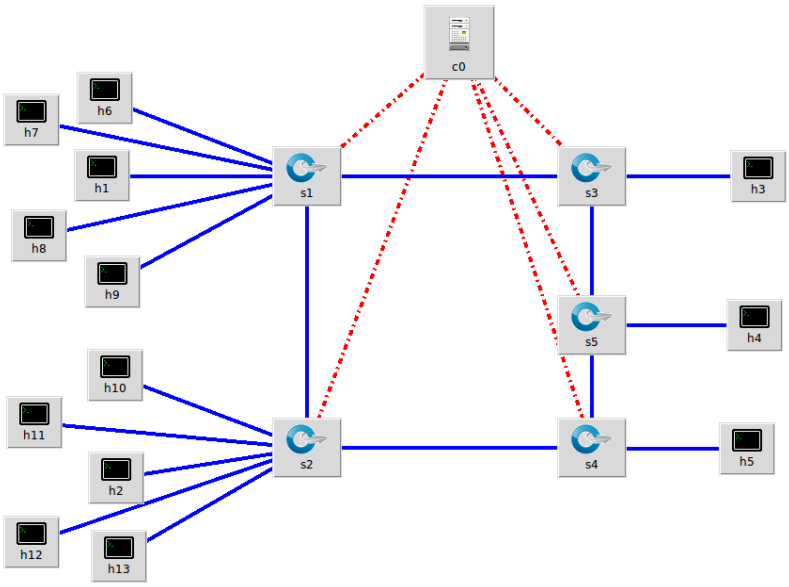
3.2 Flowchart Perancangan Sistem



Gambar 2. Flowchart Skema Sistem Perancangan

3.4 Sekenario dan Implementasi

Dalam realisasi system, akan dilakukan pengujian dengan parameter dan skenario yang akan dilakukan pengujian pada sebuah jaringan komputer dengan sekenario dan implementasi yang telah dirancang. jumlah server sebanyak 3 Pengujian.



Gambar 3. Gambaran skenario I server (3) dan client (10)

4.1 Parameter dan sekenario pengujian

Parameter dan sekenario pengujian merupakan perencanaan yang telah disusun secara rinci. Skenario pengujian dilakukan setelah perencanaan sudah dianggap siap. Sedangkan parameter pengujian adalah keluaran hasil dari scenario pengujian load balancing menggunakan algoritma optimasi koloni semut pada jaringan SDN.

4.1.1 Parameter Pengujian

Untuk menguji hasil dari Tugas Akhir yang di kerjakan, beberapa parameter yang di butuhkan untuk menunjang keberhasilannya adalah sebagai berikut :

1. Response Time

Waktu tanggap yang diberikan oleh interface ketika Client/Client mengirim permintaan ke server atau total waktu yang dibutuhkan mulai dari pengirim mengirimkan transmisi hingga respons dari penerima diterima oleh pengirim[8]. Pengujian response time ditujukan untuk mengetahui seberapa lama sebuah data berjalan dalam sebuah jaringan dari satu node ke node tujuan, dimana pada pengujian ini bertujuan untuk mengetahui reponse time dari tiap server. Perhitungan response time dilakukan dengan cara pengiriman paket HTTP dari host ke server. Pengujian dilakukan dengan parameter ini karena parameter Response Time merupakan ukuran kecepatan request dikirimkan oleh user dan respon server terhadap request dari user. Faktor yang mempengaruhi response time adalah jaringan antara user dan server. Semakin kecil nilai response time, semakin baik pula performa yang dihasilkan

T1 T2 Response Time = T1 - T2 [4] *1*

```
Time of first response(T1) - time of customer request(T2) =
Response Time
```

2. Cpu Utilization

Pada pengujian skenario ini dilakukan dengan rate low,mid,dan high dari resource data yang terkirim berdasarkan request dari client dan besar nilai CPU Usage server idle.

Pengujian ini juga untuk bertujuan monitoring besar utilitas sebuah Cpu yang terdapat di suatu server atau beberapa server yang akan di ujikan melalui request yang di berikan oleh setiap Client pada pengujian tugas akhir ini. CPU utilization dilakukan untuk mengetahui seberapa banyak resource yang digunakan oleh controller ketika menerapkan load balancing dengan algoritma yang akan diujikan. Nilai yang akan dimonitoring pada pengujian ini adalah Memory Utilization dari controller ketika controller belum diaktifkan, controller pada kondisi idle dan controller dengan mengimplementasikan algoritma ANN dan round robin. Penggunaan parameter ini berfungsi untuk mengetahui konsumsi resource pada Server ketika server menangani traffic dari tiap node pada jaringan SDN.

Monitoring penggunaan CPU pada server dapat menggunakan perintah

```
~$ mpstat
```

atau

```
~$ glance
```

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

U = 100% - CPU Idle (% of time spent in idle task)[4] 2

```
~$ Wget (Untuk Get data/ download data)
~$ host ab -n server http://ip.address/
~$ curl -X POST -d @file server:port
(Ab Apache dan Curl = untuk Analisa data pengganti
```

4.1.2 Skenario Pengujian

Skenario pengujian dilakukan dengan membuat topologi tree dengan spesifikasi system yang berbeda pada jumlah Client dan Server yang masing masing sekenario mempunyai 1 controller dan 5 switch dengan dilakukan penyesuaian host untuk membuat server dan client. Pada pengujian pertama akan diuji menggunakan skenario berisikan, Pengujian Response time dan Utilitas CPU pada Algoritma Artificial Neural Network dan Round Robin dibagi menjadi 3 pengujian yaitu:

1. **Pengujian 1**
   3 server dan 10 client dengan jumlah request 100 request
2. **Pengujian 2**
   5 server dan 10 client dengan jumlah request 100 request
3. **Pengujian 3**
   10 server dan 50 client dengan jumlah request 100 request

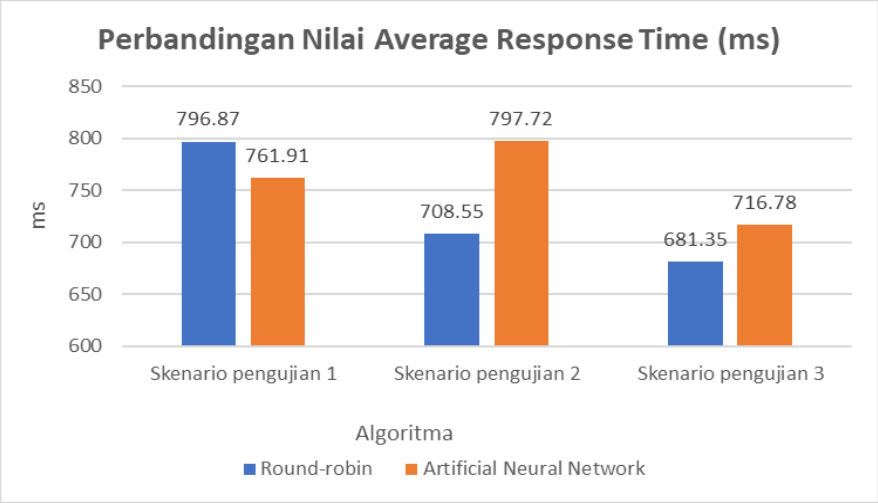
Pengujian ini bertujuan untuk melihat request yang ditujukkan dari Client kepada Server yang di Analisa melalui Path, apakah terbagi sesuai dengan kriteria dari Algoritma ANN . Pada simulasi ini server akan diberikan commend http server pada mininet melalui Xterm (emulator terminal dari mininet) dimana pada aplikasi ini server akan melayani permintaan paket HTTP melalui port 80 dan user melakukan permintaan dengan perintah

4.2.1 Skenario Pengujian

Pengujian skenario pertama menggunakan scenario 3 server dan 10 client dengan memberikan bobot bandwidth pada jalur trafik sebesar 100 Mbps, delay pada setiap switch 1 (sec) dan Host 2 (sec). Jumlah request sebanyak 100 request dimana 1 request berisi data sebesar 10485760bit atau 10MB. Tahap awal yang di lakukan untuk mendapatkan nilai data hasil Response Time dibutuh kan nilai Bandwith terlebih dahulu untuk melihat hasil time of client request . Setelah mendapatkan hasil Bandwith, maka kalkulasi response time (ms) bisa di lakukan dengan langusng menggunakan tool Ab apache dan Curl mininet.

A. Pengujian Respon Time

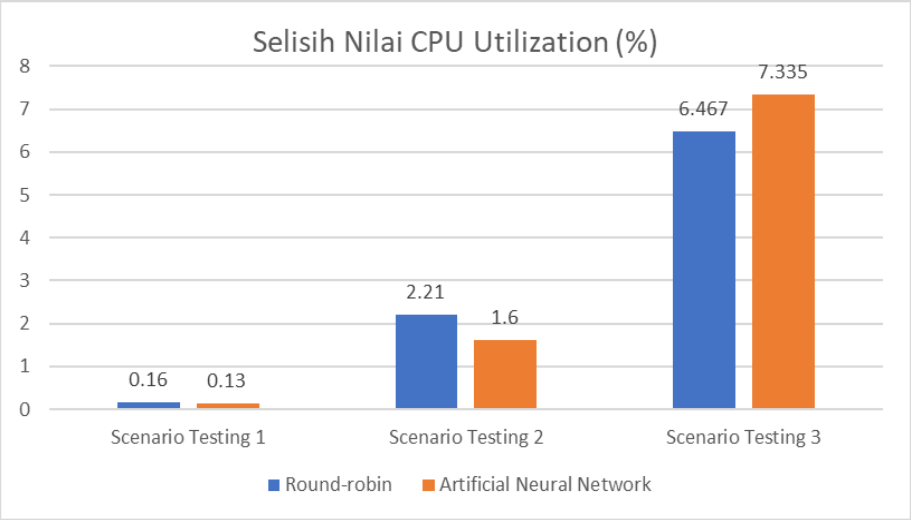
Pengujian pengambilan data Response Time dengan 100 request yang dikirimkan dari client ke server. Hasil informasi data yang diberikan mengenai Response time dapat dilihat pada terminal yang mengacu pada Ab Apache dan curl mininet.
Hasil data yang disajikan pada gambar grafik sebagai berikut:



Gambar 4. Pengujian hasil perbandingan Response Time

B. Pengujian utilitas CPU

Pengujian pengambilan data Utilitas CPU dengan 100 request yang dikirimkan dari client ke server. Hasil informasi data yang diberikan mengenai Utilitas CPU dapat dilihat pada terminal yang mengacu pada terminal mininet.
Hasil data yang disajikan pada gambar grafik sebagai berikut:



Gambar 5. Pengujian hasil selisih utilitas CPU

5.1 Kesimpulan

Berdasarkan hasil pengujian maka Analisa ini dapat disimpulkan bahwa semakin besar request yang dilakukan maka nilai dari response time semakin besar juga. Hal ini disebabkan server menerima beban yang semakin banyak sehingga diperlukan waktu yang besar untuk memproses setiap request dari client.

Dan Algoritma Artificial neural network dapat di terapkan dan di simulasikan pada Software Defined Network dengan mekanisme load balancing pada *path* menggunakan Controller POX untuk pengujian parameter, kemudian penelitian tugas akhir ini dapat di simpulkan sebagai berikut :

1. Response Time pada server dengan scenario pengujian yang beragam hanya mengalami sedikit perbedaan antar satu server dengan server yang lain.
2. Dari hasil pengujian skenario I,II, dan III yaitu semakin kecil nilai dari response time, maka semakin baik hasil pengujian,
3. Dari hasil pengujian skenario I,II, dan III yaitu semakin besar nilai bandwith dari tiap host/client maka semakin kecil nilai response time pada server.
4. Semakin banyak server, maka semakin kecil nilai dari Response Time
5. Pada scenario pengujian I, hasil pengujian response time dari algoritma ANN lebih baik dari pada algoritma RR, namun pada scenario II dan III, Response time algoritma Round-robin mendapatkan hasil yang lebih kecil yang berarti lebih baik hasilnya di banding dengan algoritma Artificial Neural Network.
6. Algoritma Artificial neural network menghasilkan nilai utilitas CPU lebih rendah dibanding algoritma Round-robin, namun lebih stabil hasil dari algoritma Round-robin.

5.2 Saran

Saran yang di harapkan guna membantu analisis kedepan agar penilitian lebih baik dan akurat yaitu :

1. Penelitian lebih lanjut di tambah algoritma ntuk perbandingan antara algoritma Artificial Neural Network (ANN) dengan algoritma lain terhadap parameter Cpu Utilization untuk memperlihatkan apakah algoritma yang lain berpengaruh terhadap *response time* dan memberikan analisis yang lebih baik
2. Apabila melakukan menggunakkan Virtualisasi OS diharapkan menggunakan komputer atau laptop dengan Spesifikasi yang memumpuni.
3. Penelitian lebih lanjut dapat ditambah jumlah *server*, *client*, dan *request*-nya.
4. Penelitian lebih lanjut dapat menggunakan monitor tambahan untuk mempermudah mengambil informasi.

Daftar Pustaka:

[1] I Made Widhi Wirawan, Komang Tris Sumarianta. 2012. IMPLEMENTASI Load Balance Pada Jaringan Multihoming Menggunakan Router Dengan Metode Round Robin. Indonesia : Jurnal Ilmu Komputer Universitas Udayana

[2] Pan Zhu1, Jiangxing Zhang. 2017. Load Balancing Algorithm for Web Server Based on Weighted Minimal Connections. Journal of Web Systems and Applications (2017) Vol. 1, Numuber 1 Clausius Scientific Press, Canada
[3] Nada M. Al Sallami, Ali Al daoud, Sarmad A. Al Alousi. 2013. Load Balancing with Neural Network. (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 4, No. 10, 2013.Jordan
[4] Cui Chen-xiao and Xu Ya-bin. 2016. Research on Load Balance Method in SDN. School of Computer, Beijing Information Science and Technology University,: International Journal of Grid and Distributed Computing Vol. 9, No. 1 (2016), pp.25-36
[5] Daphne Tuncer, Marinos Charalambides, Stuart Clayman, and George Pavlou. 2014. Adaptive Resource Management and Control in Software Defined Networks. International Paper, Department of Electronic and Electrical Engineering at University College London, UK
[6] Shavan Askar.2016. Adaptive Load Balancing Scheme For Data Center Networks Using Software Defined Network. Journal of University of Zakho, Vol. 4(A) , No.2, Pp, 2016
[7] Ivan Hidayah, Indrarini Dyah, Yuli Sun Hariyani. 2017. Implementasi Rip Pada Jaringan Berbasis Software Defined Network (Sdn). Bandung : Tugas Akhir Universitas Telkom, e-Proceeding of Applied Science : Vol.3, No.2 Agustus 2017
[8] S. Mondal, A. Bandyopadhyay. 2014. Performance Prediction of Software Defined Network Using an Artificial Neural Network. Chinese Control Decis. Conf., no. 2, pp. 1– 4, 2013.
[9] Raden Arief Setyawan. 2015. Analisis Implementasi Load Balancing dengan Metode Source Hash Scheduling pada Protocol SSL. Jurnal EECCIS Vol. 8, No. 2, Desember 2014.
[10] Fortinet, 2015. The Basics of Server Load Balancing and the Evolution to Application Delivery Controllers. E-book White Paper. Vol 5 – 2015
[11] Dwinson Sitohang, 2016. Implementasi Load-Balancing Dengan Metode Round Robin Dalam Software Defined Networking (Sdn) Menggunakan Controller Pox. University Of Sumatera Utara Institutional Repository (Usu-Ir).